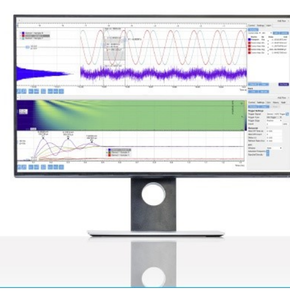
Extending Runjags: A tutorial on adding Fisher's *z* distribution to Runjags

Cite as: AIP Conference Proceedings **2329**, 060005 (2021); https://doi.org/10.1063/5.0042143
Published Online: 26 February 2021

Arifatus Solikhah, Heri Kuswanto, Nur Iriawan, Kartika Fithriasari, and Achmad Syahrul Choir



Extending Runjags: A Tutorial on Adding Fisher's *z* Distribution to Runjags

Arifatus Solikhah[1, 2], Heri Kuswanto[1, a*)], Nur Iriawan[1], Kartika Fithriasari[1], and Achmad Syahrul Choir[2]

[1]*Department of Statistics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember, Indonesia.*
[2]*BPS - Statistics Indonesia, Indonesia.*

[a)] Corresponding author: heri_k@statistika.its.ac.id

Abstract. JAGS is an open-source package to analyze graphical model that is written with extensibility in mind. The runjags package includes many enhancements to JAGS, including a custom JAGS module that contains some additional distributions in the Pareto family. A very flexible set of statistical models based on the logarithm of an F variate, the standardized Fisher's z distribution, was introduced more than 90 years ago. However, the standardized Fisher's z distribution is not yet adaptive for modeling, since the mode cannot be shifted from the zero point. This paper introduces the Fisher's z distribution, i.e., the standardized Fisher's z distribution which added a location parameter μ and a scale parameter σ. The mode of the distribution lies in μ. In this paper, we provide step-by-step instructions on how to add Fisher's z distribution to the runjags package. In order to affirm the accuracy of our implementation, we ran a comprehensive numerical experiment, using linear regression model. We conduct a simulation study to investigate the model performance compared to the normal or Gaussian error regression (GER) model. The results show that the Fisher's z error regression (ZER) model outperforms the GER model.

INTRODUCTION

JAGS (Just Another Gibbs Sampler) is a Bayesian graphics modeling program that aims for compatibility with classic BUGS (Bayesian inference Using Gibbs Sampling)[1]. The BUGS is a software package for performing Bayesian inference in which the user only needs to specify the structure of the model. In addition, BUGS uses Markov Chain Monte Carlo (MCMC) methods based on Gibbs sampling, to generate samples from the posterior distribution of the specified model [2]. JAGS is fully open source and written in C++ language [3]. Wabersich and Vandekerckhove [4] provided a very useful tutorial on writing and installing a standalone JAGS module, but it is easier to implement a shared JAGS library in an R package [3]. The configured script provided in the runjags package can be used as a template to create additional extension modules within R packages [3].

The functions in the runjags are designed to be user-friendly. The runjags package provide a number of features to make the recommended convergence and sample size checks more obvious to the end user. The runjags package also provides additional distributions including the Pareto types I, II, III and IV and other distributions such as the generalized Pareto, half-Cauchy, DuMouchel, and Lomax distributions [3]. However, the package did not explain how to add a new distribution to the package. This paper aims to discuss how to add the Fisher's z distribution to the runjags package, by modifying the package. This distribution will be used as an error term in the regression model.

Regression analysis is an important statistical tool that is commonly applied in most sciences. Among the many possible regression techniques, the least squares (LS) method has been generally adopted due to tradition and ease of computation. However, there is a widespread awareness of the dangers posed by the occurrence of outliers, which can be the result of typing errors, recording or transmission errors, misplaced decimals, exceptional phenomena such as earthquakes or strikes, or members of a different population slipping into the sample. Not only the response variable can be outlying, but also the explanatory part. Both types of outliers can completely ruin an ordinary LS analysis. To remedy outliers in regression analysis, the robust methods and the outlier diagnostics have been

International Conference on Mathematics, Computational Sciences and Statistics 2020
AIP Conf. Proc. 2329, 060005-1–060005-14; https://doi.org/10.1063/5.0042143
Published by AIP Publishing. 978-0-7354-4073-9/$30.00

developed [5]. In the context of robust methods, the interesting aspect of flexibility is represented by the possibility of adjusting the tail weight of the error term to contain outliers [6]. When the error term reaches the real number line, an interesting distribution is the Fisher's *z* distribution.

The standardized Fisher's *z* distribution was introduced by Fisher [7] as half of the logarithm of the *F*-distribution with two shape parameters d_1 and d_2. This distribution is always unimodal, has a zero-point mode, and has a symmetrical shape when the values of the two parameters are the same and has an asymmetrical shape if it is not the same. The standardized Fisher's *z* distribution is a family of the Chi-squared, Student *t*, and Normal distributions [7,8]. However, the distribution is not yet adaptive for modeling because the mode cannot be shifted from zero. Continuing the research of Fisher [7] and the previous researchers, this paper introduces the Fisher's *z* distribution, i.e., the standardized Fisher's *z* distribution which added a location parameter μ and a scale parameter σ. Furthermore, we provide step-by-step instructions on how to add the Fisher's *z* distribution to the runjags package.

THE *F* AND FISHER'S *Z* DISTRIBUTIONS

In this section, we discuss the probability density function, the cumulative distribution function, and the quantile function of the *F* and Fisher's *z* distributions.

The *F* Distribution

Let Y be a random variable distributed as an F distribution with d_1 and d_2 degrees of freedom. The probability density function (p.d.f.) of the *Y* be defined as [9]

$$f_Y\left(y;d_1,d_2\right)=\frac{\left(d_1/d_2\right)^{d_1/2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{y^{(d_1/2)-1}}{\left(1+yd_1/d_2\right)^{(d_1+d_2)/2}};\quad y>0;d_1>0;d_2>0, \tag{1}$$

and the cumulative distribution function (CDF) of the *Y* be defined as follows [9]

$$F_Y\left(y;d_1,d_2\right)=\frac{1}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\int_0^{y^*}t^{(d_1/2)-1}\left(1-t\right)^{(d_2/2)-1}dt;\quad y^*=\frac{d_1y}{d_2+d_1y}, \tag{2}$$

where $\mathrm{B}(.)$ be the beta function. The value y_p is called the *p*-quantile of the population, if $P\left(Y\le y_p\right)=p$ with $0\le p\le 1$ [10]. The quantile function (QF) of the *Y* be expressed as

$$y_p=\frac{d_2\mathrm{I}^{-1}{}_{y_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}{d_1\left(1-\mathrm{I}^{-1}{}_{y_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)\right)}, \tag{3}$$

where $I^{-1}{}_{y_p}(.)$ is the inversion of the incomplete beta function ratio.

