CSI 709/CSS 739
Verification and Validation of Models

Statistics and Visualization Techniques for Verification and Validation

Dr. Hamdi Kavak
Computational and Data Sciences Department

http://www.hamdikavak.com
hkavak@gmu.edu

Soon

Original photo: https://knowyourmeme.com/photos/1326438-ben-swolo

Soon

Original photo: https://knowyourmeme.com/photos/1326438-ben-swolo

Next week (SWA)

- Short writing assignment (SWA 1) due at noon (12 pm), Monday (Sep 18).
  - http://hamdikavak.com/course-v-and-v/docs/content/05runtime-verification/
  - Choose one of the papers and submit your document on Blackboard.

Assigned Reading (choose one for your SWA 1):

- Ahrens, J., Heitmann, K., Petersen, M., Woodring, J., Williams, S., Fasel, P., ... & Geveci, B. (2010). Verifying scientific simulations via comparative and quantitative visualization. IEEE Computer Graphics and Applications, 30(6), 16-28.
- Feldkamp, N., Bergmann, S., & Strassburger, S. (2020, January). Visualization and Interaction for Knowledge Discovery in Simulation Data. In Proceedings of the 53rd Hawaii International Conference on System Sciences.

Next week (final project proposal)

- Final project proposal due at 11:59 pm, Monday (Sep 18).
- Check what is of expected
  - http://hamdikavak.com/course-v-and-v/docs/assignments/03final-project/
- Feel free to ask questions after class or schedule a virtual meeting.

Paper presentation assignment

- See:
  Blackboard
  -> Assignments
  -> Paper Presentations (2 per student)
  -> Paper presentation assignment

Short recap of V&V terminology

Verification: have we built the model right?

Validation: have we built the right model?

Cook, D. A., & Skinner, J. M. (2005). How to perform credible verification, validation, and accreditation for modeling and simulation. *The Journal of Defense Software Engineering*, *18*(5), 20-24.

Objective of this lecture

- Getting familiar with some basic statistical and visualization techniques for verification and validation.
- Topics include
  - Uncertainty and random number generators
  - Confidence intervals
  - Visual inspection of model data

Python packages used in this lecture

- NumPy
- Matplotlib
- Scipy

Uncertainty

- uncertain: ”not known or definite” (Oxford Dictionary)

Uncertainty

- uncertain: ”not known or definite” (Oxford Dictionary)
- Many real-world systems contain uncertainty
  - E.g.: traffic, queues, weather, voting, package delivery, disease spread,…

Uncertainty

- uncertain: ”not known or definite” (Oxford Dictionary)
- Many real-world systems contain uncertainty
  - E.g.: traffic, queues, weather, voting, package delivery, disease spread,…
- Can be of varying degrees

Uncertainty

- uncertain: ”not known or definite” (Oxford Dictionary)
- Many real-world systems contain uncertainty
  - E.g.: traffic, queues, weather, voting, package delivery, disease spread,…
- Can be of varying degrees
- If one or more components of a system has uncertainty, we call such systems “**stochastic**.”

Uncertainty

- uncertain: ”not known or definite” (Oxford Dictionary)
- Many real-world systems contain uncertainty
  - E.g.: traffic, queues, weather, voting, package delivery, disease spread,…
- Can be of varying degrees
- If one or more components of a system has uncertainty, we call such systems “**stochastic**.”
- We use a **probabilistic framework** to model a stochastic system and call the model as a **stochastic model**.

Some sources of uncertainty

Imperfect knowledge

(e.g., small sample, resolution)

Changes in the environment

(e.g., weather, human decisions)

Time dependency

(e.g., different traffic patterns at different times, store visits)

Presence of noise

(e.g., measurement precision or accuracy)

Failure

(e.g., power outage, defect)

Deterministic vs. stochastic models

Deterministic

- For the same input, each model run gives the same result.
- Assumes no uncertainty.

Stochastic

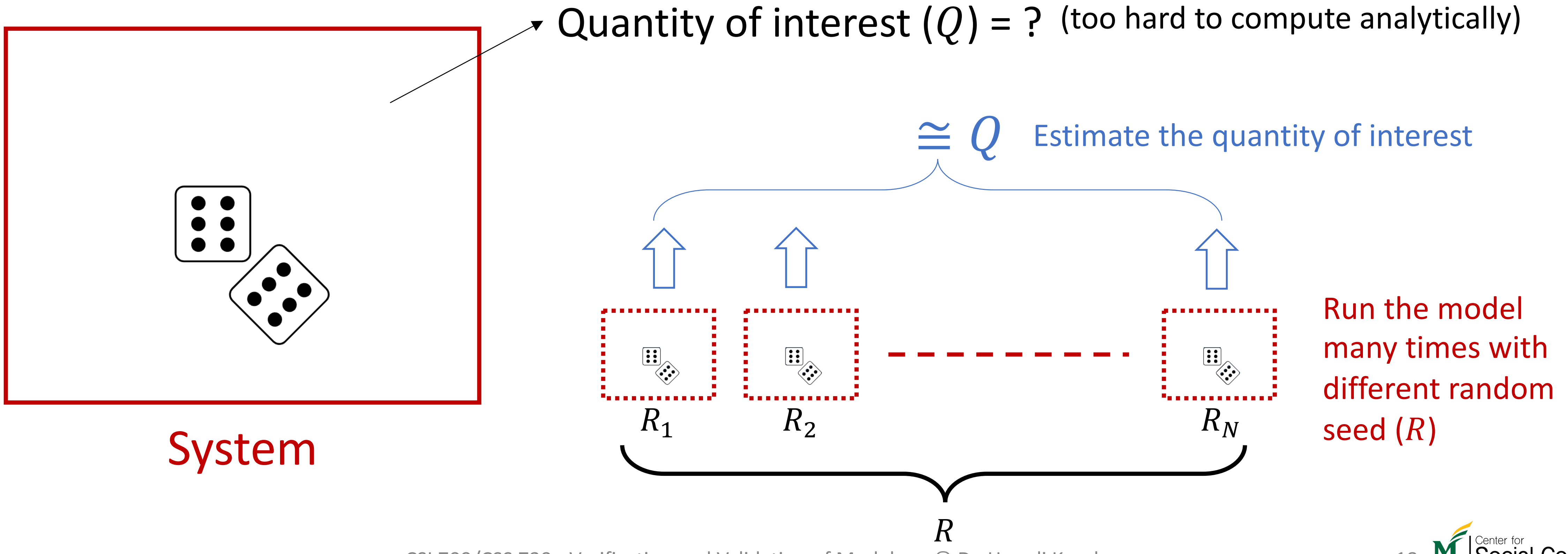
- For the same input, each model run gives a different result.
- Assumes uncertainty.
- We can still reproduce the same results if we want.

Random variables

- Stochastic systems contain components that are modeled by **random variables**.
- Random variables (e.g., X) are used to quantify the outcome of stochastic processes.
- We can technically call a stochastic model a “random variable”
- Two main types
  - Discrete (i.e., countable number of possible values)
    - E.g.: 5, 16, 8, 210, …
  - Continuous (i.e., cannot count the number of possible values)
    - E.g.: 4.5234, 99.12, 3.14, 7.78, …

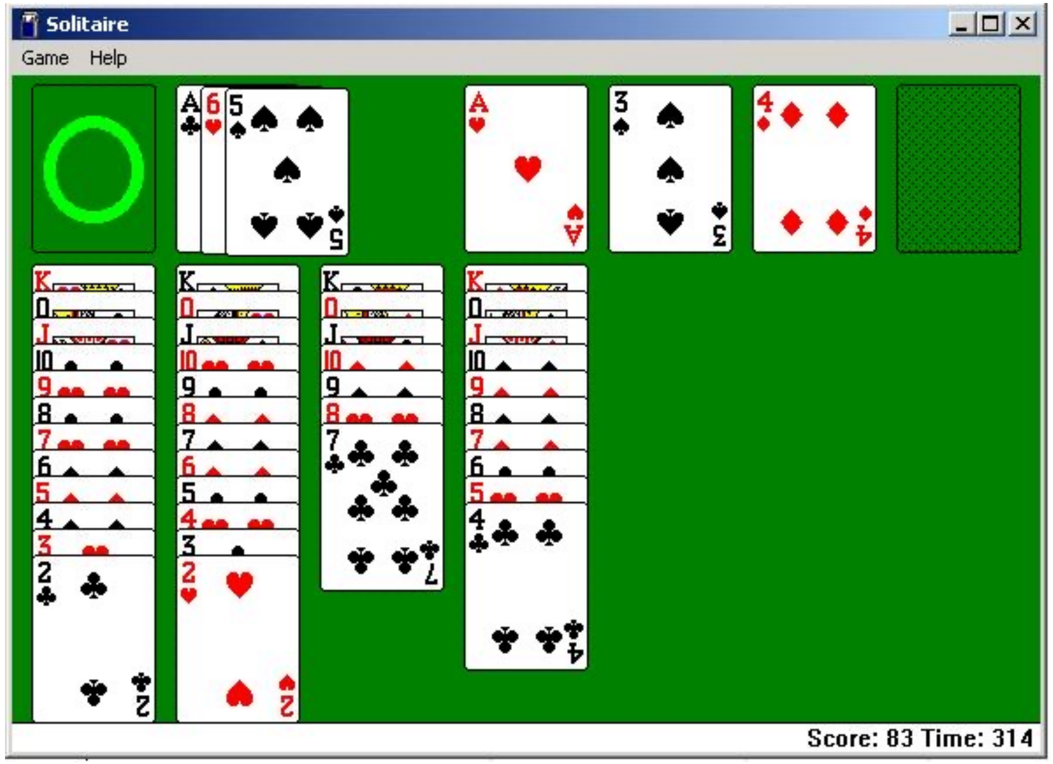
Stochastic models

- Most stochastic systems we deal do not have a closed form solution
- We approximate them by developing stochastic models and running multiple times. A.k.a.?



