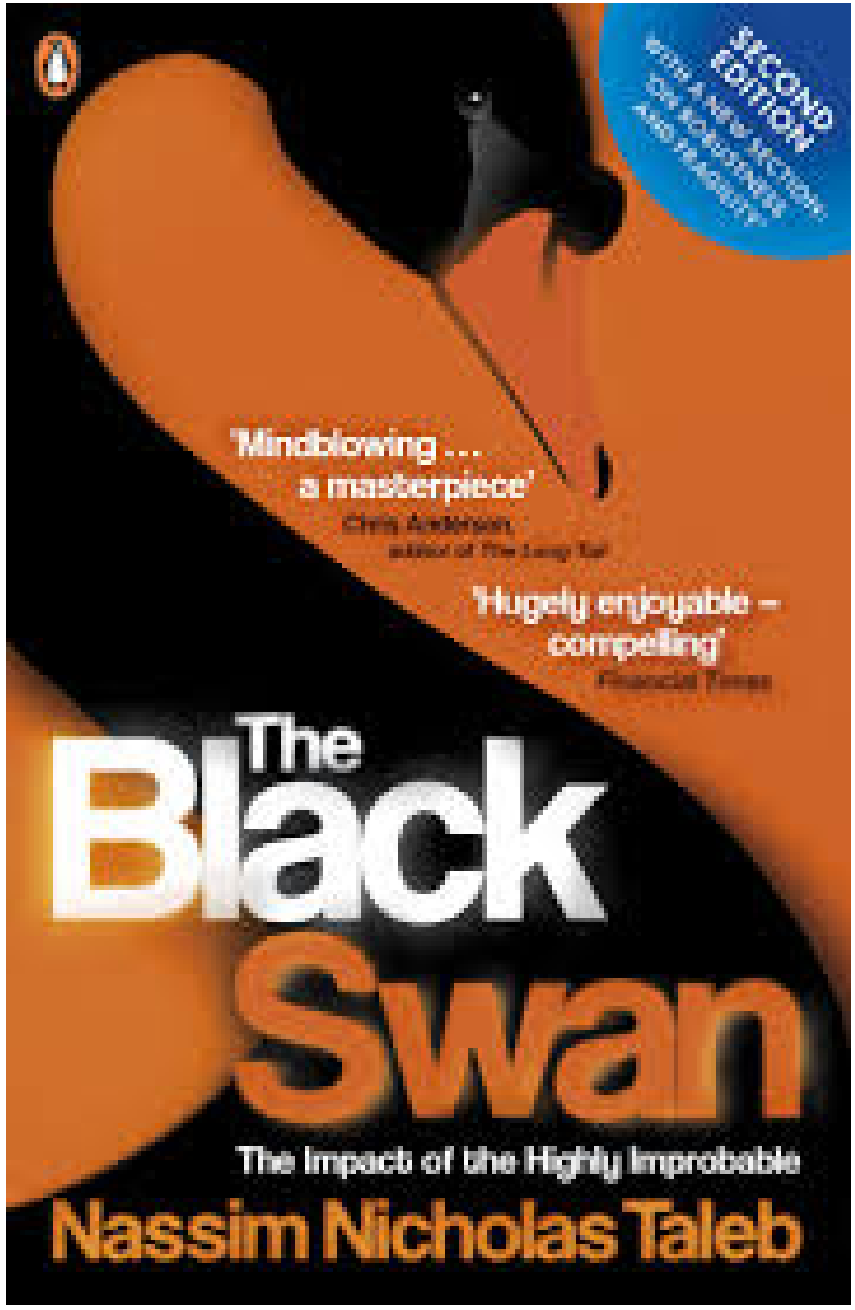
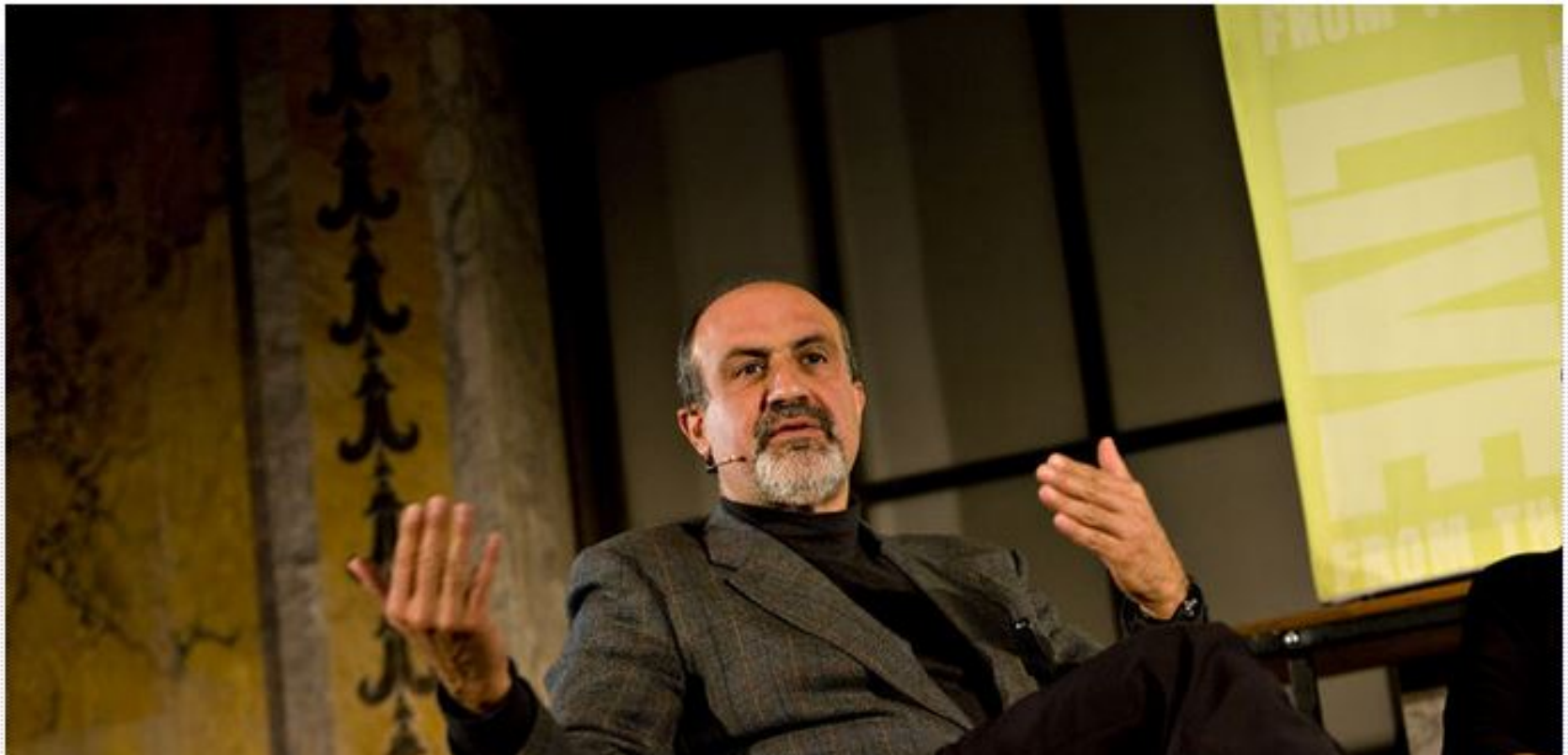

SECOND EDITION
WITH A NEW SECTION, CHAPTERS
AND PREFACE
'Mindblowing ... a masterpiece'
Chris Anderson
'Hugely enjoyable – compelling'
The Black Swan
The Impact of the Highly Improbable
Nassim Nicholas Taleb

Taleb Says Investors Should Sue Nobel Panel for Losses



Probability Models

In his 2007 bestseller "The Black Swan: The Impact of the Highly Improbable," Taleb described how unforeseen events can roil markets. He warned that bankers were relying too much on probability models and disregarding the potential for unexpected catastrophes.

"If no one else sues them, I will," said Taleb, who declined to say where or on what basis a lawsuit could be brought.

"What goes out of the window? The entire discipline of modern finance and portfolio theory (the theories named after Harry Markowitz, William Sharpe, Merton Miller), the model-based methods of Paul Samuelson, much of time series econometrics (which dont appear to predict anything), along with papers and theories that are based on optimization. These bring fragility into the system."

(Remark: The basis of his argument in this matter has little to do with his Black Swan Theory).

Probability Distributions

A probability distribution is a table or an equation that links each outcome (or range of outcomes) of a statistical experiment with its probability of occurrence.

