Measurement Errors in R

by Iñaki Ucar, Edzer Pebesma, Arturo Azcorra

April 24, 2018

Abstract

This paper presents an R package to handle and represent measurements with errors in a very simple way. We briefly introduce the main concepts of metrology and propagation of uncertainty, and discuss related R packages. Building upon this, we introduce the errors package, which provides a class for associating uncertainty metadata, automated propagation and reporting. Working with errors enables transparent, lightweight, less error-prone handling and convenient representation of measurements with errors. Finally, we discuss the advantages, limitations and future work of computing with errors.

1 Introduction

The International Vocabulary of Metrology (VIM) defines a quantity as “a property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference”, where most typically the number is a quantity value, attributed to a measurand and experimentally obtained via some measurement procedure, and the reference is a measurement unit (BIPM et al., 2012).

Additionally, any quantity value must accommodate some indication about the quality of the measurement, a quantifiable attribute known as uncertainty (also traditionally known as error). The Guide to the Expression of Uncertainty in Measurement (GUM) defines uncertainty as “a parameter, associated with the result of a measurement, that characterises the dispersion of the values that could reasonably be attributed to the measurand” (BIPM et al., 2008). Uncertainty can be mainly classified into standard uncertainty, which is the result of a direct measurement (e.g., electrical voltage measured with a voltmeter, or current measured with an amperimeter), and combined standard uncertainty, which is the result of an indirect measurement (i.e., the standard uncertainty when the result is derived from a number of other quantities by the means of some mathematical relationship; e.g., electrical power as a product of voltage and current). Therefore, provided a set of quantities with known uncertainties, the process of obtaining the uncertainty of a derived measurement is called propagation of uncertainty.

Traditionally, computational systems have treated these three components (quantity values, measurement units and uncertainty) separately. Data consisted of bare numbers, and mathematical operations applied to them solely. Units were just metadata, and error propagation was an unpleasant task requiring additional effort and complex operations. Nowadays though, many software libraries have formalised quantity calculus as method of including units within the scope of mathematical operations, thus preserving dimensional correctness and protecting us from computing nonsensical combinations of quantities. However, these libraries rarely integrate uncertainty handling and propagation (Flater, 2018).

Within the R environment, the units package (Pebesma and Mailund, 2017; Pebesma et al., 2016) defines a class for associating unit metadata to numeric vectors, which enables transparent quantity derivation, simplification and conversion. This approach is a very comfortable way of managing units with the added advantage of eliminating an entire class of potential programming mistakes. Unfortunately, neither units nor any other package address the integration of uncertainties into quantity calculus.

This article presents errors, a package that defines a framework for associating uncertainty metadata to R vectors, matrices and arrays, thus providing transparent, lightweight and automated propagation of uncertainty. This implementation also enables ongoing developments for integrating units and uncertainty handling into a complete solution.
Figure 1: Illustration of linearity in an interval ± one standard deviation around the mean.

2 Propagation of uncertainty

There are two main methods for propagation of uncertainty: the Taylor series method (TSM) and the Monte Carlo method (MCM). The TSM, also called the delta method, is based on a Taylor expansion of the mathematical expression that produces the output variables. As for the MCM, it is able to deal with generalised input distributions and propagates the error by Monte Carlo simulation.

2.1 Taylor series method

The TSM is a flexible method of propagation of uncertainty that can offer different degrees of approximation given different sets of assumptions. The most common and well-known form of TSM is a first-order TSM assuming normality, linearity and independence. In the following, we will provide a short description. A full derivation, discussion and examples can be found in Arras (1998).

Mathematically, an indirect measurement is obtained as a function of \( n \) direct or indirect measurements, \( Y = f(X_1, \ldots, X_n) \), where the distribution of \( X_n \) is unknown \textit{a priori}. Usually, the sources of random variability are many, independent and probably unknown as well. Thus, the central limit theorem establishes that an addition of a sufficiently large number of random variables tends to a normal distribution. As a result, the first assumption states that \( X_n \) are normally distributed.

