INTRODUCING A FULLY GENERALIZED NORMAL DISTRIBUTION – GENERALIZED FOR SKEWNESS AND KURTOSIS

[Accompanying Code Package: www.github.com/quantsareus/introducing_a_fully_generalized_normal_distribution]

by

H{e[ik)o S}c[hl)i[ng}ma(nn [1]

from L{eo]p(o]lds(hoe]}e

email{at]quantsareus[dot}net

2024-01-04

is Presented As Dissertation Thesis in Partial Fulfillment
of the Requirements for the Degree

Doctor / PhD of ____________________ in __________________________

at

____
(University)

[1] MBA (German Dipl. Kfm.) Faculty of Economics, University Muenster.

The Dissertation Committee for

____

certifies the approval for the following dissertation:

INTRODUCING A FULLY GENERALIZED NORMAL DISTRIBUTION –

GENERALIZED FOR SKEWNESS AND KURTOSIS

Dissertation Committee

____
Chair (Title Name Signature)

____
Committee Member (Title Name Signature)

____
Committee Member (Title Name Signature)

University/ Faculty Representative

____
University/ Faculty Dean (Title Name Signature Name of the University/ Faculty)

____
Date approved

Author’s Declaration of

Own Significant Performance

The author declares, that he has created this work (including the accompanying code package) on his own and that significant scientific assistance, if existent – especially in the central subject of the work, is truly disclosed as alien performance. Third party scientific performances, that have been taken from other author’s works by their word or by their meaning, are disclosed by their source. This is also true for formulas, tables, figures, constructional plans and other illustrating schemes. Except they are familiar scientific standard of the discipline or well known common knowledge.

Further, the author declares, that he does <u>not</u> present an old central scientific part of his own work, for which he has already successfully achieved a previous scientific degree, without significant further development as the central scientific part of this work, again.

L{eo]p(o]lds(hoe]}e, 2024-01-04

[illegible]

H{e[ik)o S}c[hl)i[ng}ma(nn

PREFACE

This is an OPEN DISSERTATION REQUEST to all data-science, statistics, econometrics, probability mathematics and informatics faculties and alike. The author does NEITHER have a PhD-father-professor NOR a university-mother in this regard, yet! The author will add a phd_request_closed file to the accompanying code package www.github.com/quantsareus/introducing_a_fully_generalized_normal_distribution , as soon a scientific faculty/ university has granted a PhD degree. As long there is no such file in the current version of the accompanying code package, the PhD request is still open. ANY university and faculty in the world is welcome to grant a PhD title or equivalent for this work.

This PhD thesis is in many aspects a work in the gap. The author has experienced, that most data-scientists would like to have a flexible distribution, that covers the typical amounts of skewness and excess kurtosis, so that a broad range of empirical data could be processed in this regard automatically. New distributions are usually invented theoretically by probability mathematicians, who have the optimum skills to do that. However, it is not possible to order a custom-made fully generalized normal distribution from probability-mathematicians, e.g. as a master or PhD thesis project. Because this would be a project with an uncertain solve, thus not suitable for a master or PhD thesis work. Following, the author as a practicing applied data-scientist has researched several years to develop a fully generalized normal distribution, himself. The author is convinced, that the research is scientifically very useful. As the development work took several years, the author also thinks the work might be worth honoring with a PhD degree. As the fully generalized normal distribution solution provided in this work is of programming and simulation nature, the work is also presented as code (instead of mathematical formulas), which does not fit not into the current scientific segmentation of data science. Especially the author has been told, that the work did contain too less mathematical performance to get accepted as a PhD thesis in probability-mathematics. On the other hand the author has also frequently been told, that the topic of thesis was too mathematical and too much programming to get accepted at a faculty of applied statistics. Further he has been told, that applied statistics faculties only adopted already invented mathematical solutions to the problems of their discipline. But they did not invent new mathematical features like distributions, their selves. Chairs for Data-Science with a predominant focus on programming are still rare. So, the scientific segmentation of data science caused, that the author has not been able to find a PhD fathering professor for the thesis, although he has tried a lot. Thus, it is finally published as an OPEN DISSERTATION REQUEST to all data-science, statistics, probability mathematics and numerical informatics faculties (and alike), who ever might find “the first brick of the quantum mechanics revolution in statistics” worth a PhD title.

DANKSAGUNG / THANKS

an / to

meine Eltern / my Parents

dafür, dass sie mir eine gute Ausbildung ermöglicht haben. /

for enabling a good education for me.

Table of Contents

List of Illustrations

List of Tables

Appendices

Legend

aka	Also known as
AQRNG	Alternative quality random number generator
c.p.	Ceteris paribus (Usually: Altering just this parameter, keeping all the others.)
CDF	Cumulated density function, the anti-derivative of the PDF
Construction location parameter	Aka location parameter
Deviation parameter	Aka scale parameter
e.g.	Exempli gratia (examples given)
ff	Further following (pages)
FGN	Fully generalized normal
FGND	Fully generalized normal distribution
GND	Generalized normal distribution
GND-V1	Nadarajah’s generalized normal distribution version 1
GND-V1a	Generalized normal distribution version 1 adopted by the author
GND-V2	Hosking’ s and Wallis’ generalized (skewness-)normal distribution version 2
Histogram	Frequency diagram
kurtosed	Containing excess-kurtosis (kurtosis<>3)
ND	Normal distribution
PDF	Probability density function
Power and kurtosis parameter	Aka shape parameter of a symmetrical distribution (free of skewness)
q. e. d.	Quod erat demonstrandum (What has been to be shown)
QRNG	Quality random number generator
RNG	Random number generator
skewed	Containing skewness

1 Abstract

The author is introducing a new, super-flexible, fully generalized normal distribution (FGND), generalized for skewness *and* kurtosis, that hopefully will build a proper foundation for automated processing of empirical data regarding skewness and kurtosis, especially for significance testing. Beside the naked PDF the author is also delivering a moment table calculator, an appropriate random number generator and an identification procedure for identifying the corresponding theoretical distribution from data, which is the central problem for the statistical application of a new complex distribution. All components of the solution have been fully worked out as *a functioning proof-of-concept demonstrator in Python*, of which whole code including the validation tests is passed over as a Python source code package and also as a code print in the Appendix, in order to get the research easily reproduced and to allow for some direct application (after a learning process).

2 Introduction

2.1 Motivation

In (applied) data-science it is an *absolute* nuisance, that inference tests, especially as taught in applied statistics, mostly remain a fragmented piece-work for the following reasons:

1. A generally acknowledged test is typically acknowledged for *one* (to validate) parameter (for instance the OLS regression intercept $beta_0$), based on its assumed *individual* distribution. Thus, the relevant test has to be chosen appropriately from a full test-universe out of hundreds individual tests, which is more difficult to automate than the other parts of a data-science survey, that are almost fully automatized, already.
2. Sometimes there simply is no inference test, that meets the empirical distribution of the computed (to validate) parameter, especially when significant amounts of skewness and kurtosis are present. This – when being strict – results, that the calculated model parameter cannot be validated fully accurate.
3. Further, there may also be more than one inference test considered relevant in statistics literature for the observed (to validate) parameter, that are founded on different distribution assumptions. Which may – for a given significance niveau alpha – sometimes provide contrary statistical results (e.g. one test significant, but the other test not significant).

Thus, in data-science practice well transparent and computing efficient parametric models are often avoided and get replaced by non-parametric ones simply for the reason to save the expensive manual work of inference testing the model parameters.[2] Therefor, the author is introducing a new super-flexible – *however also more complex* – fully generalized normal distribution, which is generalized for skewness and kurtosis, that hopefully will build a broad universal foundation for inferential (hypothesis) testing of most empirical practice data and following will hopefully enable an automated inference testing workflow for parametric model parameters.

2.2 Introduction of the Central Development Problem

The central problem for the statistical applicability of a new complex distribution is the “identifying the corresponding theoretical distribution from data” problem. Identification here means *identifying the construction parameters* of the theoretical distribution, which – in relative frequencies – is most similar to the data. Standardly the identification is performed in two steps. In the first step the PDF of the theoretical distribution is abstractly solved by an analytical solution. E.g. there are solution formulas for the power parameter and the deviation parameter of the PDF. In the second step these solution formulas are applied to the data, so that the parameters are calculated, of which PDF is most similar to the relative frequencies of the present data sample. The big dilemma here is, that the data-scientist’s favorite distribution, fully flexible in skewness and kurtosis, cannot get analytically solved any more, following cannot identify the corresponding theoretical distribution. Thus till now, a fully generalized

[2] Reflecting the author’s personal business experience.

normal distribution, flexible in both, skewness *and* kurtosis, has just been a (applied) data-scientist's dream.

2.3 General Concept of the Solution

The author has chosen another path to solve the "identifying the corresponding theoretical distribution from data" problem. Instead of solving the PDF of the new fully generalized normal distribution analytically, the solution is provided algorithmically by comparing the statistical moments skewness and kurtosis of empirical data samples to the ones most similar in the moments table of the standard fully generalized normal distribution (with deviation parameter counting 1). The design of the new FGN-distribution is this way, that standard FGNDs (with deviation parameter counting 1) are unique in skewness and kurtosis, following can be identified by them. The other half of the identification problem is accomplished by statistical z-standardization from any FGND (with any deviation parameter) to the standard FGND (with deviation parameter counting 1), which is another useful feature of the new FGND.

However, the avoiding of an analytical solution has been achieved at the costs, that the solution has become much more computing extensive. It requires several numerical and algorithmic solving procedures, such as a moment table calculator, an identification procedure and also an appropriate quality random number generator to deliver precise test data for identification. Especially the iterative precision improvement steps of the quality random number generator, as the last missing piece of a testable reproduction cycle, has required a lot of working month.

2.4 Structure and Presentation Form of the Work

The work is presented in the so called "classical straight scheme", which is pertinent for German scientific thesis. Following, the scientific discoveries are *not* presented in timely order, *nor* presented with all the difficulties that arose while achieving them. Instead the research is shown through the perfect world glasses as a lecture easy to consume, that strongly focuses on the central problems and their solutions and tries to avoid logical anticipations as much as possible. But known pitfalls are mentioned, where they are relevant for reproducing the solution.

As common for thesis following the classical straight scheme, chapter 3 is a foundation part, that begins with "The State of Research in Generalized Normal Distributions", followed by the introduction of the central tools used within the work ("Central Tools and Terms"). Chapter 4 is the first main part, which presents the development process of the solution step by step through the perfect world glasses ("Developing a Fully Generalized Normal Distribution"). Chapter 5 is the second main part, that evaluates the found solution ("Survey of the Fully Generalized Normal Solution"). It surveys the meeting of the development considerations, the performed testing setup and contains a validity evaluation of the solution. Finally, the author presents his conclusions in chapter 6. The conclusions have been split into conclusions regarding application and in conclusions regarding further research prospects.

All components of the solution have been fully worked out as a functioning proof-of-concept demonstrator in Python, of which source code (including the validation tests) is either passed over as accompanying Python source code package and also additionally attached as a code print in the Appendix.

3 State of Research and Central Tools

3.1 State of Research in Generalized Normal Distributions

3.1.1 Gauss’ Normal Distribution

Already the famous German mathematician C. F. Gauss, living in the 18th and 19th century, stated the density function of the well known normal distribution. The probability density function f of the normal distribution is

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normal Distribution (Standard Notation)

$$\Leftrightarrow f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-0.5\left(\frac{x-\mu}{\sigma}\right)^2}$$

Normal Distribution (Sensible Notation)

Where

$\sqrt{\pi}=\Gamma(0.5)$ [3]

$$\Leftrightarrow f(x)=\frac{1}{\sigma 2^{0.5}\Gamma(0.5)}e^{-0.5\left(\frac{x-\mu}{\sigma}\right)^2}$$

Normal Distribution (Gamma Notation)

Where
x: Random Variable
μ : Mean
σ : Standard Deviation (Square Root of Variance)

The statistical moments of the normal distribution are:

Expected Value: μ

Variance: σ^2
Skewness: 0
Kurtosis: 3
Excess-Kurtosis: 0

3 [Spiegel (1994)] p. 1.