Do you like Solitaire?

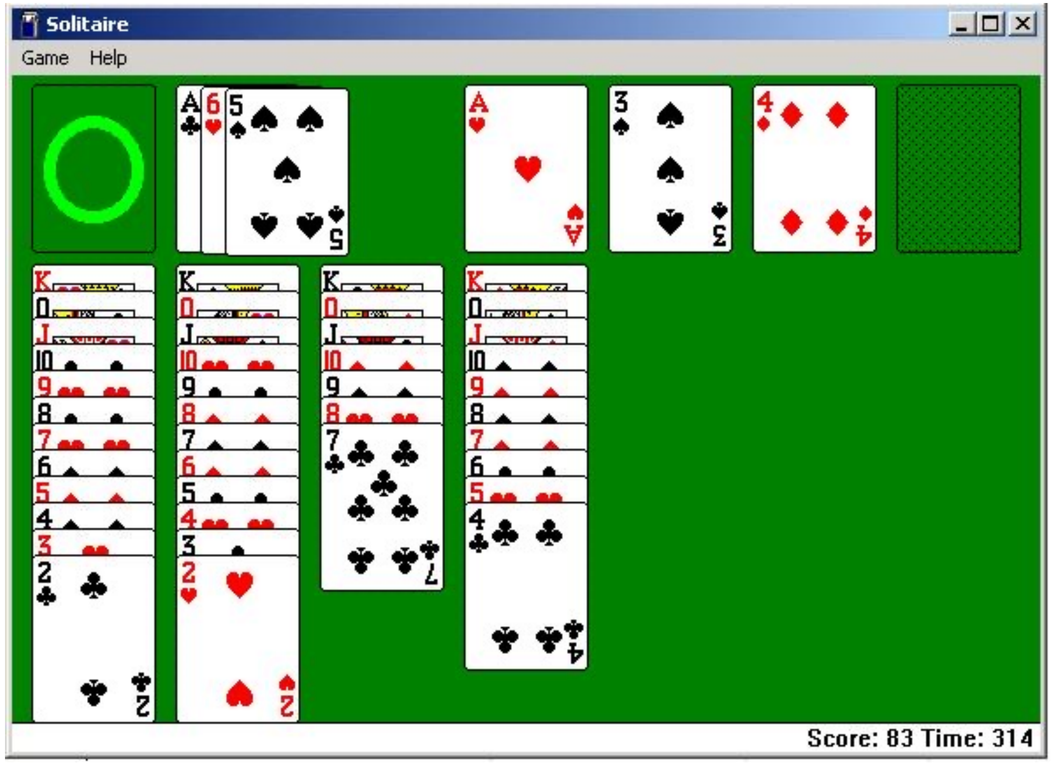
What is the chance of winning the solitaire game before seeing the cards?



Source: https://www.flickr.com/photos/bobb/43826727

Do you like Solitaire?

What is the chance of winning the solitaire game before seeing the cards?

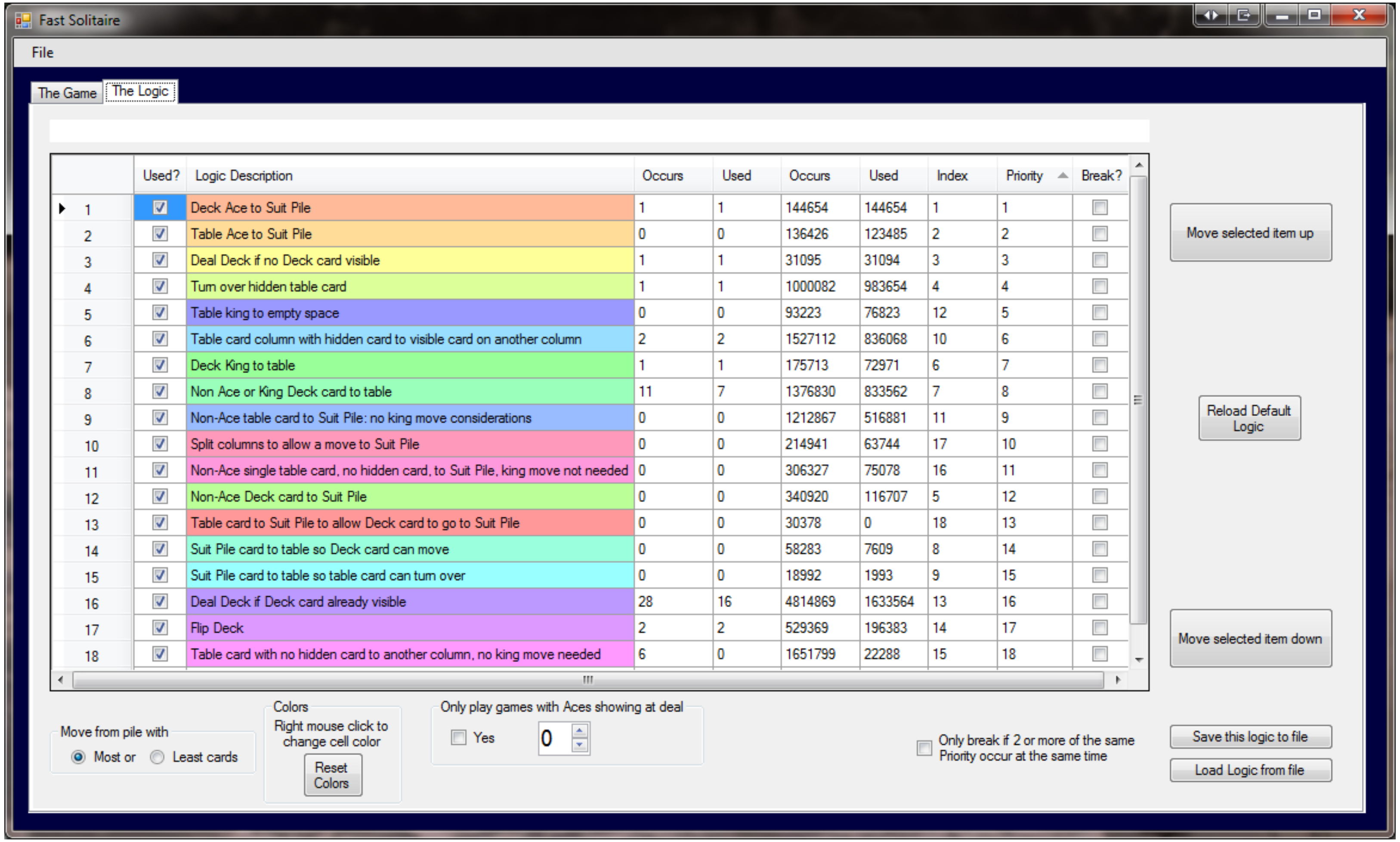


Source: https://www.flickr.com/photos/bobb/43826727

After playing 100,000,000 games, it is found that only 8.7% of cases win*.