The Fisher's *z* Distribution

Let Z be a random variable distributed as half of logarithm of an F distribution with two shape parameters d_1 and d_2, i.e., $Y = e^{2Z}$ is distibuted as *F* with the stated degrees of freedom. The density of *Z* is [7,8]

$$f_Z\left(z;d_1,d_2\right)=\frac{2d_1^{\frac{1}{2}d_1}d_2^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{d_1z}}{\left(d_1e^{2z}+d_2\right)^{(d_1+d_2)/2}};\quad -\infty<z<\infty;d_1>0;d_2>0. \tag{4}$$

Equation (4) is defined as a p.d.f of standardized Fisher's z distribution. Interchanging d_1 and d_2 is equivalent to replacing z with $-z$, so the p.d.f in equation (4) can also be defined as:

$$f_Z\left(z;d_1,d_2\right)=\frac{2d_1^{\frac{1}{2}d_1}d_2^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2z}}{\left(d_2e^{-2z}+d_1\right)^{(d_1+d_2)/2}}. \tag{5}$$

This distribution approached the standardized normal distribution as $d_1\to\infty$ and $d_2\to\infty$. If d_1 is infinete, this distribution tends to the Chi square distribution with d_2 degrees of freedom. Similarly if d_2 is infinete, it tends to the

Chi square distribution, with d_1 degrees of freedom. The standardized Fisher's z distribution approached a square of the standardized Student *t* distribution with d_1 degrees of freedom, if $d_2 = 1$ [7,8].

If Z is random variables distributed as a standardized Fisher's z, μ is a location parameter, and σ is a scale parameter, then the p.d.f of $X = \sigma Z + \mu$ is

$$f_X\left(x;d_1,d_2,\mu,\sigma\right)=\frac{2}{\sigma}\frac{d_1^{\frac{1}{2}d_1}d_2^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{x-\mu}{\sigma}\right)}}{\left(d_2e^{-2\left(\frac{x-\mu}{\sigma}\right)}+d_1\right)^{(d_1+d_2)/2}};\quad -\infty<x<\infty;\sigma>0;-\infty<\mu<\infty;d_1>0;d_2>0. \tag{6}$$

If the numerator and denominator of equation (6) are divided by $d_1^{(d_1+d_2)/2}$ then we get

$$f_X\left(x;d_1,d_2,\mu,\sigma\right)=\frac{2}{\sigma}\frac{\left(d_2/d_1\right)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{x-\mu}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{x-\mu}{\sigma}\right)+\ln\left(d_2/d_1\right)}\right)^{(d_1+d_2)/2}}. \tag{7}$$

Equation (7) is defined as a p.d.f of Fisher's z distribution, which is denoted by $\mathrm{z}(d_1, d_2, \mu, \sigma)$. The mode of the distribution lies in μ. The CDF of the Fisher's *z* distribution is expressed as

$$F_X(x;d_1,d_2,\mu,\sigma)=\frac{1}{\mathrm{B}(\frac{1}{2}d_1,\frac{1}{2}d_2)}\int_0^{x^*}t^{\frac{1}{2}d_2-1}\left(1-t\right)^{\frac{1}{2}d_1-1}dt;\quad x^*=\frac{d_2e^{-2\left(\frac{x-\mu}{\sigma}\right)}}{d_1+d_2e^{-2\left(\frac{x-\mu}{\sigma}\right)}}. \tag{8}$$

The QF of the Fisher's *z* distribution is

$$x_p=\mu+\frac{\sigma}{2}\ln\frac{d_2\mathrm{I}^{-1}_{x_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}{d_1\left[1-\mathrm{I}^{-1}_{x_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)\right]}; \tag{9}$$

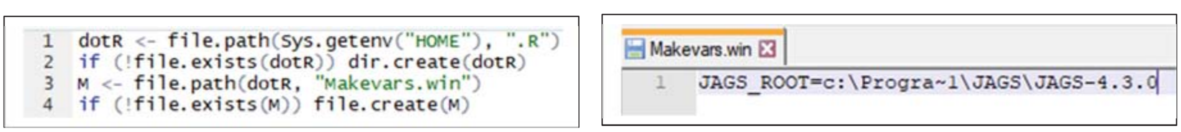
where $\left(d_2\mathrm{I}^{-1}_{x_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)\big/d_1\left[1-\mathrm{I}^{-1}_{x_p}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)\right]\right)$ is the QF of the *F*-distribution, as in equation (3).

STEPS TO ADDING THE DISTRIBUTION TO THE RUNJAGS

In this section, we describe the steps required to add a custom distribution in JAGS, by modifying the source code of the runjags package. We modify the source code with Rstudio [11], which are written in C++ programming language. The source code of the runjags package can be downloaded for free at https://cran.r-project.org/src/contrib/runjags_2.0.4-6.tar.gz. We start by installing the statistical software R [12], RStudio, Rtools, and JAGS before modifying the runjags. In the rest of this paragraph, it is assumed that R statistical software, RStudio, Rtools, and JAGS are installed in their default directories. Steps to modify the runjags package as follows:

Step 1 : Open a new project in RStudio.
Go to the `File` menu and click on `New Project`. Then select `Existing Directory`, browse to the runjags directory, click on `Open` and click on `Create Project` to modify the runjags package.

Step 2 : Create the `Makevars.win` file in the `C:/Users/user/Documents/.R` folder, with the code as shown in Fig. 1 (a).



(a) (b)

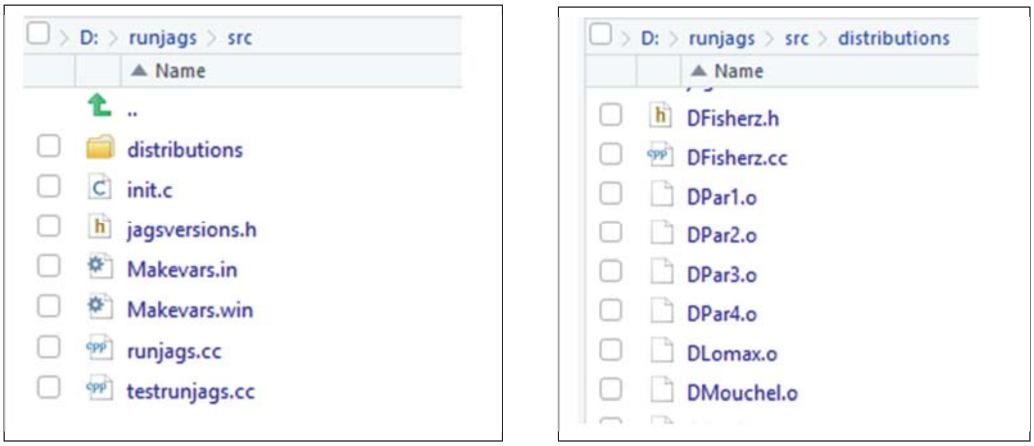
FIGURE 1. The `Makevars.win` File, (a) Creating The File (b) Modifying The File

Step 3 : Open the `Makevars.win` file in the folder `C:/Users/user/Documents/.R` and write the code on it, as shown in Fig.1 (b) (`JAGS_ROOT` = 'the location of JAGS program, that has been installed in our computer').