Two Families

- Discrete Distributions (i.e. Count Variables - Integers)
- Continuous Distributions (i.e. Continuous Variables - Real Numbers)

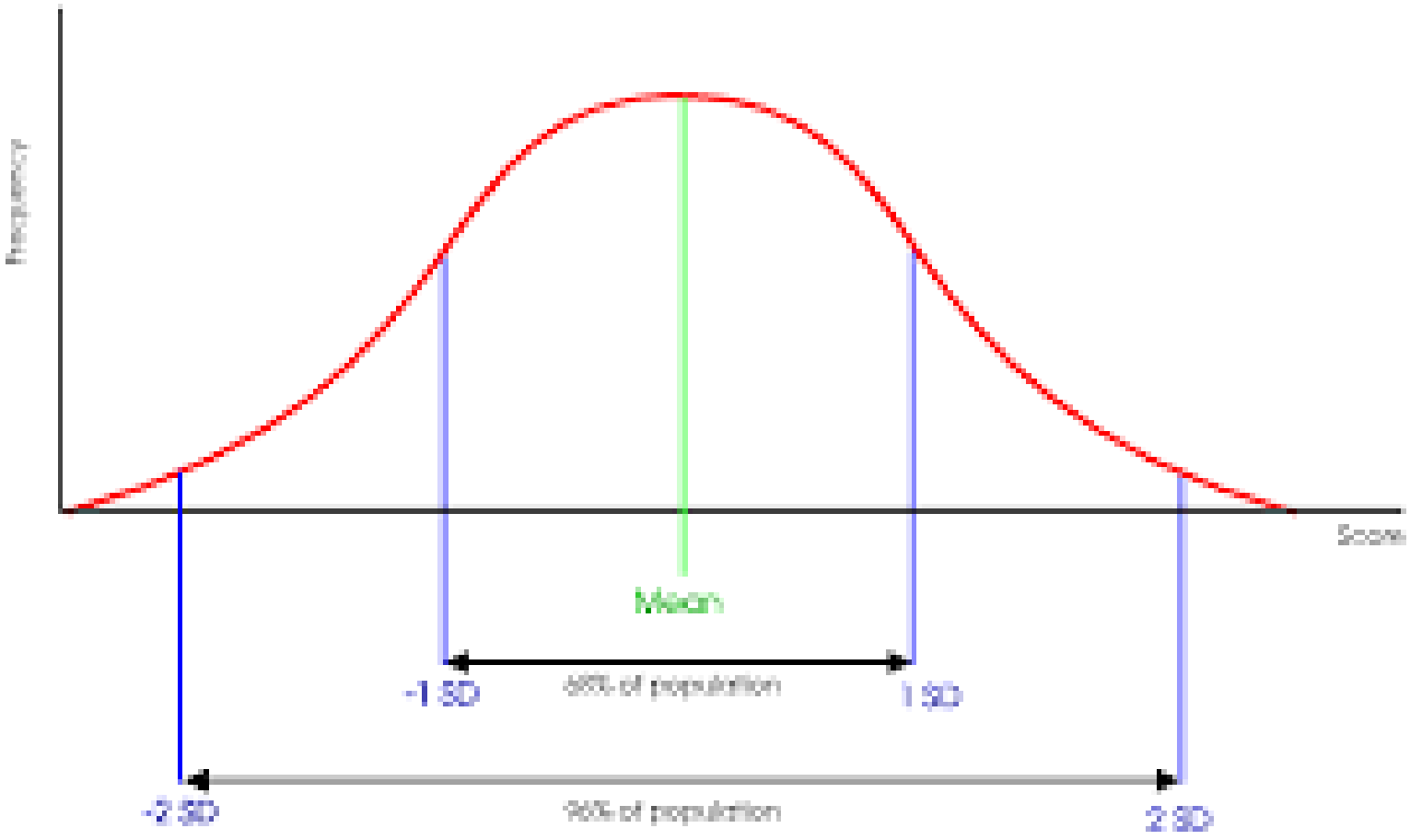
Discrete distributions

- Benford Distribution
- Bernouilli distribution
- Binomial distribution
- Hypergeometric distribution
- Geometric distribution
- Multinomial distribution
- Negative binomial distribution
- Poisson distribution
- Zipf's law

Continuous distributions

- Beta and Dirichlet distributions
- Cauchy distribution
- Chi Square distribution
- Exponential distribution
- Fisher-Snedecor distribution
- Gamma distribution
- Levy distribution
- Log-normal distribution
- Normal and related distributions
- Pareto Distributions
- Student's t distribution
- Uniform distribution
- Weibull distribution
- Extreme values and related distribution
- Distribution in circular statistics