*According to "Bill's Solitaire Tester" using the deal 3 cards and 3 times around the deck option

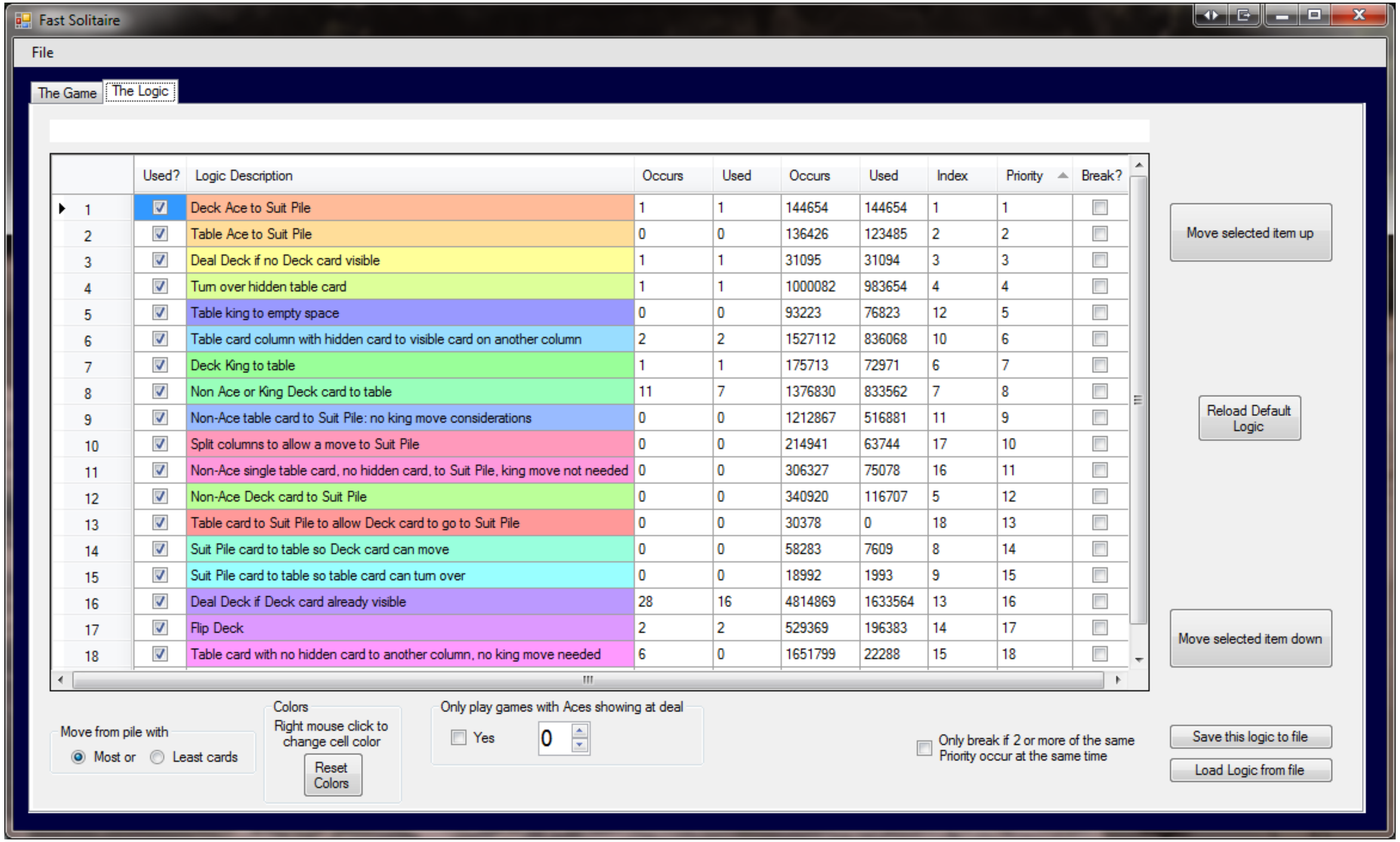
Strategy used in the solitaire game



	Used?	Logic Description	Occurs	Used	Occurs	Used	Index	Priority	Break?
1	☑	Deck Ace to Suit Pile	1	1	144654	144654	1	1	☐
2	☑	Table Ace to Suit Pile	0	0	136426	123485	2	2	☐
3	☑	Deal Deck if no Deck card visible	1	1	31095	31094	3	3	☐
4	☑	Turn over hidden table card	1	1	1000082	983654	4	4	☐
5	☑	Table king to empty space	0	0	93223	76823	12	5	☐
6	☑	Table card column with hidden card to visible card on another column	2	2	1527112	836068	10	6	☐
7	☑	Deck King to table	1	1	175713	72971	6	7	☐
8	☑	Non Ace or King Deck card to table	11	7	1376830	833562	7	8	☐
9	☑	Non-Ace table card to Suit Pile: no king move considerations	0	0	1212867	516881	11	9	☐
10	☑	Split columns to allow a move to Suit Pile	0	0	214941	63744	17	10	☐
11	☑	Non-Ace single table card, no hidden card, to Suit Pile, king move not needed	0	0	306327	75078	16	11	☐
12	☑	Non-Ace Deck card to Suit Pile	0	0	340920	116707	5	12	☐
13	☑	Table card to Suit Pile to allow Deck card to go to Suit Pile	0	0	30378	0	18	13	☐
14	☑	Suit Pile card to table so Deck card can move	0	0	58283	7609	8	14	☐
15	☑	Suit Pile card to table so table card can turn over	0	0	18992	1993	9	15	☐
16	☑	Deal Deck if Deck card already visible	28	16	4814869	1633564	13	16	☐
17	☑	Flip Deck	2	2	529369	196383	14	17	☐
18	☑	Table card with no hidden card to another column, no king move needed	6	0	1651799	22288	15	18	☐

Source: http://www.billturnbull.com/solitaire/Logic.html

Strategy used in the solitaire game



	Used?	Logic Description	Occurs	Used	Occurs	Used	Index	Priority	Break?
1	☑	Deck Ace to Suit Pile	1	1	144654	144654	1	1	☐
2	☑	Table Ace to Suit Pile	0	0	136426	123485	2	2	☐
3	☑	Deal Deck if no Deck card visible	1	1	31095	31094	3	3	☐
4	☑	Turn over hidden table card	1	1	1000082	983654	4	4	☐
5	☑	Table king to empty space	0	0	93223	76823	12	5	☐
6	☑	Table card column with hidden card to visible card on another column	2	2	1527112	836068	10	6	☐
7	☑	Deck King to table	1	1	175713	72971	6	7	☐
8	☑	Non Ace or King Deck card to table	11	7	1376830	833562	7	8	☐
9	☑	Non-Ace table card to Suit Pile: no king move considerations	0	0	1212867	516881	11	9	☐
10	☑	Split columns to allow a move to Suit Pile	0	0	214941	63744	17	10	☐
11	☑	Non-Ace single table card, no hidden card, to Suit Pile, king move not needed	0	0	306327	75078	16	11	☐
12	☑	Non-Ace Deck card to Suit Pile	0	0	340920	116707	5	12	☐
13	☑	Table card to Suit Pile to allow Deck card to go to Suit Pile	0	0	30378	0	18	13	☐
14	☑	Suit Pile card to table so Deck card can move	0	0	58283	7609	8	14	☐
15	☑	Suit Pile card to table so table card can turn over	0	0	18992	1993	9	15	☐
16	☑	Deal Deck if Deck card already visible	28	16	4814869	1633564	13	16	☐
17	☑	Flip Deck	2	2	529369	196383	14	17	☐
18	☑	Table card with no hidden card to another column, no king move needed	6	0	1651799	22288	15	18	☐

Source: http://www.billturnbull.com/solitaire/Logic.html

Credits to Bill Turnbull
www.billturnbull.com

who reminds the young generation how websites looked like in the 90s.

True legend!

Quantifying model uncertainty

Confidence intervals (CIs)

- CIs are used to quantify the uncertainty of a model based a sample of observations (or model runs).

Confidence intervals (CIs)

- CIs are used to quantify the uncertainty of a model based a sample of observations (or model runs).
- We need three quantities
  - Confidence level (e.g., 90%)
  - Point estimate (e.g., sample mean)
  - Margin of error – to be calculated
    - Use t-score when sample size < 30; use z-score otherwise.

