https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

GEOCHRONOLOGY Discussions Open Access EGU

Robust Isochron Calculation

Roger Powell[1], Eleanor C. R. Green[1], Estephany Marillo Sialer[1], and Jon Woodhead[1]

[1]School Earth Sciences, The University of Melbourne, Vic 3010, Australia

Correspondence: Roger Powell (powell@unimelb.edu.au)

Abstract. A robust statistics approach to isochron calculations is presented, accompanied by an implementation in Python. It allows isochrons to be calculated for a wider range of datasets than the standard classical statistics approach, assuming that the distribution of uncertainties on the data is slightly fatter-tailed than Gaussian. The robust approach advocated reduces to the classical approach for "good" datasets.

1 Introduction

The realm of isochron calculations has been dominated by a classical statistics approach in which data uncertainties, derived from the analytical methods, are taken to be *strictly* Gaussian-distributed (e.g. York, 1969; York et al. 2004, and references therein). This approach will be referred to as YORK. In YORK, a goodness-of-fit parameter (mean standard weighted deviates, or `mswd`, see below) is used to determine whether a dataset can be considered to have age significance, giving an isochron (e.g. Wendt & Carl, 1991). If `mswd` is too large, the dataset is deemed not to have age significance and the data fit is referred to as an errorchron.

We suggest that using `mswd` in YORK is too restrictive, excluding datasets that would seem intuitively to have age significance. Rather than being truly Gaussian, data uncertainties may well be Gaussian-distributed in their centres, but slightly fat-tailed distant from the centres. Powell et al. (2002) showed that such behaviour has a devastating effect on the value of `mswd` in simulated datasets. Note that establishing the distribution of data uncertainties is impossible in the small datasets typical of isochron work. So, taking data uncertainties to be strictly Gaussian-distributed is an *assumption*.

An isotopic dataset commonly looks intuitively acceptable if the data has a central linear "spine", in which scatter is commensurate with stated analytical uncertainty, but this spine is flanked by data of somewhat larger scatter(*extra* scatter, from the "fat tail"). Age-significance in such data manifests primarily via the position of the spine. A successful calculation method for a dataset that may not have strictly Gaussian-distributed uncertainties must, firstly, ascertain via a robust test whether or not such a spine exists in the data—and hence, in the terminology of the classical statistics approach, whether calculations yield an isochron or an errorchron. Secondly, in the case of an isochron calculation, the successful method must reliably locate the spine without being perturbed by vagaries in the more scattered data. Classical statistical methods can do neither of these things, tending to be excessively influenced by the data at the extremes of the scatter. However, the field of robust statistics offers calculation methods that can. When a dataset has little or no extra scatter, so that `mswd` passes, such methods can retrieve

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

identical results to classical statistic methods, but they provide defensive age and age-uncertainty estimates in the presence of extra scatter. This applies regardless of whether the scatter originates in the isotopic analysis or in geological disturbance.

2 An Algorithm for Isochron Calculations

An algorithm is sought that finds a robust straight line through a 2-dimensional linear data trend, while becoming coincident with the classical statistical approach of YORK for datasets with consistent scatter (i.e. `mswd` passes). This section describes the nature of the problem and the theoretical basis for the robust statistical approach that will be adopted. The resulting algorithm is then evaluated via simulated datasets, and applied to a natural dataset. The algorithm is detailed in Appendix A. A `python` implementation is given in Appendix C.

2.1 Uncertainty distributions and data fitting

Geochronological datasets are collected on the presumption that the isotopic compositions were established via an "event" the age of which is to be estimated. Given the focus here on data with linear trends, even if the effect of the event is recorded perfectly by the samples analysed—the isotopic compositions lying on a line—the actual data are measured with finite precision so the data scatter about the trend. An uncertainty probability distribution can be used to describe the form of the data scatter.

Classical statistical methods assume that the underlying uncertainty distribution of a dataset is known, typically taken to be Gaussian. Under the Gaussian assumption, if the analytical uncertainty on the measurements have been appropriately inferred, `mswd`, a classical statistics parameter used in YORK to validate an isochron, tests that the scatter of datapoints is consistent with the inferred uncertainties, given the assumption of Gaussian-distributed scatter. But, in general, there is no reason to suppose that a given analytical technique generates a truly Gaussian uncertainty distribution. If it is not, classical methods of data fitting become strictly inappropriate.

While there are many possible non-Gaussian uncertainty distributions, this paper is concerned with a situation commonly occurring in datasets, in which the datapoints form a linear spine with Gaussian-like scatter, but additional scatter is seen in the tails of the distribution. Such a dataset still encodes meaningful age information in its spine, yet it will typically fail an `mswd` test owing to its departure from a Gaussian distribution. In this work, datasets of this nature are modelled using a contaminated Gaussian uncertainty distribution, written $c\%d\mathrm{N}$, meaning that with a probability $(100-c)\%$ the distribution involves a standard deviation, σ, but with a probability $c\%$ the distribution has a standard deviation, $d\sigma$, both with a mean of zero (see Powell et al., 2002). An example is 25%3N, with $c = 25$ and $d = 3$, so that with 25% probablity the uncertainty is drawn from $\mathrm{N}(0, 3\sigma)$, and 75% probability drawn from $\mathrm{N}(0, \sigma)$, with the $\mathrm{N}(0, s)$ notation indicating a Gaussian distribution with a mean of zero and a standard deviation of s.

2.2 A robust statistics approach to isochron calculation

We seek a statistical approach to isochron calculation that is *robust* (e.g. Huber, 1981, ch.1; Hampel *et al,*, 1986), meaning that it is not excessively affected by outliers in the data, while having desirable statistical properties, for example good efficiency (see

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

below). In addition, we require the approach to converge to YORK for a "good" dataset, one with a near-Gaussian uncertainty distribution. The approach adopted will be referred to as HUBER (Maronna et al., 2006), as originally developed in Huber (1981).

In HUBER, as in YORK, a straight line is fitted to a dataset by minimising a function of the residuals, a measure of the distance of a datapoint to the line. Since isochron data are generally bivariate with correlated analytical uncertainties in x and y, the analytical uncertainty in datapoint k can be represented as an ellipse as in Fig. 1. The residual for datapoint, k, used, denoted r_k, is the scaling factor on the size of the ellipse required to expand it until it touches the best-fit line (Fig. 1). The residual r_k is derived from the x and y uncertainties in Appendix A.