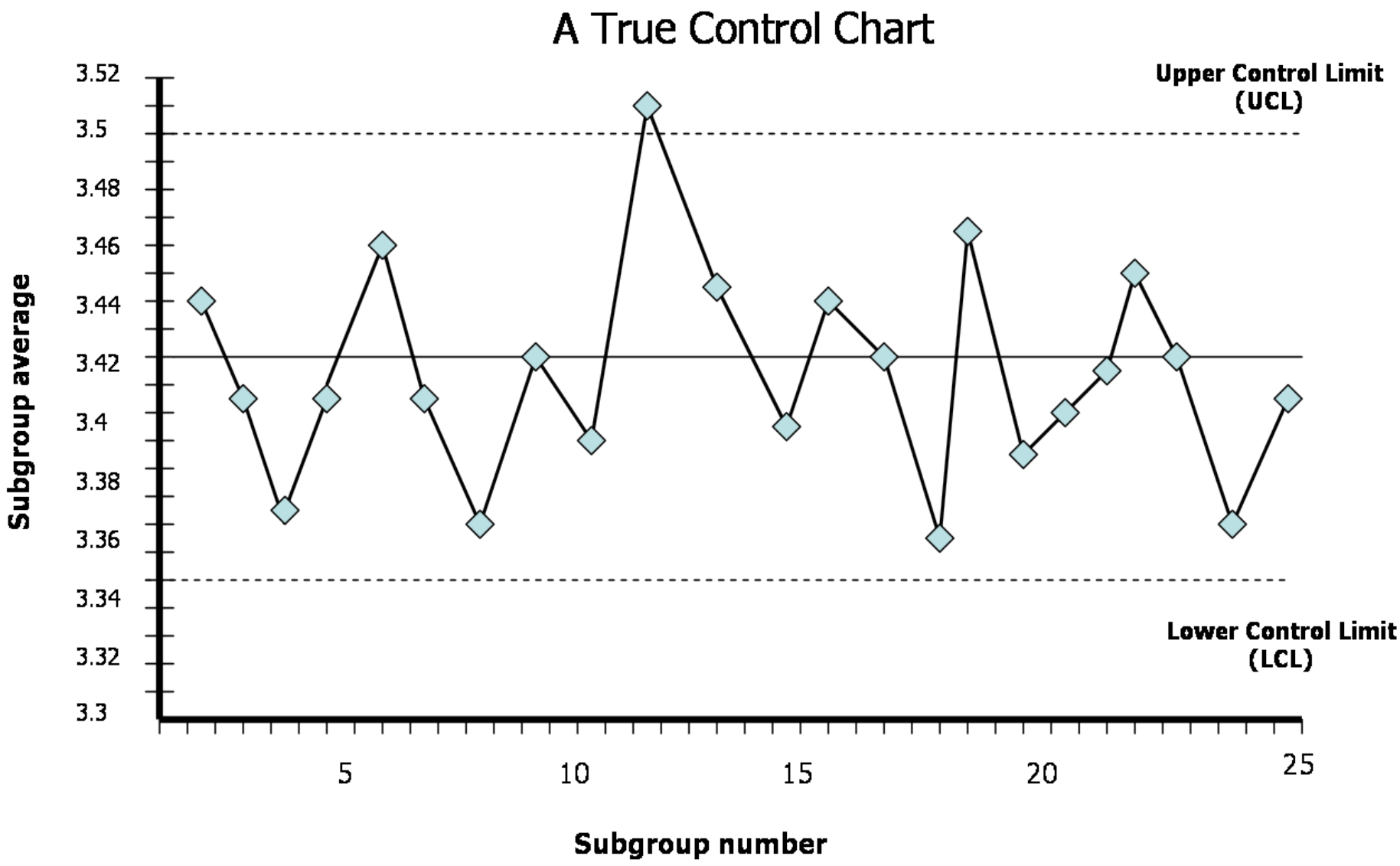
The Normal Distribution Curve



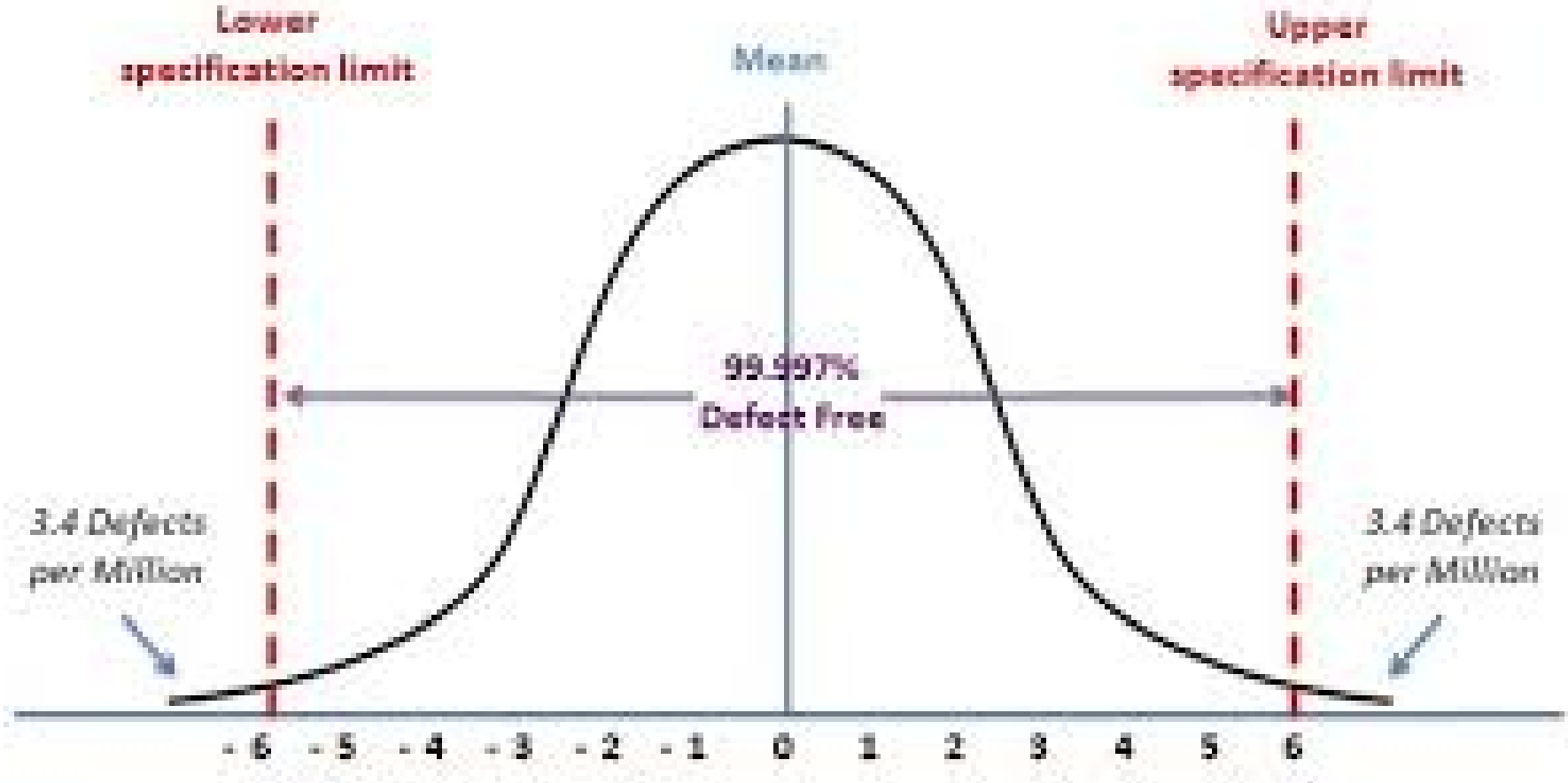
The Normal Distribution

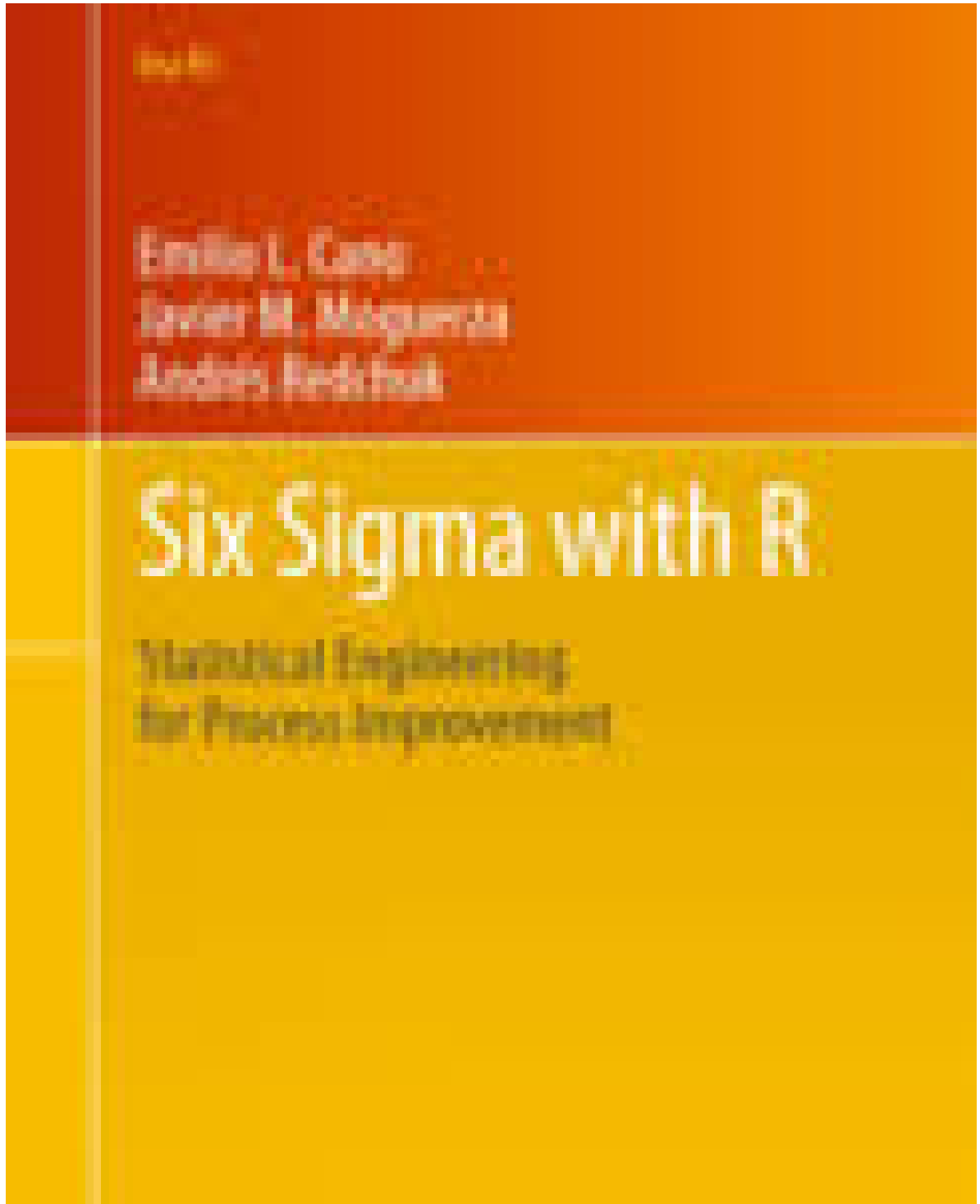
Normal Probability Distribution

- Cornerstone of every undergraduate statistics module.
- Basis of a substantial body of statistical theory
- Central Limit Theorem - basis of Statistical Inference (i.e. Hypothesis Tesing, Confidence Intervals).


A True Control Chart
Upper Control Limit
(UCL)
Lower Control Limit
(LCL)
Subgroup average
3.52
3.5
3.48
3.46
3.44
3.42
3.4
3.38
3.36
3.34
3.32
3.3
5
10
15
20
25
Subgroup number


6σ
DEFINE
MEASURE
ANALYZE
IMPROVE
CONTROL


Lower specification limit
Mean
Upper specification limit
99.997% Defect Free
3.4 Defects per Million
3.4 Defects per Million
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6


Emilio L. Cano
Javier M. Moguerza
Andrés Redchuk
Six Sigma with R
Statistical Engineering
for Process Improvement

Compendium of Probability Distributions

Compendium of Probability Distributions

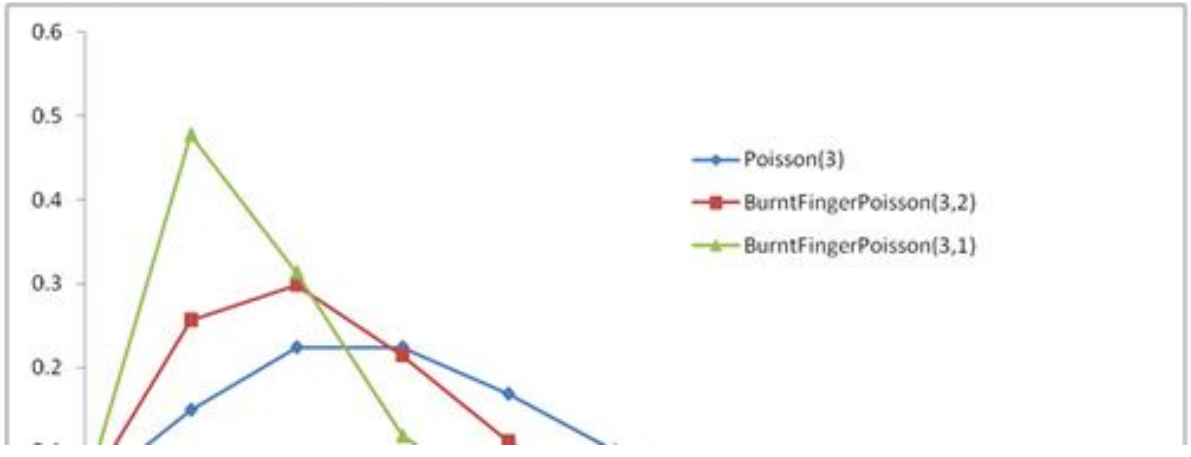
Burnt Finger Poisson Distribution

See also: Poisson distribution

Format: VoseBurntFingerPoisson(λ_1, λ_2, U)

BurntFingerPoisson equations

A BurntFingerPoisson(λ_1, λ_2) distribution returns discrete values from 0 to +infinity:



Compendium of Probability Distributions

Burnt Finger Poisson Disribution

- This type of situation occurs when, for example, an individual has an expected rate of accidents λ_1, but if an accident occurs the individual will become more careful (his/her "fingers got burned") so that for the rest of the modeled time a new, lower expected accident rate λ_1 applies.

Compendium of Probability Distributions

Burr Disribution

- The Burr distribution is a right-skewed distribution bounded at the minimum value of a. b is a scale parameter while c and d control its shape.
- It is frequently used to model insurance claim sizes, and is sometimes considered as an alternative to a Normal distribution when data show slight positive skewness.

Compendium of Probability Distributions

Burr Disribution

Burr(0,1,c,d) is a unit Burr distribution. Examples of the Burr distribution are given below:

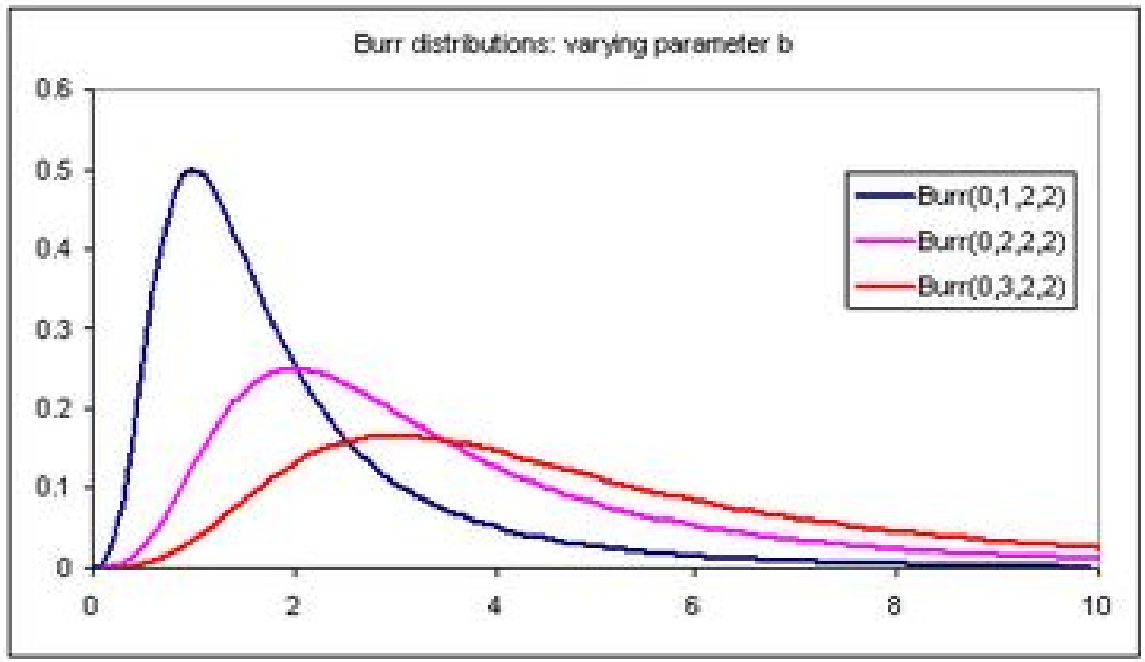


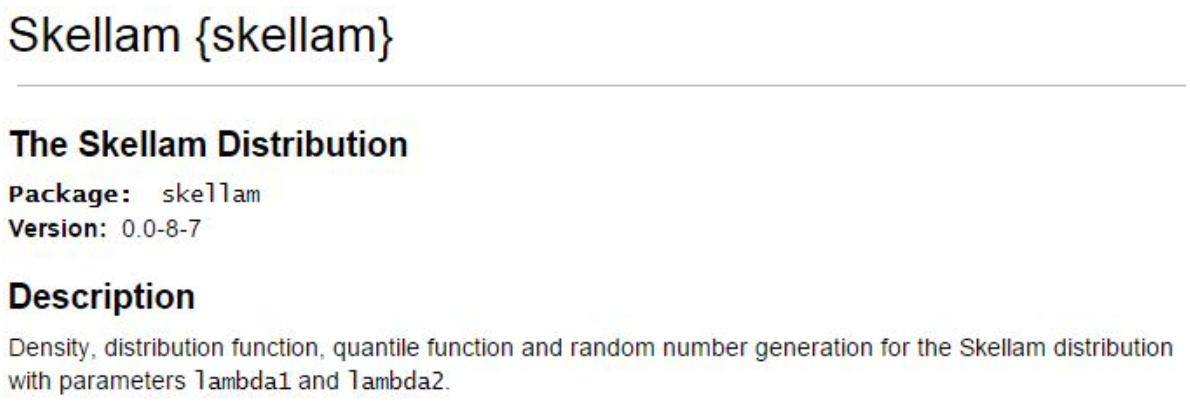
Figure:

Skellam distribution

- The **Skellam distribution** is the discrete probability distribution of the difference $n_1 - n_2$ of two statistically independent random variables N_1 and N_2 each having Poisson distributions with different expected values μ_1 and μ_2.
- It is useful in describing the statistics of the difference of two images with simple photon noise.
- It is also useful in describing the **point spread distribution** in sports where all scored points are equal, such as baseball, hockey and soccer.

Skellam distribution

Package no longer on CRAN - seemingly implemented on some other pacakges though.



Skellam {skellam}

The Skellam Distribution

Package: skellam
Version: 0.0-8-7

Description

Density, distribution function, quantile function and random number generation for the Skellam distribution with parameters lambda1 and lambda2.

Figure:

also implements a left-censored version.

- *Tweedie distribution* : the Tweedie distribution is implemented in package tweedie. Let us note that the Tweedie distribution is not necessarily continuous, a special case of it is the Poisson distribution.
- *Uniform distribution* : d, p, q, r functions are of course provided in R. See section RNG for random number generation topics.

Figure:

Figure:

Applications of Tweedie distributions include:

- actuarial studies[37][38][39][40][41][42][43]
- assay analysis [44][45]
- survival analysis[46][47][48]
- ecology [11]
- analysis of alcohol consumption in British teenagers [49]
- medical applications [50]
- meteorology and climatology [50][51]
- fisheries [52]
- Mertens function [53]
- self-organized criticality[54]

ActuDistns: Functions for actuarial scientists

Computes the probability density function, hazard rate function, integrated hazard rate function and the quantile function for 44 commonly used survival models

Version:	3.0
Depends:	R (≥ 2.15.0), reliaR, actuar, hypergeo
Published:	2012-09-14
Author:	Saralees Nadarajah
Maintainer:	Saralees Nadarajah <Saralees.Nadarajah at manchester.ac.uk>
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]
NeedsCompilation:	no
In views:	Distributions
CRAN checks:	ActuDistns results

Compendium of Probability Distributions

Braford Distribution

The theory has a lot of implications in researching and investment in periodicals: for example, how many journals an institute should subscribe to, or one should review in a study. It also gives a guide for advertising, by identifying the first third of journals that have the highest impact, helps determine whether journals on a new(ish) topic (or arena like e-journals) have reached a stabilised population, and test the efficiency of Web browsers.

Main Functions for R

- `d` - probability density function
- `p` - cumulative distribution function
- `q` - quantile function
- `r` - random number generation function

Some Probability Distributions in R

Distribution	R name	Additional Args
beta	`beta`	`shape1, shape2, ncp`
binomial	`binom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
chi-squared	`chisq`	`df, ncp`
exponential	`exp`	`rate`
F	`f`	`df1, df2, ncp`
...	...	...
normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
Student's t	`t`	`df, ncp`
uniform	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`
Wilcoxon	`wilcox`	`m, n`

Distributions in the stats package

Description

Density, cumulative distribution function, quantile function and random variate generation for many standard probability distributions are available in the **stats** package.

Details

The functions for the density/mass function, cumulative distribution function, quantile function and random variate generation are named in the form `dxxx`, `pxxx`, `qxxx` and `rxxx` respectively.

For the beta distribution see `dbeta`.

For the binomial (including Bernoulli) distribution see `dbinom`

The Birthday Problem

- How many people have to be in a room for there to be a better than 50/50 chance of two people sharing a birthday?

```
> pbirthday(2)
[1] 0.002739726
> 1/365
[1] 0.002739726
> pbirthday(23)
[1] 0.5072972
>
```

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics

CRAN Task View: Probability Distributions

Maintainer: Christophe Dutang
Contact: Christophe.Dutang at ensimag.fr
Version: 2015-03-27

For most of the classical distributions, base R provides probability distribution functions (p), density functions (d), quantile functions (q), and random number generation (r). Beyond this basic functionality, many CRAN packages provide additional useful distributions. In particular, multivariate distributions as well as copulas are available in contributed packages.

Ultimate bibles on probability distributions are:

- different volumes of N. L. Johnson, S. Kotz and N. Balakrishnan books, e.g. Continuous Univariate Distributions, Vol. 1,
- Thesaurus of univariate discrete probability distributions by G. Wimmer and G. Altmann.
- Statistical Distributions by M. Evans, N. Hastings, B. Peacock.
- Distributional Analysis with L-moment Statistics using the R Environment for Statistical Computing, Asquith (2011).

- *Zipf law* : Package zipfR provides tools for the Zipf and the Zipf-Mandelbrot distributions. VGAM also implements the Zipf distribution.
- *Further distributions* : The VGAM package provides several additional distributions, namely: Skellam, Yule-Simon, Zeta and Haight's Zeta, Borel-Tanner and Felix distribution.

The VGAM package

VGAM: Vector Generalized Linear and Additive Models

An implementation of about 6 major classes of statistical regression models. At the heart of it are the vector generalized linear and additive model (VGLM/VGAM) classes. Many (150+) models and distributions are estimated by maximum likelihood estimation (MLE) or penalized MLE, using Fisher scoring. VGLMs can be loosely thought of as multivariate GLMs. VGAMs are data-driven VGLMs (i.e., with smoothing). The other classes are RR-VGLMs (reduced-rank VGLMs), quadratic RR-VGLMs, reduced-rank VGAMs, RCIMs (row-column interaction models)—these classes perform constrained and unconstrained quadratic ordination (CQO/UQO) models in ecology, as well as constrained additive ordination (CAO). Note that these functions are subject to change, especially before version 1.0.0 is released; see the NEWS file for latest changes.

Version:	0.9-7
Depends:	R ($\geq$ 3.0.0), methods, stats, stats4, splines
Suggests:	VGAMdata, MASS

zipfR: Statistical models for word frequency distributions

Statistical models and utilities for the analysis of word frequency distributions. The utilities include functions for loading, manipulating and visualizing word frequency data and vocabulary growth curves. The package also implements several statistical models for the distribution of word frequencies in a population. (The name of this library derives from the most famous word frequency distribution, Zipf's law.)

Version:	0.6-6
Depends:	R (≥ 2.10.1)
Published:	2012-04-03
Author:	Stefan Evert, Marco Baroni
Maintainer:	Stefan Evert <stefan.evert at uos.de>

Random Number Generators:

- *Basic functionality* : R provides several random number generators (RNGs). The random seed can be provided via `set.seed` and the kind of RNG can be specified using `RNGkind`. The default RNG is the Mersenne-Twister algorithm. Other generators include Wichmann-Hill, Marsaglia-Multicarry, Super-Duper, Knuth-TAOCP, Knuth-TAOCP-2002, as well as user-supplied RNGs. For normal random numbers, the following algorithms are available: Kinderman-Ramage, Ahrens-Dieter, Box-Muller, Inversion (default). In addition to the tools above, setRNG provides an easy way to set, retain information about the setting, and reset the RNG.
- *Pseudo-randomness* : RDieHarder offers several dozen new RNGs from the GNU GSL. randtoolbox provides more recent RNGs such as SF Mersenne-Twister and WELL, which are generators of Mersenne Twister type, but with improved quality parameters. rngwell19937 provides one of the WELL generators with 53 bit resolution of the output and allows seeding by a vector of integers of arbitrary length. randaes provides the deterministic part of the Fortuna cryptographic pseudorandom number generator (AES). SuppDists implements two RNGs of G. Marsaglia.