Confidence intervals (CIs)

- CIs are used to quantify the uncertainty of a model based a sample of observations (or model runs).
- We need three quantities
  - Confidence level (e.g., 90%)
  - Point estimate (e.g., sample mean)
  - Margin of error – to be calculated
    - Use t-score when sample size < 30; use z-score otherwise.
- Confidence interval = $Sample\ statistic \pm Margin\ of\ error$

Precision vs. reliability in CIs

- For instance, confidence interval for height of adults
  - 5’8” $\pm$ 0.01” (precise but not very reliable)
  - 5’8” $\pm$ 1.00’ (not very precise but reliable)

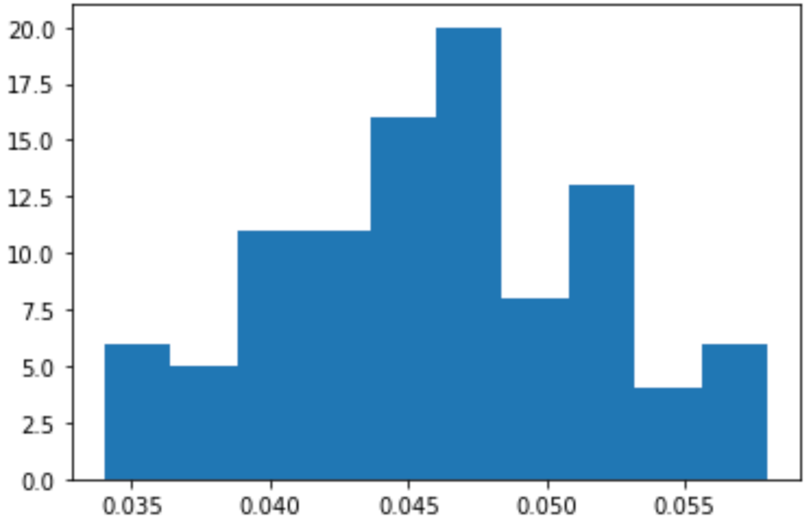
margin of error

mean

Confidence intervals example

```
model_results = np.array([0.047, 0.044, 0.043, 0.042, 0.04, 0.04
```

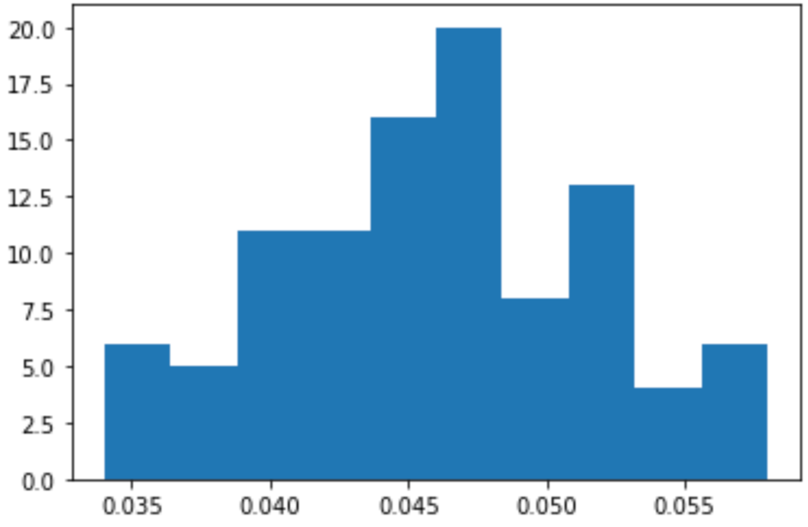
```
plt.hist(model_results)
plt.show()
```



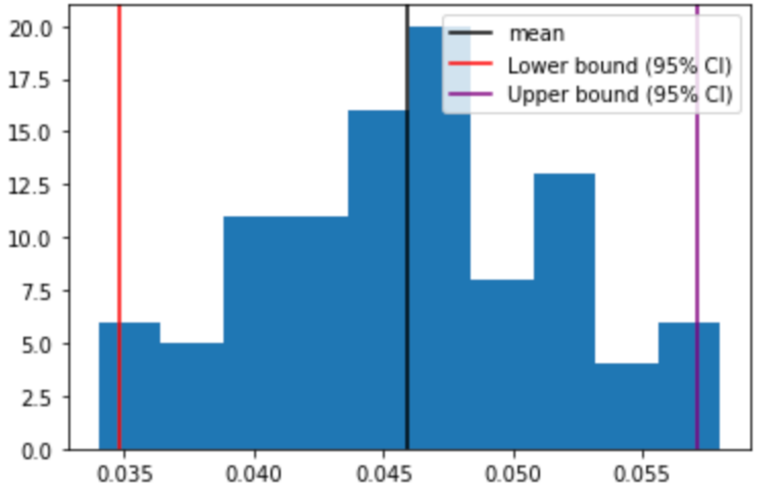
Confidence intervals example

```python
model_results = np.array([0.047, 0.044, 0.043, 0.042, 0.04, 0.04
```

```python
plt.hist(model_results)
plt.show()
```

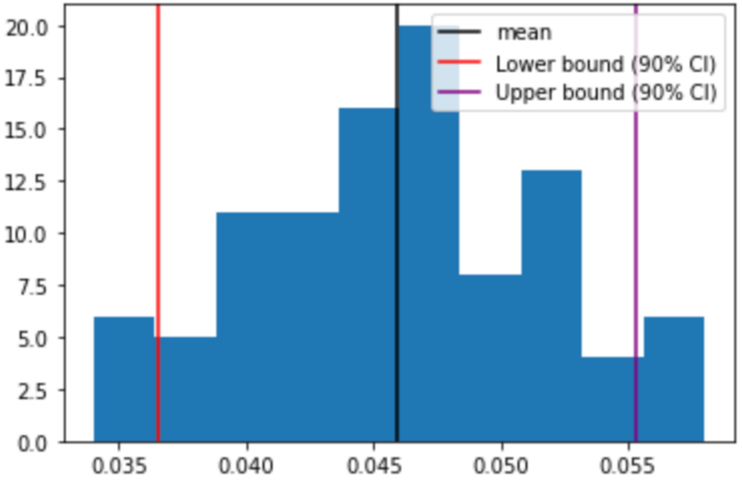



```python
# for 95% confidence interval
m = np.mean(model_results)
std = np.std(model_results)
lb = m - 1.96*std
ub = m + 1.96*std

plt.hist(model_results)
plt.axvline(x=m,color="black",label="mean")
plt.axvline(x=lb,color="red",label="Lower bound (95% CI)")
plt.axvline(x=ub,color="purple",label="Upper bound (95% CI)")
plt.legend()
plt.show()
```

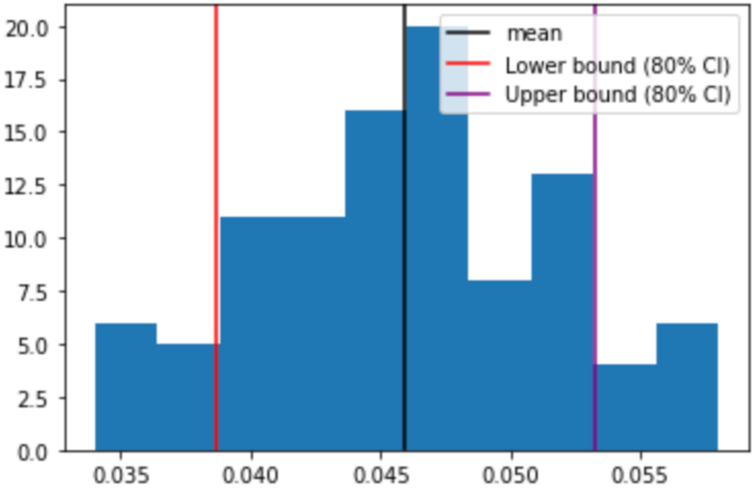


Confidence intervals example

```python
# for 90% confidence interval
m = np.mean(model_results)
std = np.std(model_results)
lb = m - 1.645*std
ub = m + 1.645*std

plt.hist(model_results)
plt.axvline(x=m,color="black",label="mean")
plt.axvline(x=lb,color="red",label="Lower bound (90% CI)")
plt.axvline(x=ub,color="purple",label="Upper bound (90% CI)")
plt.legend()
plt.show()
```




```python
# for 80% confidence interval
m = np.mean(model_results)
std = np.std(model_results)
lb = m - 1.282*std
ub = m + 1.282*std

plt.hist(model_results)
plt.axvline(x=m,color="black",label="mean")
plt.axvline(x=lb,color="red",label="Lower bound (80% CI)")
plt.axvline(x=ub,color="purple",label="Upper bound (80% CI)")
plt.legend()
plt.show()
```



Where to use confidence intervals?

- Inspect the output of the quantity of interest.
  - E.g., a **component of a model** or the **entire model output**.
- Inspect observations collected from the real world.
  - Can be related to **model input parameters** (e.g., age distribution of the population) or **expected output values** (e.g., number of people died of COVID-19).

Where to use confidence intervals?

- Inspect the output of the quantity of interest.
  - E.g., a **component of a model** or the **entire model output**.
- Inspect observations collected from the real world.
  - Can be related to **model input parameters** (e.g., age distribution of the population) or **expected output values** (e.g., number of people died of COVID-19).

- Things to check
  - How precise or reliable?
  - Difference from the real world measurements?
  - Is it chaotic?