The function that is minimised to find the best-fit line can be written $\sum \rho(r_k)$ for both YORK and HUBER. Whereas in YORK, $\rho(r_k) = r_k^2$ for all r_k (equivalent to the method of least squares), HUBER minimises $\rho(r_k) = r_k^2$ near the centre of the uncertainty distribution, then downweights datapoints for which the absolute value of the residual is greater than a cut-off value, h. Formally, the HUBER algorithm minimises $\rho(r)$ where

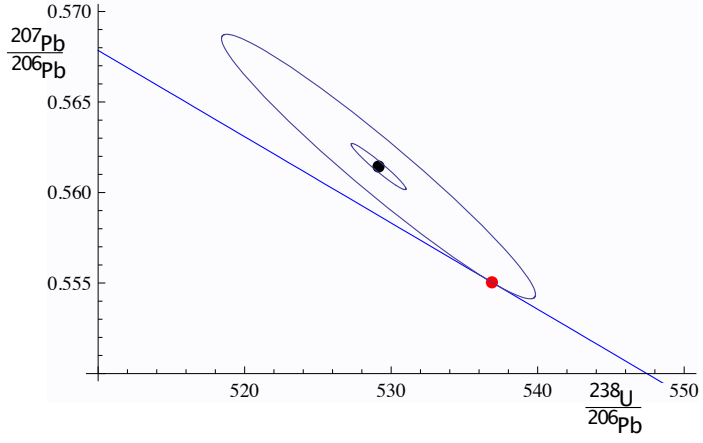

Figure 1. For an example datapoint, $\{x_k, y_k\}$, the inner ellipse is calculated with the analytical uncertainties, V_k, at the 1σ level (in black). Given a line, $y = a + bx$ (in blue), the ellipse must be drawn at the $r_k\sigma$ level (in red) to touch the line, in this case $r_k = 5.73$. The data point is $x_k = 529.14$, $y_k = 0.5614$, and $\sigma_{x_k} = 1.870$, $\sigma_{y_k} = 0.00127$ and $\rho_{x_k y_k} = -0.967$.

$$\rho(r_k) = \begin{cases} 2hr_k - h^2 & r_k < -h \\ r_k^2 & \text{if} \quad -h < r_k < h \\ 2hr_k - h^2 & r_k > h \end{cases} \tag{1}$$

as in Fig. 2. In HUBER, for smaller residuals that have an absolute value less than an adjustable constant, h, the contribution to the sum being minimised is the same as for YORK, but is linear in the residual for larger absolute value. Note that as h becomes larger and larger, HUBER converges to YORK. Although not obvious from the form of $\rho(r_k)$, HUBER is equivalent to bringing

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

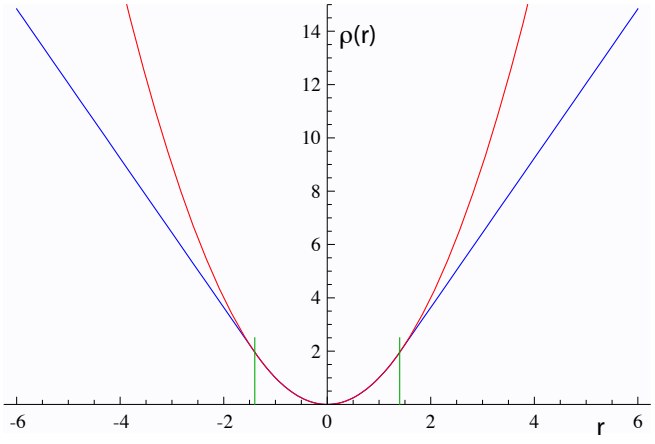

Figure 2. Plots of $\rho(r)$ against r for YORK in red (r^2), and for HUBER (eq. 1) in blue, with the two curves coincident for $|r| < h$, with $h = 1.4$ the vertical green lines. See text.

datapoints in to $\pm h$ if $|r_k| > h$, in other words truncating the residual (Maronna et al., 2006, Sect. 9.1). The value to use for h is discussed in Maronna et al. (2006), Sect. 2.2.2.

The algorithm developed in Appendix A minimises $\sum_k \rho(r_k)$ with respect to the unknown, θ, a two-element column vector, $\{a,b\}^{\mathrm{T}}$ in the line equation, $y = a + bx$. The algorithm is applicable to HUBER and also YORK. The minimisation is iterative. As a starting point, it uses an estimate for θ using least absolute deviations, L_1, as advocated by Maronna et al. (2006), Sect. 4.4.2. L_1 is a resistant estimator, meaning that it is not sensitive to extra scatter in the data. However it is much less efficient than HUBER (see below), so HUBER is a better ultimate estimator. A full iteration is envisaged in Appendix A, rather than 1-step reweighted least squares (w-form, as in Maronna et al., 2006, Sect. 9.1).

The algorithm converges in less than 5 iterations for all the simulations run (see Appendix B). Once θ is calculated, the measure of scatter used to distinguish an isochron from an errorchron can be calculated. In the case of YORK this is just `mswd`. In the case of HUBER, a robust alternative is needed, and this is developed in the next section. If an isochron is deemed to have been calculated, the uncertainty on θ, $\mathbf{V}_\theta$, can be found, as outlined in Appendix A.

2.3 Isochrons and errorchrons

In YORK, assuming that the data uncertainties are strictly Gaussian distributed, the probability distribution of `mswd` provides bounds that can be used to distinguish isochrons from errorchrons (e.g. Wendt & Carl, 1991). These bounds come from the 95% confidence interval on `mswd`. Datasets whose scatter give `mswd` outside the bounds are deemed to be errorchrons, not isochrons. Although focus is usually on `mswd` that is too large (extra scatter), `mswd` that is too small identifies the case where analytical uncertainties have been over-estimated. `Mswd` is defined as

$$\texttt{mswd} = \frac{\sum r_k^2}{n-2} \tag{2}$$

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

where the residuals, r_k, are described as in Section 2.2, depending both on the distance of datapoint k from the best-fit line, and on the datapoint's uncertainty ellipse.