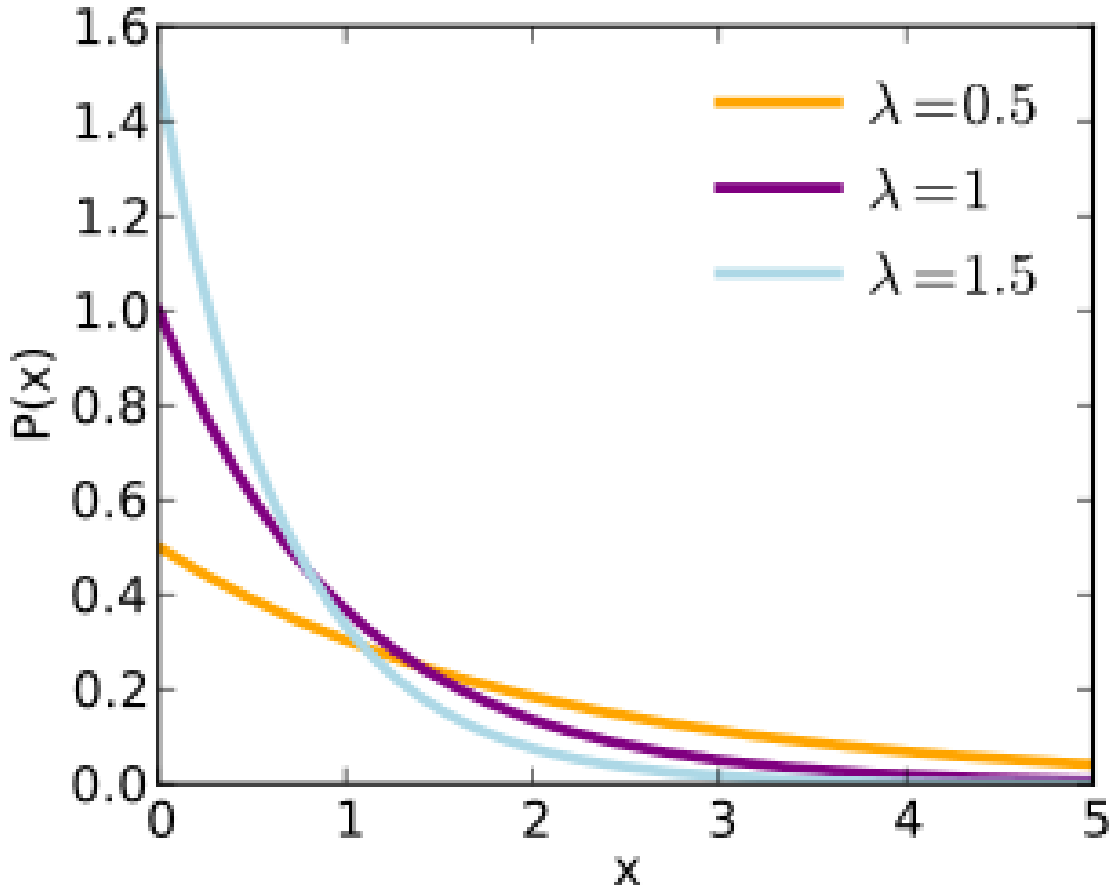
Copulas:

- *Unified approaches* : The packages fCopulae, copula, and copBasic provide a lot of general functionality for copulas. Although lacking support for many existing copulas themselves, copBasic is primarily oriented around utility functions for the general mathematics of copulas as described in the well known introduction to copulas by Nelsen.
- *Archimedean copulas* : The Frank bivariate distribution is available in VGAM, RTDE. fCopulae implements the 22 Archimedean copulas of Nelsen (1998, *Introduction to Copulas* , Springer-Verlag) including Gumbel, Frank, Clayton, and Ali-Mikhail-Haq. gumbel is a standalone package for the Gumbel copula and VGAM provides the Ali-Mikhail-Haq bivariate distribution. copula provides Ali-Mikhail-Haq, Clayton, Frank, Gumbel and Joe copulas. CDVine and VineCopula provide Clayton, Gumbel, Frank, Joe, BB1, BB6, BB7 and BB8 copulas. Generalized Archimedean copulas are implemented in the fgac package.
- *Blomqvist copula* : provided in copBasic.
- *Composition of copula* : copBasic provides functions for composition of a single symmetric copula and composition of two copulas.
- *Cubic copula* : Not yet implemented?
- *Dirichlet copula* : Not yet implemented?

The Exponential Distribution

- The exponential distribution is often used to model the waiting time X between events occurring randomly and independently in time (or space).
- Because it is a continuous distribution, the height of the exponential curve at any X refers to probability density rather than probability.
- Probability is instead represented by area under the exponential curve.

The Exponential Distribution



The Exponential Distribution

- The main assumption of the exponential distribution is that at any point in time, the probability of an event occurring in the next instant does not depend on how much time has already elapsed since the previous event.
- The parameter of the exponential distribution is the **rate parameter** at which events occur (the number of events per unit time).
- The mean of the exponential distribution is 1/rate.

The Exponential Distribution

Waiting time X to the next event has probability density

```
dexp(X, rate=1)
dexp(X, rate=1, log=TRUE)
```

The default value for the rate is 1, so you must alter to fit your circumstance.

Exponential Distribution : Problem

- The mean checkout time of a supermarket cashier is three minutes.
- Find the probability of a customer checkout being completed by the cashier in less than two minutes, three minutes and four minutes.
- (i.e. what percentage of "waiting times" are less than two, three and four minutes?)

The Exponential Distribution

Solution

- The checkout processing rate is equals to one divided by the mean checkout completion time.
- Hence the processing rate is $1/3$ checkouts per minute.
- We then apply the function `pexp()` of the exponential distribution with rate$=1/3$.

The Exponential Distribution

```
> pexp(2,rate=1/3)
[1] 0.4865829
>
> pexp(3,rate=1/3)
[1] 0.6321206
>
> pexp(4,rate=1/3)
[1] 0.7364029
>
> pexp(5,rate=1/3,lower=FALSE)
[1] 0.1888756
```

The Exponential Distribution

- What is the median waiting time? To answer this question we would use the `qexp()` function.
- Recall that the median is value of x such that $P(X \leq x) = 0.50$.
- Also determine the first and third quartiles Q_1 and Q_3.

```
> qexp(0.5,rate=1/3)
[1] 2.079442
> qexp(0.25,rate=1/3)
[1] 0.8630462
> qexp(0.75,rate=1/3)
[1] 4.158883
```

The Geometric Probability Distribution

- Used for binary data (1 or 0; success or failure, etc).
- In a sequence of trials, the trial yielding the first success (all previous trials ending in failure) has a geometric distribution.
- The assumption required is that separate trials are independent and the probability of success p is the same in every trial.
- The probability of failure in each trial is $1 - p$.

The Geometric Probability Distribution

```
# p is the probability of success
dgeom(X, prob = p)
# log of the probability instead
dgeom(X, prob = p, log=TRUE)
```

The Geometric Probability Distribution

In probability theory and statistics, the **geometric distribution** is either of two discrete probability distributions:

- The probability distribution of the number X of Bernoulli trials needed to get one success, supported on the set { 1, 2, 3, ...}
- The probability distribution of the number $Y = X - 1$ of failures before the first success, supported on the set { 0, 1, 2, 3, ... }

Which of these one calls "the" geometric distribution is a matter of convention and convenience.

The Geometric Probability Distribution

```
> dgeom(0,0.25)
[1] 0.25
> dgeom(1,0.25)
[1] 0.1875
> pgeom(1,0.25)
[1] 0.4375
>
```

Pseudo-Random Number Generation

Random Number Generation

- To a very high degree computers are deterministic and therefore are not a reliable source of significant amounts of random values. In general pseudo random number generators are used.
- The default algorithm in `R` is Mersenne-Twister but a long list of methods is available.
- See the help of `RNGkind()` to learn about random number generators.

Pseudo-Random Number Generation

`.Random.seed` is an integer vector, containing the **random number generator** (RNG) state for random number generation in `R`.
It can be saved and restored, but should not be altered by the user.

`RNGkind` is a more friendly interface to query or set the kind of RNG in use.

Pseudo-Random Number Generation

`RNGversion` can be used to set the random generators as they were in an earlier `R` version (for reproducibility).

`set.seed` is the recommended way to specify seeds.

Seed

A pseudo random number generator is an algorithm based on a starting point called "**seed**".
If you want to perform an exact replication of your program, you have to specify the seed using the function `set.seed()`. The argument of `set.seed()` has to be an integer.

```
> set.seed(1)
> runif(1)
[1] 0.2655087
> set.seed(1)
> runif(1)
[1] 0.2655087
```

Pseudo-Random Number Generation

Mersenne Twister

Mersenne Twister(MT) is a pseudorandom number generating algorithm developped by Makoto Matsumoto and Takuji Nishimura (alphabetical order) in 1996/1997.

- It is designed with consideration on the flaws of various existing generators.
- The algorithm is coded into a C-source downloadable below.
- Far longer period and far higher order of equidistribution than any other implemented generators.

Mersenne Twister

Mersenne Twister

- Fast generation.
  (Although it depends on the system, it is reported that MT is sometimes faster than the standard ANSI-C library in a system with pipeline and cache memory.)
- Efficient use of the memory.

Diehard Tests

The Diehard Tests

- The diehard tests are a battery of statistical tests for measuring the quality of a random number generator.
- They were developed by George Marsaglia over several years and first published in 1995 on a CD-ROM of random numbers.

RDieHarder R Package

RDieHarder: R interface to the dieharder RNG test suite

The RDieHarder packages provides an R interface to the dieharder suite of random number generators and tests that was developed by Robert G. Brown and David Bauer, extending earlier work by George Marsaglia and others.