Here it is important to mention, that the construction parameters of a distribution and its moments are theoretically two different things. As the construction location parameter counts the same as the moment expected value in the normal distribution, and the construction parameter for deviation counts the same as the moment variance, it is a common standard to simply denote the normal distribution short-cut-wise by the moment symbols μ and σ . However, this short-cut-notation will not be possible for more complex (fully) generalized normal distributions any more, *as the construction parameters and the moments do fall apart!*

When the construction location parameter $\mu=0$ and the deviation parameter $\sigma=1$, the normal distribution simplifies to the standard normal distribution.

$$f(x)=\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-0.5x^2}$$

Standard Normal Distribution

Where
x: Random Variable

3.1.2 Nadarajah's Generalized Normal Distribution Version 1

[Nadarajah (2005)] introduced a general normal distribution generalized for power/ kurtosis (only). Nowadays it is either simply referenced as generalized normal distribution or referenced as generalized normal distribution version 1 (GND-V1). The GND-V1 is highly relevant in this work, as it has been the starting point for the author's fully generalized normal distribution. The PDF f(x) of Nadarajah's GND-V1 is

$$f(x)=\frac{k}{2z\Gamma(\frac{1}{k})}e^{-(\frac{|x-c|}{z})^k}$$

Nadarajah's GND-V1

Where
x: Random Variable
c: Construction Location Parameter
z: Deviation Parameter
k: Power and Kurtosis Parameter
$\Gamma()$: Gamma Function

3.1.3 Hosking' s and Wallis' Generalized Normal Distribution Version 2

[Hosking, Wallis (1997)] invented a skewness-normal distribution generalized for skewness (only), which is nowadays referenced as skewness-normal distribution or – more common – simply as generalized normal distribution version 2 (GND-V2). The

GND-V2 is just mentioned once for completeness, here. It has not played any role in the development of the fully generalized normal distribution.

3.1.4 Generalized Gaussians

Generalized Gaussians are a generalized normal distribution family, which is mainly used in the domains of physics, electrical engineering and signal processing. From the point view of a statistical categorization a plain vanilla Generalized Gaussian is a composed distribution out of Nadarajah’s generalized normal distribution version 1 (without norming pre-factor) and a uniform distribution, which has two parameters to level the two distributions against each other. The GND-V1 part represents a signal, the uniform distribution part represents the noise. However, the first Generalized Gaussian effectively has been invented much more early than Nadarajah’s generalized normal distribution version 1. [4] [5]

Although there are some modifications and alternative versions of the plain vanilla Generalized Gaussian, the author did not get aware, that among those there would exist one to generalize for skewness, as well. Nor did the Generalized Gaussians play any role in the development of the fully generalized normal distribution. So, the Generalized Gaussians are just mentioned once for a global orientation in the world of generalized normal distributions, here.

3.1.5 State of Research Summary

The full universe of all distributions ever invented is very large. Following, it is not possible to research the full universe exhaustively; also not possible because a significant proportion of those distributions has not been digitalized. Instead the author has queried scholar.google.com, google.com, bing.com, ecosia.com, duckduckgo.com and relevant scientific article databases for the keywords {“fully generalized normal distribution”, “normal distribution generalized skewness kurtosis”, “normal distribution generalized skewness power”, “generalized normal distribution”, “generalized gaussian”}. Based on these keywords and search methods the author did not find a fully generalized normal distribution, which is generalized for skewness *AND* kurtosis as well.

3.2 Central Tools and Terms

3.2.1 Probability Density Function (PDF)

3.2.1.1 PDF Definition

As probability density functions (PDFs) are a familiar standard of the data-science discipline, the author is just presenting the relevant parts and the critical points. Basically, a PDF is the mathematical representation of an abstract theoretical distribution. Com-

[4] [Varanasi, Aazhang (1989)].

[5] [Holzner S. ((2013)] p. 88.

monly acknowledged a PDF f(x) defines the probability of a random variable x within the interval [a, b] as follows:[6]

$$P(a \leq x \leq b) = \int_a^b f(x)dx$$

Equivalently expressed in the cumulative distribution function F(x), the anti-derivative of the PDF:[7]

$$\Leftrightarrow P(a \leq x \leq b) = F(b) - F(a)$$

Please notice, that in the commonly acknowledged PDF definition the probability of a certain x-interval corresponds to the area under the PDF curve, respectively the difference of the CDF, but not to the PDF value itself.[8] Thus, PDF-values can – for a limited x-interval – be larger than one; the frequently found statement, that PDF values could be maximum 1, is wrong.[9]

3.2.1.2 PDF Axioms

Kolmogorov postulated, that the probability of an event E_i out of set of all possible outcomes E with n elements has to fulfill the following axioms:[10]

Kolmogorov’s Probability Axiom No. 1

$$P(E_i) \geq 0$$

Kolmogorov’s Probability Axiom No. 2

$$\sum_{i=1}^{n} P(E_i) = 1$$

Translating the two axioms to a continuous function a PDF has to fulfill the following PDF axioms:[11]

PDF Axiom No. 1 (Non-Negativity Axiom)

$$f(x) \geq 0 \quad \text{for all} \quad x \in \mathbb{R}$$

PDF Axiom No. 2 (Norming Axiom)

$$\int_{-\infty}^{\infty} f(x) = 1$$

3.2.1.3 Direct PDF Interpretation

[Fahrmeir, Kuenstler, Pigeot (2004)] p. 87 ff interpret a PDF as “an infinitesimal frequency histogram for the values of an independent variable x. Thus, real visual – not

6 [Spiegel (1994)] p. 256.
7 [Spiegel (1994)] p. 257.
8 [Ushakov (2001)].
9 [Wikipedia, Wahrscheinlichkeitsdichtefunktion (2022)].
10 [Kolmogorov (1933)].
11 [Stirzaker (2007)].

infinitesimal narrow – frequency bar charts overlap the PDF this way, that the positive and negative difference areas between the top of the bar and the PDF are equal size.”[12] The author generally follows this interpretation. However, it has to be pointed out clearly here, that such direct PDF frequencies for x-values are *not* normed probabilities (relative frequencies), but just some frequency *weights*, that do *not* sum up to 1. The author has explicitly checked in a simulation experiment, that drawing a *large* number of independent pseudo-continuous x-values with PDF-value-weights generates a data population with the 4 moments mean, variance, skewness and kurtosis, that follows the theoretical moments of the PDF calculated by numerical integration.

This simulation experiment, which originally was just meant to check [Fahrmeir, Kuenstler, Pigeot (2004)] p. 87 ff direct PDF interpretation by statistical operationalization, years later has led the author to the alternative quality random number generator in section “Alternative Quality Random Generator”. Regarding the PDF norming axiom an alternatively generated random variable drawn from an independent pseudo-continuous variable x with a constant small step-size $\bar{\Delta} x$ has to fulfill the axiom straight forward as follows:

PDF Axiom No. 2 (Norming Axiom)

$$\sum_{i=1}^{n} f(x_i) \cdot \bar{\Delta} x \simeq 1$$

3.2.1.4 PDF Design

The design of a PDF f(x) is usually started with some core function f’(x), that has one maximizing extremum (no further maxima, minima or saddle points) and converges to 0 for x→ -inf and x → +inf. The left and right ends of the PDF are called tails. To fulfill the PDF norming axiom mentioned above, namely to sum up to 1 in total, the core function is divided by a preliminary norming term, that counts the integral of the core function f’(x) from -inf to +inf.

E.g. in the normal distribution the core function is

$$f'(x) = \mathrm{e}^{-0.5\left(\frac{x-\mu}{\sigma}\right)^2}$$

Core Function of the Normal Distribution

and the preliminary norming term containing the value of the norming integral is

$$\frac{1}{\sigma\sqrt{2\pi}}$$

Preliminary Norming Term of the Normal Distribution

The common acknowledged normal distribution is only a proper PDF regarding the norming axiom, if

[12] Translated from originally German text.

$$\int_{-\infty}^{\infty} e^{-0.5\left(\frac{x-\mu}{\sigma}\right)^2} = \sigma\sqrt{2\pi}$$

.

When two PDFs should to be compared, it is usually sufficient to compare their core functions. As the preliminary norming term given the parameters is a constant, which just norms the integral of the core function to 1, their characteristic properties lay in their core functions.

3.2.2 Statistical Moments

Statistical moments are another familiar standard of the data-science discipline. Statistical moments can be calculated for data and for a PDF. Statistical moments are ordered by grades:[13]

1. Expected Value/ Mean
2. Variance/ Standard Deviation
3. Skewness
4. Kurtosis

Statistical moments are linked to mathematical moments, so the statistical moments of a probability density function (PDF) can be calculated using a moment generating function.[14] Mean and variance are the lower moments. Skewness and kurtosis are the higher central moments, which are derived a by moment generating function of grade 3 and 4.[15]

The terms “central”and “higher” here express, that the PDF/ the data is normalized for location (re-centered) and for variance, before it gets analyzed for the higher central moments. Thus, the lower moments and the higher moments are independent, which especially includes, that a z-transformation of a PDF/ the data does not effect the higher moments skewness and kurtosis.[16]

Regarding statistical moments calculated from *data*, beginning with moments of grade two (variance) and ongoing, there also exist sample size adjusted versions.[17] However, sample size adjustments of moments can be ignored in large samples (over #100).[18] Small samples sizes below #100 do not appear in this work. Thus, only unadjusted statistical moments are presented here.

3.2.2.1 Statistical Moments from Data

3.2.2.1.1 Expected Value

$$\mu=\frac{1}{n}\sum_{i=1}^{n} x_i$$

Where

[13] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.
[14] [Frees (2004)] p. 380.
[15] [Wikipedia, Wahrscheinlichkeitsdichtefunktion (2022)].
[16] [Wikipedia, Wahrscheinlichkeitsdichtefunktion (2022)].
[17] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.
[18] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.

n: Total number of observations
i: Index
x: Random Variable

3.2.2.1.2 Variance

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

Where
n: Total number of observations
i: Index
x: Random Variable
μ : Mean

3.2.2.1.3 Skewness

$$\gamma_S = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^3$$

Where
n: Total number of observations
i: Index
x: Random Variable
μ : Mean
σ : Standard deviation (square root of variance)

Interpretation [19]

$\gamma_S < 0$: Left skewed distribution

$\gamma_S = 0$: Symmetric distribution without skewness

$\gamma_S > 0$: Right skewed distribution

3.2.2.1.4 Kurtosis

$$\gamma_K = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^4$$

Where
n: Total number of observations
i: Index
x: Random Variable
μ : Mean
σ : Standard Deviation (Square Root of Variance)

[19] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.

Interpretation [20]

$\gamma_K < 3$: Less peaked than the normal distribution

$\gamma_K = 3$: As peaked as the normal distribution

$\gamma_K > 3$: More peaked than the normal distribution

More peaked distributions are also more heavy tailed (aka fat tailed), meaning that they carry a larger proportion of relative frequency in the tails than the normal distribution does.[21]

3.2.2.1.5 Excess-Kurtosis

In US statistics packages and sometimes also in international statistics libraries kurtosis gets denoted as excess-kurtosis γ_{K_E} .[22]

$$\gamma_{K_E} = \frac{1}{n}\left[\sum_{i=1}^{n}\left(\frac{x_i - \mu}{\sigma}\right)^4\right] - 3$$

Where
n: Total number of observations
i: Index
x: Random Variable
μ : Mean
σ : Standard Deviation (Square Root of Variance)

Interpretation

$\gamma_{K_E} < 0$: Less peaked than then the normal distribution

$\gamma_{K_E} = 0$: As peaked as the normal distribution

$\gamma_{K_E} > 0$: More peaked than the normal distribution

[20] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.
[21] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 75, 76.
[22] Python library scipy.stats.kurtosis(), accessed 2021.

3.2.2.2 Statistical Moments from the PDF by Integration

As already mentioned above, instead from data, statistical moments can also be calculated from a PDF f(x) by integration. [23] [24] [25] Their interpretation is exactly the same as the interpretation of the version from data.

3.2.2.2.1 Expected Value

$$\mu=\int_{-\infty}^{\infty} x f(x) dx$$

Where
x: Random Variable
f(x): PDF for x

3.2.2.2.2 Variance

$$\sigma^2=\int_{-\infty}^{\infty} (x-\mu)^2 f(x) dx$$

Where
x: Random Variable
f(x): PDF for x
μ : Mean

3.2.2.2.3 Skewness

$$\gamma_S=\int_{-\infty}^{\infty} \left(\frac{x-\mu}{\sigma}\right)^3 f(x) dx$$

Where
x: Random Variable
f(x): PDF for x
μ : Mean
σ : Standard Deviation (Square Root of Variance)

3.2.2.2.4 Kurtosis

$$\gamma_K=\int_{-\infty}^{\infty} \left(\frac{x-\mu}{\sigma}\right)^4 f(x) dx$$

Where
x: Random Variable
f(x): PDF for x
μ : Mean
σ : Standard Deviation (Square Root of Variance)

[23] [Frees (2004)] p. 380.
[24] [Wikipedia, Wahrscheinlichkeitsdichtefunktion (2022)].
[25] [Kaas, Goovaerts, Dhaene (2008)] p. 334.