Step 4 : Modifying the `runjags.cc` module, which is located in the `/runjags/src` folder, as shown in Fig. 2 (a). The modification steps to modify `the runjags.cc` file are as follows:

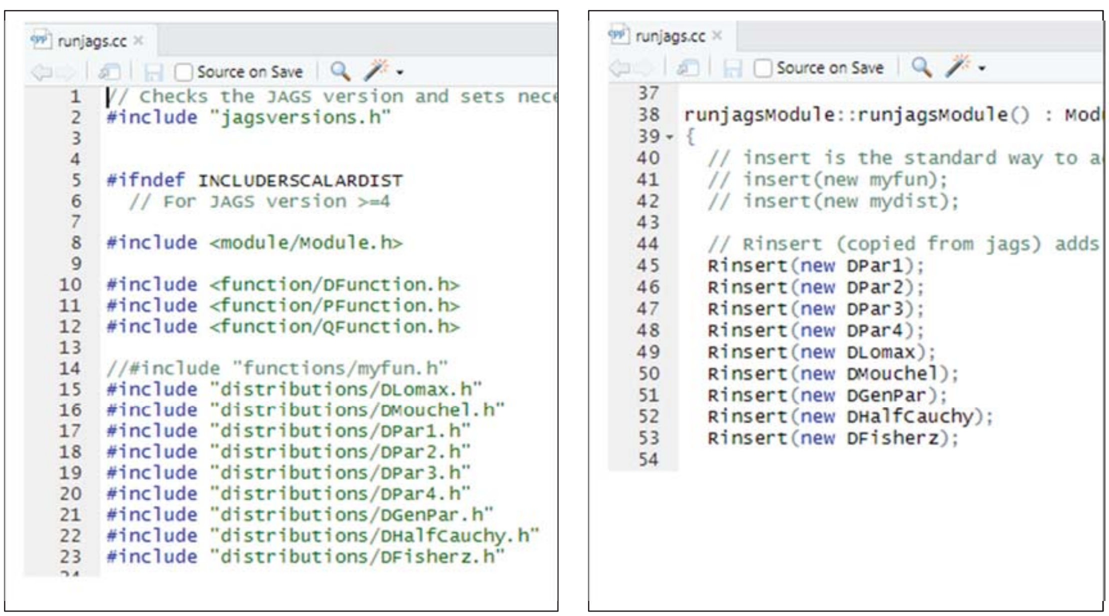
a. Write the distribution headers with the code `#include "distributions/DFisherz.h"` as shown in Fig. 3 (a) line 23.

b. Add the Fisher's *z* distribution in the constructor function `runjagsModule::runjagsModule() : Module("runjags")` with the code `Rinsert(new DFisherz);` as shown in Fig. 3 (b) line 53.



(a) (b)

FIGURE 2. The Folder Structure, (a) `/runjags/src` (b) `/runjags/src/distribution`



(a) (b)

FIGURE 3. The `runjags.cc` File, (a) Lines 1 to 24 (b) Lines 37 to 54

```
#ifndef DFisherz_H_
#define DFisherz_H_
// Checks the JAGS version and sets necessary macros:
#include "../jagsversions.h"
#ifndef INCLUDERSCALARDIST
#include <distribution/RScalarDist.h>
namespace jags {
#else
#include "jags/RScalarDist.h"
#endif         /* INCLUDERSCALARDIST */
namespace runjags {
class DFisherz : public RScalarDist {
 public:
  DFisherz(); // Constructor

  double d(double x, PDFType type,
     std::vector<double const *> const &parameters,
     bool give_log) const;
  double p(double q, std::vector<double const *> const &parameters, bool lower,
     bool give_log) const;
  double q(double p, std::vector<double const *> const &parameters, bool lower,
     bool log_p) const;
  double r(std::vector<double const *> const &parameters, RNG *rng) const;
     bool checkParameterValue(std::vector<double const *> const &parameters) const;
};
}              // namespace runjags
#ifndef INCLUDERSCALARDIST
}              // namespace jags
#endif         /* INCLUDERSCALARDIST */
#endif         /* DFisherz_H_ */
```

FIGURE 4. The Code of `DFisherz.h` Scalar Distribution Class Header File

```
#include "DFisherz.h"

#include <util/nainf.h>
#include <rng/RNG.h>
#include <cmath>
#include <cfloat>
#include <JRmath.h>
using std::vector;
using std::exp;
using std::log;

#define D1(par) (*par[0])
#define D2(par) (*par[1])
#define MU(par) (*par[2])
#define SIGMA(par) (*par[3])

#ifndef INCLUDERSCALARDIST
namespace jags {
#endif  /* INCLUDERSCALARDIST */

namespace runjags {

DFisherz::DFisherz()
  : RScalarDist("dz",4, DIST_UNBOUNDED)

{}

bool DFisherz::checkParameterValue (vector<double const *> const &par) const
{
  return (SIGMA(par) > 0 && D1(par) > 0 && D2(par) > 0 );
}
```

FIGURE 5. The Code of `DFisherz.cc` File, Lines 1 to 32

Step 5 : Create the Fisher's z functions file.

The functions file consists of two files, namely the `DFisherz.h` and the `DFisherz.cc`. These two files are placed in the `/runjags/src/distribution` folder, as shown in Fig. 2 (b).

a. The `DFisherz.h` scalar distribution class header file

   Figure 4 shows the prototypes of the constructor function and the four functions required, namely the d, p, q, and r functions.

b. The `DFisherz.cc` file

   The code in the `DFisherz.cc` file is shown in Fig. 5 through 7. We need to include the `util/nainf.h` and `rng/RNG.h` functions from the JAGS library, to provide the RNG struct and the `JAGS_*` constants, as well as the `jags_*` functions. We also need to include the `cmath` for standard math operations and need to include the `cfloat` for the characteristics of floating types for the specific system and compiler implementation used. Furthermore, the `JRmath.h` is needed to provide many basic functions that can be useful for writing extensions.

   We now need to implement the four functions and the prototyped constructor function in the `DFisherz.h`. The implementations of the required functions are provided, in Fig. 5, 6, and 7.

   1) The p.d.f of a Fisher's z distributed random variable X is in equation (7), and the log code of the p.d.f is written as shown in Fig. 6, lines 33 to 48. The code uses the function $\text{log1pexp}(u)$, more stable than by literally adding 1 to e^u and taking the logarithm, where $u = -2\left(\frac{x-\mu}{\sigma}\right) + \ln d_2 - \ln d_1$.
   2) The cumulative distribution function (CDF) of the Fisher's z distribution is in equation (8), which relates to the CDF of the F-distribution in equation (2). With the result that the CDF code in the `Fisherz.cc` file is written as shown in Fig. 6, lines 50 to 63.
   3) The quantile function (QF) of the Fisher's z distribution is in equation (9), which relates to the QF of the F-distribution in equation (3). The QF code in the `Fisherz.cc` file is written as shown in Fig. 7, lines 65 to 80.
   4) Generating random variates with the inversion method is exact when an explicit form of the QF is known [13]. In other cases, the QF of Fisher's z distribution is not an explicit form, so the random numbers are generated using some other method. The code to generate random number in the `Fisherz.cc` file is written as shown in Fig. 7, lines 82 to 93.

```
33 //Computing pdf
34
35 double
36 DFisherz::d(double x, PDFType type, vector<double const *> const &par, bool give_log) const
37 {
38   double mu = MU(par);
39   double sigma = SIGMA(par);
40   double d1 = D1(par);
41   double d2 = D2(par);
42   double z;
43   double lp;
44   z=(x-mu)/sigma;
45   lp= log(2)+0.5*d2*(log(d2)-log(d1))-d2*(x-mu)/sigma
46     -log(sigma)-lbeta(0.5*d1,0.5*d2)-(d1+d2)/2*log1pexp((-2*(x-mu)/sigma)+log(d2)-log(d1));
47   return give_log?lp: exp(lp);
48 }
49
50 //Computing CDF
51 double
52 DFisherz::p(double x, vector<double const *> const &par, bool lower, bool give_log)
53 const
54 {
55   double mu = MU(par);
56   double sigma = SIGMA(par);
57   double d1 = D1(par);
58   double d2 = D2(par);
59   double z;
60   z=(x-mu)/sigma;
61   double y = exp(2*z);
62   return pF(y,d1,d2,lower,give_log);
63  }
```