Version:	0.1.3
Depends:	R (≥ 2.5.0)
Published:	2014-02-21
Author:	Dirk Eddelbuettel
Maintainer:	Dirk Eddelbuettel <edd at debian.org>
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]
URL:	http://code.google.com/p/rdieharder/
NeedsCompilation:	yes

Vignette on CRAN

RDieHarder: An R interface to the Die*Harder* suite of Random Number Generator Tests

Dirk Eddelbuettel
Debian
edd@debian.org

Robert G. Brown
Physics, Duke University
rgb@phy.duke.edu

Initial Version as of May 2007

Rebuilt on February 20, 2014 using RDieHarder 0.1.3

1 Introduction

Random number generators are critically important for computational statistics. Simulation methods are becoming ever more common for estimation; Monte Carlo Markov Chain is but one approach. Also, simulation methods such as the Bootstrap have long been used in inference and are becoming a standard part of a rigorous analysis. As random number generators are at the heart of the simulation-based methods used throughout statistical computing, 'good' random numbers are therefore a crucial aspect of a statistical, or

Diehard Tests

Birthday spacings: Choose random points on a large interval. The spacings between the points should be asymptotically exponentially distributed. The name is based on the birthday paradox.

Overlapping permutations: Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.

Diehard Tests

Ranks of matrices: Select some number of bits from some number of random numbers to form a matrix over 0,1, then determine the rank of the matrix. Count the ranks.

Monkey tests: Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that don't appear should follow a known distribution. The name is based on the *infinite monkey* theorem.

Diehard Tests

Count the 1s: Count the 1 bits in each of either successive or chosen bytes.
Convert the counts to "letters", and count the occurrences of five-letter "words".

Parking lot test: Randomly place unit circles in a 100×100 square. If the circle overlaps an existing one, try again.
After 12,000 tries, the number of successfully "parked" circles should follow a certain normal distribution.

Diehard Tests

Minimum distance test: Randomly place 8,000 points in a 10,000 x 10,000 square, then find the minimum distance between the pairs.
The square of this distance should be exponentially distributed with a certain mean.

Random spheres test: Randomly choose 4,000 points in a cube of edge 1,000.
Center a sphere on each point, whose radius is the minimum distance to another point.
The smallest sphere's volume should be exponentially distributed with a certain mean.

Diehard Tests

The squeeze test: Multiply 231 by random floats on (0,1) until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a certain distribution.

Overlapping sums test: Generate a long sequence of random floats on (0,1).
Add sequences of 100 consecutive floats.
The sums should be normally distributed with characteristic mean and sigma.

Diehard Tests

Runs test: Generate a long sequence of random floats on (0,1). Count ascending and descending runs. The counts should follow a certain distribution.

The craps test: Play 200,000 games of craps, counting the wins and the number of throws per game.
Each count should follow a certain distribution.

Power Laws

Many man made and naturally occurring phenomena, including city sizes, incomes, word frequencies, and earthquake magnitudes, are distributed according to a power-law distribution. A power-law implies that small occurrences are extremely common, whereas large instances are extremely rare.

- Bradford's Law
- Benford's Law
- Zipf's Distribution

Benford's Law

- Benford's law, also called the **First-Digit Law**, refers to the frequency distribution of digits in many (but not all) real-life sources of data.
- In this distribution, 1 occurs as the leading digit about 30% of the time, while larger digits occur in that position less frequently: 9 as the first digit less than 5% of the time.
- Benford's law also concerns the expected distribution for digits beyond the first, which approach a uniform distribution.
- It is named after physicist Frank Benford, who stated it in 1938, although it had been previously stated by Simon Newcomb in 1881.

Benford's Law

- It has been shown that this result applies to a wide variety of data sets, including electricity bills, street addresses, stock prices, population numbers, death rates, lengths of rivers, physical and mathematical constants, and processes described by power laws (which are very common in nature).
- It tends to be most accurate when values are distributed across multiple orders of magnitude.

Benford Distribution

```
> # The Benford Distribution is the distribution of
> # the first digit of a number.
>
> library(VGAM)
>
> dbenf(c(1:9))
[1] 0.30103000 0.17609126 0.12493874
[4] 0.09691001 0.07918125 0.06694679
[7] 0.05799195 0.05115252 0.04575749
```

Motivation [edit]

Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc. For example, in the Brown Corpus of American English text, the word "the" is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf's Law, the second-place word "of" accounts for slightly over 3.5% of words (36,411 occurrences), followed by "and" (28,852). Only 135 vocabulary items are needed to account for half the Brown Corpus.[4]

A "Startlingly Neat & Simple" Rule & Five Graphs About Patterns That Might Surprise You

February 17, 2015

By Plotly

(This article was first published on **Plotly**, and kindly contributed to R-bloggers)

George Zipf popularized an idea—Zipf's Law—that approximates populations of cities, distribution of money in counties, and how frequently words are used. Nobel Prize-winning columnist Paul Krugmans wrote of Zipf's Law that

> *"the usual complaint about economic theory is that our models are*

Zipf Distribution

George Zipf popularized an idea- "Zipfs Law" - that approximates populations of cities, distribution of money in counties, and how frequently words are used.

Nobel Prize-winning columnist Paul Krugmans wrote of Zipfs Law that

> *the usual complaint about economic theory is that our models are oversimplified that they offer excessively neat views of complex, messy reality. [In the case of Zipfs law] the reverse is true: we have complex, messy models, yet reality is startlingly neat and simple.*

Gambler's Ruin

- Consider a gambler who starts with an initial fortune of \$1 and then on each successive gamble either wins \$1 or loses \$1 independent of the past with probabilities p and q = 1-p respectively.
- Suppose the gambler has a starting kitty of A.
- This gambler will place bets with the Banker, who has an initial fortune B. The Banker is, by convention, the richer of the two.
- We will look at the game from the perspective of the gambler only.

Gambler's Ruin

- Probability of successful gamble for gambler : p
- Probability of unsuccessful gamble for gambler : q (where $q = 1 - p$)
- Ratio of success probability to failure success: $s = p/q$
- Conventionally the game is biased in favour of the Banker (i.e. $q > p$ and $s < 1$)
- Total Jackpot : A+B
- Gambling concludes if Gambler , or Banker, loses everything.

Gambler's Ruin

Let R_n denote the Gamblers total fortune after the $n-$th gamble.

- If the Gambler wins the first game, his wealth becomes $R_n = A + 1$.
- If he loses the first gamble, his wealth becomes $R_n = A - 1$.
- The entire sum of money at stake is the Jackpot i.e. $A + B$.
- The game ends when the Gambler wins the Jackpot ($R_n = A + B$) or loses everything ($R_n = 0$).

Gambler's Ruin

Simulation a Single Gamble

To simulate one single bet, compute a single random number between 0 and 1.

```
runif(1)
```

- Lets assume that the game is biased in favour of the Banker p $= 0.45$, q $= 0.55$.
- If the number is less than 0.45, the gamble wins.
- Otherwise the Banker wins.

Gambler's Ruin

```
> runif(1)
[1] 0.1251274
>#Gambler Loses
>
> runif(1)
[1] 0.754075
>#Gambler wins
>
> runif(1)
[1] 0.2132148
>#Gambler Loses
>
> runif(1)
[1] 0.8306269
```

Gambler's Ruin

The vector R_n records the gambler's worth on an ongoing basis. At the start, The first value is A.

```
A=20;B=100;p=0.47
Rn=c(A)

probval = runif(1)

if (probval < p)
 {
 A = A+1; B =B-1
 }else{A=A-1;B=B+1}

#Save the values from each bet
Rn=c(Rn,A)
```

Gambler's Ruin

Should the Gambler win the entire jackpot (A+B). The game would also cease. We include a `break` statement to stop the loop if the gambler wins the entire jackpot. A break statement will stop a loop if a certain logical condition is met.

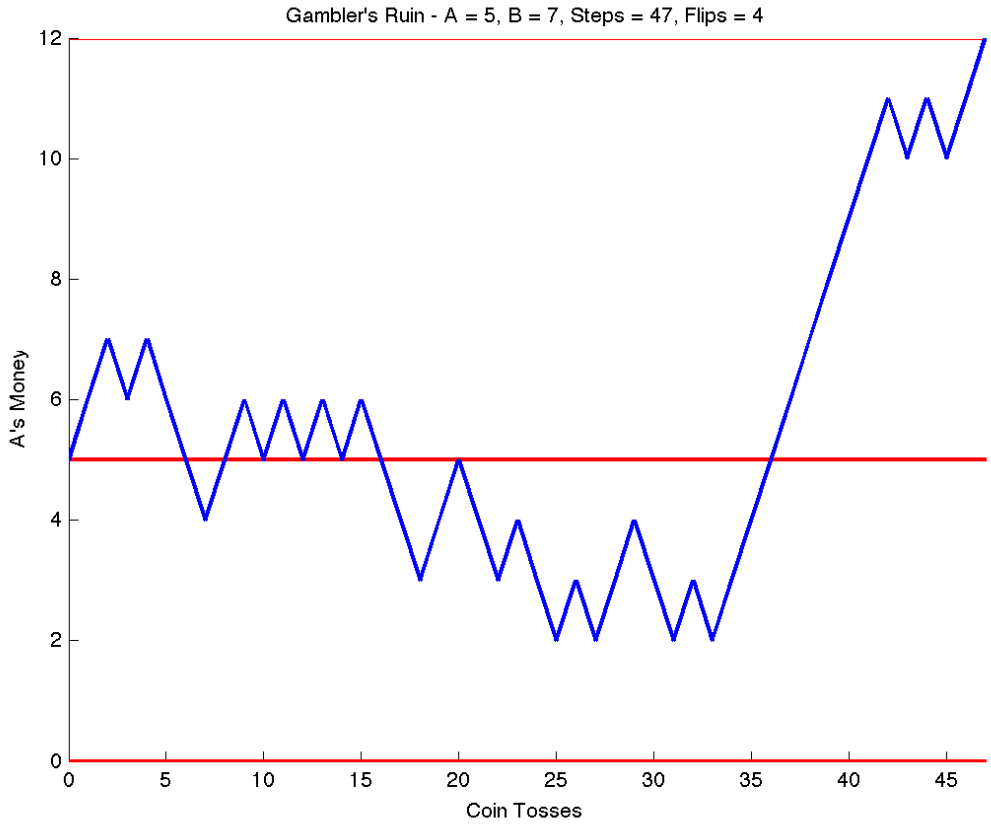

Gambler's Ruin - A = 5, B = 7, Steps = 47, Flips = 4
A's Money
Coin Tosses
0
2
4
6
8
10
12
0
5
10
15
20
25
30
35
40
45

Figure:

Gambler's Ruin

```
A=20;B=100;p=0.47
Rn=c(A)
Total=A+B

while(A>0)
  {
  UnifVal=runif(1)

  if(UnifVal <= p)
     {
     A = A+1; B =B-1
     }else{A=A-1;B=B+1}
  Rn=c(Rn,A)
  if(A==Total){break}
  }
```

Gambler's Ruin

- We can construct a plot to depict the gambler's ongoing fortunes in the game.
- We use a `for` loop, that implements the game, recording the duration of the game each time.
- The duration of each game is the dimension of the `Rn` vector, i.e. `length(Rn)`.