The second assumption presumes linearity, i.e., that \( f \) can be approximated by a first-order Taylor series expansion around \( \mu_{X_n} \) (see Figure 1). Then, given a set of \( n \) input variables \( X \) and a set of \( m \) output variables \( Y \), the first-order error propagation law establishes that

\[
\Sigma_Y = J_X \Sigma_X J_X^T
\]  

(1)

where \( \Sigma \) is the covariance matrix and \( J \) is the Jacobian operator.

Finally, the third assumption supposes independency among the uncertainty of the input variables. This means that the cross-covariances are considered to be zero, and the equation above can be simplified into the most well-known form of the first-order TSM:

\[
(\Delta y)^2 = \sum_i \left( \frac{\partial f}{\partial x_i} \right)^2 (\Delta x_i)^2
\]  

(2)

In practice, as recommended in the GUM (BIPM et al., 2008), this first-order approximation is good even if \( f \) is non-linear, provided that the non-linearity is negligible compared to the magnitude of the uncertainty, i.e., \( \mathbb{E}[f(X)] \approx f(\mathbb{E}[X]) \). Also, this weaker condition is distribution-free: no assumptions are needed on the probability density functions (PDF) of \( X_n \), although they must be reasonably symmetric.
2.2 Monte Carlo method

The MCM is based on the same principles underlying the TSM. It is based on the propagation of the PDFs of the input variables $X_n$ by performing random sampling and evaluating them under the model considered. Thus, this method is not constrained by the TSM assumptions, and explicitly determines a PDF for the output quantity $Y$, which makes it a more general approach that applies to a broader set of problems. For further details on this method, as well as a comparison with the TSM and some discussion on the applicability of both methods, the reader may refer to the Supplement 1 of the GUM (BIPM et al., 2008).

3 Reporting uncertainty

The GUM (BIPM et al., 2008) defines four ways of reporting standard uncertainty and combined standard uncertainty. For instance, if the reported quantity is assumed to be a mass $m_S$ of nominal value 100 g:

1. $m_S = 100.02147 \text{ g with (a combined standard uncertainty) } u_c = 0.35 \text{ mg}$.
2. $m_S = 100.02147(35) \text{ g, where the number in parentheses is the numerical value of } u_c$ referred to the corresponding last digits of the quoted result.
3. $m_S = 100.02147(0.00035) \text{ g, where the number in parentheses is the numerical value of (the combined standard uncertainty) } u_c$ expressed in the unit of the quoted result.
4. $m_S = (100.02147 \pm 0.00035) \text{ g, where the number following the symbol } \pm$ is the numerical value of (the combined standard uncertainty) $u_c$ and not a confidence interval.

Schemes (2, 3) and (4) would be referred to as parenthesis notation and plus-minus notation respectively throughout this document. Although (4) is a very extended notation, the GUM explicitly discourages its use to prevent confusion with confidence intervals.

4 Related work

Several R packages are devoted to or provide methods for propagation of uncertainties. The car (Fox and Weisberg, 2016, 2011) and msm (Jackson, 2016, 2011) packages provide the functions deltaMethod() and deltamethod() respectively. Both of them implement a first-order TSM with a similar syntax, requiring a formula, a vector of values and a covariance matrix, thus being able to deal with dependency across variables.

```r
library(msm)
vals <- c(5, 1)
err <- c(0.01, 0.01)
deltamethod(~ x1 / x2, vals, diag(err**2))
#> [1] 0.0509902
```