If, instead, data uncertainties are $c\%d$N, with unknown c and d, then there is no equivalent of the `mswd` argument to say which datasets should give isochrons rather than errorchrons. The approach advocated here is to use a measure that reflects whether the dataset has a linear spine of "good" data within it. The measure suggested, s, coined the spine width, is robust, and is defined as

$$s = \texttt{nmad}(\mathbf{r}) \tag{3}$$

in which $\texttt{nmad}(\mathbf{r})$ is the median of the absolute values of r, normalised to be the same as the standard deviation for Gaussian-distributed r. Given that s is based on a *median*, its magnitude depends on that half of the data that have the smallest absolute values of r , in other words those that would define a spine. If the data were in fact Gaussian-distributed, it is expected that s should be about 1, given that r already involves the analytical uncertainties. The larger is s, greater than 1, the less pronounced is the linear spine in the data (or the uncertainties have been underestimated). If s, less than 1, is small it suggests that the uncertainties have been overestimated. Whereas the 95% confidence interval (95%ci) on `mswd` for Gaussian-distributed uncertainties comes from a well-established probability distribution (e.g. Wendt & Carl, 1991), the confidence interval on s needs to be found by simulation (see Appendix B). The intervals are given in this table, Table 1:

	$\sqrt{\texttt{mswd}}$	95%ci	s	95%ci
n	low	high	low	high
5	0.268	1.765	0.16	1.65
6	0.348	1.669	0.25	1.63
8	0.454	1.552	0.32	1.59
10	0.522	1.480	0.37	1.56
15	0.621	1.379	0.45	1.51
30	0.739	1.260	0.60	1.41
60	0.818	1.181	0.71	1.30

For example, for a dataset with 10 datapoints ($n = 10$), the dataset is deemed to yield an isochron if the observed s lies in the range 0.37 to 1.56. If s is outside this range the dataset gives an errorchron. For isochrons, the age uncertainty is calculated as in Appendix A. For errorchrons, the age uncertainty is not calculated.

2.4 Application of HUBER to simulated datasets

Assessing algorithms for data fitting is best done using simulated datasets. Datasets were generated by drawing data points from a range of uncertainty distributions, all centred on a linear trend reflecting an age of 4 Ma. Full details are provided in Appendix B. Two features of the datasets are varied: the number of datapoints in the dataset, and the uncertainty structure adopted, the latter via varying c and d in $c\%d$N. The algorithm is assessed in terms of its ability to retrieve the specified age of the linear trend on which the simulated datasets are built, and on the uncertainty in the age.

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

Given that the datasets investigated have fat-tailed contaminated-Gaussian uncertainty distributions, the focus is on the effect of *extra* scatter in the data, in other words, data scatter over and above what is expected for Gaussian data uncertainties. Nevertheless a small proportion of datasets do have small scatter, giving s which is below the lower bound for that number of datapoints.

The analysis below compares the results of YORK, applied only to those simulated datasets that lie within the `mswd` bounds, with the results of HUBER, applied to those datasets that lie within the spine width (s) bounds. The greatest majority of the former are included in the latter, e.g. $> 97\%$ for $n = 10$). However, HUBER typically identifies the age information in many more datasets than YORK. In the following table, $m\%$excl and $s\%$excl are the percentage of simulated datasets excluded on the basis of the `mswd` and s bounds, respectively:

n	N		5%3N		25%3N		10%10N	
	$m\%$excl	$s\%$excl	$m\%$excl	$s\%$excl	$m\%$excl	$s\%$excl	$m\%$excl	$s\%$excl
5	2.5	2.5	8.7	4.0	30.2	9.8	32.5	9.5
6	2.5	2.5	9.6	3.9	34.6	13.8	37.3	10.9
8	2.5	2.5	12.7	4.2	44.7	14.5	46.0	10.4
10	2.5	2.5	14.2	4.0	51.8	15.2	53.5	9.7
15	2.5	2.5	17.4	4.2	65.2	17.1	68.2	9.1

Note that, for example, for $n = 10$, datasets drawn from 5%3N, in fact have $100(10^{0.95}) = 59.9\%$ of the datasets having all uncertainties Gaussian, and 40.1% having at least one uncertainty drawn from 3 times Gaussian (3N). For 25%3N, 5.6% are Gaussian only, and for 10%10N, 34.9%. The leftmost columns are 2.5% by definition.

The 95% confidence interval on age using HUBER is the same or slightly less than from using YORK for all dataset sizes and uncertainty structures studied, noting that this is from a (much) larger proportion of the dataset simulations. Such a 95% confidence interval is calculated from an ordered list of the ages, with the lower limit at the 2.5% point in the list, and the upper limit at the 97.5% point.

Even if the age comparison is favourable, it might be expected that the age uncertainty suffers from the extra scatter in the data. This appears not to be the case, but there is a small degradation in the age uncertainties retrieved caused by an unavoidable efficiency loss. Efficiency relates to the number of datapoints that is required in a dataset in order to estimate the age to a given uncertainty. HUBER has optimal efficiency for all $|r_k| < h$, when it is identical to YORK, but there is an efficiency loss associated with using HUBER for an isochron-yielding dataset with any $|r_k| > h$.

Figure 3 illustrates the efficiency loss via kernel density estimate (`kde`) plots of the age uncertainties calculated for simulated datasets with $n = 10$. `Kde` plots are probability distributions akin to smoothed histograms (Wand & Jones, 1995). The red curve is the `kde` for datasets that have all $|r_k| < h$, for which efficiency is optimal. The blue curve is the `kde` for all datasets with at least one $|r_k| > h$. The efficiency loss is seen in the displacement of the blue curve to slightly higher age uncertainty than the red curve. The overall `kde`, in black, is the `kde` of all of the datasets in the red and blue `kde`, in proportion about 30% to 70%.