Gambler's Ruin

```
A.ini=20;B=100;p=0.47;M=1000
RnDist=numeric();Total=A+B
for (i in 1:M)
    {
    Rn=numeric();    Rn[1]=A;    A=A.ini
    while(A>0)
        {
        UnifVal=runif(1)

        if(UnifVal <= p)
            {
            A = A+1; B =B-1
            }else{A=A-1;B=B+1}
        Rn=c(Rn,A)
        if(A==Total){break}
        }
```

Distribution of Durations

Simpler Approach

```
A=50;B=200;p=0.47
GAME = A+cumsum(sign(p-runif(1000)))

which(GAME==0)[1]
 # [1] 402

which(GAME==(A+B))[1]
 # [1] NA
```

Gambler's Ruin

```
T1 <- Sys.time()
A <- 50 ; B<- 100 ; p <- 0.48
M=50000
Duration = c()
GamblerWins = c()
for (i in 1:M)
{
GAME = A+cumsum(sign(p-runif(20000)))
Duration <- c(Duration, which(GAME==0)[1])
GamblerWins <- c(GamblerWins, which(GAME==(A+B))[1])
}
T2 <- Sys.time()
T2-T1
```

Gambler's Ruin

```
+ GamblerWins <- c(GamblerWins, which(GAME==(A+B))[1])
+ }
> T2 <- Sys.time()
> T2-T1
Time difference of 1.901574 mins
>
```

Quick enough, but could have faster implementation using Julia or RCPP.

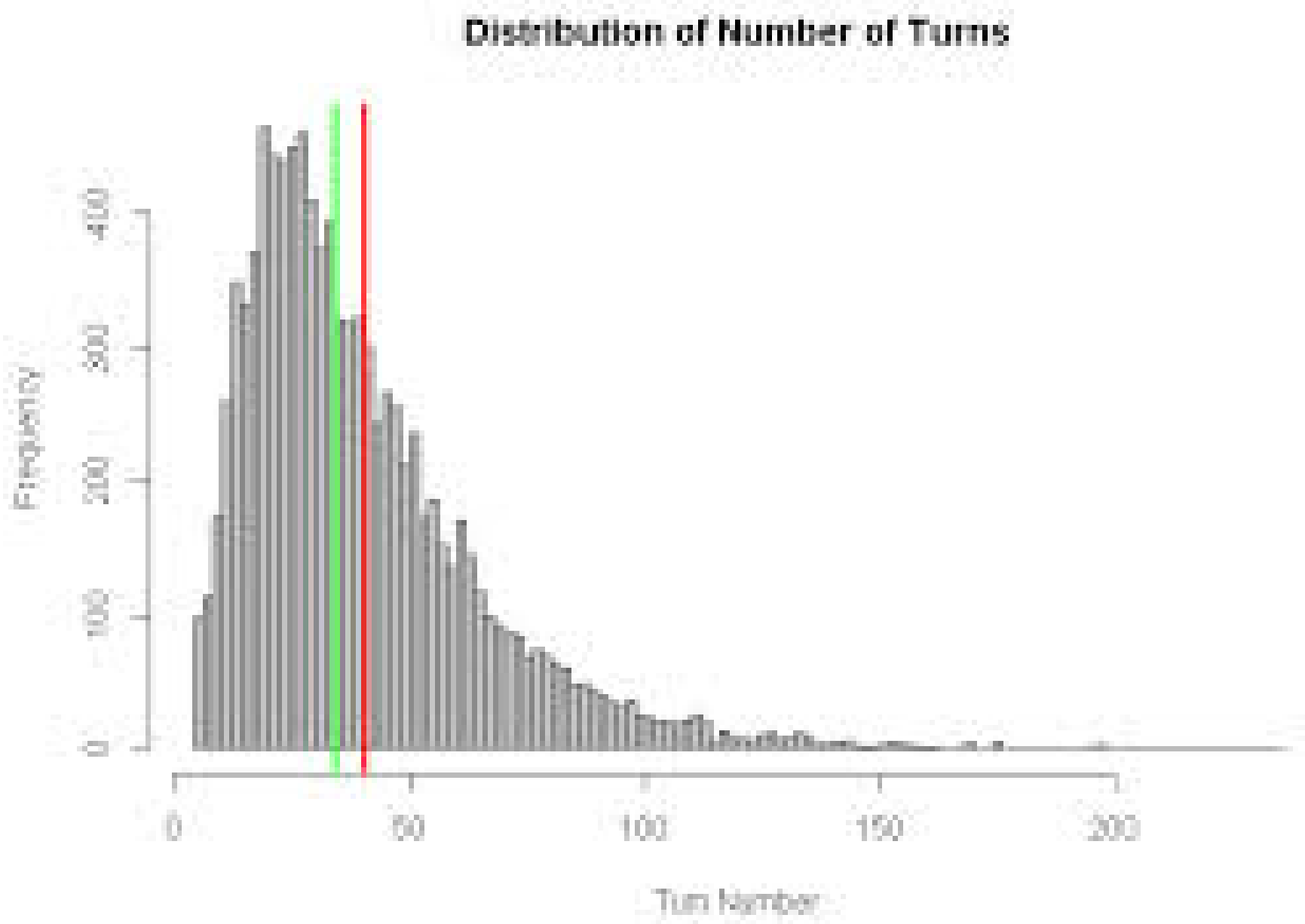
Gambler's Ruin

Did the Gambler Ever Win?

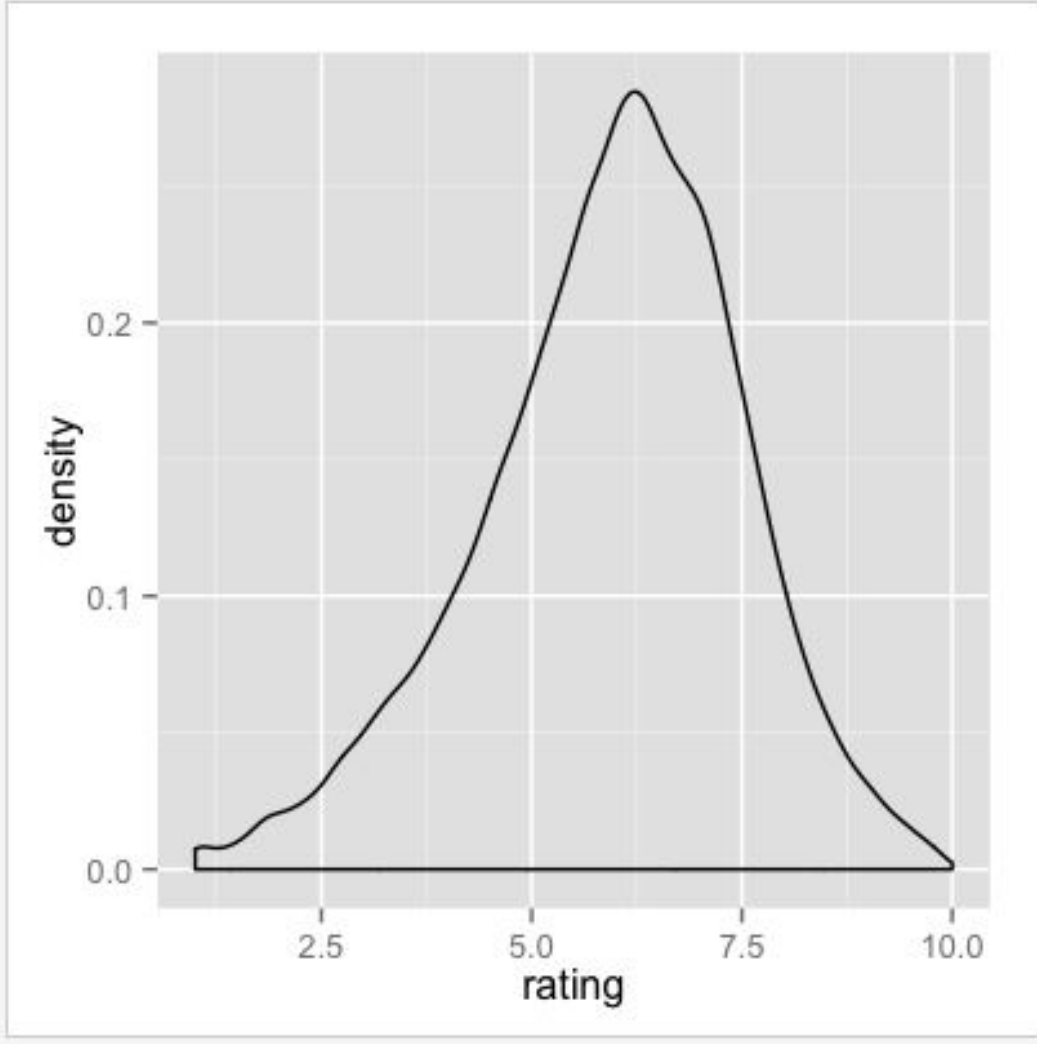
```
GamblerWins[!is.na(GamblerWins)]
which(!is.na(GamblerWins))

Keep <- which(!is.na(GamblerWins))
cbind(Duration[Keep],GamblerWins[Keep])
```