Introducing uncertainty in models

Pseudo-random number generators (RNGs)

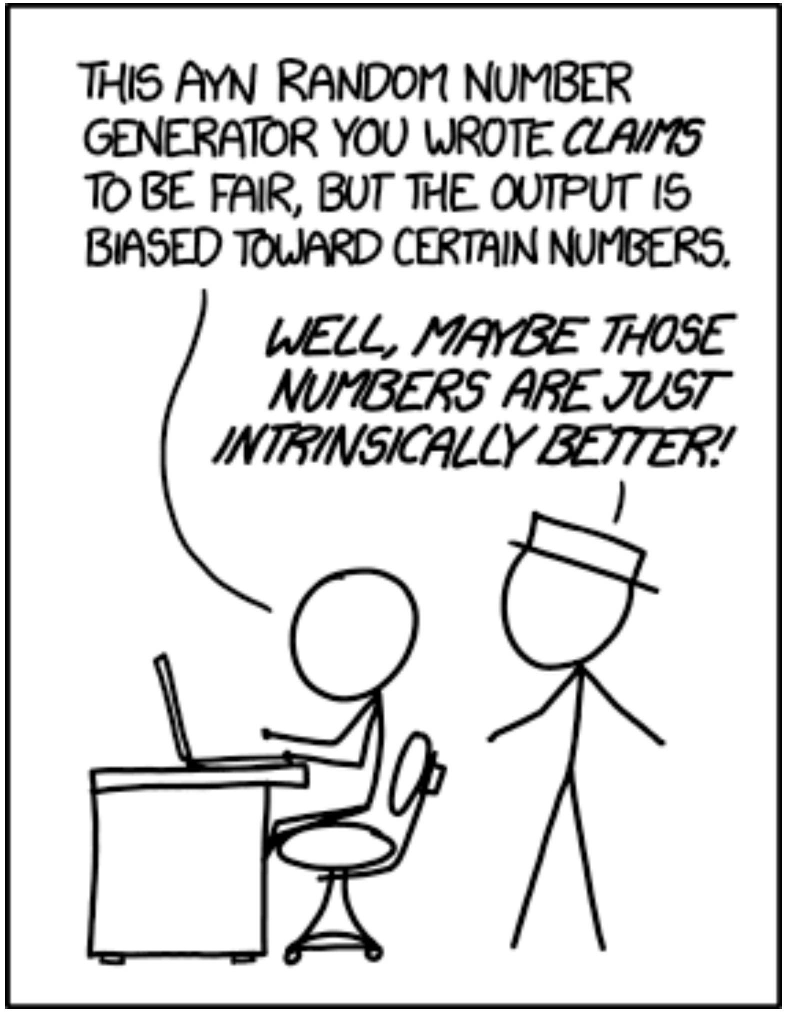
- A family of algorithms that can generate "random-looking" numbers.
- Different from physical random number generators (e.g., hardware that use thermal noise to generate random number).
- Given initial parameters, one can reproduce the entire sequence of numbers to be generated.
- All major programming languages provide RNGs.

Important features of RNGs

Randomness	• not biased towards certain values or distributions
Controllability	• “ability to reproduce its output, if desired”
Portability	• ”ability to produce the same output on a variety of computer systems”
Efficiency	• “speed, with minimal computer-resource requirement”
Documentation	• “theoretical analysis and extensive testing”

Leemis and Park (2006)

RNGs



Source: https://www.explainxkcd.com/wiki/index.php/1277:_Ayn_Random

RNGs

- Given a ***seed*** value, we can regenerate the exact sequence of numbers.
- A simple technique: Linear Congruential Generator (LCG) – Lehmer (1951).
  - $x_i = (a\, x_{i-1} + c)\ mod\ m$
    - a: multiplier
    - m: modulus
    - c: increment
    - x_0: the seed
  - One of the standard RNG in programming languages
    - Java still uses it w/ parameters $a = 25214903917, m = 2^{48}, c = 11$
- Another popular one is Mersenne Twister (Matsumoto 1998)

RNGs example

- Python's NumPy package and built-in random package implement the Mersenne Twister

Import the NumPy package

```
import numpy as np
```

RNGs example

- Python’s NumPy package and built-in random package implement the Mersenne Twister

Import the NumPy package

```python
import numpy as np
```

Generate a value between 0 and 1 = U(0,1)

```python
np.random.random()
```

```
0.5488135039273248
```

RNGs example

- Python's NumPy package and built-in random package implement the Mersenne Twister

Import the NumPy package

```python
import numpy as np
```

Generate a value between 0 and 1 = U(0,1)

```python
np.random.random()
```

```
0.5488135039273248
```

Generate an array of five values U(0,1)

```python
np.random.random(5)
```

```
array([0.71518937, 0.60276338, 0.54488318, 0.4236548 , 0.64589411])
```

RNGs example

- Python's NumPy package and built-in random package implement the Mersenne Twister

Import the NumPy package

```
import numpy as np
```

Generate a value between 0 and 1 = U(0,1)

```
np.random.random()
```

```
0.5488135039273248
```

Generate an array of five values U(0,1)

```
np.random.random(5)
```

```
array([0.71518937, 0.60276338, 0.54488318, 0.4236548 , 0.64589411])
```

Generate a matrix U(0,1)

```
np.random.random((2,3))
```

```
array([[0.43758721, 0.891773  , 0.96366276],
       [0.38344152, 0.79172504, 0.52889492]])
```

How critical RNGs in model reproducibility

A personal story

Statistical distributions

- Can be ***continuous*** or ***discrete***

(*NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/)

Statistical distributions

- Can be ***continuous*** or ***discrete***
- Usually, we fit a statistical distribution to data to understand the underlying mechanism that generates the data.
  - Often need to try many different distributions and check the **goodness of fit**.

(*NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/)

Statistical distributions

- Can be ***continuous*** or ***discrete***
- Usually, we fit a statistical distribution to data to understand the underlying mechanism that generates the data.
  - Often need to try many different distributions and check the **goodness of fit**.
- ***Probability density function (PDF)*** is used for continuous statistical distributions to describe the likelihood of sampling values from the value space.
  - ***Probability mass function (PMF)*** is used for discrete distributions

(*NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/)