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

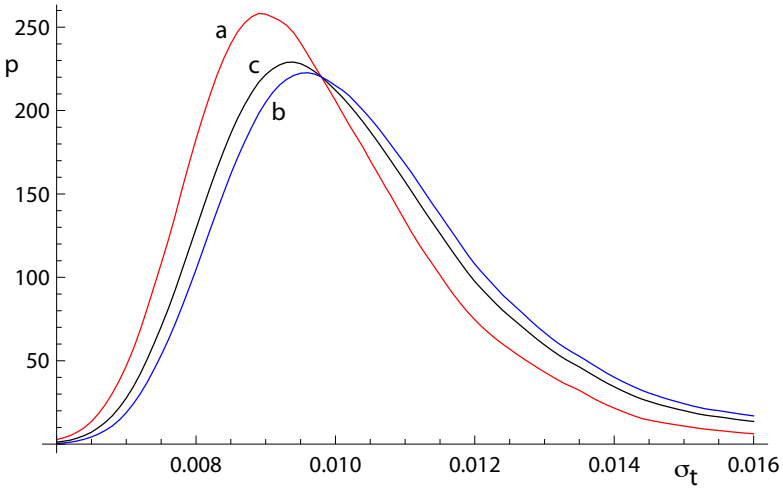

Figure 3. Kernel density estimates (`kde`) for age uncertainty calculated with HUBER on 10,000 simulated datasets with n=10 and Gaussian-distributed uncertainties. (a) Those datasets for which all $|r_k|<h$ (in red); (b) those datasets for which at least one $|r_k|>h$ (in blue), and (c) overall result combining a and b in observed proportion (in black).

The relationships shown in Figure 3 for $n = 10$ can be seen for other n in Figure 4. The pairs of red and black lines, correspond to, and have the same meaning as the red and black lines in Figure 3. As expected, the distribution of age uncertainties moves towards larger values as the sample size n decreases.

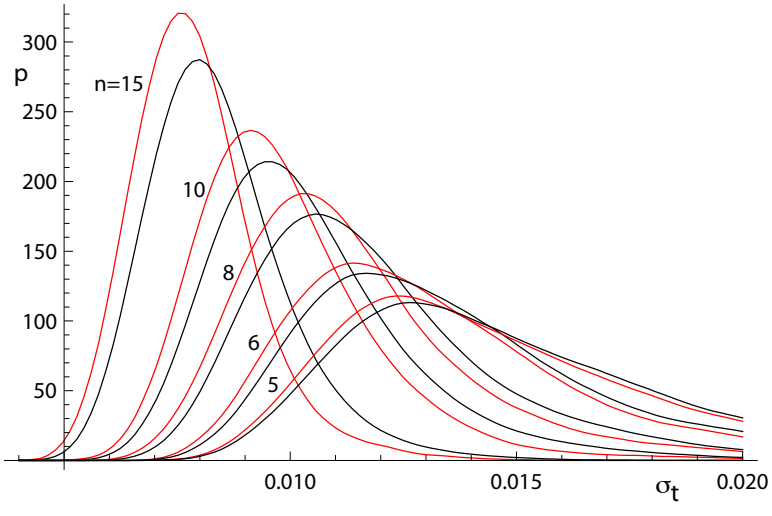

Figure 4. Kernel density estimates for age uncertainty calculated with HUBER on 10,000 simulated datasets with a range of n values and Gaussian-distributed uncertainties. In each case, the `kde` for those datasets for which all $|r_k|<h$ is in red and the overall `kde` is in black.

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

The ability of HUBER to retrieve age uncertainty information for data with uncertainties from contaminated Gaussian distributions varies with the probability and scale of the contamination, as shown in Figure 5. Not unexpectedly the more seriously contaminated distributions (25%3N and 10%10N) involve a greater displacement of the `kde` to higher age uncertainty than the more weakly contaminated 5%3N distribution. Although the displacement of the blue curves from the black curve is real, nevertheless the ability of HUBER to retrieve age uncertainties from datasets with contaminated distributions is good.

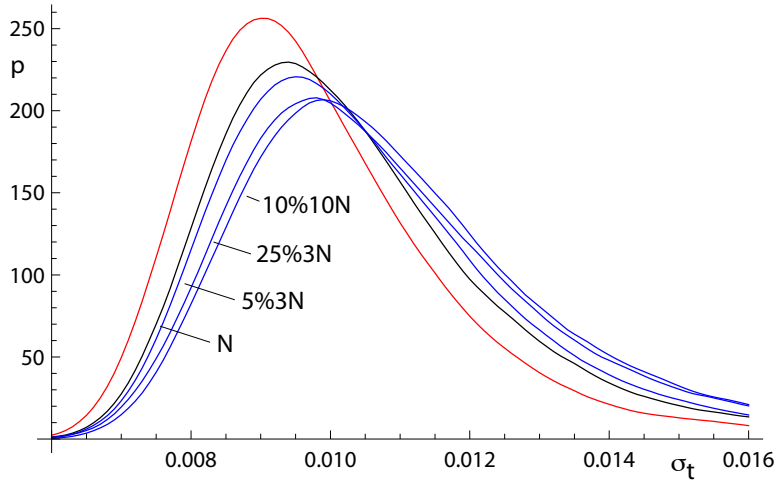

Figure 5. Kernel density estimates for age uncertainty calculated with HUBER on 10,000 simulated datasets with n=10 and several uncertainty structures. The `kde` for those datasets for which all $|r_k|<h$ is in red, the `kde` for datasets with Gaussian-distributed uncertainties is in black, and the `kde` for all of the datasets for 5%3N, 25%3N and 10%10N respectively are in blue.

2.5 Application of HUBER to a natural dataset

Sample 0708 is a carbonate flowstone from the Riversleigh World Heritage fossil site in Queensland, Australia. Isotope dilution U-Pb data for the bulk sample were previously published by Woodhead et al. (2016) providing a Model 2 isochron with an age of 13.72 ± 0.12 Ma with a `mswd` of 3.7. The new data presented here were obtained by laser ablation icpms using methods outlined in Woodhead & Petrus (2019). Such datasets are typically larger with little error correlation (round error ellipses), but with larger uncertainties than isotope dilution data. These data define an errorchron under the YORK assumptions, with `mswd` = 1.68. With HUBER, $s = 1.19$, well within the s range for an isochron. The age is 13.69 ± 0.26 Ma ($\pm$ is 2σ). The data for 0708 is plotted in Fig. 6, with 95% confidence ellipses on the datapoints.

