investr: An R Package for Inverse Estimation

by Brandon M. Greenwell and Christine M. Schubert Kabban

Abstract Inverse estimation is a classical and well-known problem in regression. In simple terms, it involves the use of an observed value of the response to make inference on the corresponding unknown value of the explanatory variable. To our knowledge, however, statistical software is somewhat lacking the capabilities for analyzing these types of problems. In this paper[1], we introduce **investr** (which stands for **inv**erse **est**imation in **R**), a package for solving inverse estimation problems in both linear and nonlinear regression models.

Introduction

Consider the regression model $\mathcal{Y}_i = f(x_i; \boldsymbol{\beta}) + \epsilon_i \ (i = 1, \ldots, n)$, where f is a known expectation function (called a *calibration curve*) that is monotonic over the range of interest and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. A common problem in regression is to predict a future response $\mathcal{Y}_0$ from a known value of the explanatory variable x_0. Often, however, there is a need to do the reverse; that is, given an observed value of the response $\mathcal{Y} = y_0$, estimate the unknown value of the explanatory variable x_0. This is known as the *calibration problem*, though we refer to it more generally as inverse estimation. In this paper, we consider only *controlled calibration*, where the values of the explanatory variables are fixed by design. A more thorough overview of the calibration problem, including Bayesian approaches and multivariate calibration, is given in Osborne (1991).

There are three main functions available in the **investr** package:

- `calibrate`;
- `invest`;
- `plotFit`.

`calibrate` operates on objects of class `lm` and can only be used when the expectation function has the form $f(x_i; \boldsymbol{\beta}) = \beta_0 + \beta_1 x_i$ (i.e., the simple linear regression model), where closed-form solutions are available for calculating confidence intervals for x_0. For more complicated models (e.g., polynomial and nonlinear regression), closed-form expressions are usually not available and iterative techniques will be required. This is the purpose of the function `invest`, which calculates the point estimate and confidence intervals for x_0 by calling the function `uniroot` from the **stats** package. The function `plotFit` produces a scatterplot of the data with fitted regression curve and the option to add confidence/prediction bands for the response (pointwise or adjusted). It can be used with single-predictor objects of class `lm` or `nls`; however, for objects of class `nls`, confidence/prediction bands are based on the linear approximation and can be misleading (Bates and Watts, 1988, pg. 65). The development version of **investr** can be found on GitHub at `https://github.com/w108bmg/investr`. To report bugs or issues, contact the main author or submit them to `https://github.com/w108bmg/investr/issues`. and can easily be installed using the the **devtools** package (Wickham and Chang, 2013):

```
# Install development version from GitHub
install.packages("devtools")
devtools::install_github(repo = "bgreenwell/investr")
```

Calibration for straight line regression

Consider the most common calibration model, $\mathcal{Y}_i = \beta_0 + \beta_1 x_i + \epsilon_i \ (i = 1, \ldots, n)$, where x_i is fixed and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. Suppose we obtain a series of m observations $y_{01}, \ldots, y_{0m}$ which are associated with the same (unknown) x_0. It can be shown (Graybill, 1976) that the maximum likelihood (ML) estimator of x_0, also known as the *classical estimator*, is

$$\widehat{x}_0 = \frac{\bar{y}_0 - \widehat{\beta}_0}{\widehat{\beta}_1}, \tag{1}$$

[1]The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Navy, Department of Defense, or the U.S. Government.

where $\widehat{\beta}_0$ and $\widehat{\beta}_1$ are the usual ML estimators of β_0 and β_1, respectively, and $\bar{y}_0 = \sum_{j=1}^{m} y_{0j}/m$. The classical estimator in general is computed as $\widehat{x}_0 = f^{-1}\left(\bar{y}_0; \widehat{\boldsymbol{\beta}}\right)$; that is, by inverting the fitted calibration curve at $\bar{y}_0$ (Eisenhart, 1939). Since $\widehat{x}_0$ is a ratio of jointly normal random variables, it is not surprising to learn that it does not have any finite moments (think of a standard Cauchy distribution). The sampling distribution of $\widehat{x}_0$ is complicated, but fortunately, not required for setting an exact $100(1-\alpha)\%$ confidence interval on x_0. This is discussed in the next section.