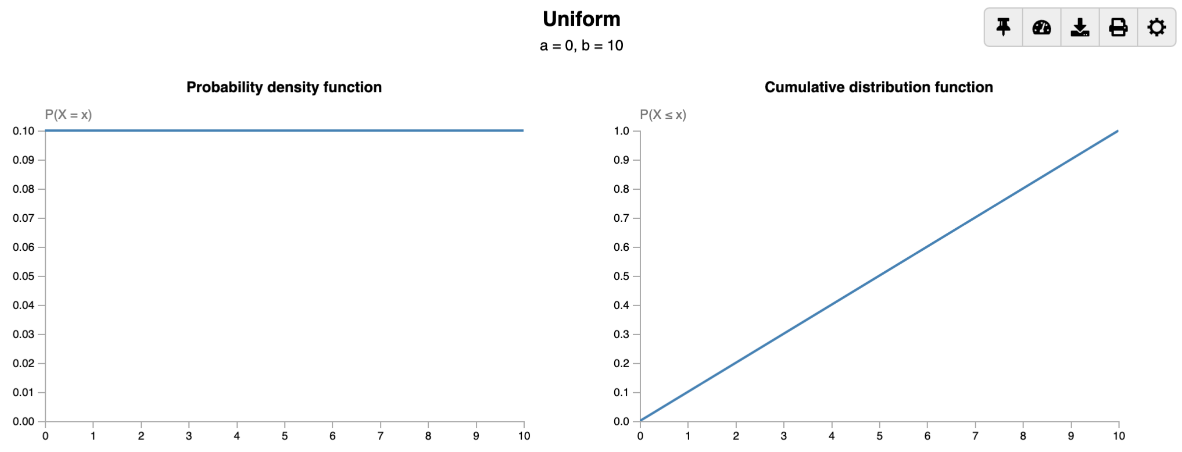
Statistical distributions

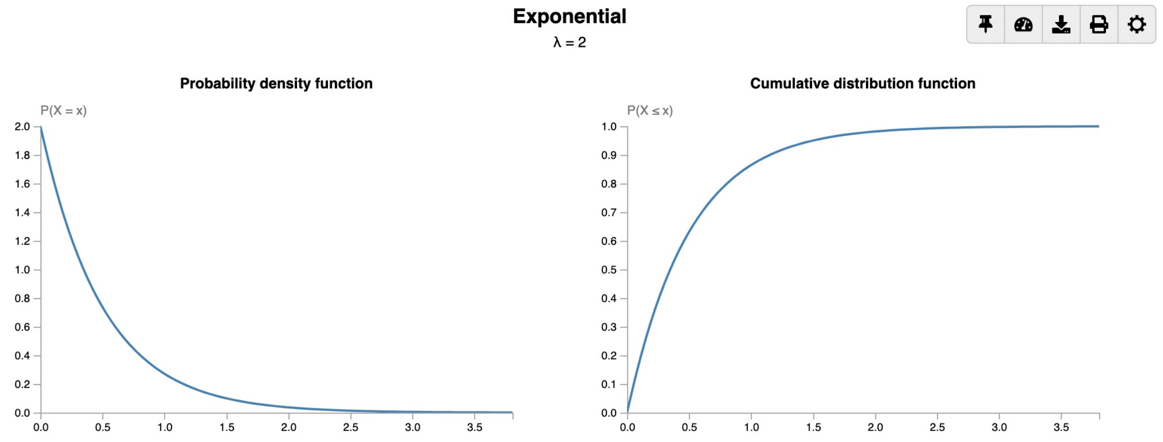
- Can be ***continuous*** or ***discrete***
- Usually, we fit a statistical distribution to data to understand the underlying mechanism that generates the data.
  - Often need to try many different distributions and check the **goodness of fit**.
- ***Probability density function (PDF)*** is used for continuous statistical distributions to describe the likelihood of sampling values from the value space.
  - ***Probability mass function (PMF)*** is used for discrete distributions
- ***Cumulative distribution function (CDF)*** is to describe "the probability that the variable takes a value less than or equal to" given value. In the continuous case, CDF can be calculated by integrating its PDF.

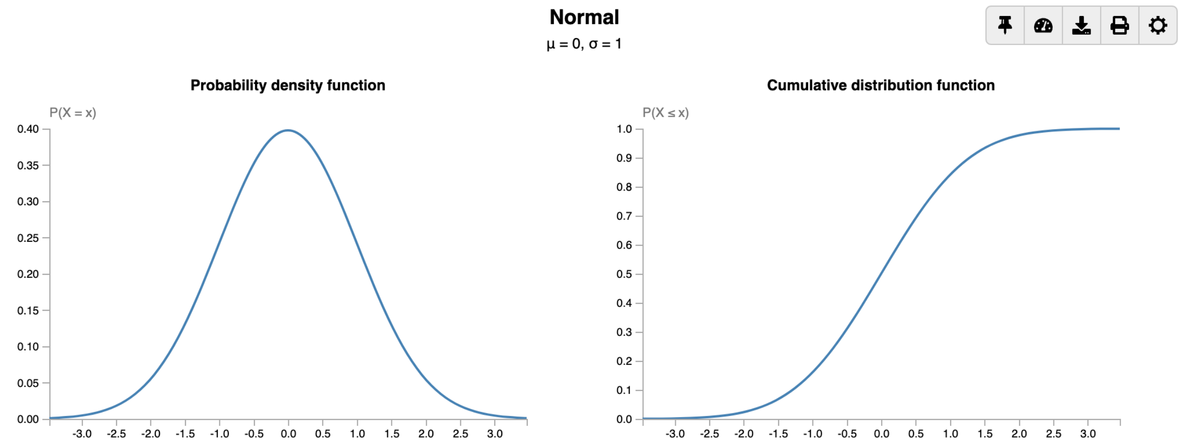
(*NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/)

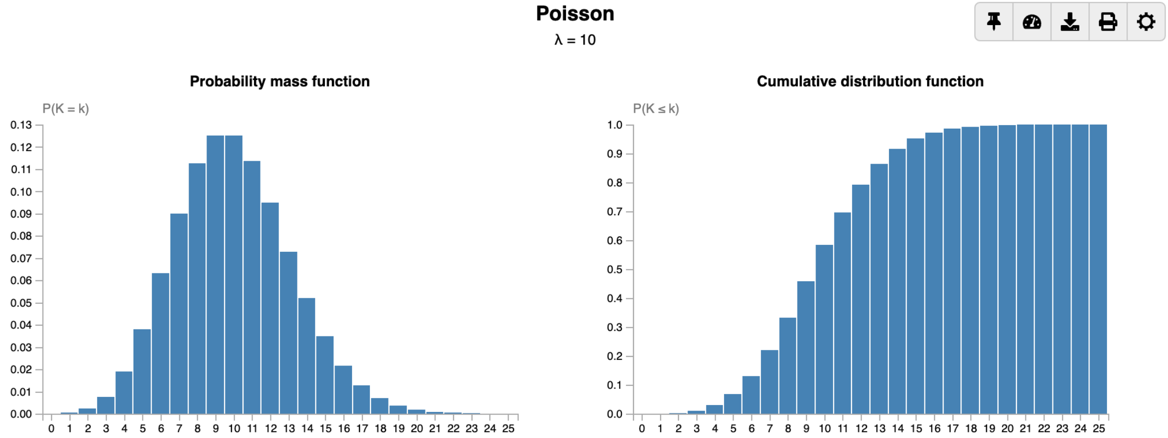
PDF (PMF) and CDF examples

For more: https://statdist.com/









Sampling from statistical distributions example

Uniform distribution within 0-10 interval

```
np.random.uniform(0,10,size=10)
```

```
array([3.78653645, 9.68804991, 8.22695094, 2.36785104, 8.32063758,
       0.15156402, 7.01776218, 6.63535641, 0.73149233, 7.88035412])
```

Sampling from statistical distributions example

Uniform distribution within 0-10 interval

```
np.random.uniform(0,10,size=10)
```

```
array([3.78653645, 9.68804991, 8.22695094, 2.36785104, 8.32063758,
       0.15156402, 7.01776218, 6.63535641, 0.73149233, 7.88035412])
```

Normal distribution with mean=0 and standard deviation=5.

```
np.random.normal(0,5,size=10)
```

```
array([-0.34837998,  1.85523413,  4.78192638,  0.00827601, -0.42549727,
       -2.91218348, -5.89590159,  1.03210198,  2.46722228, -3.45734937])
```

Sampling from statistical distributions example

Uniform distribution within 0-10 interval

```
np.random.uniform(0,10,size=10)
```

```
array([3.78653645, 9.68804991, 8.22695094, 2.36785104, 8.32063758,
       0.15156402, 7.01776218, 6.63535641, 0.73149233, 7.88035412])
```

Normal distribution with mean=0 and standard deviation=5.

```
np.random.normal(0,5,size=10)
```

```
array([-0.34837998,  1.85523413,  4.78192638,  0.00827601, -0.42549727,
       -2.91218348, -5.89590159,  1.03210198,  2.46722228, -3.45734937])
```

Poisson distribution with lambda=10.

```
np.random.poisson(10,size=10)
```

```
array([ 6, 20,  6, 11,  8,  4, 12, 12,  6, 10])
```

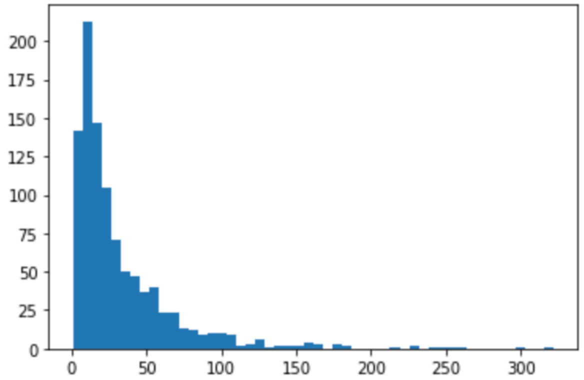
Other distributions you can sample from

- beta
- binomial
- chisquare
- dirichlet
- exponential
- f
- gamma
- geometric
- gumbel
- hypergeometric
- laplace
- logistic
- lognormal
- …

More comprehensive index: https://numpy.org/doc/stable/reference/random/index.html
Older version: https://docs.scipy.org/doc/numpy-1.14.0/reference

```
s = np.random.lognormal(3.0, 1.0, size=1000)
```

```
plt.hist(s,50)
plt.show()
```



Where do we use statistical distributions

- Input data modeling
- Output data modeling
- Compare model's output distribution against the real system's distribution.
  - Z-test
  - Kolmogorov–Smirnov (KS) test
  - …

Multiple competing models

- How would you pick the best one?

George Mason University
Center for Social Complexity

Ways to compare model performance

- Kullback–Leibler (K – L) divergence if we have a distribution of quantities (e.g., firm size distribution).
  - Describes the difference between two probability distributions.
  - Works for both continuous and discrete data.
- Wasserstein metric
- Pearson correlation
- Rank correlation (e.g., Kendall's tau)
- Various measures
  - MAE
  - MSE
  - RMSE
  - …

Visual techniques to inspect model output

Basic definitions

Let $X = \{x_1, x_2, \dots, x_n\}$ be a set of **ordered** values gathered from n model runs.