FIGURE 6. The Code of `DFisherz.cc` File, Lines 33 to 63

Step 6 : Modifying the `Makevars.in` module as shown in Fig. 8 line 24, which is located in the `/runjags/src` folder,

```
PKG_LIBS=-L@JAGS_LIB@ -ljags @JAGS_RPATH@
```

should be changed to

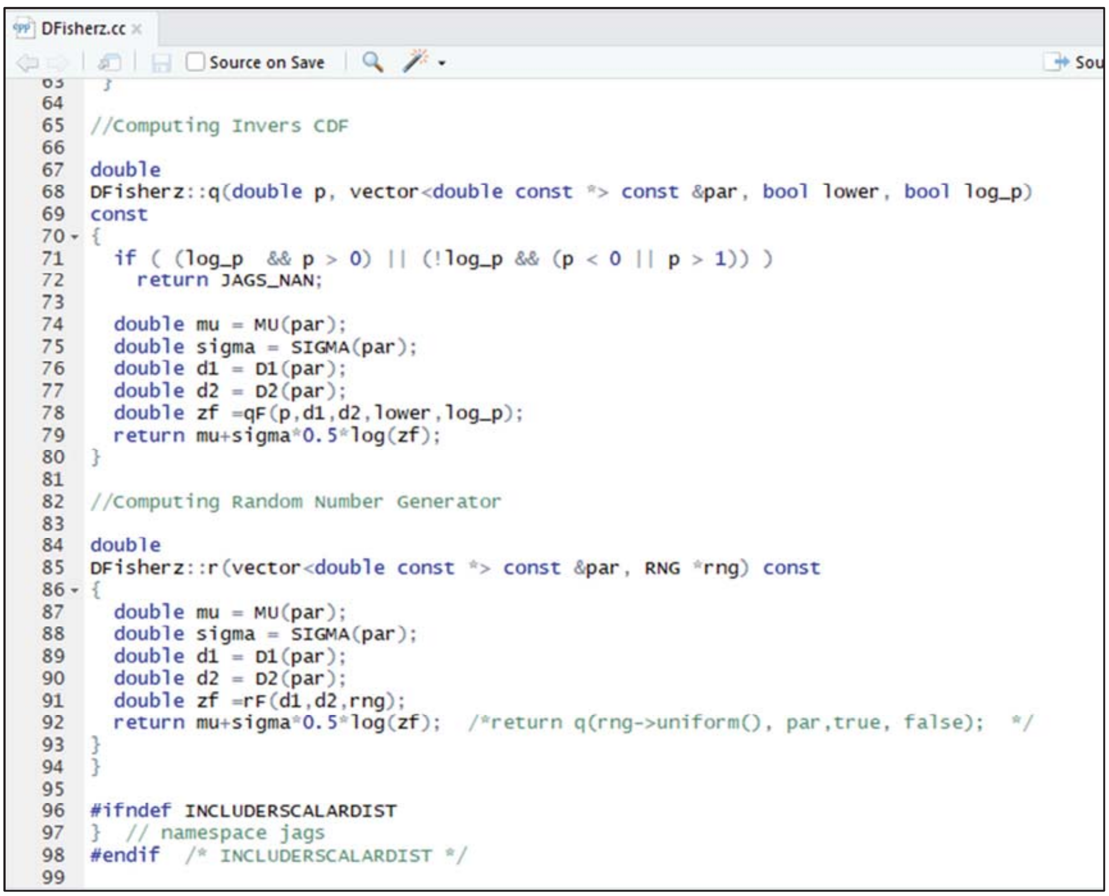
```
PKG_LIBS=-L@JAGS_LIB@ -ljags -ljrmath @JAGS_RPATH@
```

and line 35 of Fig. 8

```
OBJECTS = distributions/jags/DFunction.o distributions/jags/DPQFunction.o
distributions/jags/PFunction.o distributions/jags/QFunction.o
distributions/jags/RScalarDist.o distributions/DPar1.o distributions/DPar2.o
distributions/DPar3.o distributions/DPar4.o distributions/DLomax.o
distributions/DMouchel.o distributions/DGenPar.o distributions/DHalfCauchy.o
init.o runjags.o testrunjags.o
```

should be changed to

```
OBJECTS = distributions/jags/DFunction.o distributions/jags/DPQFunction.o
distributions/jags/PFunction.o distributions/jags/QFunction.o
distributions/jags/RScalarDist.o distributions/DPar1.o distributions/DPar2.o
distributions/DPar3.o distributions/DPar4.o distributions/DLomax.o
distributions/DMouchel.o distributions/DGenPar.o distributions/DHalfCauchy.o
distributions/DFisherz.o init.o runjags.o testrunjags.o
```



```
    }

//Computing Invers CDF

double
DFisherz::q(double p, vector<double const *> const &par, bool lower, bool log_p)
const
{
  if ( (log_p  && p > 0) || (!log_p && (p < 0 || p > 1)) )
    return JAGS_NAN;

  double mu = MU(par);
  double sigma = SIGMA(par);
  double d1 = D1(par);
  double d2 = D2(par);
  double zf =qF(p,d1,d2,lower,log_p);
  return mu+sigma*0.5*log(zf);
}

//Computing Random Number Generator

double
DFisherz::r(vector<double const *> const &par, RNG *rng) const
{
  double mu = MU(par);
  double sigma = SIGMA(par);
  double d1 = D1(par);
  double d2 = D2(par);
  double zf =rF(d1,d2,rng);
  return mu+sigma*0.5*log(zf);   /*return q(rng->uniform(), par,true, false);   */
}
}

#ifndef INCLUDERSCALARDIST
}  // namespace jags
#endif   /* INCLUDERSCALARDIST */
```