3.2.2.2.5 Excess-Kurtosis

$$\gamma_{K_E} = \int_{-\infty}^{\infty} \left(\frac{x-\mu}{\sigma}\right)^4 f(x)\,dx - 3$$

Where
x: Random Variable
f(x): PDF for x
μ : Mean
σ : Standard Deviation (Square Root of Variance)

3.2.3 Random Number Generation

In random number generation (RNG) one has chosen a certain theoretical distribution and one wants to produce a *data series* of a random variable x, that follows the distribution's relative frequency, which is determined by the PDF.[26] Generally, random number generation (aka random number generator) is required for statistical simulation and for (sub-)sampling. Specifically, without a random number generator no (artificial) data samples can be produced, that follow a new developed distribution. Thus, random number generation is absolutely essential to make a new distribution statistically applicable.

There is a wide range of techniques for random number generation. Most popular statistical RNGs generate uniform random numbers in the first step and then transform them to the desired distribution in the second step. In most statistical packages and libraries currently the standard algorithm for generating pseudo-random uniform numbers within the interval [0, 1] is Merisenne-Twister.[27] For the second transform into the desired distribution the inversion method (aka probability integral transform method) and the rejection method are most popular.[28]

However, an applied data-scientist is primarily interested in the quality of generated random data. Among others [Knuth (1997)] has well presented the general spectrum of abstract, generally acknowledged quality tests for random number generators, respectively their random data output. However, the author does not have a general abstract research interest in random number generator quality. Instead the author has a specific quality interest regarding proper moments of generated random number data, as later in this work the author wants to develop a distribution identification procedure, that is based on the moments variance, skewness and kurtosis. Therefor, [Knuth (1997)] 's list of random number generator quality tests has been limited helpful, as most of these tests are in form of significance tests, that most possibly exhibit another significance sensitivity as the author's planned identification procedure. Thus, the author has tested the random number quality in the moments variance, skewness and kurtosis directly and defined the standard errors in the moments variance, skewness and kurtosis as relevant quality criteria.[29]

26 [Wikipedia, Pseudo-random number sampling (2022)].
27 Python library numpy.random, accessed 2021.
28 [Kaas, Goovaerts, Dhaene (2008)] p. 334.
29 In this practical random number quality view the standard error(s) of the central statistical feature(s), that effectively get applied to produce the solution, make up the relevant quality criteria. E.g. in a portfolio diversification simulation – not the standard error of the moments – but the standard error

3.2.4 Z-Standardization

The x_t-values of a distribution following a PDF with a core function f'(x_t) of the form

$$f'(x_t) = e^{-(\frac{|x_t - c|}{z * u})^k}$$

can be transformed with division by u

$$x_t \rightarrow x = x_t / u \quad ,$$

so that x follows the core function $f'(x)$

$$f'(x) = e^{-(\frac{|x - c|}{z})^k} \quad .$$[30]

Basically, a z-standardization of a generalized normal distribution performs the same mathematical operation as standardizing the normal distribution.[31] [32] However, the result is slightly different. While the normal distribution gets standardized to standard deviation 1 (, which is the second *moment*), a generalized normal distribution gets standardized to deviation construction *parameter* z=1 (aka scale parameter), while its standard deviation moment also depends on the power-and-kurtosis parameter k.

When a z-standardization is applied in the opposite direction away form z=1, it is called z-transformation. The direction of the alternation of the construction deviation parameter z can be up or down.

of the Pearson correlation between generated random (return) numbers would be the relevant criterion.

30 [Tabachnick, Fidell (2000)] p. 80.

31 [Fahrmeir, Kuenstler, Pigeot (2004)] p. 291.

32 Z-standardization of a normal distribution is also the reason, why it is sufficient to table the standard normal distribution (and not any normal distribution).

4 Developing A Fully Generalized Distribution Solution

4.1 Development Considerations

4.1.1 PDF Axioms

Any new distribution has to fulfill the PDF axioms to make up a proper distribution. Thus a new FGND has to fulfill the following PDF axioms:[33] [34]

PDF Axiom No. 1 (Non-Negativity Axiom):
$f(x) \geq 0$ for all $x \in \mathbb{R}$

PDF Axiom No. 2 (Norming Axiom):
$\int_{-\infty}^{\infty} f(x) = 1$

4.1.2 Flexibility in Skewness and Kurtosis

Adding flexibility in skewness and kurtosis should especially include the ability to alter skewness away from zero and kurtosis away from 3, *simultaneously*. It should be possible to accomplish the alternations in seamless degrees. For handy control of skewness and kurtosis, they ideally should be steered separately. Meaning, ideally there is one parameter to steer skewness and there is another parameter to steer kurtosis.

4.1.3 Identifying the Corresponding Theoretical Distribution

Identifying the corresponding theoretical distribution means exactly the opposite of random number generation.[35] One has got a data sample of a variable x with (empirical) relative frequencies and one wants to find that theoretical distribution with the most similar relative frequencies to it. Typically out of a preselected set of distributions; e.g. one considers only normal distributions with alternative construction parameters.

Distribution identification is required for the major part of statistical procedures, e.g. for significance testing. So, a new developed distribution without an “identifying the corresponding theoretical distribution from data feature” is not much worth. Thus, the identification problem is the most central problem for developing a new statistically applicable distribution.

[33] Compare section “PDF Axioms”.
[34] [Stirzaker (2007)].
[35] Compare section “Random Number Generation”.

4.1.4 Further Mathematical Properties

4.1.4.1 Uni-Modal PDF

In statistics universal distributions are uni-modal distributions.[36] The term mode in modal refers to the point of maximum frequency. Regarding the design of the PDF this means, it has to be to be a single maximum function. This includes, the PDF is free of kinks and free of local extrema, thus does not have local maxima and minima.

4.1.4.2 Steady PDF Derivatives

A lot of statistical methods internally apply first and second derivatives of the PDF. E.g. the maximum likelihood parameters of a GLM regression are analytically found by setting the first derivation of the PDF to zero and solving the relevant part for the parameters.[37] Numerical procedures, e.g. Newton’s Method and Fisher-Scoring, also apply second order derivatives.[38] As these methods require derivatives, steady PDF derivatives are an important mathematical feature for a new distribution.

4.1.4.3 Core Function – Easy Analytical Handling versus Easy Application

When mathematicians create a new generalized normal distribution they prefer a pre-factor free core function in the PDF. E.g. there is no pre-factor in the core function of Nadarajah’s GND-V1 and there also is none in the Generalized Gaussian. The reason for this obvious. Every further (factor) term needs to be treated in the analytical derivation of the PDF. But in the core function of the Normal distribution there is a pre-factor – 0.5.[39] In statistics, especially in applied statistics, the Normal distribution is the outstanding central distribution. Not surprisingly kurtosis, the forth moment of the normal distribution, which counts 3, has been relabled to excess-kurtosis counting zero. As the normal distribution is outstanding and every first semester statistics student gets to know its moment properties, the author has decided about the trade-off in favor for easy application. So, the core function of the new to create fully generalized normal distribution will also have a pre-factor 0.5. Thus, it will be directly downward compatible to the Normal distribution.

4.1.5 Quality Random Number Generation

When one wants to prove to be capable to identify the corresponding theoretical distribution by a data-simulation instead of solving the distribution analytically, one has to generate that specific data using a random number generator, in the first step. However, there is no random number generator for the new FGND, yet. Thus, needs to be developed as well.

[36] Although there are also multi-modal distributions known in statistics, these ones are mostly used for very special scenarios. E.g. the multi-modal Tweedie distribution is used in insurance business to model the loss severity resulting from a multi path process (damage report, policy coverage/ no policy coverage).

[37] [Eliason (1993)] p. 12.

[38] [Kaas, Goovaerts, Dhaene (2008)] p. 313 – 315.

[39] The pre-factor in the core function will be discussed in detail the next section “Deriving the Probability Density Function”.

Second, the more flexible the distribution is, that should be identified, the more elevated quality is required for its random number generator. If the the data is not generated in high accuracy, the following re-identification trial will identify the neighbor distribution, to be specific the construction parameters of the neighbor distribution, instead.

4.2 Deriving the Probability Density Function

4.2.1 Adopted Generalized Normal Version 1

Basically, Nadarajah’s GND-V1 is a prospectus candidate to get fully generalized for skewness and kurtosis. However, the author has decided for directly downward compatible to the Normal distribution. Thus, the pre-factor 0.5 must be added to the core function.

The core function f’ of the normal distribution is

$$f'(x)=\mathrm{e}^{-0.5\left(\frac{x-\mu}{\sigma}\right)^2}$$

Normal Distribution Core Function

Where
x: Random Variable
μ : Expected Value
σ : Standard Deviation (Square Root of Variance)

The core function f’ of Nadarajah’s Generalized Normal Distribution Version 1 is

$$f'(x)=e^{-\left(\frac{|x-c|}{z}\right)^k}$$

Nadarajah's GND-V1 Core Function

Where
x: Random Variable
c: Construction Location Parameter
z: Deviation Parameter
k: Power and Kurtosis Parameter

For instance to reproduce the standard normal distribution with Nadarajah’s GND-V1 its power must be set k=2 and the deviation parameter must be set to $z=\sqrt{2}$. This is already another deviation parameter as the one from the ND, $\sigma=1$. Further, if the power k in GND-V1 is altered to 2.5 (to generate a power-deformation of the normal distribution), also the deviation parameter z needs to be adjusted to $z=\sqrt[2.5]{2}$. Especially this is not handy for numerically unexperienced users.

Therefor, Nadarajah’s GND-V1 core function is adopted to a directly downward compatible one with the same pre-factor 0.5 as the normal distribution. Thus, its deviation parameter z directly corresponds to the Normal distribution's deviation parameter σ , now.

Adopted GND-V1 core function f':

$$f'(x)=e^{-0.5\left(\frac{|x-c|}{z}\right)^k}$$

Adopted GND-V1 Core Function

To sum the total probability up to one again, the preliminary norming integral also needs to be adopted.[40] The author has figured out by numerical integration and further simulation experiments, that

$$\int_{-\infty}^{\infty} e^{-0.5\left(\frac{|x-c|}{z}\right)^k} dx=\frac{2z\Gamma\left(\frac{1}{k}\right)}{k0.5^{\frac{1}{k}}}=\lambda$$

Where
x: Random Variable
c: Construction Location Parameter
z: Deviation Parameter
k: Power and Kurtosis Parameter
$\Gamma()$: Gamma Function

When the value of the norming integral is assigned to λ , making up the denominator of the new preliminary norming factor, then the new adopted "GND-V1a" denotes:

$$f(x)=\frac{1}{\lambda}e^{-0.5\left(\frac{|x-c|}{z}\right)^k}$$

GND-V1a

Where

$$\lambda=\frac{2z\Gamma\left(\frac{1}{k}\right)}{k0.5^{\frac{1}{k}}}$$

Norming Term

x: Random Variable
c: Construction Location Parameter
z: Deviation Parameter
k: Power and Kurtosis Parameter
$\Gamma()$: Gamma Function

[40] Compare section "PDF Axioms".

4.2.2 Plugging Two Halfnormals with Different Powers Together

In data-science, when the two halfs of a normal distribution are used separately, they are called half-normals. Theoretically, if a left half-normal and a right half-normal with *different power parameter k* were plugged together to a new PDF, they would generate skewness. However, a plugged-together-PDF is required be a continuous function (as any other PDF).

At $x=c$ the GND-V1a core function f'

$$f'(x)=e^{-0.5\left(\frac{|x-c|}{z}\right)^{k}}$$

GND-V1a Core Function

always values 1, *regardless of the power parameter k*. Thus, when two GND-V1a core functions with different powers k are plugged together at $x=c$, they will generate a new *continuous* function.

In order to accomplish separate control of skewness and kurtosis, the requirement from section “Aptness Focus for Generalization”, there is a new asymmetry parameter a introduced to generate different powers in the left and in the right half-normal. In the left branch the original power parameter k is substituted by $a\cdot k$, in the right branch the original k is substituted by a/k . Following, an asymmetry parameter $a>1$ creates a right skewed function.

As before, when the core function of a PDF is altered, also the preliminary norming denominator needs to be adopted to sum the area under the curve up to 1, again. This is pretty easy here. The preliminary norming denominators λ_1 and λ_2 simply have to be applied with the weight of each half-normal, namely each with weight 0.5.