- Mean: $\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$
- Variance: $s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n-1}$
- Standard deviation: $s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n-1}}$

- First quartile*: median of the numbers from start to general median.
- Median**: $\begin{cases} x_{(n+1)/2} & if\ n\ is\ odd \\ \frac{(x_{n/2} + x_{n/2+1})}{2} & if\ n\ is\ even \end{cases}$
- Third quartile***: median of the numbers from general median to end.

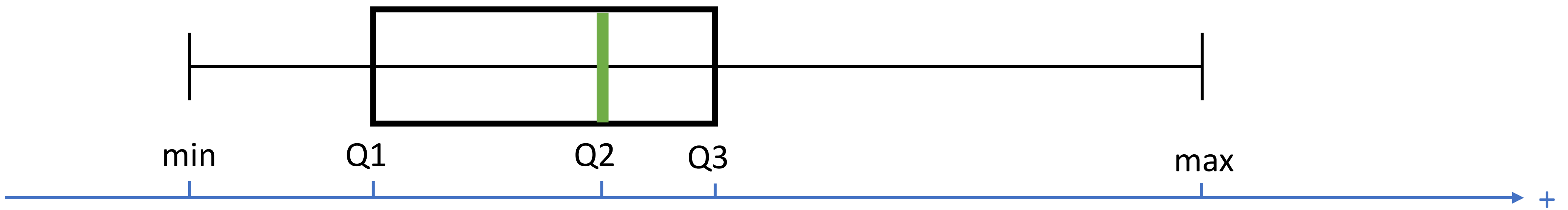
* First quartile(Q_1) or 25th percentile

** Second quartile(Q_2) or 50th percentile

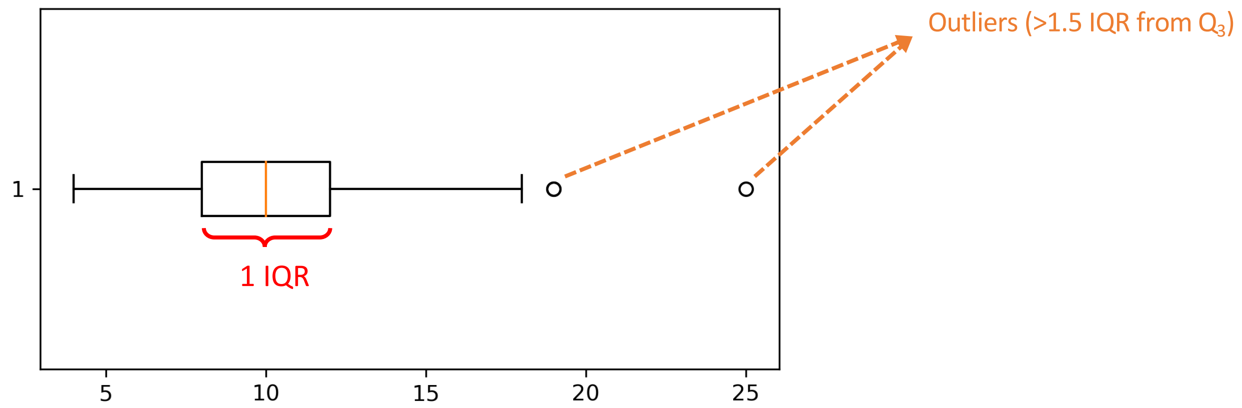
*** Third quartile(Q_3) or 75th percentile

Box and Whiskers plots

- Used to visually display the spread of data




```
[1]: import matplotlib.pyplot as plt
     import numpy as np

[2]: np.random.seed(2020)
     values = np.random.poisson(10,200)

[3]: plt.figure(figsize=(6,3))
     plt.boxplot(values, vert=False)
     plt.savefig('boxplot.png', dpi=300)
```



Probability-Probability (P-P) Plots

- Used to compare a sample ($X = \{x_1, x_2, \dots, x_n\}$) to a theoretical distribution (T).
- Steps
  1. Order your sample ascending.
  2. $\forall x_i \in X$, determine $p_i = P(X < x_i)$ using the CDF of T.
  3. Scatter plot p_i against $(i - 0.5)/n$
  4. If the constructed plot looks like $y = x$, we can say they are similar.

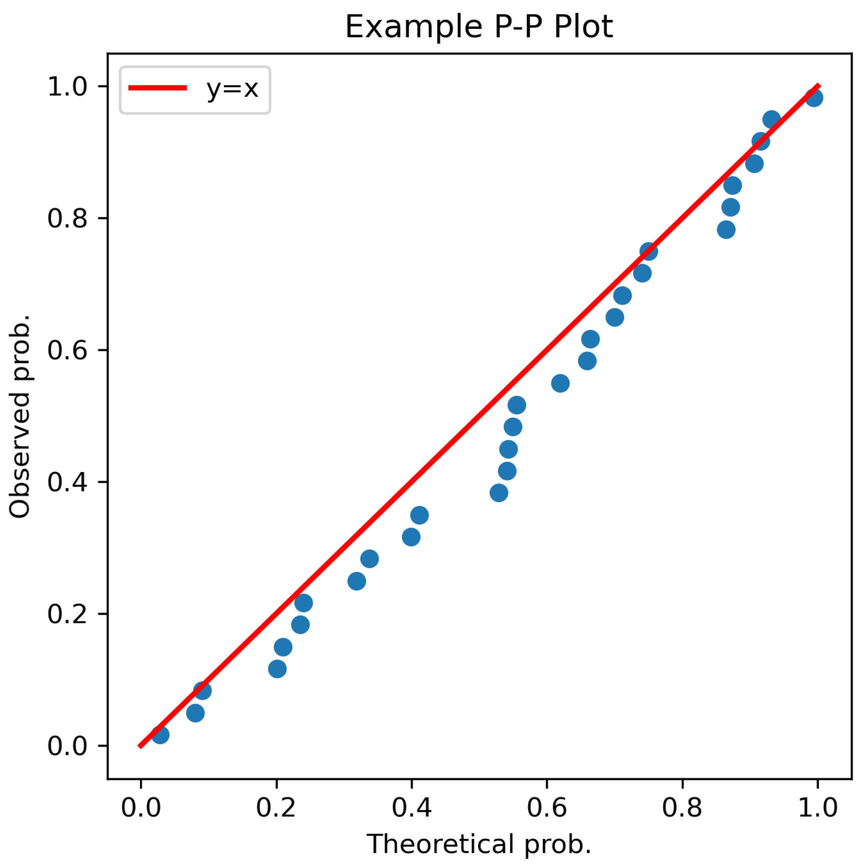
Probability-Probability (P-P) Plots

```
# generate 30 random number w/ mean=10, std dev=5
n = 30
mean = 10
std = 5
sample_X = np.random.normal(mean,std,n)
```