3 Discussion

This work was motivated by the belief that many isochron datasets contain meaningful age information that cannot be identified using classical statistical methods. In such datasets, the age information is contained in a linear spine in the data, but the dataset

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

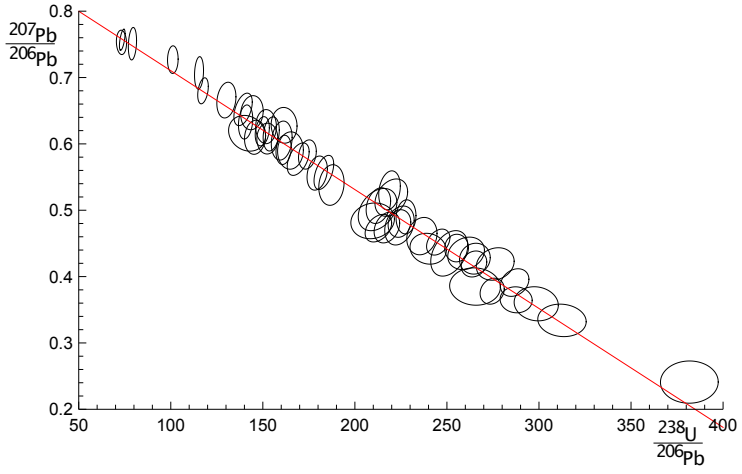

Figure 6. Laser ablation example 0708. See text.

also contains additional scatter that is inconsistent with a Gaussian uncertainty distribution. A suitably calibrated, statistically-robust method is able to identify this age information.

Contaminated Gaussian distributions provide a model for a type of dataset with extra scatter relative to a strictly Gaussian-distributed one. The robust isochron method presented in this work can however be applied *in general* to data which is Gaussian-distributed only in the central spine of the uncertainty distribution, with non-Gaussian scatter occurring in the tails, arising from analytical or geological uncertainty.

In most robust statistics data fitting approaches, the formal uncertainties output during data measurement are ignored. Instead, the scale used in the data fitting is derived from the scatter in the data themselves, an approach used by Powell et al. (2002). The approach advocated here *does* include the data measurement uncertainties, and this allows the results to converge on those of the YORK method when the data have little excess scatter.

Appendix A: HUBER algorithm

The HUBER algorithm involves minimising $\sum_k \rho(r_k)$ with respect to the unknown, θ, a two-element column vector, $\{a, b\}^{\mathrm{T}}$ in the line equation, $y = a + bx$, in order to fit the data. The residual, r_k on datapoint k and the function ρ are defined in the following below. The HUBER algorithm subsumes YORK.

Writing the kth datapoint as $\{x_k, y_k\}$, generally the isotopic data used in age calculations involve uncertainties in both x_k and y_k, and commonly the x_k and y_k are also correlated. These can be represented by a covariance matrix, V_k,

$$V_k = \begin{bmatrix} \sigma_{x_k}^2 & \sigma_{x_k}\sigma_{y_k}\rho_{x_k y_k} \\ \sigma_{x_k}\sigma_{y_k}\rho_{x_k y_k} & \sigma_{y_k}^2 \end{bmatrix} \tag{A1}$$

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

in which σ_{x_k} is the standard deviation on x_k, σ_{y_k} the standard deviation on y_k, $\sigma_{x_k}\sigma_{y_k}\rho_{x_k y_k}$ the covariance between x_k and y_k, and $\rho_{x_k y_k}$ the correlation coefficient between x_k and y_k. The covariance matrix can be represented by an ellipse around the data point in an x–y diagram, as illustrated in Fig. 1. The residual, r_k, a measure of the distance of the point $\{x_k, y_k\}$ to the line, is calculated from the coordinates of the data points, $\{x_k, y_k\}$, and their uncertainties in V_k, by

$$r_k = \frac{e_k}{\sigma_{e_k}} \tag{A2}$$

in which e_k is the distance of the datapoint from the line, $e_k = a + b x_k - y_k$, and σ_{e_k} is the standard deviation on e_k. The standard deviation, σ_{e_k}, is calculated by error propagation using V_k:

$$\sigma_{e_k}^2 = \left\{ \frac{\partial e_k}{\partial x_k}, \frac{\partial e_k}{\partial y_k} \right\} V_k \left\{ \frac{\partial e_k}{\partial x_k}, \frac{\partial e_k}{\partial y_k} \right\}^{\mathrm{T}} = b^2 \sigma_{x_k}^2 + \sigma_{y_k}^2 - 2 b \sigma_{x_k} \sigma_{y_k} \rho_{x_k y_k} \tag{A3}$$

with the term in curly brackets evaluating to $\{b, -1\}$. The residual is then

$$r_k = \frac{e_k}{\sigma_{e_k}} = \frac{a + b x_k - y_k}{\sqrt{b^2 \sigma_{x_k}^2 + \sigma_{y_k}^2 - 2 b \sigma_{x_k} \sigma_{y_k} \rho_{x_k y_k}}} \tag{A4}$$

The minimisation of $\sum_k \rho(r_k)$ in the HUBER algorithm is iterative, starting from a resistant estimate of the line, for example using least absolute deviations, L_1, as advocated by Maronna et al. (2006). At each iteration, the algorithm provides an update of θ, $\Delta\theta$, so that at the ith iteration, $\theta_i = \theta_{i-1} + \Delta\theta$.

The minimisation of $\sum \rho(r_k)$ is undertaken using the fact that, at the minimum, the derivative of $\sum \rho(r_k)$ with respect to θ is zero. Defining

$$2\psi(r_k) = \frac{\partial \rho(r_k)}{\partial r_k} \tag{A5}$$

this quantity, for HUBER, is

$$\psi(r_k) = \begin{cases} -h & \\ r_k & \text{if} \quad -h < r_k < h \\ h & \end{cases} \begin{array}{l} r_k < -h \\ \\ r_k > h \end{array}$$

For YORK, $\psi(r_k) = r_k$, equivalent to HUBER with large h. Then, at the minimum

$$\sum_k \frac{\partial \rho(r_k)}{\partial \theta} = 0 = \sum_k \left(\frac{\partial \rho(r_k)}{\partial r_k} \right) \left(\frac{\partial r_k}{\partial \theta} \right) = \sum_k \psi(r_k) \left(\frac{\partial r_k}{\partial \theta} \right) \tag{A6}$$

Defining the kth row of a matrix $\mathbf{C}$, C_k, to be the derivative of r_k with respect to θ then

$$C_k = \frac{\partial r_k}{\partial \theta} = \frac{1}{\sigma_{e_k}} \frac{\partial e_k}{\partial \theta} - r_k \frac{\partial \sigma_{e_k}}{\partial \theta} = B_k - r_k \frac{\partial \sigma_{e_k}}{\partial \theta} \tag{A7}$$

with B_k the kth row of $\mathbf{B}$, given by $1/\sigma_{e_k}\, \partial e_k / \partial \theta$, then at the minimum

$$\sum_k \psi(r_k) \frac{\partial r_k}{\partial \theta} = \sum_k \psi(r_k)\, C_k = 0 \tag{A8}$$

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

or in matrix form, $\mathbf{C}^{\mathrm{T}}\psi(\mathbf{r}) = 0$, in which $\psi(\mathbf{r})$ is a column vector whose kth element is $\psi(r_k)$. This constitutes two non-linear equations requiring iteration to solve.