Finally, the new fully generalized normal distribution denotes:

$$f(x)=\left|\begin{matrix}\frac{2}{\lambda_1+\lambda_2}e^{-0.5\left(\frac{|x-c|}{z}\right)^{ak}} for\, x-c<0\\ \frac{2}{\lambda_1+\lambda_2}e^{-0.5\left(\frac{|x-c|}{z}\right)^{\frac{k}{a}}} for\, x-c>0\end{matrix}\right|$$

Fully Generalized Normal Distribution

Where

$$\lambda_1=\frac{2z\Gamma\left(\frac{1}{ak}\right)}{ak\,0.5^{\frac{1}{ak}}}$$

$$\lambda_2=\frac{2z\Gamma\left(\frac{a}{k}\right)}{\frac{k}{a}0.5^{\frac{a}{k}}}$$

x: Random Variable
c: Construction Location Parameter
z: Deviation Parameter
k: Power and Kurtosis Parameter
a: Asymmetry Parameter
$\Gamma()$: Gamma Function

4.3 Abbreviated FGND Reference

As the full PDF formula of the FGND, especially including the preliminary norming term, is long, the author is introducing an abbreviated notation of the FGND as follows:[41]

$$s(c,z,k,a) \Leftrightarrow f(x) = \left\langle \begin{array}{l} \frac{2}{\lambda_1+\lambda_2} e^{-0.5\left(\frac{|x-c|}{z}\right)^{ak}} \quad for\; x-c<0 \\ \frac{2}{\lambda_1+\lambda_2} e^{-0.5\left(\frac{|x-c|}{z}\right)^{\frac{k}{a}}} \quad for\; x-c>0 \end{array} \right\rangle$$

Where

$$\lambda_1 = \frac{2z\Gamma\left(\frac{1}{ak}\right)}{ak\,0.5^{\frac{1}{ak}}}$$

$$\lambda_2 = \frac{2z\Gamma\left(\frac{a}{k}\right)}{\frac{k}{a}0.5^{\frac{a}{k}}}$$

4.4 Calculating the Moments Table

The “moments from the PDF method “from section “Statistical Moments” has been used to calculate the moments mean, standard deviation, skewness and kurtosis of a standard FGN distribution s(0, 1, k, a). Specifically, the moments have been calculated for the construction parameter space:

- $c=0$
- $z=1$
- $k \in [1.5, 3.0] \wedge k_i = k_{i-1} + 0.01$
- $a \in [0.66, 1.50] \wedge a_i = a_{i-1} + 0.01$

The resulting moments table “table.csv” has been computed using the Python program tablecalc.py, which is also provided as code print in the Appendix. The moments table in total contains 15251 lines of parameter combinations and their corresponding mo-

[41] The letter f is already taken for a general (unspecific) PDF.

ments. The resolution regarding the kurtosis parameter k and the asymmetry parameter a in the moments table is two decimals (0.01), which has been the acknowledged precision standard of statistical distributions in paper tables for decades since around year 2000. The moment table’s columns are c, z, k, a, mean, standard-deviation, skewness and kurtosis.

The beginning and the end of the table look as follows:

```
         c    z    k        a      mean       std      skew      kurt
0      0.0  1.0  1.5  1.000000  0.000000  1.364138  0.000000  3.761954
1      0.0  1.0  1.5  1.010000  0.018212  1.364489  0.045893  3.764744
2      0.0  1.0  1.5  1.020000  0.036261  1.365526  0.091321  3.773000
3      0.0  1.0  1.5  1.030000  0.054168  1.367232  0.136280  3.786559
4      0.0  1.0  1.5  1.040000  0.071950  1.369590  0.180765  3.805255
...    ...  ...  ...       ...       ...       ...       ...       ...
15246  0.0  1.0  3.0  0.684932 -0.157334  0.836123 -0.452104  2.822165
15247  0.0  1.0  3.0  0.680272 -0.160836  0.838773 -0.461483  2.838418
15248  0.0  1.0  3.0  0.675676 -0.164352  0.841470 -0.470863  2.854963
15249  0.0  1.0  3.0  0.671141 -0.167884  0.844214 -0.480245  2.871800
15250  0.0  1.0  3.0  0.666667 -0.171431  0.847004 -0.489629  2.888929

[15251 rows x 8 columns]
```

Table 1: Moments Table Preview

4.5 Identification

4.5.1 Unique Skewness and Kurtosis

Examining the moments table from the previous section has exhibited, that each standard fully generalized normal distribution s(0, 1, k, a) has a corresponding *unique* combination of the moments skewness and kurtosis. The validity of the unique-property will be delivered in section “Generation and Re-Identification Tests” ff. Meanwhile, it will – for the direct following chapters – be assumed, that standard FGNDs s(0, 1, k, a) can be identified by their unique moments skewness and kurtosis.

4.5.2 Z-Standardization

As already mentioned in section “Central Terms and Tools” a PDF with core function f’(x_t) of the form

$$f'(x_t)=e^{-\left(\frac{|x_t-c|}{z}\right)^k} \quad ,$$

by applying the z-standardization

$$x_t \rightarrow x = x_t / z \quad ,$$

can be reduced to the core function of the standard-PDF

$$f'(x)=e^{-\left(\frac{|x-c|}{1}\right)^{k}} \quad .$$

The new FGND, however, is a plugged-together-PDF, out of *two* core functions not one. The 16 thousand dollar question is, if z-standardization can be applied on the new FGND or not. The author has figured out in several simulation experiments, that a FGND s(c, z, k, a) has exactly z-times the standard deviation of a standard FGND s(0, 1, k, a). The validity of the z-transformation assumption for the new FGND will be proven in section “Generation and Re-Identification Tests” ff. Meanwhile, it will – for the direct following chapters – be assumed, that the deviation construction parameter z of a FGND (c, z, k, a) can be calculated from its standard deviation and the standard deviation of the corresponding standard FGND s(0, 1, k, a) as follows:

$$z=\frac{\sigma(s(c,z,k,a))}{\sigma(s(0,1,k,a))}$$

4.5.3 Identification Procedure

Plugging the identification of standard-FGNDs by unique skewness-kurtosis-combination and the z-standardization together, an identification procedure for a niveau-free FGN distribution s(0, z, k, a) is constructed, that in the first step identifies that standard FGND in the moments table, which is most similar in skewness and kurtosis to the test data. “Most similar” is mathematically operationalized as argmin (tol_skewness+ tol_kurtosis). In the second step it applies z-standardization to conclude from the two standard deviations to the estimated construction parameter z_. This logic has been implemented into code in the identification function findparams(). The full code of modul.findparams() is also granted as code print in the Appendix.

4.6 Alternative Quality Random Number Generator

First of all, when bringing up the new distribution, there simply has not been a Python random generator for it. Second, the author has performed moment quality pretest on the popular random numbers provided in Python (and R). These pretests were – especially regarding the elevated quality in the moments skewness and kurtosis required for successful re-identification – already unsuccessful.

The author did remember the simulation experiment, he had made years ago to check [Fahrmeir, Kuenstler, Pigeot (2004)] p. 87 ff direct PDF interpretation by statistical operationalization.[42] This had turned out, that it is possible to create a set of random numbers, that follows the 4 true theoretical statistical moments of the distribution,[43] by drawing independent pseudo-continuous x-values with the simple (non-relative) frequency *weights* provided by the PDF f(x) for x. This procedure has in a lot precision

[42] Compare section “Direct PDF Interpretation”.

[43] *Theoretical* statistical moment equivalent “Moment from the PDF” in section “Statistical Moments”.

improvement iterations been refined to the precise population generator mkpop(). The basic method in mkpop() is, that the decimal frequency *weights* provided by the PDF f(x) are multiplied by a constant, casted to an integer frequency and finally each pseudo-continuous x-value is drawn multiple times the integer frequency for it. This basic procedure is refined by cumulating the residual decimal frequency, that cannot not get transferred to an integer frequency directly, and further one-time drawing a pseudo-continuous x-value, where the residual decimal frequency cumulates up to 0.5.

In the second step, the population generator mkpop() is extended to the alternative quality random number generator qrng() by adding a random sampling of the population. This gets installed by the function sample(). Generally, it is possible to use the parameter popsizefac to create the population popsizefac times big then the sample. However, the author usually takes a full random sample, which statistically is a permutation of the population, because partial samples contain higher errors in true statistical moments, which is contra-productive for re-identification. The Python codes of modul.mkpop(), modul.sample() and modul.qrng() are also granted as code prints in the Appendix.

As there is still an open proof for the norming axiom, a further check is installed inside the function mkpop(). The check provides, that only pseudo-continuous distributions fulfilling the norming axiom can be generated, as follows:

```
dcontr= np.sum(d) *ustep
dtol= abs(1- dcontr)
if dtol < dtolmax:
       ...
```

Summarizing, the function mkpop() and following also the quality random number generator qrng(), are capable to generate population data with outstanding exact moments mean, standard deviation, skewness and kurtosis. Over the thumb, the standard errors in the moments of mkpop() and qrng() are 5 times smaller than the ones of the numpy standard random number generator.[44] This – from the author's perspective – sets a new quality standard for the moment quality of random numbers. Unfortunately, the outstanding high moment quality to a significant portion depends on a large sample size (good quality starting around #10'000) and on optimization of the step-size of the independent pseudo-continuous variable (in code: ustep) to the individual frequency distribution of the PDF.

[44] The author has experienced, that the moment errors produced by the standard numpy random number generator *slightly* varied by machine. One machine had a microcode update versus AEPIC CPU vulnerability, the other machine did not. (Both CPU has been from Intel Sunny Cove architecture.) The Intel micro-code patch for the AEPIC-vulnerability might be relevant this way, that it includes a patch of the APIC interrupt controller, which coordinates multiple Intel Sunny Cove cores. This might cause CPU time – on a nanosecond level – *not be read fully equal distributed* at the initial start of a random number generation. However, the author did not investigate the finding further.

5 Survey of the Fully Generalized Normal Solution

5.1 Meeting the Development Considerations

5.1.1 PDF-Axioms

Fulfill of PDF axiom no. 1 (non-negativity axiom)

$f(x) \geq 0$ for all $x \in \mathbb{R}$

Where

$$f(x) = \left\langle \begin{array}{l} \frac{2}{\lambda_1 + \lambda_2} e^{-0.5(\frac{|x-c|}{z})^{ak}} \quad for\, x - c < 0 \\ \frac{2}{\lambda_1 + \lambda_2} e^{-0.5(\frac{|x-c|}{z})^{\frac{k}{a}}} \quad for\, x - c > 0 \end{array} \right\rangle$$

is given as follows

1.
$2 > 0$

2.
$$\lambda_1 = \int_{-\infty}^{\infty} e^{-0.5(\frac{|x-c|}{z})^{ka}} dx > 0$$

3.
$$\lambda_2 = \int_{-\infty}^{\infty} e^{-0.5(\frac{|x-c|}{z})^{\frac{k}{a}}} dx > 0$$

4.
$e^{-t} > 0$

The fulfill PDF axiom no. 2 (norming axiom)

$$\int_{-\infty}^{\infty} f(x) = 1$$

would be satisfied by proving that the solution of the integral

$$\int_{-\infty}^{\infty} e^{-0.5(\frac{|x-c|}{z})^{k}} = \frac{2z\Gamma(\frac{1}{k})}{k\, 0.5^{\frac{1}{k}}}$$

is true,

which the author has found by numerical integration and simulation experiments.[45] However, the full mathematical proof of the integral may be difficult and complex, thus out of the scope of a data-science work. Instead, the integral has been listed as determined numerically in “Appendix I” and the PDF norming axiom is guaranteed by the total probability check inside the alternative quality random number generator procedure, so that only properly normed distributions can be generated. That has already been as presented in the previous section “Alternative Quality Random Number Generator”. The other procedures in this work do not require the PDF norming axiom property.

5.1.2 Flexibility in Skewness and Kurtosis

Simultaneous flexibility in skewness and kurtosis has been built in by power-and-kurtosis-parameter k and asymmetry parameter a. These can be altered seamless, thus enable seamless degrees of skewness and kurtosis. Separate control has been achieved for kurtosis. Altering the power-and-kurtosis-parameter k only changes the kurtosis and leaves skewness unchanged. However, separate control of skewness has not been reached. When altering the asymmetry parameter a, either skewness and kurtosis get changed.[46]

5.1.3 Identifying the Corresponding Theoretical Distribution

The demonstration of the generation and the re-identification of a theoretical distribution from data is – because of its high importance – sourced out to the following separate section “Generation and Re-Identification Tests”, where they will be tested and validated in detail.