FIGURE 7. The Code of `DFisherz.cc` File, Lines 64 to 99

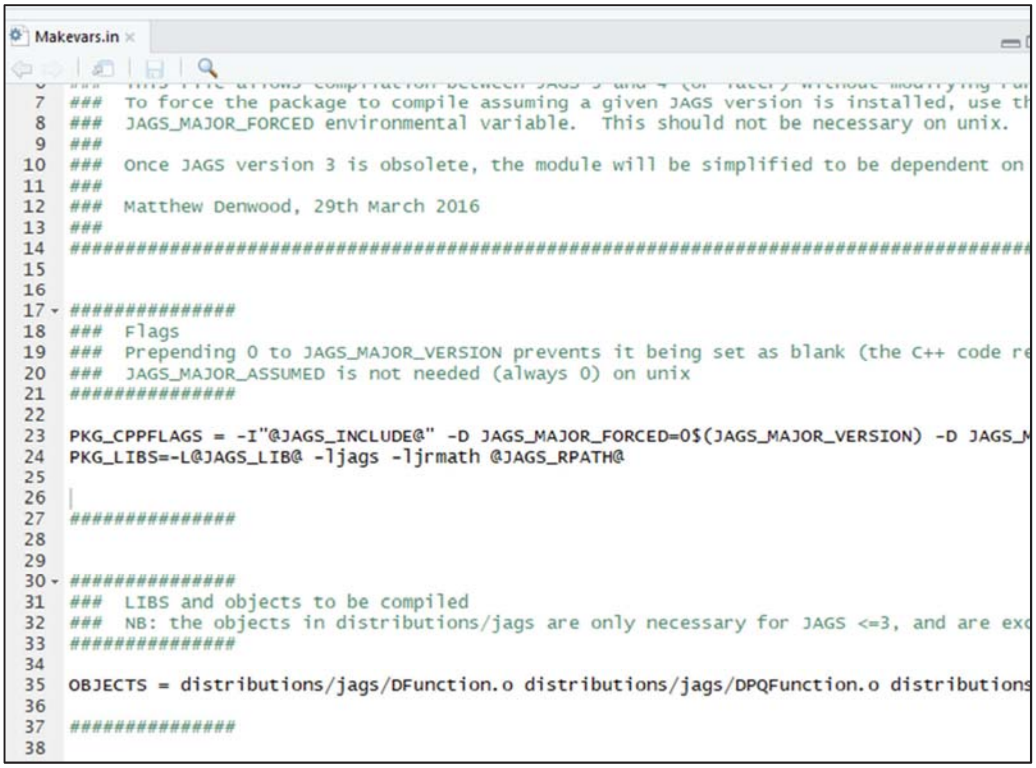


FIGURE 8. The Code of `Makevars.in` File, Lines 7 to 38

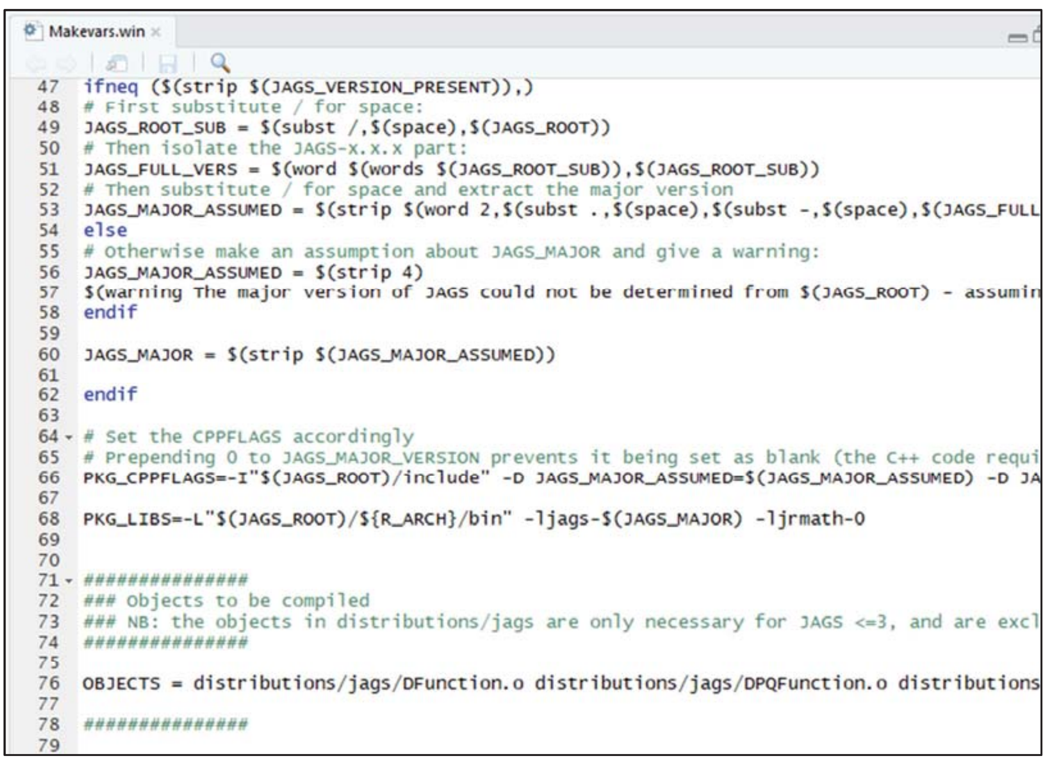


FIGURE 9. The Code of `Makevars.win` File, Lines 47 to 79

Step 7 : Modifying the `Makevars.win` module as shown in Fig. 9 line 68, which is located in the `/runjags/src` folder,

```
PKG_LIBS=-L"$(JAGS_ROOT)/${R_ARCH}/bin" -ljags-$(JAGS_MAJOR)
```

should be changed to

```
PKG_LIBS=-L"$(JAGS_ROOT)/${R_ARCH}/bin" -ljags-$(JAGS_MAJOR) -ljrmath-0
```

and line 76 of Fig. 9

```
OBJECTS = distributions/jags/DFunction.o distributions/jags/DPQFunction.o
distributions/jags/PFunction.o distributions/jags/QFunction.o
distributions/jags/RScalarDist.o distributions/DPar1.o distributions/DPar2.o
distributions/DPar3.o distributions/DPar4.o distributions/DLomax.o
distributions/DMouchel.o distributions/DGenPar.o distributions/DHalfCauchy.o
init.o runjags.o testrunjags.o
```

should be changed to

```
OBJECTS = distributions/jags/DFunction.o distributions/jags/DPQFunction.o
distributions/jags/PFunction.o distributions/jags/QFunction.o
distributions/jags/RScalarDist.o distributions/DPar1.o distributions/DPar2.o
distributions/DPar3.o distributions/DPar4.o distributions/DLomax.o
distributions/DMouchel.o distributions/DGenPar.o distributions/DHalfCauchy.o
distributions/DFisherz.o init.o runjags.o testrunjags.o
```

Step 8 : Installing the runjags package which has been modified. Go to the `Build` menu and click on `Install and Restart`. The successful installation can be confirmed by loading the runjags module in runjags package. To do this, type the following syntax in the R works:

```
> library(runjags)
> load.runjagsmodule()
module runjags loaded
```

NUMERICAL IMPLEMENTATION

To confirm the accuracy of the changes made, we performed a comprehensive numerical experiment, using the Fisher's *z* error regression (ZER) model with one predictor variable. The model is defined as follows