A few times! One game in every three thousand approx


Distribution of Number of Turns
Frequency
0
100
200
300
400
0
50
100
150
200
Turn Number

```
m <- ggplot(movies, aes(x = rating))
m + geom_density()
```



Kernel Density Plot

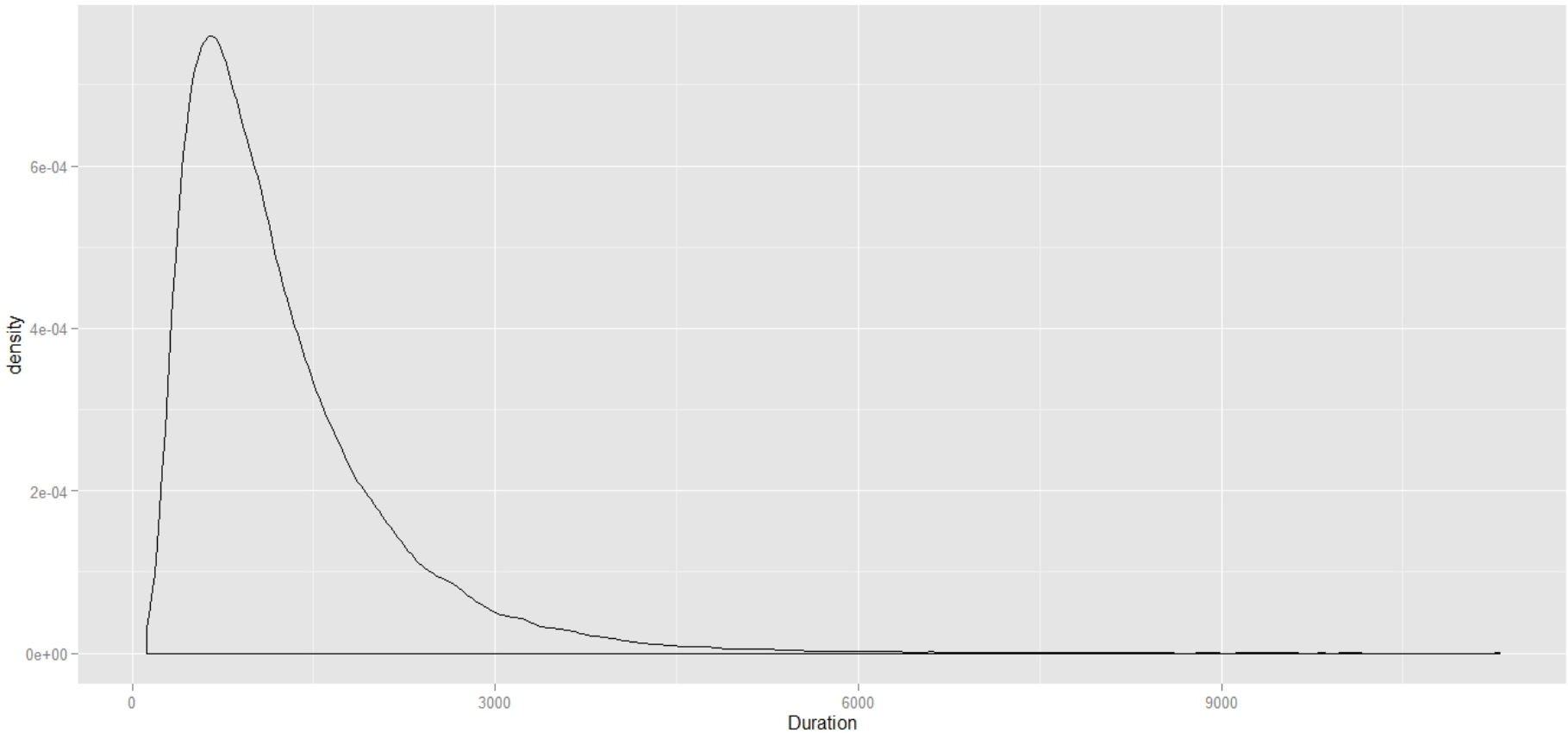
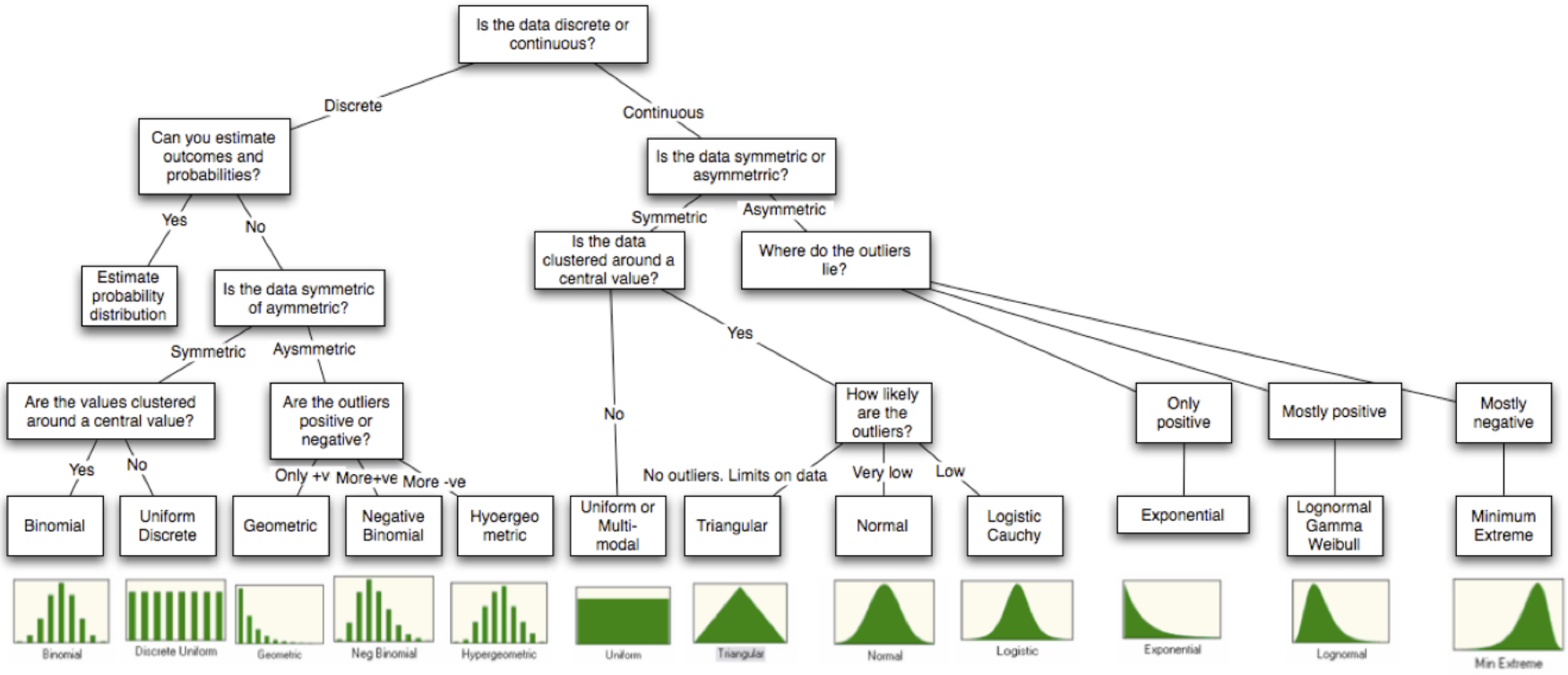


Figure 6A.15: Distributional Choices



fitdistrplus: Help to Fit of a Parametric Distribution to Non-Censored or Censored Data

Extends the fitdistr function (of the MASS package) with several functions to help the fit of a parametric distribution to non-censored or censored data. Censored data may contain left censored, right censored and interval censored values, with several lower and upper bounds. In addition to maximum likelihood estimation method the package provides moment matching, quantile matching and maximum goodness-of-fit estimation methods (available only for non-censored data).

Version:	1.0-4
Depends:	R (≥ 2.10.0)
Imports:	stats, survival
Suggests:	actuar, rgenoud, mc2d, gamlss.dist
Published:	2015-02-23

```
> summary(fitg)
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
        estimate  Std. Error
shape 4.00825257 0.341336047
rate  0.05441911 0.004935468
Loglikelihood:  -1253.625   AIC:  2511.25   BIC:  2518.325
Correlation matrix:
          shape      rate
shape 1.0000000 0.9384381
rate  0.9384381 1.0000000

>
```

Figure:

Extreme Value Theory

- The Extreme Value Theory (EVT) is extensively used for modelling very large and/or very small events.
- Usually the focus of the analysis is the estimation of very extreme quantiles or tail probabilities.
- It is widely used in several areas, such as environment, insurance, weather and hydrology.
- Extreme value theory is important for assessing risk for highly unusual events, such as 100-year floods.
- Several R packages have been developed for fitting models in this framework.

Extreme Value Theory

- The field of extreme value theory was pioneered by Leonard Tippett ($1902 - 1985$).
- Tippett was employed by the British Cotton Industry Research Association, where he worked to make cotton thread stronger. In his studies, he realized that the strength of a thread was controlled by the strength of its weakest fibers.
- With the help of R. A. Fisher, Tippet obtained three asymptotic limits describing the distributions of extremes. The German mathematician Emil Julius Gumbel codified this theory in his 1958 book **Statistics of Extremes**, including the Gumbel distributions that bear his name.

evir: Extreme Values in R

Functions for extreme value theory, which may be divided into the following groups; exploratory data analysis, block maxima, peaks over thresholds (univariate and bivariate), point processes, gev/gpd distributions.

Version:	1.7-3
Depends:	stats
Published:	2012-07-26
Author:	Bernhard Pfaff [aut, cre], Alexander McNeil [aut] (S original (EVIS)), Alec Stephenson [trl] (R port of EVIS)
Maintainer:	Bernhard Pfaff <bernhard at pfaffikus.de>
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]