Inversion interval

It can be shown (see, for example, Draper and Smith (1998, pp. 47-51)) that an exact $100(1-\alpha)\%$ confidence interval for x_0 is given by

$$\widehat{x}_0 + \frac{(\widehat{x}_0 - \bar{x})\, g \pm \left(t\widehat{\sigma}/\widehat{\beta}_1\right)\sqrt{(\widehat{x}_0 - \bar{x})^2 / S_{xx} + (1-g)\left(\frac{1}{m} + \frac{1}{n}\right)}}{1-g}, \tag{2}$$

where $S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2$, $g = \left(t^2\widehat{\sigma}^2\right) / \left(\widehat{\beta}_1^2 S_{xx}\right)$ and $t = t_{\alpha/2,n+m-3}$ is the upper $1-\alpha$ percentile of a Student's t-distribution with $n+m-3$ degrees of freedom. The inversion interval (2) can also be obtained using a fiducial argument, as in Fieller (1954). For the special case $m = 1$, the inversion interval is equivalent to inverting a $100(1-\alpha)\%$ prediction interval for the response. In other words, if one draws a horizontal line through a scatterplot of the data at y_0, then the abscissas of its intersection with the usual (pointwise) prediction band for f correspond to the endpoints of the inversion interval (2). This interval should only be used when the usual $\mathcal{F}$ test for testing $\mathcal{H}_0 : \beta_1 = 0$ versus $\mathcal{H}_1 : \beta_1 \neq 0$ can be rejected at the specified α level; in other words, when the regression line is not "too" flat. If $\mathcal{H}_0$ is not rejected, then such a confidence interval for x_0 may result in either the entire real line or two semi-infinite intervals (Graybill and Iyer, 1994, p. 429) — see Figure 2. The `plotFit` function in the **investr** package can be used for drawing scatterplots with the fitted model and confidence/prediction bands. To calculate the inversion interval for the linear calibration problem, we use the `calibrate` function and specify the option `interval = "inversion"` (the default) as in the following example.

The data frame `arsenic` contains the true amounts of arsenic present in 32 water samples (Graybill and Iyer, 1994). Also present, is the amount of arsenic measured by some field test, which is subject to error. A new water sample is obtained and subjected the field test producing a reading of 3.0 μg/ml. It is desired to infer the true amount of arsenic present in the sample. The following code fits a simple linear regression model to the arsenic data and then uses the `calibrate` function to compute the ML estimate and corresponding 90% calibration interval based on Equation (2).

```
library(investr)
mod <- lm(measured ~ actual, data = arsenic)
(res <- calibrate(mod, y0 = 3, interval = "inversion", level = 0.9))

## estimate    lower    upper
## 2.931449 2.603537 3.258658

# Figure 1
plotFit(mod, interval = "prediction", level = 0.9, shade = T,
        col.pred = "skyblue")
abline(h = 3, v = c(res$lower, res$estimate, res$upper), lty = 2)
```

Running the above block of code produces Figure 1 and the following output to the R console:

```
  estimate    lower    upper
    2.9314   2.6035   3.2587
```

where `estimate` is the ML estimate (1), and `lower`/`upper` correspond to the lower/upper bounds of the 90% inversion interval for x_0 (2). If instead the new water sample was subjected to the field test three times, thereby producing three response values corresponding to x_0, say 3.17, 3.09, and 3.16 μg/ml, we would simply supply `calibrate` with a vector of these values as in

```
calibrate(mod, y0 = c(3.17, 3.09, 3.16), interval = "inversion", level = 0.9)

## estimate    lower    upper
## 3.073191 2.884300 3.261588
```