5.1.4 Further Mathematical Properties

The author thinks, that the two mathematical properties steady derivatives and uni-modal distribution are met. Except for effective powers a*k <= 1 (left half-normal), respectively a/k <= 1 (right half-normal), where the first derivation of FGND will get discontinuous at x=c (just like the Laplace distribution), the FGND is considered to have continuous first and second derivatives. The author assumes, that the new FGND is also a uni-modal distribution (with a single maximum extremum core function). However, a profound mathematical discussion of the further mathematical properties is left over to an in depth mathematical survey from other data-scientists. Direct downward compatibility to the Normal distribution, however, has obviously been reached by building in preliminary factor 0.5 of the normal distribution into the FGND’s core function. [47]

[45] An internet search for the integral was without result.
[46] This may be a point for further improvement of the fully generalized normal distribution.
[47] In detail compare section “Adopted Generalized Normal Distribution Version 1”.

5.2 Generation and Identification Tests

5.2.1 Defining A Test-Strategy

Now the central “identifying the corresponding theoretical distribution from data” capability of the new fully generalized normal distribution solution gets explicitly tested.

Each elementary generation and re-identification test experiment is performed in four steps:

1. Generating random number data for a FGND with the construction parameters c, z, k and a using the alternative quality random number generator from section “Alternative Quality Random Generator”.
2. Measuring standard deviation, skewness and kurtosis of the data using the statistical moments from section “Statistical Moments”.
3. Re-Identifying the original construction parameters z, k, a by the findparams() function from section “Identification”.
4. Evaluating, if the true construction parameters have been found, storing the success or non-success of the re-identification test to the variable match (success: 1, non-success: 0) and writing a report line out of the original parameters a, z, k, a, the re-identified parameters a_, z_, k_, a_ and variable match to the file results.csv.

Each elementary generation and re-identification experiment as mentioned above is repeated one thousand times with alternative construction parameters. A FGND consists out of the 4 construction parameters c, z, k and a. The construction parameter space for the test scenario has been defined as follows:

- $c=0$
- $z\in[0.5,2.0]\wedge z_i=z_{i-1}+0.1$
- $k\in[1.5,3.0]\wedge k_i=k_{i-1}+0.01$
- $a\in[0.75,1.33]\wedge a_i=a_{i-1}+0.01$

The full join of each allowed parameter definition resulted in a total parameter space out of 83904 parameter combinations. Out of this total parameter space a random sample with the size of 1000 is drawn to feed each of the 1000 elementary tests with a single set of construction parameters c, z, k, a. Thus, the sample covers a proportion of 1.2% of total parameter space.

The test values of the construction location parameter c and deviation parameter z have been chosen very restrictively. c=0 and z around 1. This has be done on purpose. First of all, to show consistence between the standard FGND moment table and the moments of the data samples generated by the alternative quality random number generator. Second, to keep the total parameter space small, so that the random sample covers a significant proportion of it. Third, the author expects other data-scientists, who apply the new FGND, to z-reduce their data to z=1. Thus, a validated deviation space $z\in[0.5,2.0]$ is thought to be most helpful.

5.2.2 General Test Setup

- Moments table “table.csv” with parameter precision one digit (0.01), calculated by tablecalc.py as presented in section “Calculating the Moments Table”.
- The re-identification is performed by the function findparams(), as presented in section “Identification”.
- Random numbers are generated using the functions mkpop() and sample() as presented in section “Alternative Random Number Generator” with the technical parameters
    - #n= 100’000,
    - ustep= 0.01.
- Intel CPU from Sunny Cove Architecture with *unpatched AEPIC-Vulnerability.*[48]

5.2.3 Test Results

```
        c    z     k         a   c_        z_    k_        a_  matched  trial
0     0.0  0.5  2.00  1.000000  0.0  0.499998  2.00  1.000000      1.0    1.0
1     0.0  1.0  2.00  1.000000  0.0  0.999990  2.00  1.000000      1.0    1.0
2     0.0  2.0  2.00  1.000000  0.0  1.999988  2.00  1.000000      1.0    1.0
3     0.0  0.6  2.28  0.833333  0.0  0.599995  2.28  0.833333      1.0    1.0
4     0.0  1.0  2.32  1.040000  0.0  0.999994  2.32  1.040000      1.0    1.0
...   ...  ...   ...       ...  ...       ...   ...       ...      ...    ...
998   0.0  1.7  2.22  1.090000  0.0  1.699990  2.22  1.090000      1.0    1.0
999   0.0  0.8  2.06  1.290000  0.0  0.799992  2.06  1.290000      1.0    1.0
1000  0.0  1.9  2.26  0.787402  0.0  1.899987  2.26  0.787402      1.0    1.0
1001  0.0  0.7  1.74  0.781250  0.0  0.699989  1.74  0.781250      1.0    1.0
1002  0.0  0.6  2.32  1.090000  0.0  0.599997  2.32  1.090000      1.0    1.0

[1003 rows x 10 columns]
```

Table 2: Generation and Re-Identification Test Results Preview

```
########################################################################
###                        Test Results Summary                      ###
########################################################################

# Total Trials: 1003
# Among these # trials with z==1: 65.0 ; # trials with z!=1: 938.0

# Matched: 1003
```

Proportion Matched: 1.0

Table 3: Generation and Re-Identification Test Results Summary

[48] For a guessed relevance of Intel AEPIC microcode patches for random number generation compare the author’s footnote remarks in section “Alternative Quality Random Number Generator”.

5.3 Validity Evaluation

5.3.1 Hypothesis Test of the Generation and Re-Identification Cycle

For the observed events x of a generation and re-identification cycle

$$x_i = \left\langle \begin{matrix} 0, if\ reidenification\ unsuccessfull \\ 1, if\ reidenification\ successfull \end{matrix} \right\rangle$$

a binomial hypothesis test can constructed with the hypothesis

$H_A : \pi > 0.99$ versus $H_0 : \pi \leq 0.99$

where π is the true successful proportion of an unknown population with size N

$$\pi = \frac{1}{N} \sum_{i=1}^{N} x_i$$

and π' is the observed successful proportion of the tested sample with size n

$$\pi' = \frac{1}{n} \sum_{i=1}^{n} x_i \quad .$$ [49]

The proportion in the hypothesis does not count the exact observed proportion 1.0 for the following reason. In background of binomial test a binomial distribution with the observed proportion π' is built.[50] A binomial distribution with proportion of exactly 1.0 would not be a proper distribution with more than (multiple times) one outcome, any more. Commonly alternative hypothesis H_A (to prove) are of form "<", ">" or "<>".[51]

The binomial test has resulted as follows:

```
#########################################################################
###                     Hypothesis Test Result                       ###
#########################################################################

BinomTestResult(k=1003, n=1003, alternative='greater', proportion_estimate=1.0,
pvalue=4.1889018191313905e-05)
```

P-Value: 4.1889018191313905e-05

Table 4: Hypothesis Test Results of Generation and Re-Identification Cycle

[49] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 404 ff.
[50] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 411 ff.
[51] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 415.

Generally, the p-value denotes the probability that the null-hypothesis is true.[52] Here, the resulted p-value counts lower than the typical statistical confidence niveau $\alpha = 0.05$, thus the null-hypothesis $H_0 : \pi \leq 0.99$ is rejected. Instead, automatically the alternative hypothesis $H_A : \pi > 0.99$ is accepted.

Interpretation:

The reproduction cycle of the generation and re-identification test is accepted to calculate correctly

- at the test data parameter space
  - $c = 0$
  - $z \in [0.5, 2.0]$
  - $k \in [1.5, 3.0]$
  - $a \in [0.75, 1.33]$ and
- at the precision of the moments table, that is second decimal in construction parameters c, z, k, a. (0.01), which has been the acknowledged precision standard of statistical distributions in paper tables for decades since around year 2000.

q.e.d.

5.3.2 Validity of the Procedures Involved Inside the Reproduction Cycle

Before the generation and re-identification tests the following points have been open for proof:

- The validity of the alternative quality random number generator process (AQRNG) from section "Alternative Quality Random Generator", which involves:
  - The fulfill of the PDF norming axiom from section "Reach of Aptness Focus/ Basic Requirements"
  - Respectively, the correctness of the numerically found norming integral for the FGND from section "Adopted Generalized Normal Distribution Version 1".
- The validity of the identification procedure performed by the function findparams(), which consists out of:
  - Equivalence ("=" operation) of FGND-skewness and FGND-kurtosis to standard-FGND-skewness and standard-FGND-kurtosis
  - z-standardization ("/z" operation) regarding standard deviation (and mean[53])

The critical procedures random number generation and identification are harnessed into the generation and re-identification cycle. Every successful generation and re-

[52] [Fahrmeir, Kuenstler, Pigeot (2004)] p. 420.

[53] z-standardization of the mean is not shown here, as subject of a FGND GLM regression, thus out of the scope of the work.

identification test is a round trip from the original construction parameters c, z, k, a to data and from data back to the construction parameters, thus makes up a reproduction cycle. This reproduction cycle is best illustrated by the following graphic:

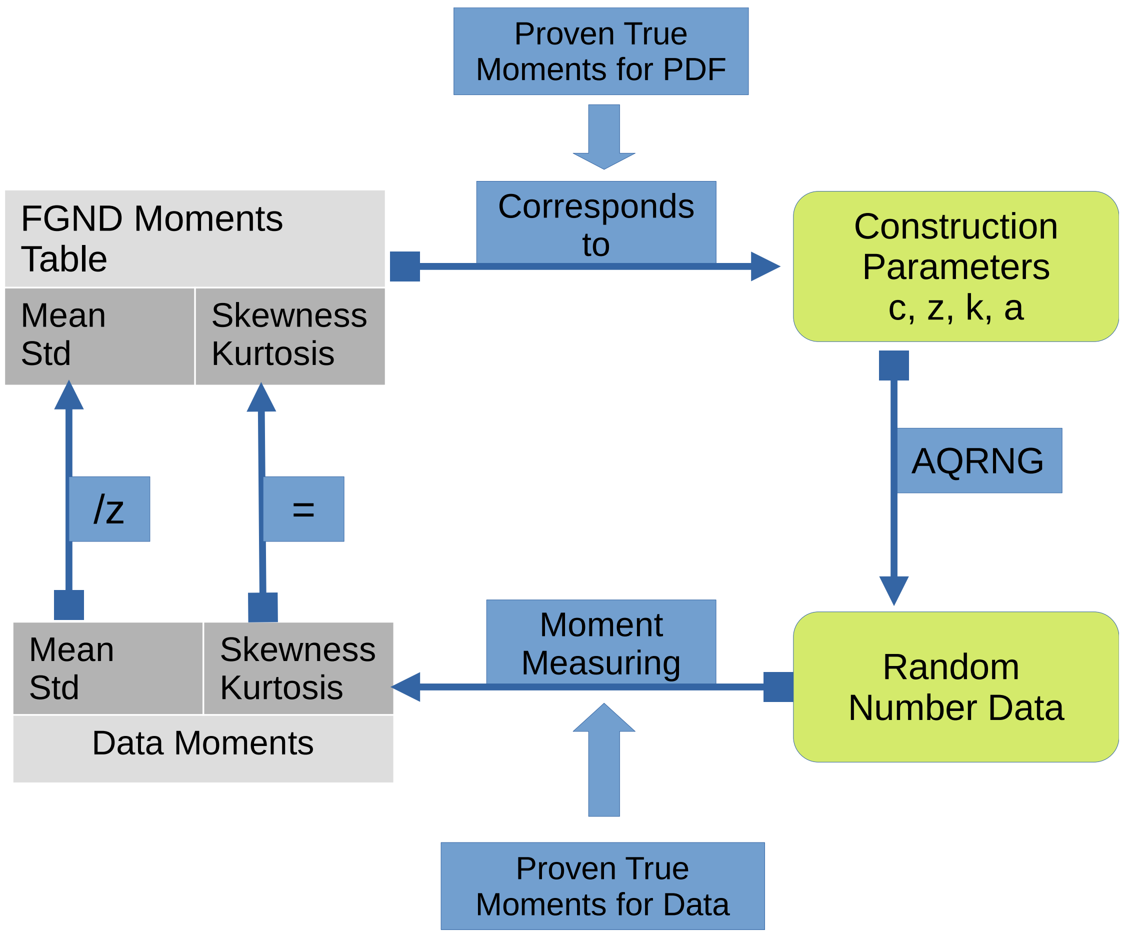

Figure 1: Generation and Re-Identification Reproduction Cycle

The generation and re-identification reproduction cycle has four central specifications:

1. Every generation and re-identification round trip passes through and effectively involves 4 procedures:
   1. Alternative random number generation,
   2. Moment measuring,
   3. Identification and
   4. Moments to construction parameter correspondence.