Now, at iteration i, progressing towards the minimum

$$\psi_i(r_k)|_{|r_k|<h} = B_k(\theta_{i-1} + \Delta\theta) - \frac{y_k}{\sigma_{e_k}} = \psi_{i-1}(r_k) + B_k\,\Delta\theta \tag{A9}$$

and $\psi_i(r_k)|_{|r_k|>h} = \psi_{i-1}(r_k)$ otherwise. This can be written

$$\psi_i(r_k) = \psi_{i-1}(r_k) + \dot{\psi}_{i-1}(r_k)\,B_k\,\Delta\theta \tag{A10}$$

in which $\dot{\psi}(r_k) = \partial\psi(r_k)/\partial r_k$. So, for HUBER, $\dot{\psi}(r_k) = 1$ for $|r_k| < h$, and $\dot{\psi}(r_k) = 0$ otherwise. Substituting (A10) into (A8) gives

$$\sum(\psi_{i-1}(r_k) + \dot{\psi}_{i-1}(r_k)B_k\Delta\theta)\,C_k = 0 \tag{A11}$$

or, in matrix form, dropping iteration subscripts

$$\mathbf{C}^{\mathrm{T}}(\mathbf{I}'\,\mathbf{B}\,\Delta\theta + \psi(\mathbf{r})) = 0 \tag{A12}$$

with $\mathbf{I}' = \mathrm{diag}(\dot{\psi}(\mathbf{r}))$ a modified identity matrix with its kkth element equal to $\dot{\psi}(r_k)$. Equation (A12) can then be rearranged to give $\Delta\theta$ at the current iteration

$$\Delta\theta = -(\mathbf{C}^{\mathrm{T}}\mathbf{I}'\,\mathbf{B})^{-1}\mathbf{C}^{\mathrm{T}}\psi(\mathbf{r}) \tag{A13}$$

The iteration works because the changes in $\mathbf{B}$ and $\mathbf{C}$ between iterations are small, particularly when a good starting guess is used at the beginning of the iterations. This is the iteration implemented in the python code.

Accepting that an isochron has been calculated, the covariance matrix of θ, $\mathbf{V}_\theta$, can be calculated by error propagation of $\mathbf{r}$ to θ

$$\mathbf{V}_\theta = \left(\frac{\partial\theta}{\partial\mathbf{r}}\right)\mathbf{V}_\mathbf{r}\left(\frac{\partial\theta}{\partial\mathbf{r}}\right)^{\mathrm{T}} \tag{A14}$$

assuming that θ is approximately linear in r_k around the minimum in $\sum\rho(r_k)$. Then, using (A14) with (A13)

$$\mathbf{V}_\theta = \left(\mathbf{C}^{\mathrm{T}}\mathbf{I}'\,\mathbf{B}\right)^{-1}\mathbf{C}^{\mathrm{T}}\,\mathbf{I}'\,\mathbf{V}_\mathbf{r}\,\mathbf{I}'\,\mathbf{C}\left(\mathbf{C}^{\mathrm{T}}\mathbf{I}'\,\mathbf{B}\right)^{-\mathrm{T}} \tag{A15}$$

If it is assumed that the uncertainty on a datapoint has the form $c\%d\mathrm{N}$, with unknown c and d, then $\mathbf{V}_r$ is not specified. However those residuals with $|r_k| > h$ are likely to be those where $d > 1$, but these residuals are the ones with $\dot{\psi}(r_k) = 0$. Therefore a good approximation involves taking $\mathbf{V}_r = \mathbf{I}'$. This is identically true in the case of YORK, when $\mathbf{V}_r = \mathbf{I}$ as then $\mathbf{I}' = \mathbf{I}$. Then (A15) becomes

$$\mathbf{V}_\theta = \left(\mathbf{C}^{\mathrm{T}}\mathbf{I}'\,\mathbf{B}\right)^{-1}\mathbf{C}^{\mathrm{T}}\,\mathbf{I}'\,\mathbf{C}\left(\mathbf{C}^{\mathrm{T}}\mathbf{I}'\,\mathbf{B}\right)^{-\mathrm{T}} \tag{A16}$$

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

In YORK (or if all $|r_k| < h$ in HUBER), then $\mathbf{I}' = \mathbf{I}$ and $\psi(\mathbf{r}) = \mathbf{r}$. So

$$\Delta\theta = -(\mathbf{C}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{r} \tag{A17}$$

$$\mathbf{V}_\theta = (\mathbf{C}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{B})^{-\mathrm{T}} \tag{A18}$$

If, in addition, all $\sigma_{x_k} = 0$ then $\mathbf{C} = \mathbf{B}$ and, as in this case, $\mathbf{B}$ does not depend on θ, iteration is not involved, $\mathbf{r}$ is replaced by $-\mathbf{y}$, and

$$\theta = (\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{y} \tag{A19}$$

$$\mathbf{V}_\theta = (\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1} \tag{A20}$$

These are the results for fitting data by simple weighted least squares.

Appendix B: Simulation setup

This work was originally motivated by the dating of speleothems using the lower intercept with a U-Pb Concordia in Tera-Wasserburg style plots (e.g. Woodhead et al. 2012). This paper therefore discusses $\{x, y\}$ data with the expectation that $x = {}^{238}\mathrm{U}/{}^{206}\mathrm{Pb}$ and $y = {}^{207}\mathrm{Pb}/{}^{206}\mathrm{Pb}$, but the logic and the algorithm are in no way restricted to this system.