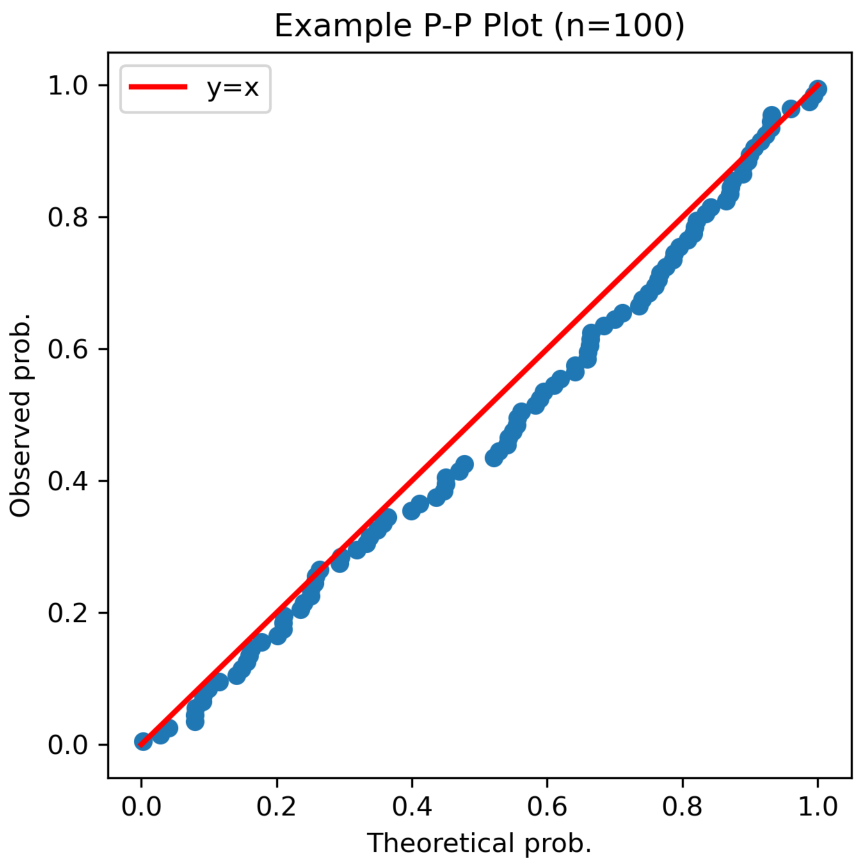
```
sample_X.sort()
```

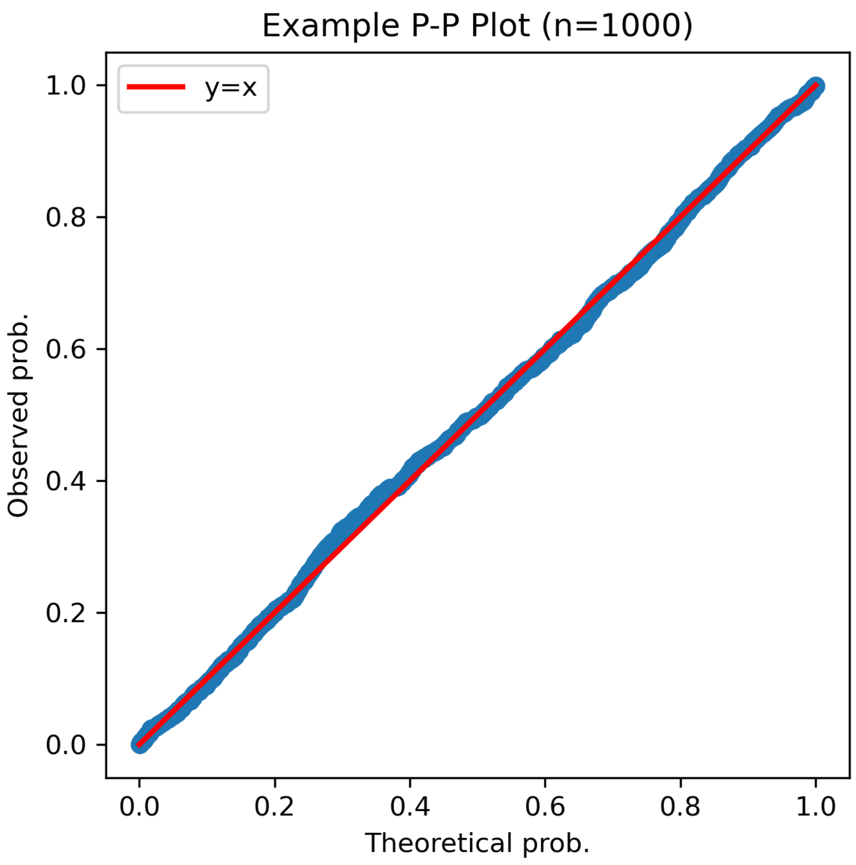
```
x = sci.norm.cdf(sample_X, loc=mean, scale=std)
y = (np.arange(1,n+1)-0.5)/n
```

```
plt.figure(figsize=(5,5))
plt.plot(np.linspace(0,1),np.linspace(0,1),'r',lw=2,label="y=x")
plt.scatter(x,y)
plt.xlabel('Theoretical prob.')
plt.ylabel('Observed prob.')
plt.title('Example P-P Plot')
plt.legend()
plt.savefig('qq.png', dpi=300)
```



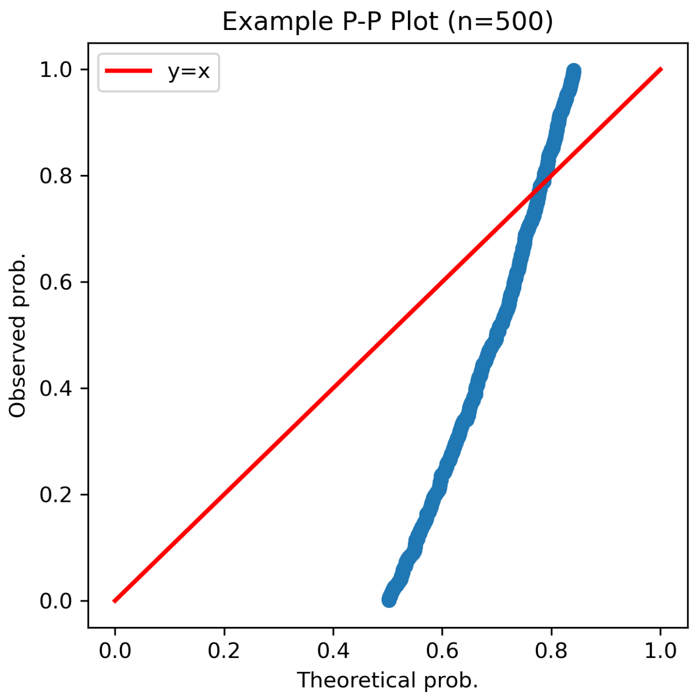
Probability-Probability (P-P) Plots



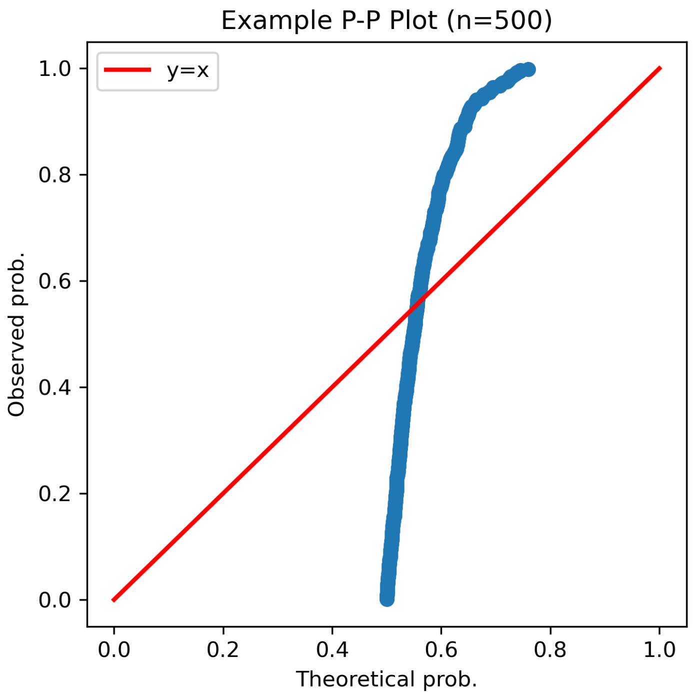


Probability-Probability (P-P) Plots

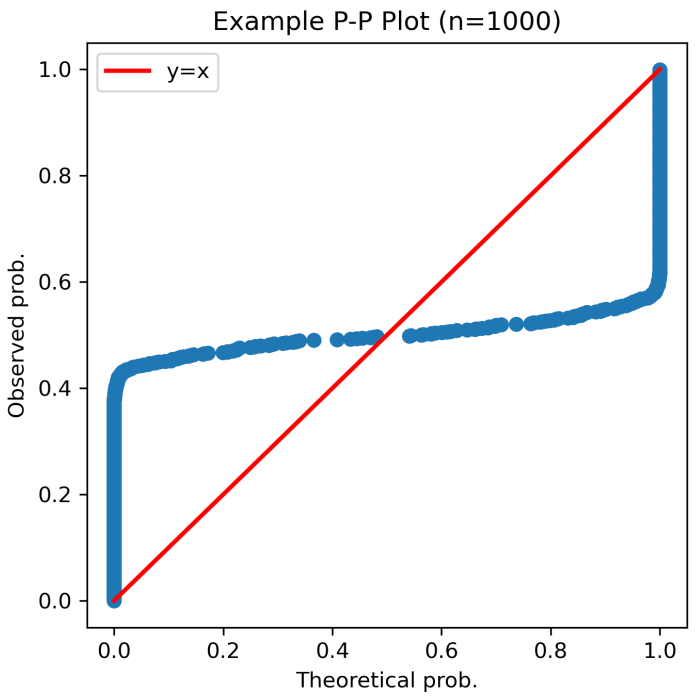
Uniform (0,1) vs. Standard Normal



Beta (1,5) vs. Standard Normal



Normal (0,10) vs. Standard Normal



Sources

- Collins, A. MSIM 651 Lecture Slides. Old Dominion University.
- Leemis, L. M., & Park, S. K. (2006). *Discrete-event simulation: A first course*. Upper Saddle River, NJ: Pearson Prentice Hall.
- Lehmer, D. H. (1951). Mathematical methods in large-scale computing units. *Annu. Comput. Lab. Harvard Univ., 26*, 141-146.
- Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS), 8*(1), 3-30.
- *NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/