2. Of which 2 are proven true:
   1. The “moment measuring” follows the generally acknowledged “statistical moments calculation” from section “Statistical Moments”. Which works as a “proven true support”, here.

2. The “moments to construction parameter correspondence” within the moments table has been calculated by the “moments from PDF method” from section “Statistical Moments”. This is also a generally acknowledged method, that is a “proven true support”.

3. One portion of the parameter space of the test data *has deviation parameter z=1*, thus directly overlaps with the parameters space of the moment table, which also is deviation parameter z=1. This portion of test data only requires one half of the identification process, namely the equivalence operation regarding skewness and kurtosis, but not z-standardization.

4. The other portion of the parameter space of the test data is *different from deviation parameter z=1*, thus requires the equivalence operation regarding skewness and kurtosis and also requires z-standardization to get identified.

Summing it up, the procedures remaining critical for proof – random number generation and identification – in the reproduction cycle work as inverse operations of each other. As all the other procedures involved in the reproduction cycle are already proven true by generally acknowledged “proven true supports”, the random number generation and the identification directly cross check each other. That means, *if they are wrong*, while the whole reproduction cycle is true (which has been proven by a formal hypothesis given the parameter space of the test data) and while there are no further degrees of freedom in the reproduction cycle (in form of further procedures), *they have to exactly counteract each other’s error*.

“Fortunately”, the author has created 3 additional fixed test data sets at the beginning of the test data with the parameters of *plain vanilla normal distributions* with the standard deviations 0.5, 1.0 and 2.0, that exclude this possibility, because they are always present (, namely do not get sampled).[54] Commonly acknowledged the normal distribution has power parameter 2 (no asymmetry $\rightarrow$ a=1), skewness 0 and kurtosis 3. If, these 3 test data sets pass the reproduction cycle and get successfully re-identified *on skewness=0 and kurtosis=3* by the identification procedure, also the last two critical procedures must be true. And this is case, here. The additional 3 normal distribution parameter sets at the beginning of the test data have been successfully re-identified.[55]

Interpretation:

Also every procedure within the generation and re-identification reproduction cycle is accepted to calculate correctly

- at the test data parameter space
  - $c=0$
  - $z\in[0.5,2.0]$
  - $k\in[1.5,3.0]$
  - $a\in[0.75,1.33]$ and
- at the precision of the moments table, that is second decimal in construction parameters c, z, k, a. (0.01), which has been the acknowledged precision stan-

54 This is also the reason, why the test data includes 1003 test experiments instead of 1000.

55 In doubt compare the table “Generation and Re-Identification Test Results Preview” or the file “testresults.csv” inside the Python source code package, directly.

dard of statistical distributions in paper tables for decades since around year 2000.

q.e.d.

5.4 Limits of the Solution

The effectively validated parameter space by test data is:

- $c=0$
- $z\in[0.5,2.0]$
- $k\in[1.5,3.0]$
- $a\in[0.75,1.33]$.

The author has simply chosen a straight cube form for the test data parameter space, because this one can be most easiest created. So, the true validatable parameter space does not necessarily really have a cube form and its full extent is considered to be larger than that. Thus, one might want to extend the validated parameter space. However, the author during his FGND work already noticed some limits, over which most possibly cannot get extended. Such limits are subject of this section.

5.4.1 PDF Range

From the author’s point of view, the parameters in the PDF of the fully generalized normal distribution do not have further theoretical limits than k > 0 and a > 0.

However, when z, a or 1/a become very large or k becomes low, then the distribution gets very wide spread and the relative frequencies of x-values get low. This might arise practical computing limits by the machine precision of float values.

Further, when effective powers a*k and k/a get equal 1 or lower, the derivatives of the distribution get discontinuous (like in the Laplace distribution). This might arise mathematical computing limits, as far methods are used, that (internally) apply first and second derivations.

5.4.2 Distribution Identification by Moments

The author thinks, that there are – despite of the practical computing limits already mentioned above – no theoretical limits in moment calculation (from the PDF) and no theoretical limits in identification. However, this is still in research.

5.4.3 Random Number Generator

However, there are effective limits in random random generation. Data samples are only generated precise enough to get correctly re-identified on precision 0.01 at the validated parameter space, if the number of generated observations n is at least

100’000 and the step size of the independent pseudo-continuous variable to draw from is 0.01. To achieve a good precision also the technical step-size of the independent pseudo-continuous variable (in code: ustep) has to be optimized this way, that it fits to the individual deviation characteristic of the distribution (narrow or widespread).[56] However, even if the technical step-size is not optimally configured, the general functionality still remains. Then the achievable re-identification precision will just be just lower, e.g. one decimal precision instead of two decimals.

[56] A sophisticated optimization procedure is not known, yet.

6 Conclusions for Application and Research Prospects

6.1 Application Options

Up to this work, in data-science practice well transparent and computing efficient parametric models were often avoided and got replaced by non-parametric ones simply for the reason to save the expensive manual work of inference testing the model parameters.[57] Now, the author has introduced a new more flexible fully generalized normal distribution, which is generalized for skewness and kurtosis, that may build a broad universal foundation for inferential (hypothesis) testing of most (empirical) practice data and following may enable an automated inference testing workflow for parametric model parameters.

The direct application gains delivered by the work consist in

1. the PDF of the new FGN distribution
2. identifying the new fully generalized normal distribution from data and
3. generating FGN distributed random data

at the effectively validated parameter space

- $c=0$
- $z\in[0.5,2.0]$
- $k\in[1.5,3.0]$
- $a\in[0.75,1.33]$ and

at the precision of the used moments table

- second decimal (0.01) in construction parameters c, z, k, a.

Applying the new FGND within this ranges it is expected, that there will not appear statistical problems with the new fully generalized normal distribution and its accompanying procedures.

Regarding identifying the new FGND (from already existing data) it should from the author's point of view, however without effective validation so far, be possible to extend the explicitly validated parameter space for the deviation parameter z as follows:

1. Calculate the moments of the data to process.
2. Execute the findparams() procedure to find an approximate z_ estimator. (If necessary, calculate a more detailed moments table running tablecalc.py.)
3. Perform z-transformation (using z_) to transform the data to $z\simeq 1$.

This extension, from the author's free estimation, will already cover roughly 98 % of a data-scientist's common data, thus will make about 98% of common data available for automated inference testing, which especially will include automated hypothesis testing of parametric model parameters.[58]

[57] Reflecting the author's personal experience.

[58] Universal, fully automated hypothesis testing on the foundation of the new FGND may be performed as follows: 1. Deriving the calculated parameter's true distribution by bootstrapping. 2. Modeling its distribution with the new FGND. 3. Inference testing it on the new FGND.

Regarding generating FGN distributed random data, however, extending the effectively validated parameter space is much more difficult. This will from author's experience either require a significant (about factor 10) increase of the sample size (from currently 100'000) and/ or an optimization of the technical step-size of the independent pseudo-continuous variable to the distribution's individual deviation characteristics. It is only recommended for research purposes.

Summing it up, the current operational decision in favor or against the practice application of the new fully generalized normal distribution in its present development state will mainly depend on the data-scientist's individual need to treat skewness and excess-kurtosis in his/ her data automated, respectively the demand for this kind of simulation experiments. An interested data-scientist should check first, which of the desired statistical methods would require an adoption for skewness-and-kurtosis-capability. Where a FGND adoption is required, it has – current development state – be performed from scratch. The necessary adoption may in some cases be easy, in other cases complicated and in a third case class – given the current early development status – the proper adoption may still be unknown.

Thus, some data-scientists will already start to build automated applications for skewed and kurtosed data, that have not been possible before. Others might consider the application of the new FGND – in its present development status – as too basic functionality, too experimental or simply treat it as a qualified order of an analytical solution for it (respectively as a qualified order of another similar, analytically solved FGND).

6.2 Research Prospects

Despite from a direct practice application the author is expecting, that this work will cause a wake-up in data-science research regarding the proper treatment of skewness and excess-kurtosis, that will massively intensify the research in fully generalized (normal) distributions. This will especially include the revalidation of the current standard methods by stress-testing them with skewed FGND data. And also include the research of the moment qualities of popular random number generators, regarding their standard errors in mean, standard-deviation, skewness and kurtosis, in comparison to the author's alternative quality random number generator.

Further, other data-scientists, statisticians and probability-mathematicians will both validate and also improve the presented first fully generalized normal distribution. And also will adopt higher level statistical procedures from the normal distribution to this (or another improved) fully generalized normal distribution, additional to the low level statistical foundation presented here.

So, the author already sees on the far horizon, that this new fully generalized normal distribution will in long term work as "the first brick of a quantum mechanics revolution in data-science" and lead to a fully generalized data-science universe in the future, with e.g. new significance tests on the foundation of this (or another improved) FGND, revalidation/ enhancement of correlation and glm regression for skewed and kurtosed data, analytical (portfolio) diversification of skewed and kurtosed variables and so on.

If everything works as planned, a follow-up FGND discussion may later be found on www.quantsareus.net .

7 Appendices

Appendix I: Numerically Determined Integrals

It is a common practice to state numerically determined integrals separately in the appendix. Primarily, to disclose the remaining uncertainty of proof in it. Secondary, to deliver mathematicians numerically determined integrals for mathematical survey and analytical solution.

In the work these integrals have been determined numerically as follows :

$$\int_{-\infty}^{\infty} e^{-0.5\left(\frac{|x-c|}{z}\right)^{k}} = \frac{2\,z\,\Gamma\left(\frac{1}{k}\right)}{k\,0.5^{\frac{1}{k}}}$$

Appendix II: Source Code

Appendix II.I: tablecalc.py:

```
#! /usr/bin/env python3


"""
Description:
The "tabelcalc.py" python script computes a new moments table "table.csv",
which is used by the findparams() identification procedure. Thus, computing a
new larger moments table can enhance the identification parameter space of find-
params().

The new "table.csv" will be written at the current work directory. In case make a
backup of "table.csv", first, before running tablecalc.py.
"""


def version_changes():
        """
        Version changes from V0.17 to V0.20:
        - Code clean up, writing comments
        """
        pass


### Setup:

import numpy as np
# import numpy.random
# from numpy.linalg import inv

import scipy.integrate as sp_it
# from scipy.special import gamma

from sympy.functions.special.gamma_functions import gamma

import pandas as pd

# import matplotlib as mpl

import time


### Functions:

def fgnd(x):
        """
        Author's Fully Generalized Normal Distribution appropriate for one-dimen-
sional numerical integration
        """
        norm1= (2 *z *gamma(1/(k*a))) /(k*a *0.5**(1/(k*a)))
        norm2= (2 *z *gamma(1/(k/a))) /(k/a *0.5**(1/(k/a)))
        norm= (norm1 + norm2) /2
        # print("norm:", norm)
        if x-c <0:
                return np.exp(-0.5 *(abs(x-c)/z)**(k*a) ) /norm
        else:
                return np.exp(-0.5 *(abs(x-c)/z)**(k/a) ) /norm

def mean_fct(x):
        return fgnd(x)* x

def variance_fct(x):
        return fgnd(x)* (x-mean)**2
```

```
def skewness_fct(x):
        return fgnd(x)* ((x-mean)/std)**3

def kurtosis_fct(x):
        return fgnd(x)* ((x-mean)/std)**4


### Creating FGND parameter matrix to get tabled:

c= np.array([0], dtype=np.float128)

z= np.array([1], dtype=np.float128)

k= pd.Series(np.arange(1.5, 3.01, 0.01, dtype=np.float128))

a= pd.Series(np.arange(1.0, 1.51, 0.01, dtype=np.float128))
# adding 1/a values for left skewed:
a= pd.Series(a).append(pd.Series((1/a)[1:]))


c= pd.DataFrame({"key": np.repeat(0, c.shape[0]), "c": pd.Series(c)})
z= pd.DataFrame({"key": np.repeat(0, z.shape[0]), "z": pd.Series(z)})
k= pd.DataFrame(({"key": np.repeat(0, k.shape[0]), "k": pd.Series(k)}))
a= pd.DataFrame(({"key": np.repeat(0, a.shape[0]), "a": pd.Series(a)}))
params= pd.merge(c, z)
params= pd.merge(params, k)
params= pd.merge(params, a)
params= np.array(params, dtype=np.float128)
# Deleting key:
params= params[:,1:]


print("")
print("Machine precision of params table containing c,z,k,a,:")
print(params.dtype)
print("")
print("parameter preview:")
print(params)
print("")


### Calculating moments table:

table= np.zeros((params.shape[0], 8), dtype= np.float128)

# for i in range (0, 1, 1):
for i in range (0, params.shape[0], 1):
        print("iteration:", i, "/", params.shape[0], "  time:", time.asctime(time.lo-
caltime()) )

        c= params[i, 0]
        z= params[i, 1]
        k= params[i, 2]
        a= params[i, 3]

        mean= sp_it.quad(mean_fct, -np.inf, np.inf)[0]
        variance= sp_it.quad(variance_fct, -np.inf, np.inf)[0]
        std= np.sqrt(variance)
        skewness= sp_it.quad(skewness_fct, -np.inf, np.inf)[0]
        kurt= sp_it.quad(kurtosis_fct, -np.inf, np.inf)[0]

        table[i, 0:4]= params[i, 0:4]
        table[i, 4:8]= np.array([mean, std, skewness, kurt])


tabledf= pd.DataFrame(table, columns=
["c","z","k","a","mean","std","skewness","kurt"])
```

```
if tabledf.to_csv("table.csv", index= False, header= True):
        print("")
        print("writing table.csv has failed")
else:
        print("")
        print("the moments table (preview):")
        print(tabledf)
        print("")
        print("has been written succesfully")
```