10,000 simulated datasets, each containing 5, 6, 8, 10 and 15 datapoints, respectively, were used to assess the HUBER algorithm. Each dataset corresponds to an age of 4 Ma, with an underlying trend chosen to be $y = 0.811 - 0.000474737x$. For each dataset, the x-values were drawn from a uniform probability distribution with bounds, $\{400, 1100\}$ (so the x are not equi-spaced). Datapoints are assigned uncertainty with $\sigma_{x_k} = 0$ and a fixed $\sigma_{y_k} = 0.00125$, the latter representing the analytical uncertainty, propagated from both the x and y measurement into y. In a real $\{{}^{238}\mathrm{U}/{}^{206}\mathrm{Pb}, {}^{207}\mathrm{Pb}/{}^{206}\mathrm{Pb}\}$ dataset, σ_{x_k} and σ_{y_k} would be finite and correlated. However, this makes no difference to the calculations once data is processed into r_k form as in Fig. 1. For a given dataset, scatter is introduced into the data by drawing the y values from an uncertainty distribution, centred on the underlying trend, that may be either Gaussian (N) or one of three contaminated Gaussian distributions—5%3N, 25%5N, or 10%10N—as in Powell et al. (2002). For $n = 10$ and gaussian-distributed uncertainties, the age uncertainty obtained is approximately $\sigma_t = 0.01$ Ma.

Results are presented in terms of kernel density estimates using an Epanetchnikov kernel (Wand & Jones, 1995). Kernel density estimates (`kde`) are a way of presenting data that could otherwise be plotted as a histogram, normally normalised so that—like a probability distribution—the area under the `kde` curve is 1. The smoothness of the `kde` is controlled by a smoothing constant whose value was chosen to be just large enough for the `kde` to appear smooth, given that 10,000 datasets are used in each `kde`.

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

Appendix: HUBER code

```
import sys
import datetime
import numpy as np

out = open("out.txt", "w") # opening output file
screen = sys.stdout
standard = [screen, out]   # default where print goes

defaulth = 1.4             # default h in huber

def nmad(e):
    return 1.4826 * np.median(np.absolute(e))

def lsq(data):
    X = [[1, xk] for xk in data[:,0]]
    Y = data[:,2]
    inv = np.linalg.inv(np.dot(np.transpose(X), X))
    theta = np.dot(inv, np.dot(np.transpose(X), Y))
    e = np.dot(X, theta) - Y
    sigfit2 = np.dot(e, e)/(e.shape[0] - 2)
    return (theta, sigfit2 * inv)

def lad(data): # sadovski (1974)
    n = data.shape[0]
    (x, sdx, y, sdy, cor) = np.transpose(data)
    rr = 1e-8 * np.random.random(n-1) # used for naive breaking of x ties
    bi = np.empty(n); bi.fill(False)
    k = 0; i = 0; i1 = 0; i2 = 0;
    while i != -1 and k < 12:
        i2 = i1; i1 = i; k += 1
        o = np.delete(np.arange(n), i);
        x1 = np.delete(x - x[i], i) + rr
        y1 = np.delete(y - y[i], i)
        oo = np.argsort(y1/x1)
        x2 = np.abs(x1[oo]);  mid = sum(x2)/2
        sx = 0; j = 0
        while sx < mid: sx += x2[j]; j += 1
        i = o[oo][j-1]
        if i == i2: i = -1
        elif bi[i]: i = -1
        else: bi[i] = True
    return np.array(((x[i1] * y[i2] - x[i2] * y[i1])/(x[i1] - x[i2]),
```

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

GEOCHRONOLOGY Discussions Open Access EGU

```
                    (y[i1] - y[i2])/(x[i1] - x[i2])))

def calcage(theta, covtheta = None, method = 1):
# function to be added for isotope system of interest

def rhohub(r, h = 1.4):
    v = [rk**2 if abs(rk) < h else 2 * h * abs(rk) - h **2 for rk in r]
    return np.array(v)

def psihub(r, h = 1.4):
    v = [rk if abs(rk) < h else np.sign(rk) * h for rk in r]
    return np.array(v)

def dpsihub(r, h = 1.4):
    v = [1 if abs(rk) < h else 0 for rk in r]
    return np.array(v)

def sumrho(data, theta, h = 1.4):
    (a, b) = theta
    (x, sdx, y, sdy, cor) = np.transpose(data)
    e = a + b * x - y
    sde = np.sqrt(sdy**2 + b**2 * sdx**2 - 2*b*cor*sdx*sdy)
    r = e / sde
    return np.sum(rhohub(r, h))

def sumpsi2(data, theta, h = 1.4):
    (a, b) = theta
    (x, sdx, y, sdy, cor) = np.transpose(data)
    e = a + b * x - y
    sde = np.sqrt(sdy**2 + b**2 * sdx**2 - 2*b*cor*sdx*sdy)
    r = e / sde
    c = np.transpose([1/sde, (x - r * (b*sdx**2 - cor*sdx*sdy)/sde)/sde])
    pc = np.dot(np.transpose(c), psihub(r, h))
    return np.sqrt(np.dot(pc, pc))

def halving(data, theta, deltheta, m, h = 1.4):
# naive interval halving to get a better steplength in huber
# assumes step will be good for step < 1
    kmax = 16
    step1 = 0
    s1 = sumrho(data, theta, h)
    step2 = 1
    s2 = sumrho(data, theta + step2 * deltheta, h)
    k = 1
```

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

GEOCHRONOLOGY Discussions Open Access EGU

```
    while (k < m or step1 < 1e-10) and k < kmax:
        if s1 > s2:
            step1 = (step1 + step2)/2
            s1 = sumrho(data, theta + step1 * deltheta, h)
        else:
            step2 = (step1 + step2)/2;
            s2 = sumrho(data, theta + step2 * deltheta, h)
        (steps, ss) = (step2, s2) if s1 > s2 else (step1, s1);
        k += 1
    return (steps, ss)

def huber(data0, h = 1.4):
#   huber line-fitter
    n = data0.shape[0]
    itmax = 12; minsump = 1e-5; mindel = 1e-8; mincond = 1e-12
    code = 0;

    (x, sdx, y, sdy, cor) = np.transpose(data0)
    avx = np.dot(x, np.ones(n))/n;  avy = np.dot(y, np.ones(n))/n;
    div = np.array([1/avy, avx/avy])

    data = np.copy(data0)
    (x, sdx, y, sdy, cor) = np.transpose(data)
    x /= avx; sdx /= avx;  y /= avy; sdy /= avy

    th0 = (lad(data), lsq(data)[0])
    sr = np.array([sumrho(data, thetak) for thetak in th0])
    theta = th0[np.argsort(sr)[0]]

    sump = 1e10; step = 1; deltheta = (1e10, 1e10)
    k = 0; bb = 0
    while k < itmax and (sump > minsump or \
                         np.sqrt(np.dot(deltheta,deltheta)) > mindel):
        k += 1
        (a, b) = theta
        e = a + b * x - y
        sde = np.sqrt(b**2*sdx**2 - 2*b*cor*sdx*sdy + sdy**2)
        r = e/sde

        sum = sumrho(data, theta, h)

        c = np.transpose([1/sde, (x - r * (b*sdx**2 - cor*sdx*sdy)/sde)/sde])
        rs = psihub(r, h)
        pc = np.dot(np.transpose(c), rs)
```