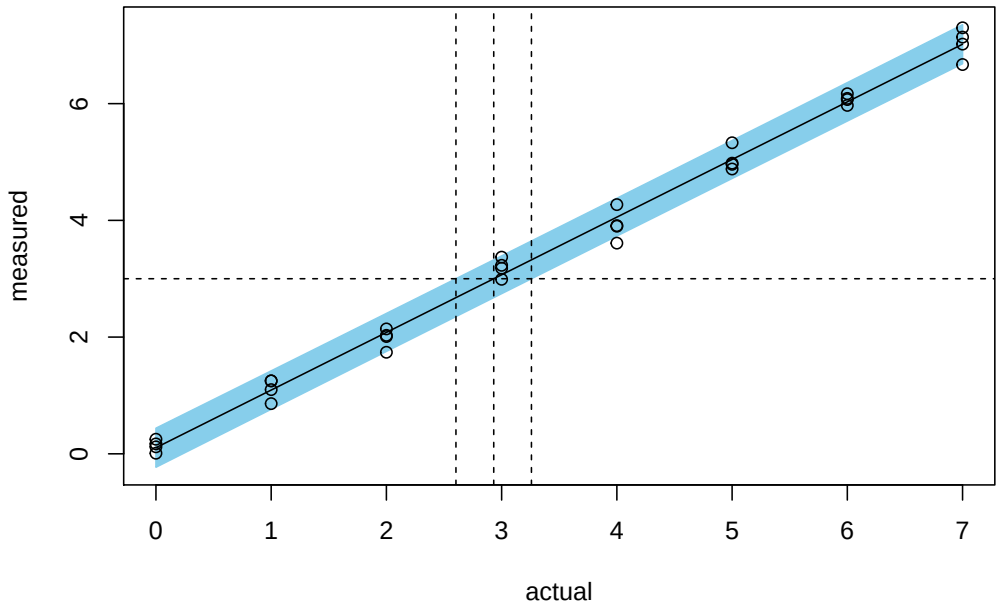

Figure 1: Scatterplot of the arsenic data with fitted calibration line and pointwise prediction band. A horizontal reference line is drawn through the observed value $y_0 = 3$. The vertical lines identify the position of the point estimate and 90% confidence bounds for x_0.

If `interval = "inversion"`, and the slope of the model is not ignificant at the specified α level, then finite confidence limits for x_0 will not be produced. For example, suppose `badfit` is an `lm` object for which the slope is not significant at the $\alpha = 0.1$ level. Then, as illustrated in Figure 2,

```
calibrate(badfit, y0 = 10, level = 0.9)
```

will either produce two semi-infinite intervals, e.g.,

```
 Error: The calibration line is not well determined.
Returning two semi-infinite intervals:
( -Inf , -282.0006 ) and ( 393.1267 , Inf )
```

or the entire real line, e.g.,

```
estimate    lower    upper
-97.5987     -Inf      Inf
Warning message:
The calibration line is not well determined.
```

Wald interval

Another common approach to computing calibration intervals is to use the delta method (Dorfman, 1938). It is easy to show that an approximate standard error for the ML estimator (1), based on a first-order Taylor series approximation, is given by

$$\widehat{\text{se}}\,(\widehat{x}_0) = \frac{\widehat{\sigma}}{\left|\widehat{\beta}_1\right|}\sqrt{\frac{1}{m} + \frac{1}{n} + \frac{(\widehat{x}_0 - \bar{x})^2}{S_{xx}}}. \tag{3}$$

Assuming large sample normality for $\widehat{x}_0$ leads to an approximate $100(1-\alpha)\%$ Wald confidence interval for x_0 of

$$\widehat{x}_0 \pm t_{\alpha/2,n+m-3}\frac{\widehat{\sigma}}{\left|\widehat{\beta}_1\right|}\sqrt{\frac{1}{m} + \frac{1}{n} + \frac{(\widehat{x}_0 - \bar{x})^2}{S_{xx}}}. \tag{4}$$