Appendix II.II: modul.py:

```
#####! /usr/bin/env python3

"""
Description:
The fgnd module provides the first fully generalized normal distribution (FGND),
generalized for skewness and kurtosis. [To be precise also provides a normal dis-
tribution (nd) and an ADOPTED generalized normal version 1 distribution (gnd-
v1a)]. The naked distributions are accompanied by foundation identification and
random number generation methods. For a more detailed introduction of the new
FGND refer to http://www.quantsareus.net/hs/publ/introducing_a_fully_general-
ized_normal_distribution.pdf
"""

def version_changes():
	"""
	Version changes from V0.17 to V0.20
	- Renamed skewness() to skewness()
	- Code clean up, writing comments
	- findinstalledpath()
	- setupcheck()
	"""
	pass


### General Setup:

import numpy as np
# import numpy.random as np_rd
# from numpy.linalg import inv

# import scipy.integrate as sp_it
# from scipy.special import gamma

from sympy.functions.special.gamma_functions import gamma

import pandas as pd

# import matplotlib as mpl

import sys
import os


### Distributions:

def nd(u, c, z):
	"""
	Gauss' Normal Distribution
	"""
	norm= np.float128( z* (2* np.pi)**0.5 )
	d= 1 /norm *np.float128( np.exp(-0.5 *((u-c)/z)**2) )
	return d

def gndv1a(u, c, z, k):
	"""
	Nadarajah's Generalized Normal Distribution Version 1
	ADOPTED to direct downward compatibility to Gauss' Normal Distribution
by the author
	"""
	norm= np.float128( (2 *z *gamma(1/k)) /(k *0.5**(1/k)) )
	d= 1 /norm *np.float128( np.exp(-0.5 *(abs(u-c)/z)**k) )
	return d
```

```
def fgnd(u, c, z, k, a):
	"""
	Author's Fully Generalized Normal Distribution
	"""
	norm1= np.float128( (2 *z *gamma(1/(k*a))) /(k*a *0.5**(1/(k*a))) )
	norm2= np.float128( (2 *z *gamma(1/(k/a))) /(k/a *0.5**(1/(k/a))) )
	norm= np.float128( np.array((norm1 + norm2) /2) )
	d= np.float128( np.exp(-0.5 *(abs(u-c)/z)**(k*a) ) )
	d2= np.float128( np.exp(-0.5 *(abs(u-c)/z)**(k/a) ) )
	d[u-c > 0]= d2[u-c > 0]
	d= 1/ norm *d
	return d


### Methods:

def skewness(x):
	"""
	Statistical skewness without sample size correction
	"""
	n= x.shape[0]
	# return n /((n-1)*(n-2)) *np.sum( ((x- np.mean(x))/np.std(x))**3)
	return 1/n* np.sum( ((x- np.mean(x))/np.std(x))**3)

def kurtosis(x):
	"""
	Statistical kurtosis without sample size correction
	"""
	n= x.shape[0]
	# return n*(n+1) /((n-1)*(n-2)*(n-3)) *np.sum( ((x
-np.mean(x))/np.std(x))**4)
	return 1/n* np.sum( ((x -np.mean(x))/np.std(x))**4)

def mkpop(u, ustep, n, distr, c, z, k, a, dtolmax):
	"""
	Method to create a population
	from an independent variable u with stepsize (resolution) ustep
	that follows a distribution "distr" with parameters c, z, k, a,
	at maximum allowed density tolerance dtolmax (from 1)
	"""
	l= u.shape[0]
	if distr== "nd":
		d= np.float128( nd(u, c, z) )
	if distr== "gndv1a":
		d= np.float128( gndv1a(u, c, z, k) )
	if distr== "fgnd":
		d= np.float128( fgnd(u, c, z, k, a) )

	# PDF norming axiom test:
	dcontr= np.sum(d) *ustep
	dtol= abs(1- dcontr)
	if dtol < dtolmax:
		dcum= np.float128(np.cumsum(d) *ustep *n)
		dresid= np.float128(0.0)
		dint= np.int64(np.zeros(l))
		for i in range(1, l):
			if round(dcum[i] -dcum[i-1] +dresid) >= 0.5:
				dint[i]= round(dcum[i] -dcum[i-1] +dresid)
				dresid= (dcum[i] -dcum[i-1] +dresid) - round(dcum[i]
-dcum[i-1] +dresid)
			else:
				dresid= dresid+ dcum[i] -dcum[i-1]
			# print(d[i], dcum[i], dint[i], dresid)
		pop= u.repeat(dint)
		return pop
```

```
        else:
                print("Error: mkpop cannot create the population on independent u
within allowed dtolmax")
                print("pdf test for norming axiom negative")
                print("density sum:", dcontr)
                print("current density tolerance:", dtol, " > ", "allowed density toler-
ance:", dtolmax)
                print("")

def sample(pop, n):
        """
        Method to create a sample
        with size n
        from population pop
        """
        samp= pop.take(np.random.permutation(len(pop))[:n])
        return samp

def qrng(n, popsizefac, distr, c, z, k, a, dtolmax):
        """
        Method to create a population "popsizefac" times large in the first step
        from an independent variable u with stepsize (resolution) ustep
        that follows a distribution "distr" with parameters c, z, k, a,
        at maximum allowed density toleracance (from 1) dtolmax
        in the second step
        to create a sample
        with size n
        """
        popul= mkpop(u, ustep, n *popsizefac, distr, c, z, k, a, dtolmax)
        e= sample(popul, n)
        return e

def findparams(mean, std, skewness, kurt):
        """
        Method to find the construction parameters z, k, a of a FGND distribution
from the moments std, skewnes and kurtosis
        """
        installedpath= findinstalledpath()
        if installedpath!= "":
                tabledf= pd.read_csv(installedpath +"table.csv")
        else:
                if os.path.exists("fgnd/table.csv"):
                        tabledf= pd.read_csv("fgnd/table.csv")
                else:
                        if os.path.exists("table.csv"):
                                tabledf= pd.read_csv("table.csv")
                        else:
                                print ("\nERROR: The moments table table.csv has
not been found")
                                print ("Most possibly is FGND neither installed (glob-
ally) nor is a local FGND executed from its local folder")
                                print ("The current work directory is:", os.getcwd(), "\
n")

        table= np.array(tabledf)
        tol= (abs(table[:, 6] -skewness)) +(abs(table[:, 7] -kurt))
        params= table[tol==min(tol), :]
        # Calculating true z using z-reduction std(z=1)= std(z) / z
        z= std/ params[:, 5]
        # Setting z, the first and second moment to the proper values correpond-
ing to z!=1 using z-transform
        params[:, 1]= z
        params[:, 4]= params[:, 4] *z
        params[:, 5]= params[:, 5] *z
        return params[:]
```

```
def findinstalledpath():
	# List of runtime pathes for Python commands
	syspath= sys.path
	syspathlen= len(syspath)

	# Searching own installed path
	installedpath=""
	for i in range(0, syspathlen, 1):
		installedpath= syspath[i] +"/fgnd/"
		if os.path.exists(installedpath):
			break
		else:
			installedpath= ""
	return installedpath

def setupcheck():
	"""
	Check of proper fgnd setup
	by constructing a normal distribution with std 1
	and trying to re-identifying it on precision level 0.01
	"""
	params=np.array([0, 1, 2, 1])
	c= params[0]
	z= params[1]
	k= params[2]
	a= params[3]

	# Creating testdata from original c,z,k,a parameters:
	testdata= qrng(100000, 1, "fgnd", 0, z, k, a, 1e-4)

	mean= np.mean(testdata)
	std= np.std(testdata)
	skewness= skewness(testdata)
	kurt= kurtosis(testdata)

	# Finding parameter estimates c_, z_, k_, a_
	params_= findparams(mean, std, skewness, kurt)

	c_= 0
	z_= params_[0, 1]
	k_= params_[0, 2]
	a_= params_[0, 3]

	diffsum= abs(c -c_) +abs(z -z_) +abs(k -k_) +abs(z -z_)
	if diffsum < 1e-2:
		print("FGND Setup Check SUCCESSFULL")
		print("")
		print("Parameter Space Limits")
		print("- in Re-Identifiaction by findparams() are: f(x) in s(c=0, z=z,
1.5<=k<=3.0, 0.66<=a<=1.50) @ precision 0.01")
		print("- in Random Number Generation by qrng() are: f(x) in s(c=0,
0.5<=z<=2, 1.5<=k<=3.0, 0.75<=a<=1.33) @ precision 0.01")
		print("out-of-the-box.\n\n\n")
	else:
		print("FGND Setup Check FAILED")

def interpretercheck():
	print("Python Interpreter Check: Hello Worlds \n")

def tableprintf():
	"""
	Printing a preview of the moments table to file
	"""
	tabledf= pd.read_csv("table.csv")
	fobj= open("tablepreview.txt", "w")
	print(tabledf, file=fobj)
	fobj.close
```

```
##################################################
#####################################
### Active Technical Parameters for Random Number Generation

# ustep= 0.1
# glm setting (n=10'000):
# ustep= 0.05
# ustep= 0.025
# ustep= 0.01
# genreidtest setting (n=100'000, 0.5<z<=2.0, 1.5<k<=3.0, 0.75<a<=1.33):
ustep= 0.005
# ustep= 0.0025
# ustep= 0.001

# u= np.arange(-10, 10, ustep)
# u= np.arange(-25, 25, ustep)
# genreidtest setting (n=100'000, 0.5<z<=2.0, 1.5<k<=3.0, 0.75<a<=1.33):
u= np.arange(-50, 50, ustep)
# u= np.arange(-100, 100, ustep)
# u= np.arange(-250, 250, ustep)
# u= np.arange(-500, 500, ustep)
# u= np.arange(-1000, 1000, ustep)


##################################################
#####################################
### Check Procedures on Startup

interpretercheck()

print("Installation Check: The FGND package is installed to ", "'", findinstalled-
path(),"'\n" )

setupcheck()


##################################################
#####################################
# INACTIVE further parameters FOR TESTING PURPOSES ONLY (Must be fully
commented out for application!)

# number of testdata observations
# n= 100
# n= 1000
# glm setting:
# n= 10000
# generation and re-identification test setting:
# n= 100000
# n= 1000000
# n= 10000000

# construction parameters:
# c= 0

# z=0.5
# z=0.75
# z= 1
# z= 1.5
# z= 2
# z= 3
# z= 4
# z= 5

# k= 1.5
# k= 2
# k=2.5
# k= 3

# a= 1
# a= 1.1
# a= 1.2
# a= 1.3
# a= 1/a


#################################################
####################################
# INACTIVE Test Calls FOR TESTING PURPOSES ONLY (Must be fully com-
mented out for application!)

# testdata= mkpop(u, ustep, n, "nd", 0, z, 0, 0, 1e-4)
# testdata= mkpop(u, ustep, n, "gndv1a", 0, z, k, 0, 1e-4)
# testdata= mkpop(u, ustep, n, "fgnd", 0, z, k, a, 1e-4)

# testdata= qrng(n, 2, "nd", 0, z, 0, 0, 1e-4)
# testdata= qrng(n, 2, "gndv1a", 0, z, k, 0, 1e-4)
# testdata= qrng(n, 2, "fgnd", 0, z, k, a, 1e-4)

# print("mean:", np.mean(testdata))
# print("estd:", np.std(testdata))
# print("skewness:", skewness(testdata))
# print("kurtosis:", kurtosis(testdata))

# print(findparams(0.0, 1.0, 0.5, 3.3).shape)

# tableprintf()
```