The metRology (Ellison, 2017) and propagate (Spiess, 2014) packages stand out as very comprehensive sets of tools specifically focused on this topic, including both TSM and MCM. The metRology package implements TSM using algebraic or numeric differentiation, with support for correlation. It also provides a function for assessing the statistical performance of GUM uncertainty (TSM) using attained coverage probability. The propagate package implements TSM up to second order. It provides a unified interface for both TSM and MCM through the propagate() function, which requires an expression and a data frame or matrix as input. In the following example, the error of $x/y$ is computed, where $x = 5.00(1)$ and $y = 1.00(1)$. The first result corresponds to the TSM method and the second one, to the MCM.

```r
library(propagate); set.seed(42)
x <- c(5, 0.01)
y <- c(1, 0.01)
propagate(expression(x/y), data=cbind(x, y), type="stat", do.sim=TRUE)
```

Unfortunately, as in the case of car and msm, these packages are limited to work only with expressions, which does not solve the issue of requiring a separate workflow to deal with uncertainties.

The spup package (Sawicka and Heuvelink, 2017) focuses on uncertainty analysis in spatial environmental modelling, where the spatial cross-correlation between variables becomes important. The uncertainty is described with probability distributions and propagated using MCM.

Finally, the distr package (Kohl, 2017; Ruckdeschel et al., 2006) takes this idea one step further by providing an S4-based object-oriented implementation of probability distributions, with which one can operate arithmetically or apply mathematical functions. It implements all kinds of probability distributions and has more methods for computing the distribution of derived quantities. Also, distr is the base of a whole family of packages, such as distrEllipse, distrEx, distrMod, distrRmetrics, distrSim and distrTeach.

All these packages provide excellent tools for uncertainty analysis and propagation. However, none of them addresses the issue of an integrated workflow as units does for unit metadata. Moreover, measurements with uncertainty must be properly reported, with an appropriate number of significant digits.

5 Automated uncertainty handling in R: the errors package

The errors package aims to provide easy and lightweight handling of measurement with errors, including propagation using the first-order TSM presented in the previous section and a formally sound representation. Errors, given as (combined) standard uncertainties, can be assigned to numeric vectors, matrices and arrays, and then all the mathematical and arithmetic operations are transparently applied to both the values and the associated errors. Following the example from the Related Work section,

```r
library(errors)

x <- c(5, 1)
errors(x) <- 0.01
x

#> Errors: 0.01 0.01
#> [1] 5 1

x[1] / x[2]

#> 5.00(5)
```

The errors() function assigns or retrieves a vector of errors, which is stored as an attribute of the class errors.

```r
str(x)

#> Class 'errors' atomic [1:2] 5 1
#> ..- attr(*, "errors")= num [1:2] 0.01 0.01
```

The set_errors() function was envisaged as a replacement of errors() for use in conjunction with the pipe operator (%>% from the magrittr (Bache and Wickham, 2014) package.

```r
library(magrittr)

(x <- 1:8 %>% set_errors(., . / 30))

#> Errors: 0.03333333 0.06666667 0.10000000 0.13333333 0.16666667 ... 
#> [1] 1 2 3 4 5 6 7 8
```

Internally, errors provides S3 methods for the generics belonging to the groups Math, Ops and Summary, plus additional operations such as subsetting ([, [<-, [[, [[<-), concatenation (c()), differentiation (diff), row and column binding (rbind, cbind), or coercion to data frame and matrix.
df <- as.data.frame(x)
df["3x"] <- 3*x
df["x**2"] <- x**2
df["sin(x)"] <- sin(x)
df["cumsum(x)"] <- cumsum(x)
df
#> x 3x x**2 sin(x) cumsum(x)
#> 1 1.00(3) 3.0(1) 1.00(7) 0.84(2) 1.00(3)
#> 2 2.00(7) 6.0(2) 4.0(3) 0.91(3) 3.00(7)
#> 3 3.0(1) 9.0(3) 9.0(6) 0.1(1) 6.0(1)
#> 4 4.0(1) 12.0(4) 16(1) -0.76(9) 10.0(2)
#> 5 5.0(2) 15.0(5) 25(2) -0.96(5) 15.0(2)
#> 6 6.0(2) 18.0(6) 36(2) -0.3(2) 21.0(3)
#> 7 7.0(2) 21.0(7) 49(3) 0.7(2) 28.0(4)
#> 8 8.0(3) 24.0(8) 64(4) 0.99(4) 36.0(5)