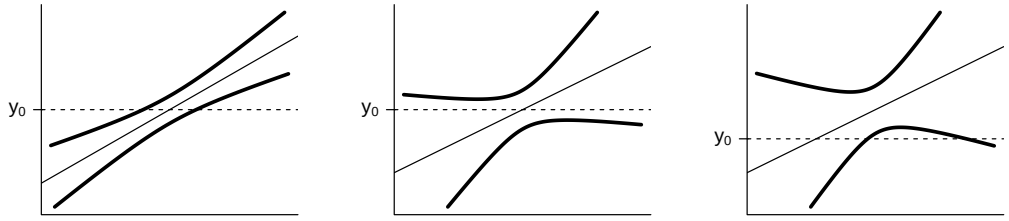

Figure 2: Hypothetical $1-\alpha$ (pointwise) prediction bands. *Left*: Horizontal line at y_0 intersects the prediction band at two points, resulting in a finite interval. extitMiddle: Horizontal line at y_0 does not intersect the prediction band at all resulting in an infinite interval. *Right*: Horizontal line at y_0 only intersects with one side of the prediction band resulting in two semi-infinite intervals.

This is equivalent to taking $g = 0$ in the inversion interval (2). Unlike the inversion interval, though, the Wald interval always exists and is symmetric about $\widehat{x}_0$. This symmetry is attractive, but not always realistic, such as when the calibration curve f is nonlinear and has horizontal asymptotes. To obtain a Wald-type interval we specify `interval = "Wald"` in the call to `calibrate`. For instance,

```
calibrate(mod, y0 = 3, interval = "Wald", level = 0.9)

##  estimate     lower     upper        se
## 2.9314491 2.6039901 3.2589080 0.1929338
```

produces the output

```
estimate    lower    upper       se
  2.9314   2.6040   3.2589   0.1929
```

The point estimate remains unchanged (as expected) but the calibration interval is slightly different. The major benefit of using the delta method to compute calibration intervals is that it always exists and provides us with an asymptotic estimate of the standard error, which in this case $\widehat{\text{se}} = 0.1929$. The bootstrap, as discussed in Section 2.4, also provides calibration intervals and an estimate of the standard error, but does not require large samples or specific distribution assumptions.

Confidence interval for a specified mean response

Instead of inferring the value of x_0 that corresponds to an observed value of the response y_0, suppose we want to infer the value of x_0 that corresponds to a specified value of the mean response, say $f(x_0; \boldsymbol{\beta}) = \mu_0$. The obvious ML estimate of x_0 is

$$\widehat{x}_0 = \frac{\mu_0 - \widehat{\beta}_0}{\widehat{\beta}_1}, \tag{5}$$

which is the same as Equation (1) but with $\bar{y}_0$ replaced with μ_0, a fixed population parameter. This is analogous to the difference between (i) predicting a future value of the response corresponding to a given value of the explanatory variable and (ii) estimating the mean response that corresponds to a particular value of the explanatory variable. Thus, the point estimates (1) and (5) are the same but the former has greater variability inherited from the variance of $\overline{\mathcal{Y}}_0$. This is sometimes referred to as *regulation* (as opposed to calibration) and is described in more detail in Graybill and Iyer (1994, chap. 6). The confidence interval formulas for x_0 corresponding to a specified mean response (i.e., regulation) are the same as those given in Equations (2) and (4), but with $1/m = 0$ and $t_{\alpha/2,n+m-3}$ replaced with $t_{\alpha/2,n-2}$. For instance, the Wald interval for x_0 corresponding to μ_0 is simply

$$\widehat{x}_0 \pm t_{\alpha/2,n-2} \frac{\widehat{\sigma}}{\left|\widehat{\beta}_1\right|} \sqrt{\frac{1}{n} + \frac{(\widehat{x}_0 - \bar{x})^2}{S_{xx}}},$$

where $\widehat{x}_0$ is calculated as in Equation (5). To obtain calibration intervals corresponding to a specified mean response, use the option `mean.response = TRUE`.