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020


```
        sump = np.sqrt(np.dot(pc, pc)) # same as given by sumpsi2

        drs = dpsihub(r, h)
        d = np.transpose([drs/sde, drs * x/sde])

        cd = np.dot(np.transpose(c),d)
        (uu, sv, vv) = np.linalg.svd(cd)
        if mincond * sv[0] > sv[1]: code = 2; break

        inv = np.dot(np.dot(np.transpose(vv), np.diag(1/sv)), np.transpose(uu))
        bb = np.dot(inv, np.transpose(c))

        deltheta = np.dot(bb, -rs)

        sum1 = sumrho(data, theta + deltheta, h);
        (step, sum1) = (1, sum1) if sum1 < 1.01 * sum else \
                                    halving(data, theta, deltheta, 4, h)
        if step == 0: code = 3; break

        theta += step * deltheta

    sump = sumpsi2(data, theta, h)

    if step == 0 and sump < np.sqrt(minsump):
        code = 0 # not fully converged, but nearly good: ok?

    if sump > 10: code = 1

    rbb = np.dot(bb, np.diag(drs))

    theta /= div
    covtheta = np.dot(rbb, np.transpose(rbb)) / \
        np.array([[div[0]**2, div[0] * div[1]], [div[0] * div[1],div[1]**2]])

    return (code, theta, covtheta, sump, k, sv[1])

def recipe(title, data, where = [screen]):
#   simple calculation driver
    h = defaulth
    (x, sdx, y, sdy, cor) = np.transpose(data)
    n = data.shape[0]

    today = datetime.datetime.now();
    pr("============================================================\n"+ \
```

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020


```
        "running huber.py on "+today.ctime())

    res = huber(data)
    if res[0] != 0: return res(0) # exit if not converged
    (a, b) = theta = res[1]
    ucovtheta = res[2]
    (age, sdage) = calcage(theta, ucovtheta)

    e = a + b * x - y
    sde = np.sqrt(b**2 * sdx**2 - 2 * b * sdx * sdy * cor + sdy**2)
    s = nmad(e/sde); slim = 2 - 0.17 * np.log(3 + n)
    iso = ": isochron " if s < slim else ": errorchron "

    pr(("sample "+title+": s = %0.2f"+ iso + "age = %0.3f +/- %0.3f Ma") % \
       (s, age, 1.96 * sdage), printto = where)

    return [0, age, sdage, theta]

def pr(s, e="\n", printto=standard):
# prints a string
    for pr in printto: print(s, end=e, file=pr)

def pra(x, f, s="", e="\n", printto=standard):
# prints an array
    for pr in printto:
        print(s, end='', file=pr)
        for xk in np.array(x).flatten():
            print(f % xk, end='', file=pr)
        print(e, file=pr)

# data rows:  x  sdx  y  sdy  cor
data2 = np.loadtxt("data0708.txt", delimiter=",")
recipe("0708", data2, where=standard)
```

Example output, running on the command line

```
running huber.py on Tue Sep  3 09:47:51 2019
sample 0708: s = 1.19: isochron age = 13.685 +/- 0.260 Ma
```

Author contributions. Roger Powell created the algorithm and coded the Python script; Eleanor Green helped validate the maths/statistics and write the paper; Tephy Marillo Sialer helped with the simulations; and Jon Woodhead oversaw the applicability of the approach.

Competing interests. The authors declare that they have no conflict of interest.

https://doi.org/10.5194/gchron-2020-4
Preprint. Discussion started: 14 February 2020

References

Hampel, F.R., Rousseeuw, P.J., Ronchetti, E.M., and Stahel, W.A.: Robust statistics. Wiley and Sons, New York, 502pp, 1986.

Huber, P.J.: Robust Statistics, John Wiley and Sons, Inc., New York: 305pp, 1981.

Maronna, R.A., Martin, D., and Yohai, V.J.: Robust statistics. John Wiley and Sons, Chichester. 403pp, 2006.

McLean, N.M.: Straight line regression through data with correlated uncertainties in two or more dimensions. Geochimica et Cosmochimica Acta, 124, 237–249, 2014.

Powell, R., Woodhead, J, and Hergt, J.: Improving isochron calculations with robust statistics and the bootstrap. Chemical Geology 185, 191–204, 2002.

Wand, M.P. and Jones, M.C.: Kernel smoothing. Chapman and Hall, London. 212pp, 1995.

Wendt, I., and Carl, C.: The statistical distribution of the mean squared weighted deviation. Chemical Geology (Isotope Geosciences Section) 86, 275–285, 1991.

Woodhead, J., Hand, S.J., Archer, M., Graham, I., Sniderman, K., Arena, D.A., Black, K.H., Godhelp, H., and Price, E.: Developing a radiometrically-dated chronologic sequence for Neogene biotic change in Australia, from the Riversleigh World Heritage Area of Queensland. Gondwana Research 29, 153–167, 2016.

Woodhead, J., Hellstrom, J., Pickering, R., Drysdale, R., Paul, B., and Bajo, P.: U and Pb variability in older speleothems and strategies for their chronology. Quaternary Geochronology 14, 105–113, 2012.

Woodhead, J. and Petrus, J., Exploring the advantages and limitations of in situ U-Pb carbonate geochronology using speleothems, Geochronology 1, 67–84,
https://doi.org/10.5194/gchron-1-69-2019.

York, D.: Least squares fitting of a straight line with correlated errors. Earth and Planetary Science Letters 5, 320–324, 1969.

York, D., Evensen, N.M., Martinez, M.L., and Delgado, J.D.: Unified equations for the slope, intercept, and standard errors of the best straight line. American Journal of Physics 72, 367–375, 2004.