It is worth noting that both values and errors are stored with all the digits. However, when a single measurement or a column of measurements in a data frame are printed, the output is properly formatted to show a single significant digit for the error. This is achieved by providing S3 methods for `format()` and `print()`. The parenthesis notation is used by default, but this can be overridden through the appropriate option in order to use the plus-minus notation instead.

```
options(errors.notation = "plus-minus")
df
#> x 3x x**2 sin(x) cumsum(x)
#> 1 1.00 ± 0.03 3.0 ± 0.1 1.00 ± 0.07 0.84 ± 0.02 1.00 ± 0.03
#> 2 2.00 ± 0.07 6.0 ± 0.2 4.0 ± 0.3 0.91 ± 0.03 3.00 ± 0.07
#> 3 3.0 ± 0.1 9.0 ± 0.3 9.0 ± 0.6 0.1 ± 0.1 6.0 ± 0.1
#> 4 4.0 ± 0.1 12.0 ± 0.4 16 ± 1 -0.76 ± 0.09 10.0 ± 0.2
#> 5 5.0 ± 0.2 15.0 ± 0.5 25 ± 2 -0.96 ± 0.05 15.0 ± 0.2
#> 6 6.0 ± 0.2 18.0 ± 0.6 36 ± 2 -0.3 ± 0.2 21.0 ± 0.3
#> 7 7.0 ± 0.2 21.0 ± 0.7 49 ± 3 0.7 ± 0.2 28.0 ± 0.4
#> 8 8.0 ± 0.3 24.0 ± 0.8 64 ± 4 0.99 ± 0.04 36.0 ± 0.5
```

Additionally, other useful summaries are provided, namely, the mean, the weighted mean and the median. The error of any measurement of central tendency cannot be smaller than the error of the individual measurements. Therefore, the error assigned to the mean is computed as the maximum between the standard error of the mean and the mean of the individual errors (weighted, in the case of the weighted mean). As for the median, its error is computed as $\sqrt{\frac{\pi}{2}} \approx 1.253$ times the error of the mean, where this constant comes from the asymptotic variance formula (Hampel et al., 2011).

The default representation displays a single significant digit for the error. However, it can be overridden using the `digits` option in `format()` and `print()`, and it is globally controlled with the option `errors.digits`.

```
options(errors.notation = "parenthesis")
# the elementary charge
e <- set_errors(1.6021766208e-19, 0.0000000098e-19)
print(e, digits=2)
#> 1.6021766208(98)e-19
```

Finally, `errors` also facilitates error plotting. In the following, we first assign a 2% of error to all the numeric variables in the `iris` data set and then we plot it using base graphics and ggplot2 (Wickham and Chang, 2016; Wickham, 2009). The result is shown in Figure 2.

```
library(dplyr)

iris_e <- iris %>%
mutable_at(vars(-Species), funs(set_errors(., .*0.02)))

head(iris_e)
```
Figure 2: Base plot with errors (left) and ggplot2’s version (right).

```r
#> Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1 5.1(1) 3.50(7) 1.40(3) 0.200(4) setosa
#> 2 4.9(1) 3.00(6) 1.40(3) 0.200(4) setosa
#> 3 4.70(9) 3.20(6) 1.30(3) 0.200(4) setosa
#> 4 4.60(9) 3.10(6) 1.50(3) 0.200(4) setosa
#> 5 5.0(1) 3.60(7) 1.40(3) 0.200(4) setosa
#> 6 5.4(1) 3.90(8) 1.70(3) 0.400(8) setosa

plot(iris_e["Sepal.Length"], iris_e["Sepal.Width"], col=iris_e["Species"])
legend(6.2, 4.4, unique(iris_e["Species"]), col=1:length(iris_e["Species"]), pch=1)
library(ggplot2)

ggplot(iris_e, aes(Sepal.Length, Sepal.Width, color=Species)) +
  geom_point() + theme_bw() + theme(legend.position=c(0.6, 0.8)) +
  geom_errorbar(aes(ymin=errors_min(Sepal.Width), ymax=errors_max(Sepal.Width))) +
  geom_errorbarh(aes(xmin=errors_min(Sepal.Length), xmax=errors_max(Sepal.Length)))
```