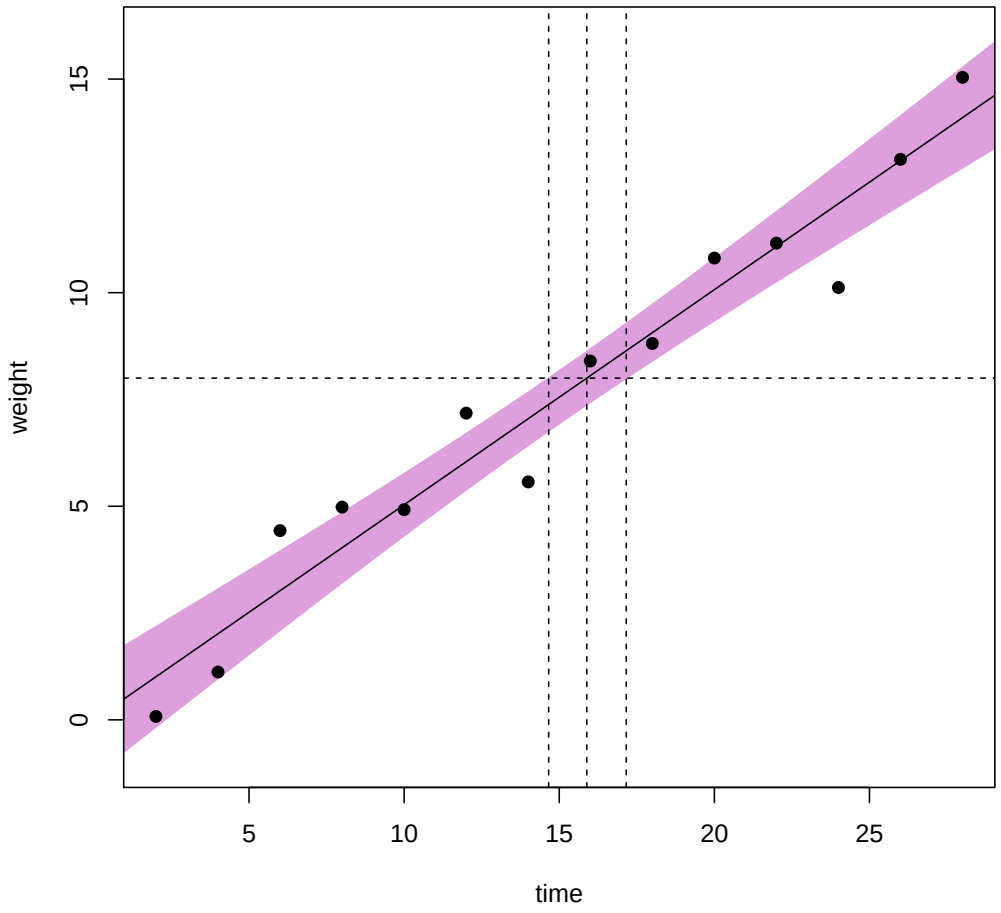

Figure 3: Scatterplot of the crystal growth data with fitted calibration line and pointwise confidence band. A horizontal reference line is drawn at $\mu_0 = 8$. The vertical lines identify the position of the point estimate and 95% confidence bounds for the growth time x_0.

To illustrate, consider the crystal growth data taken from Graybill and Iyer (1994, pg. 119). These data are from an experiment in which the weight in grams of 14 crystals were recorded after letting the crystals grow for different (predetermined) amounts of time in hours. The weight of each crystal is plotted against time in Figure 3. Suppose we want to estimate the growth time in hours that corresponds to an average weight of 8 grams; that is, we want to estimate x_0 corresponding to $\mu_0 = 8$. The following code chunk fits a simple linear regression model to the crystal growth data and then computes the ML estimate and a 95% calibration interval for x_0.

```
mod <- lm(weight ~ time, data = crystal)
(res <- calibrate(mod, y0 = 8, mean.response = T))

## estimate    lower    upper
## 15.88820 14.65896 17.15963

# Figure 3
plotFit(mod, interval = "confidence", pch = 19, shade = T,
        col.conf = "plum", extend.range = T)
abline(h = 8, v = c(res$lower, res$estimate, res$upper), lty = 2)
```