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i; \qquad i = 1, 2, \ldots, n \tag{10}$$

where y_i is the value of the response variable in the *i*th trial; β_0 and β_1 are the parameters; x_i is the value of the predictor variable in the *i*th trial; ε_i is a random error term, $\varepsilon_i \sim \mathrm{z}(d_1, d_2, 0, \sigma)$. The likelihood of the model, with the parameters $\boldsymbol{\theta} = (d_1, d_2, \sigma, \beta_0, \beta_1)$, can be formed by

$$\mathrm{L}(\boldsymbol{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{2}{\sigma} \frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1, \frac{1}{2}d_2\right)} \frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1 + e^{-2\left(\frac{\varepsilon_i}{\sigma}\right) + \ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}; \tag{11}$$

where $\varepsilon_i = y_i - (\beta_0 + \beta_1 x_i)$. In this paper, we conduct a Bayesian method to estimate the parameters $\boldsymbol{\theta}$, using MCMC with the Gibbs sampling algorithm. The Bayesian analysis requires the joint posterior density $\pi(\boldsymbol{\theta}|\mathbf{y}, \mathbf{x})$, which is defined by

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{x}) \propto \mathrm{L}(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) \tag{12}$$

where $\pi(\boldsymbol{\theta})$ are the prior of the parameters model, with

$$\pi(\boldsymbol{\theta}) = \pi(d_1)\pi(d_2)\pi(\sigma)\pi(\beta_0)\pi(\beta_1). \tag{13}$$

The $\pi(d_1)$, $\pi(d_2)$, $\pi(\sigma)$, $\pi(\beta_0)$ and $\pi(\beta_1)$ are priors for the d_1, d_2, σ, β_0, and β_1 parameters. For the priors of the d_1, d_2 and σ, we take the singly truncated Student *t* distributions (positive values only) [14], with the degrees of freedoms v_1, v_2 and v_3, the location parameters m_1, m_2 and m_3, the scale parameters $s_1{}^2, s_2{}^2$ and $s_3{}^2$, respectively. Therefore, $d_1 \sim t_{v_1}(m_1, s_1{}^2)I(0,\infty)$, $d_2 \sim t_{v_2}(m_2, s_2{}^2)I(0,\infty)$, and $\sigma \sim t_{v_3}(m_3, s_3{}^2)I(0,\infty)$. For the priors of the β_0 and β_1, we take the normal distributions [15], with the location parameters m_4, m_5 and the scale parameters $s_4{}^2, s_5{}^2$, thus $\beta_0 \sim N(m_4, s_4{}^2)$ and $\beta_1 \sim N(m_5, s_5{}^2)$. With the above configuration of the prior distributions, the joint posterior distribution of the model is given by

$$\pi(\boldsymbol{\theta}\mid\mathbf{y},\mathbf{x})\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\Delta_1\Delta_2\Delta_3\Delta_4\Delta_5; \tag{14}$$

$$\Delta_1=\left\{\left(1+\frac{1}{\nu_1}\left(\frac{d_1-m_1}{s_1}\right)^2\right)^{-\frac{(\nu_1+1)}{2}}\right\};\Delta_2=\left\{\left(1+\frac{1}{\nu_2}\left(\frac{d_2-m_2}{s_2}\right)^2\right)^{-\frac{(\nu_2+1)}{2}}\right\};$$

$$\Delta_3=\left\{\left(1+\frac{1}{\nu_3}\left(\frac{\sigma-m_3}{s_3}\right)^2\right)^{-\frac{(\nu_3+1)}{2}}\right\};\Delta_4=\exp\left(-\frac{1}{2}\left(\frac{\beta_0-m_4}{s_4}\right)^2\right);\Delta_5=\exp\left(-\frac{1}{2}\left(\frac{\beta_1-m_5}{s_5}\right)^2\right).$$

The full conditional distributions for all parameters must be derived to implement the Gibbs sampling algorithm for the joint posterior distribution in equation (14). The full conditional distributions for $d_1,\ d_2,\ \sigma,\ \beta_0$ and β_1 are given by

$$\pi(d_1\mid\mathbf{y},\mathbf{x},d_2,\sigma,\beta_0,\beta_1)\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\left\{\left(1+\frac{1}{\nu_1}\left(\frac{d_1-m_1}{s_1}\right)^2\right)^{-\frac{(\nu_1+1)}{2}}\right\}; \tag{15}$$

$$\pi(d_2\mid\mathbf{y},\mathbf{x},d_1,\sigma,\beta_0,\beta_1)\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\left\{\left(1+\frac{1}{\nu_2}\left(\frac{d_2-m_2}{s_2}\right)^2\right)^{-\frac{(\nu_2+1)}{2}}\right\}; \tag{16}$$

$$\pi(\sigma\mid\mathbf{y},\mathbf{x},d_1,d_2,\beta_0,\beta_1)\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\left\{\left(1+\frac{1}{\nu_3}\left(\frac{\sigma-m_3}{s_3}\right)^2\right)^{-\frac{(\nu_3+1)}{2}}\right\}; \tag{17}$$

$$\pi(\beta_0\mid\mathbf{y},\mathbf{x},d_1,d_2,\sigma,\beta_1)\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\exp\left(-\frac{1}{2}\left(\frac{\beta_0-m_4}{s_4}\right)^2\right); \tag{18}$$

$$\pi(\beta_1\mid\mathbf{y},\mathbf{x},d_1,d_2,\sigma,\beta_0)\propto\prod_{i=1}^{n}\frac{2}{\sigma}\frac{(d_2/d_1)^{\frac{1}{2}d_2}}{\mathrm{B}\left(\frac{1}{2}d_1,\frac{1}{2}d_2\right)}\frac{e^{-d_2\left(\frac{\varepsilon_i}{\sigma}\right)}}{\left(1+e^{-2\left(\frac{\varepsilon_i}{\sigma}\right)+\ln(d_2/d_1)}\right)^{(d_1+d_2)/2}}\exp\left(-\frac{1}{2}\left(\frac{\beta_1-m_5}{s_5}\right)^2\right). \tag{19}$$

Each draw can be performed using the Adaptive Rejection Metropolis Sampler, implemented in JAGS. A tutorial to illustrate how to use the JAGS can be found in the JAGS manual [16].

For the numerical implementation, some simulated datasets were generated with known parameter values of the small, moderate, and large sample sizes, and then were followed by fitting the model. Suppose that $\mathbf{x}=(x_1,x_2,\ldots,x_n)'$ is a vector of independent variable which has taken the values 1 to n; where $n=20,30,100$ and suppose that $\beta_0=10,\ \beta_1=2$. We generated some datasets from the model of equation (10) where $\varepsilon_i\sim\mathrm{z}(d_1,d_2,0,\sigma)$. The scenarios were considered by applying five types of error terms. The scenarios are as follows:

- Scenario 1: $d_1=1,\ d_2=10,\sigma=8$, represent the highly skewed left [17] and fat-tailed regression models, where the skewness and excess kurtosis of -1.43 and 3.67, respectively;