In base graphics, the error bars are automatically plotted when an object of class `errors` is passed. Additionally, we provide the convenience functions `errors_min(x)` and `errors_max(x)` for obtaining the boundaries of the interval in `ggplot2` and other plotting packages, instead of writing `x - errors(x)` and `x + errors(x)` respectively.

6 Discussion

The `errors` package provides the means for defining numeric vectors, matrices and arrays with errors in R, as well as to operate with them in a transparent way. Propagation of errors implements the commonly used first-order TSM formula from Equation (2). This method has been pre-computed and expanded for each operation in the S3 groups `Math` and `Ops`, instead of differentiating symbolic expressions on demand or using functions from other packages for this task. The advantages of this approach are twofold. On the one hand, it is faster, as it does not involve simulation nor symbolic computation, and very lightweight, as `errors` has no dependencies. On the other hand, it sorts out some potential notation mistakes. For instance, let us consider the following:

```r
2*x
```

```r
#> Errors: 0.06666667 0.13333333 0.20000000 0.26666667 0.33333333 ...
#>  [1] 2 4 6 8 10 12 14 16
x + x
``` 

```r
#> Errors: 0.04714045 0.09428090 0.14142136 0.18856181 0.23570226 ...
#> [1] 2 4 6 8 10 12 14 16
```

Any method based on symbolic differentiation would yield the same result for both the above expressions, but it would be incorrect. This fact can be easily understood with a real-world example.
For instance, let us imagine that we want to measure the width of a table, but we have a ruler that is only about half its width. So we manage to put a mark, by the means of some method (using a string, for instance), approximately at about half of the width of the table. Then, there are two ways of proceeding: 1) to measure the first half and multiply it by two, or 2) to measure both halves and sum them.

Intuitively, 1), which corresponds to the \(2 \times x\) case, has a larger uncertainty, because we are not measuring the second half of the table (and note that this is exactly what we obtained before!). But in 2), even if the result of the second measurement matches the first one, \(x + x\) is an abuse of notation: they are different measurements, so we should specify \(x + y\) instead, and the derived uncertainty is smaller. In essence, adding a measurement to itself has no physical meaning. Nevertheless, \texttt{errors}\ provides the right results for the propagation of the error even if the expression provided is malformed in a physical sense, as we have shown in the example above.

Another advantage of \texttt{errors}\ is the built-in consistent and formally sound representation of measurements with errors, rounding the error to one significant digit by default and supporting two widely used notations: \textit{parenthesis} (e.g., \(5.00(1)\)) and \textit{plus-minus} (e.g., \(5.00 \pm 0.01\)). These notations are applied for single numbers and data frames, as well as \texttt{tbl\_df} data frames from the \texttt{tibble} (Wickham et al., 2017) package.

Full support is provided for both \texttt{data\_frame} and \texttt{tbl\_df}, as well as matrices and arrays. However, non-supported operations may drop errors. The reason is that R is aggressive with custom classes and attributes in general: many base functions simply drop classes and attributes and return numeric vectors, so that it is hard to cover all the possible scenarios and manipulations.

### 7 Summary and future work

We have introduced \texttt{errors}, a lightweight R package for managing numeric data with associated standard uncertainties. The new class \texttt{errors}\ provides numeric operations with automated propagation of uncertainty through a first-order TSM, and a formally sound representation of measurements with errors. Using this package makes the process of computing indirect measurements easier and less error-prone.