The output for this code chunk should be

```
  estimate    lower    upper
   15.8882  14.6590  17.1596
```

Thus, in order to produce crystals with an average weight of 8 grams, they should be grown for an estimated 15.8882 hours. A 95% confidence interval for the growth time is (14.6590, 17.1596). Obviously, if `mean.response = TRUE`, then `y0` can only take a single value; otherwise, an error will be displayed as in the following:

```
calibrate(mod, y0 = c(8, 9), mean.response = T)

## Error in calibrate.default(cbind(x, y), ...): Only one mean response value allowed.
```

which displays the message

```
Error in calibrate.default(cbind(x, y), ...) :
  Only one mean response value allowed.
```

This type of calibration problem is similar to computing the *median effective dose* ($ED_{0.5}$), or more generally ED_p where $0 < p < 1$, in binary response models. In R, these models are usually fit with the `glm` function from the **stats** package. The function `dose.p` from the **MASS** package (Venables and Ripley, 2002) can then be used to compute the ML estimate of ED_p for specified p. An estimate of the asymptotic standard error based on the delta method is also given which can be used to calculate a Wald-based confidence interval for ED_p. A future release of **investr** will likely make an inversion-type interval available for these models as well (i.e. by inverting a confidence interval for the mean response on the link scale). The package **drc** (Ritz and Streibig, 2005), for fitting dose-response curves, may also be of interest.

Simultaneous calibration intervals

The calibration intervals discussed so far are one-at-a-time intervals. If k new values of the response are observed that each correspond to a different unknown, say $x_{01}, \ldots x_{0k}$, then we need to adjust the critical value used in the inversion and Wald intervals accordingly. The simplest approaches are of course, the *Bonferroni* and *Scheffé* procedures. These can be computed by specifying the `adjust` option which can take any of the following arguments: `"none"` (the default), `"bonferroni"`, or `"scheffe"`. The value k also needs to be specified. See Miller (1981, chap. 3) for more details.

Nonlinear and polynomial calibration

In application, many relationships are nonlinear (e.g., dose-response curves). The classical estimator along with the inversion and Wald intervals can easily be extended to such nonlinear calibration curves. However, classical inference in these models (such as prediction intervals) are based on large samples and linear approximations (see Bates and Watts (1988, chap. 2)). Thus, for nonlinear calibration curves, the inversion interval provides only an approximate $100(1-\alpha)\%$ confidence interval for x_0 as does the Wald interval. Calibration in nonlinear models is discussed in further detail in Schwenke and Milliken (1991), Seber and Wild (2003, pp. 245-250), and Huet (2004, chap. 5). For calibration in polynomial models, see Brown (1993, pp. 47-88) and Seber and Lee (2003, pg. 172).

The `invest` function can be used for inverse estimation with any univariate regression model in R that inherits from class `lm` or `nls` (with the exception of weighted fits). For instance, consider the regression model $\mathcal{Y}_i = f(x_i; \boldsymbol{\beta}) + \epsilon_i$ $(i = 1, \ldots, n)$, where f may or may not be linear in $\boldsymbol{\theta}$. If we wish to estimate x_0 given an observation y_0, then the point estimate $\widehat{x}_0$ is given by solving $y_0 = f\left(x; \widehat{\boldsymbol{\theta}}\right)$ for x. The solution will be unique as long as f is monotonic in the region of interest. The `invest` function computes $\widehat{x}_0$ by calling the **stats** functions `predict` and `uniroot` to solve

$$y_0 - f\left(x; \widehat{\boldsymbol{\theta}}\right) = 0$$

numerically for x.