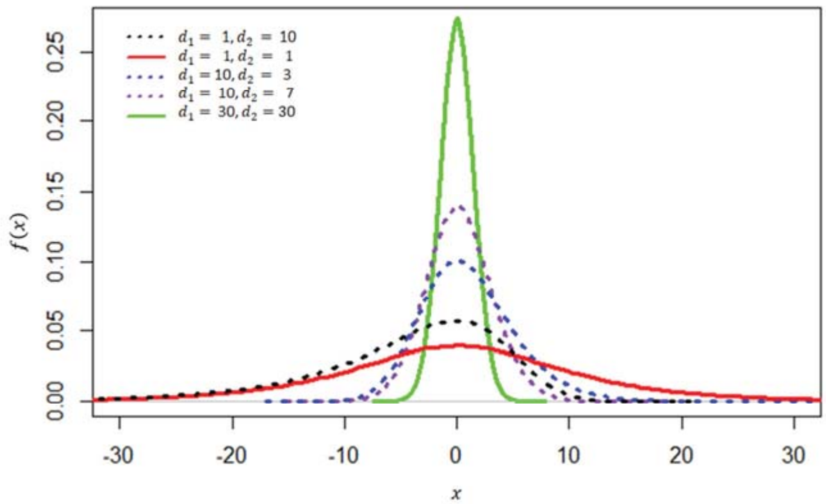


FIGURE 10. Probability Density Functions (p.d.f) of The Fisher's z Distribution when $\mu = 0$, σ=8 at Various Choices of d_1 and d_2

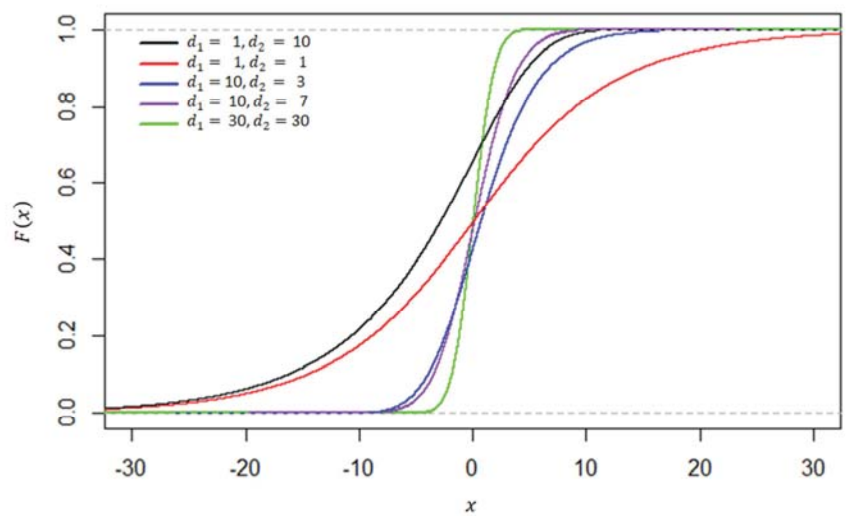


FIGURE 11. Cumulative distribution functions (CDF) of The Fisher's z Distribution when $\mu = 0$, σ=8 at Various Choices of d_1 and d_2

- Scenario 2: $d_1 = 1$, $d_2 = 1, \sigma = 8$, represent the fairly symmetrical [17] and fat-tailed regression models, the error term being lighter than the previous scenario, where the skewness and excess kurtosis are 0 and 2, respectively;
- Scenario 3: $d_1 = 10$, $d_2 = 3, \sigma = 8$, represent the moderately skewed [17] and fat-tailed regression models, the error term being lighter than the previous scenario, where the skewness and excess kurtosis are 0.63 and 1.07, respectively;
- Scenario 4: $d_1 = 10$, $d_2 = 7, \sigma = 8$, represent the fairly symmetrical and fat-tailed regression models, the error term being lighter than the previous scenario, where the skewness and excess kurtosis are 0.14 and 0.30, respectively;
- Scenario 5: $d_1 = 30$, $d_2 = 30, \sigma = 8$, represent the fairly symmetrical and fat-tailed regression models, the error term is the lightest, where the skewness and excess kurtosis are 0 and 0.07, respectively.

The comparison of graphical visualizations for the error terms in scenarios 1 to 5 can be seen in Fig. 10 and Fig.11. Furthermore, we have fitted and compared the performance of the ZER model with the Gaussian error regression

(GER) model. To compare the performance of the models, we use the Widely Applicable Information Criterion (WAIC) [18–20]. Vehtari et al. [19] implemented the WAIC calculations in the R package, called the loo package.

For scenario 1 with large samples $n = 100$, the generated data is shown in Fig. 12, which is generated by using the following code:

```
> n <- 100
> x <- seq(1, n, by = 1)
> data<-list(x=x,n=n)
> parameters<-c( "e","y")
> models<-"
+ model
+ {
+    for (i in 1:n){
+      e[i] ~ dz(1, 10 , 0, 8)
+      y[i]=10+2*x[i]+e[i]
+    }
+ }"
> generate <-run.jags( method=c("rjags"),
+                          data=data,
+                          inits=list(.RNG.name="base::Super-Duper",.RNG.seed=1),
+                          model=models,
+                          monitor=parameters ,
+                          n.chains=1,
+                          sample=1,
+                          thin=1,
+                          summarise=FALSE,
+                          modules=c("runjags"))
```

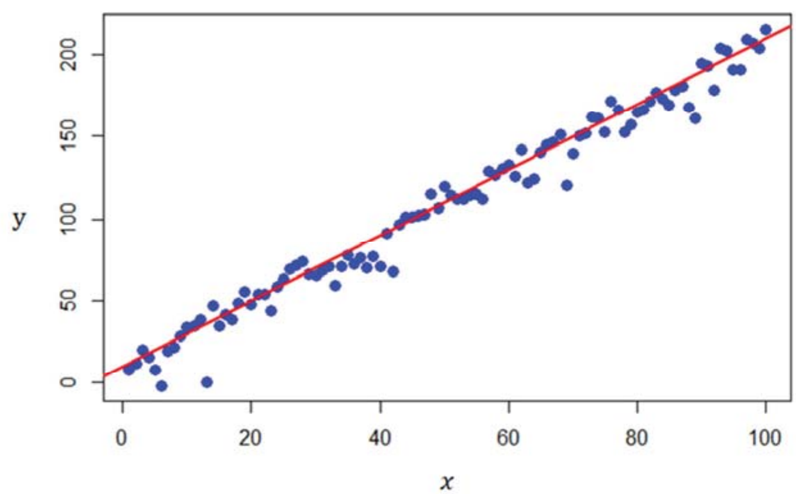


FIGURE 12. Scatterplot of **y** versus **x**

The estimator, i.e., the posterior mean obtained after running two chains for 1,000 iterations, using 30 thin intervals and discarding the first 40000 as burn-in and adapting, for a total of 2000 samples. The code is written as follows:

```
> gen<-as.vector(generate$mcmc[[1]])
> e  <- gen[1:100]
> y  <- gen[101:200]
> n <-length(y)
> data <-list(y=y,n=n, x=x)
> parameters<-c("d1","d2", "sigma","beta0","beta1")
> init1<-list(d1=1, d2=10, sigma=8, beta1=2, beta0=10, .RNG.name="base::Super-Duper",
.RNG.seed=1)
```

```
> init2<-list(d1=1, d2=10, sigma=8, beta1=2, beta0=10, .RNG.name="base::Wichmann-
Hill", .RNG.seed=2)
> models<-"
+   model
+   {
+   for (i in 1:n){
+     y[i] ~  dz(d1,d2,mu[i],sigma)
+     mu[i]<- beta0 + beta1 * x[i]
+   }
+   beta0 ~ dnorm(10, 1)
+   beta1 ~ dnorm(2, 1)
+   d1    ~ dt( 1, 0.1,3)T(0.0001,)
+   d2    ~ dt(10, 0.1,3)T(0.0001,)
+   sigma ~ dt( 8, 0.1,3)T(0.0001,)
+   }"
> obj <-run.jags( method=c("rjags"),
+                     data=data,
+                     inits=list(init1,init2),
+                     model=models,
+                     monitor=parameters ,
+                     adapt=20000 ,
+                     burnin=20000 ,
+                     sample=1000 ,
+                     thin=30,
+                     summarise=TRUE,
+                     n.chains=2,
+                     modules=c("runjags"))
```