Future work includes importing and exporting data with uncertainties, and providing the user with an interface for plugging uncertainty propagation methods from other packages. Finally, \texttt{errors}\ enables ongoing developments for integrating \texttt{units} and uncertainty handling into a complete solution for quantity calculus. Having a unified workflow for managing measurements with units and errors would be an interesting addition to the R ecosystem with very few precedents in other programming languages.

### References

K. Arras. An introduction to error propagation: Derivation, meaning and examples of \(cy = fx \times cx \times fx\'. Sep 1998. Technical Report Nr. EPFL-ASL-TR-98-01 R3, Autonomous Systems Lab, EPFL, September 1998. [p]

S. M. Bache and H. Wickham. \textit{magrittr: A Forward-Pipe Operator for R,} 2014. URL \url{https://CRAN.R-project.org/package=magrittr}. R package version 1.5. [p]

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement, 1st edn. JCGM 100:2008. \textit{Joint Committee for Guides in Metrology,} 2008. URL \url{https://www.bipm.org/en/publications/guides/gum.html}. [p]

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. The International Vocabulary of Metrology – Basic and General Concepts and Associated Terms (VIM), 3rd edn. JCGM 200:2012. \textit{Joint Committee for Guides in Metrology,} 2012. URL \url{http://www.bipm.org/vim}. [p]

S. L. R. Ellison. \textit{metRology: Support for Metrological Applications,} 2017. URL \url{https://CRAN.R-project.org/package=metRology}. R package version 0.9-26-2. [p]

D. Flater. Architecture for software-assisted quantity calculus. \textit{Computer Standards & Interfaces,} 56:144–147, 2018. ISSN 0920-5489. doi: 10.1016/j.csi.2017.10.002. [p]

J. Fox and S. Weisberg. \textit{An R Companion to Applied Regression.} SAGE Publications, second edition, 2011. ISBN 9781412975148. [p]
J. Fox and S. Weisberg. *car: Companion to Applied Regression*, 2016. URL https://CRAN.R-project.org/package=car. R package version 2.1-4. [p]

F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN 9781118150689. [p]

C. Jackson. *msm: Multi-State Markov and Hidden Markov Models in Continuous Time*, 2016. URL https://CRAN.R-project.org/package=msm. R package version 1.6.4. [p]

C. H. Jackson. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 38(8):1–29, 2011. doi: 10.18637/jss.v038.i08. [p]

M. Kohl. *distr: Object Oriented Implementation of Distributions*, 2017. URL https://CRAN.R-project.org/package=distr. R package version 2.6.2. [p]

E. Pebesma and T. Mailund. *units: Measurement Units for R Vectors*, 2017. URL https://CRAN.R-project.org/package=units. R package version 0.4-4. [p]

E. Pebesma, T. Mailund, and J. Hiebert. Measurement Units in R. *The R Journal*, 8(2):486–494, 2016. URL https://journal.r-project.org/archive/2016/RJ-2016-061/index.html. [p]

P. Ruckdeschel, M. Kohl, T. Stabla, and F. Camphausen. S4 classes for distributions. *R News*, 6 (2):2–6, May 2006. [p]

K. Sawicka and G. Heuvelink. *spup: Uncertainty Propagation Analysis*, 2017. URL https://CRAN.R-project.org/package=spup. R package version 0.1-0. [p]

A.-N. Spiess. *propagate: Propagation of Uncertainty*, 2014. URL https://CRAN.R-project.org/package=propagate. R package version 1.0-4. [p]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. [p]

H. Wickham and W. Chang. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL https://CRAN.R-project.org/package=ggplot2. R package version 2.2.1. [p]

H. Wickham, R. Francois, and K. M"uller. *tibble: Simple Data Frames*, 2017. URL https://CRAN.R-project.org/package=tibble. R package version 1.3.0. [p]