Appendix II.III: genreidtest.py:

```
#! /usr/bin/env python3


"""
Description:
The generation and re-identification test is a python script, that verifies the valid-
ity of the new fully generalized normal distribution (FGND) for the FGND con-
struction parameters within the test parameter matrix created further following.

The result files are written at the current work directory.
"""

def version_changes():
	"""
	Version changes from V0.17 to V0.20:
	- Code clean up, writing comments
	"""
	pass


### Setup:

# Proper import for installed FGND package:
import fgnd.modul as modul
# Proper import for running from local FGND folder:
# import modul

import numpy as np
# print("numpy Version:", np.__version__)
# import numpy.random
# from numpy.linalg import inv

# import scipy.integrate as sp_it
# from scipy.special import gamma
import scipy.stats as sp_stats
# print("scipy Version:", sp.__version__)

from sympy.functions.special.gamma_functions import gamma
# import sympy as sy
# print("sympy Version:", sy.__version__)

import pandas as pd
# print("pandas Version:", pd.__version__)

# import matplotlib as mpl

import time


### Creating test parameter matrix to perform tests from:

# number of testdata observations
# n= 100
# n= 1000
# glm setting:
# n= 10000
# genreidtest setting:
n= 100000
# n= 1000000
# n= 10000000

c= np.array([0], dtype=np.float128)

z= np.arange(0.5, 2.1, 0.1, dtype=np.float128)
```

```
k= pd.Series(np.arange(1.50, 3.02, 0.02, dtype=np.float128))

# Out of bounds:
# a= pd.Series(np.arange(1.0, 1.38, 0.01, dtype=np.float128))
a= pd.Series(np.arange(1.0, 1.34, 0.01, dtype=np.float128))
# Adding 1/a values for left skewed:
a= pd.Series(a).append(pd.Series((1/a)[1:]))


c= pd.DataFrame({"key": np.repeat(0, c.shape[0]), "c": pd.Series(c)})
z= pd.DataFrame({"key": np.repeat(0, z.shape[0]), "z": pd.Series(z)})
k= pd.DataFrame(({"key": np.repeat(0, k.shape[0]), "k": pd.Series(k)}))
a= pd.DataFrame(({"key": np.repeat(0, a.shape[0]), "a": pd.Series(a)}))
params= pd.merge(c, z)
params= pd.merge(params, k)
params= pd.merge(params, a)
params= np.array(params, dtype=np.float128)
# Deleting key:
params= params[:,1:]


print("Total number of parameter combinations:", params.shape[0])
print("")


# Generating 3 normal distribution parameter sets
paramsnd= np.array([[0, 0.5, 2, 1],[0, 1, 2, 1], [0, 2, 2, 1]], dtype=np.float128)

# Drawing random samples from params
paramsindexold= np.arange(0, params.shape[0], 1)
# paramsindexnew= modul.sample(paramsindexold, 100)
paramsindexnew= modul.sample(paramsindexold, 1000)
paramssamp= params[paramsindexnew, :]

paramssamp= pd.DataFrame(paramsnd).append(pd.DataFrame(paramssamp))
paramssamp= np.array(paramssamp)


# print(params)
# print(paramsnd)
print("# Test Samples:", paramssamp.shape[0])
print("")
print("Machine precision of sample parameters c,z,k,a,:", paramssamp.dtype)
print("")


### Performing Generating and Re-Identification Tests:

results= np.zeros((paramssamp.shape[0], 10), dtype= np.float128)
# matched:
results[:, 8]= -1.0
# trial:
results[:, 9]= 1.0

# for i in range (0, 1, 1):
for i in range (0, paramssamp.shape[0], 1):
        print("iteration:", i, "/", paramssamp.shape[0], "  time:",
time.asctime(time.localtime()) )

        c= paramssamp[i, 0]
        z= paramssamp[i, 1]
        k= paramssamp[i, 2]
        a= paramssamp[i, 3]

        # Creating testdata from original c,z,k,a parameters:
        testdata= modul.qrng(n, 1, "fgnd", 0, z, k, a, 1e-4)
        mean= np.mean(testdata)
        std= np.std(testdata)
        skewness= modul.skewness(testdata)
        kurt= modul.kurtosis(testdata)

        # Finding parameter estimates c_, z_, k_, a_
        paramssamp_= modul.findparams(mean, std, skewness, kurt)

        c_= 0
        z_= paramssamp_[0, 1]
        k_= paramssamp_[0, 2]
        a_= paramssamp_[0, 3]

        diffsum= abs(c -c_) +abs(z -z_) +abs(k -k_) +abs(z -z_)
        if diffsum < 1e-2:
                matched= 1.0
        else:
                matched= 0.0

        results[i, 0:4]= paramssamp[i, 0:4]
        results[i, 4:9]= np.array([c_, z_, k_, a_, matched])


### Writing Test Results:

testresultsdf= pd.DataFrame(results, columns= ["c","z","k","a","c_","z_","k_","a_",
"matched", "trial"])

if testresultsdf.to_csv("./testresults.csv", index= False, header= True):
        print("")
        print("writing testresults.csv has failed")
        print("")
else:
        print("")
        print("testresults.csv has been written succesfully")
        print("")
        fobj= open("testrespreview.txt", "w")
        print(testresultsdf, file=fobj)
        fobj.close


### Test Results Summary:

testresults= np.array(testresultsdf)
matchedsum= np.sum(np.int64(testresults[ :, 8]))
trialsum= np.sum(np.int64(testresults[ :, 9]))
matchedprop= matchedsum/ trialsum


print("")
print("")
print("##############################################
########################################")
print("###                Test Results Summary                          ###")
print("##############################################
########################################")
print("")
print("# Total Trials:" , trialsum )
print("# Among these # trials with z==1:", np.sum(testresults[np.round(testre-
sults[ :, 1], 1) == 1.0, 9 ]), "; # trials with z!=1:",
np.sum(testresults[np.round(testresults[ :, 1], 1) != 1.0, 9 ]) )
print("")
print("# Matched:", matchedsum)
print("")
print("Proportion Matched:", matchedprop)
print("")
print("")
print("############################################
########################################")
print("###                    Hypothesis Test Result                    ###")
print("############################################
########################################")
print("")
print(sp_stats.binomtest(matchedsum, trialsum, (matchedprop -0.01), "greater") )
print("")
print("P-Value:", sp_stats.binom_test(matchedsum, trialsum, (matchedprop -0.01),
"greater") )
print("")
# print("Tests failed:")
# print( testresults[np.round(testresults[ :, 8], 1)== 0.0, : ])


### File Versions of above:

fobj= open("testressum.txt", "w")
print("", file=fobj)
print("############################################
########################################", file=fobj)
print("###                    Test Results Summary                    ###",
file=fobj)
print("############################################
########################################", file=fobj)
print("", file=fobj)
print("# Total Trials:" , trialsum, file=fobj )
print("# Among these # trials with z==1:", np.sum(testresults[np.round(testre-
sults[ :, 1], 1) == 1.0, 9 ]), "; # trials with z!=1:",
np.sum(testresults[np.round(testresults[ :, 1], 1) != 1.0, 9 ]), file=fobj )
print("", file=fobj)
print("# Matched:", matchedsum, file=fobj)
print("", file=fobj)
print("Proportion Matched:", matchedprop, file=fobj)
print("", file=fobj)
fobj.close

fobj= open("hyptestres.txt", "w")
print("", file=fobj)
print("############################################
########################################", file=fobj)
print("###                    Hypothesis Test Result                    ###",
file=fobj)
print("############################################
########################################", file=fobj)
print("", file=fobj)
print(sp_stats.binomtest(matchedsum, trialsum, (matchedprop -0.01), "greater"),
file=fobj )
print("", file=fobj)
print("P-Value:", sp_stats.binom_test(matchedsum, trialsum, (matchedprop -0.01),
"greater"), file=fobj )
print("", file=fobj)
fobj.close
```

Appendix III: Licensing

Non-Commercial Use GPLv3 License
For non-commercial use all the source codes and algorithms are licensed under Gnu-Public-License Version 3 (GPLv3). Which more or less means: Free to use and free to redistribute for non-commercial purpose. Free to recycle the code, as far the new code is granted with a GPLv3 license (or higher), again.

Commercial Use Licenses
Requests for commercial use licenses are welcome. One exclusive or multiple non-exclusive licenses will be sold as wholesale packages for further reselling. They will be sold on the basis, that the non-commercial GPLv3 license granted above is remaining.

Commercial vs. Non-Commercial Usage
Every application of the source code/ the algorithms, receiving a direct or indirect monetary compensation for, is commercial. A monetary payment is received indirectly, when bundling this license with other licenses to a larger package or flatrate. The classification of commercial and non-commercial does <u>not</u> depend on explicit code/ binary usage, software-as-a-service usage or electronic silicon circuiting. All commercial software-as-a-service, cloud computing and electronic circuiting usage , for which a monetary compensation is received, requires a commercial license.

To avoid highly fragmented licensing piecework the providers of a <u>larger commercial statistical solution</u> of any kind (software, software-as-a-service, electronic circuit), in which the author’s fully generalized normal distribution only plays a <u>marginal role</u>, to be specific less than 1% of the typical use, are provisionally granted a fee-less commercial license. Provisionally means, till the license use by the larger commercial statistical solution is called for detailed investigation. Up to the point of time of such license investigation call, usually no license fees are billed (backward in time from the call). License fees are only billed forward in time from the investigation call, except the usage before the investigation call was <u>obviously</u> over the 1% hurdle rate. The provisionally fee-less users/ appliers can be obligated to protocol the type of usage by the license investigation call.

The split into commercial and non-commercial usage is a non-discriminating differentiation regarding the GPLv3 license conditions. Every person and group has the free choice to use the code and the algorithms either non-commercially or commercially. For legal evaluation especially, the original creator of the software/ the algorithms performs the split into non-commercial and commercial usage before any user/ applier has received any license rights in the software/ the algorithms at all.

Thesis Paper
The thesis paper itself is licensed under the typical academic terms. It can be used for free, as far the source of the information is cited properly.

Appendix IV: More Broad University Coverage of Data-Science, Please!

From the author’s point of view the role of a hands-on data-science **programmer** is not only often missing in business practice but often is also missing in university data-science research, itself. A lot of theoretical data-science problems can get insighted and be researched by **programmed** simulation experiments. And a little portion of them can also get solved by numerical programming, when cleverly integrating statistics and informatics methods. *Que era demonstrando en esto trabajo cientifico.*

In my honest opinion university faculties covering data-science thus should offer a broader range of data-science studies with alternative emphasizes of mathematical and hands on numerical **programming** skills, also including PhD studies. They may more orientate on F. Brooks famous words:

“**The programmer**, like a poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and to rework, so readily capable of realizing grand conceptional structures.”

Bibliography

Eliason (1993): Eliason, S. R., Maximum Likelihood Estimation - Logic and Practice, 1993

Fahrmeir, Kuenstler, Pigeot (2004): Fahrmeir L., Kuenstler R., Pigeot, I., Statistik - Der Weg zur Datenanalyse, 2004

Frees (2004): Frees E.W., Longitudinal and Panel Data, 2004

Holzner S. ((2013): Holzner S., Quantum Physics For Dummies, 2013

Hosking, Wallis (1997): Hosking J. R. M., Wallis J. R. , Regional Frequency Analysis - An approach based on L- Moments, 1997

Kaas, Goovaerts, Dhaene (2008): Kaas R., Goovaerts M., Dhaene J., Modern Actuarial Risk Theory, 2008

Knuth (1997): Knuth D.E., Seminumerical Algorithms, 1997

Kolmogorov (1933): Kolmogorov A. N., Grundbegriffe der Wahrscheinlichkeitsrechnung, 1933

Nadarajah (2005): Nadarajah S., A Generalized Normal Distribution , 2005

Spiegel (1994): Spiegel, M.R., Mathematical Handbook Of Formulas And Tables, 1994

Stirzaker (2007): Stirzaker D., Elementary Probability, 2007

Tabachnick, Fidell (2000): Tabachnick B. G., Fidell L.S., Using Multivariate Statistics, 2000

Ushakov (2001): Ushakov N.G., Density of a probablity distribution, 2001

Varanasi, Aazhang (1989): Varanasi M. K., Aazhang B., Parametric Generalized Gaussian Density Estimation, 1989

Wikipedia, Pseudo-random number sampling (2022): Div., Pseudo-random number sampling, 2022, en.wikipedia.org/wiki/Pseudo-random_number_sampling

Wikipedia, Wahrscheinlichkeitsdichtefunktion (2022): Div., Wahrscheinlichkeitsdichtefunktion, 2022, de.wikipedia.org/wiki/Wahrscheinlichkeitsdichtefunktion