The results of the simulation can be examined using the default print method as follows:

```
> obj

JAGS model summary statistics from 2000 samples (thin = 30; chains = 2; adapt+burnin
= 40000):

      Lower95  Median Upper95   Mean       SD Mode      MCerr MC%ofSD SSeff
d1     0.6475  1.2277   2.181  1.302  0.41932   --    0.00938     2.2  2000
d2     3.8143 10.5320  18.836 11.159  4.48220   --    0.10022     2.2  2000
sigma  6.0981  9.0592  13.354  9.326  1.90940   --    0.04443     2.3  1847
beta0  8.5353 10.2320  11.773 10.239  0.83176   --    0.01922     2.3  1874
beta1  1.9731  2.0083   2.044  2.008  0.01806   --    0.00040     2.2  2006

          AC.300    psrf
d1     -0.029562  1.0001
d2      0.002014  1.0007
sigma  -0.026960  1.0001
beta0  -0.022871  1.0003
beta1  -0.003622  1.0013

Total time taken: 3.9 minutes
```

The results show that the posterior mean values of all parameters are similar to the value of each parameter at the model setting in the first scenario. The potential scale reduction factor (psrf) values are less than 1.01 and the effective sample size (Sseff) values are greater than 400, indicating the convergences of the MCMC chains [20].

Table 1 shows the simulation result for all scenarios, indicating that the ZER model is better than the GER model. WAIC values for the ZER model are the smallest in all scenarios and for all sample sizes. For the first scenario, where the dataset is generated using the highly skewed to left and fat-tail error term, the WAIC values of the ZER model for the sizes of the twenty, thirty, and one hundred samples are 143.7, 207.3, and 701.6, respectively. While, for the same scenario, the WAIC values of the GER model for the sizes of the twenty, thirty and one hundred samples are 156.5, 223.5 and 732.6, respectively. Likewise for the other scenarios, the WAIC values of the ZER model are smaller than the GER model.

TABLE 1. Comparison the Fisher's z error regression (ZER) model and the Gaussian error regression (GER) model using widely applicable information criterion (WAIC) at several scenarios

Scenarios	Skewness	Excess kurtosis	$n = 20$		$n = 30$		$n = 100$	
			ZER	GER	ZER	GER	ZER	GER
1	-1.43	3.67	143.7*	156.5	207.3*	223.5	701.6*	732.6
2	0,00	2.00	168.3*	171.5	245.4*	248.8	800.3*	820.4
3	0.63	1.07	124.8*	129.8	187.7*	202.3	588.9*	628.2
4	0.14	0.30	105.1*	121.7	154.6*	172.1	497.5*	516.9
5	0,00	0.07	75.8*	109.8	110.9*	140.1	357.7*	363.8

Scenario 1: $\varepsilon_i \sim \mathrm{z}(\ 1, 10, 0, 8)$;
Scenario 2: $\varepsilon_i \sim \mathrm{z}(\ 1,\ \ 1, 0, 8)$;
Scenario 3: $\varepsilon_i \sim \mathrm{z}(10,\ \ 3, 0, 8)$;
Scenario 4: $\varepsilon_i \sim \mathrm{z}(10,\ \ 7, 0, 8)$;
Scenario 5: $\varepsilon_i \sim \mathrm{z}(30, 30, 0, 8)$.
* indicates the smallest value.

CONCLUSION

This paper introduces the Fisher's z distribution, i.e., the standardized Fisher's z distribution which added a location parameter μ and a scale parameter σ. We provide step-by-step instructions on how to adding Fisher's z distributions to the runjags package, by modifying the package. In order to affirm the accuracy of our implementation, we ran a comprehensive numerical experiment, using linear regression model. Some simulated datasets were generated with known parameter values of the ZER model, with the small, moderate, and large sample sizes. We compared the performance of the ZER model with the GER model. The results show that the ZER model is better than the GER model.

REFERENCES

1. M. Plummer, in Proceedings of the 3rd International Workshop on Distributed Statistical Computing (Vienna, Austria, 2003), pp. 1–10.
2. I. Ntzoufras, Bayesian Modeling Using WinBUGS (John Wiley & Sons, 2009).
3. M. J. Denwood, Journal of Statistical Software **71**, 1 (2016).
4. D. Wabersich and J. Vandekerckhove, Behavior Research Methods **46**, 15 (2014).
5. P. J. Rousseeuw and A. M. Leroy, Robust Regression and Outlier Detection (John wiley & sons, 1987).
6. A. Azzalini, in Recent Advances in Robust Statistics: Theory and Applications (Springer, 2016), pp. 1–16.
7. R. A. Fisher, in Proceedings of the International Congress of Mathematics (Toronto, 1924), pp. 805–813.
8. L. A. Aroian, The Annals of Mathematical Statistics **12**, 429 (1941).
9. N. L. Johnson, S. Kotz, and N. Balakrishnan, Continuous Univariate Distributions, Second (John Wiley&Sons, 1995).
10. W. Gilchrist, Statistical Modelling with Quantile Functions (CRC Press, 2000).
11. RStudio Team, RStudio: Integrated Development Environment for R (RStudio, Inc., Boston, MA, 2019).
12. R Core Team, R: A Language and Environment for Statistical Computing (R Foundation for Statistical Computing, Vienna, Austria, 2020).
13. L. Devroye, Non-Uniform Random Variate Generation (Springer-Verlag, New York Inc., 1986).
14. H.-J. Kim, Journal of the Korean Statistical Society **37**, 81 (2008).
15. N. L. Johnson, S. Kotz, and N. Balakrishnan, Continuous Univariate Distributions Volume 1, Second (John Wiley & Sons, Ltd, 1994).
16. M. Plummer, JAGS Version 4.3.0 User Manual (2017).
17. M. G. Bulmer, Principles of Statistics (M.I.T. PRESS, 1967).
18. S. Watanabe, Journal of Machine Learning Research **11**, 3571 (2010).
19. A. Vehtari, A. Gelman, and J. Gabry, Statistics and Computing **27**, 1413 (2017).
20. A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P. C. Bürkner, Bayesian Analysis (2020) https://doi.org/10.1214/20-BA1221.