Approximate confidence intervals

Equation (2) gives a closed-form expression for the inversion interval for the case of simple linear regression. In more complicated cases, such an expression is not available and the interval must be found numerically. It can be shown (see, for example, Seber and Wild (2003, pp. 245-246)) that

$$\mathcal{T}_{x_0} = \frac{\mathcal{Y}_0 - f\left(x_0; \widehat{\boldsymbol{\theta}}\right)}{\left\{\widehat{\sigma}^2 + \widehat{S}^2_{x_0}\right\}^{1/2}} \dot{\sim} t_{n-p}, \tag{6}$$

where $\widehat{S}^2_{\widehat{x}_0}$ is the estimated variance of $f\left(x_0;\widehat{\boldsymbol{\theta}}\right)$. An approximate $100(1-\alpha)\%$ confidence interval for x_0 is then given by the set

$$\left\{x : t_{\alpha/2,n-p} < \mathcal{T}_x < t_{1-\alpha/2,n-p}\right\}. \tag{7}$$

Essentially, `invest` finds the lower and upper limits for this interval by solving the equations

$$\mathcal{T}_x - t_{\alpha/2,n-p} = 0 \quad \text{and} \quad \mathcal{T}_x - t_{1-\alpha/2,n-p} = 0$$

numerically for x_0. For the special case of the simple linear regression model, These limits coincide with Equation (2) and the coverage probability is exactly $1-\alpha$.

$$\Pr\left[t_{\alpha/2,n-p} < \mathcal{T} < t_{1-\alpha/2,n-p}\right] \approx 1-\alpha$$

The Wald interval for x_0 is also easily extended. It has the basic form: $\widehat{x}_0 \pm t_{\alpha/2,n-p}\widehat{\text{se}}\left(\widehat{x}_0\right)$, where p is the dimension of $\boldsymbol{\theta}$. The estimated standard error $\widehat{\text{se}}\left(\widehat{x}_0\right)$ is based on a first-order Taylor Series approximation. For the special case of the simple linear regression model, this approximation results in Equation (3).

Since the point estimate and confidence intervals for x_0 are obtained numerically using `uniroot`, `invest` has the additional options `lower`, `upper`, `tol`, and `maxiter`. See the reference manual for details.

To get an idea of how `invest` works, consider the data in Figure 4 which were generated from the model $\mathcal{Y} = 5 + x - \sin(x) + 1.5\mathcal{Z}$, where $\mathcal{Z}$ is a standard normal random variable. Suppose we want to estimate x_0 given a new observation, say $y_0 = 22$. Fitting a model of the form $f(x;\theta) = \theta_1 + \theta_2\left[x - \sin(x)\right]$ to the sample, we obtain $\widehat{f}(x) = 5.490 + 0.941\left[x - \sin(x)\right]$, which is certainly invertible, but $\widehat{f}^{-1}$ cannot be expressed in closed-form using a finite number of terms; thus, $\widehat{x}_0 = \widehat{f}^{-1}(y_0 = 22)$ must be obtained numerically. The following chunk of code generates the sample data, calculates $\widehat{x}_0$ and an approximate 95% Wald interval for x_0 using the `invest` function.

```
set.seed(101)  # for reproducibility
x <- rep(seq(from = 0, to = 25, by = 2), each = 2)
d <- data.frame(x, y = 5 + x - sin(x) + rnorm(length(x), sd = 1.5))
mod <- lm(y ~ I(x-sin(x)), data = d)
res <- invest(mod, y0 = 22, interval = "Wald")

# Figure 4
plotFit(mod, interval = "prediction", shade = TRUE, col.pred = "seagreen1",
        extend.range = TRUE)
abline(h = 22, v = res$estimate, lty = 2)
```

The point estimate is $\widehat{x}_0 = 16.7053$ with an estimated standard error of 0.8909. The code used by `invest` for obtaining the point estimate is essentially

```
uniroot(function(x) predict(mod, newdata = list("x" = x)) - 22,
        lower = min(d$x), upper = max(d$x))$root

## [1] 16.70528
```

The code for obtaining the standard error used in the Wald interval is slightly more involved:

```
# Write x0.hat as function of parameters (theta1.hat, theta2.hat, Y0)
x0.fun <- function(params, object = mod) {
  object$coefficients <- params[1:2]
  uniroot(function(x) predict(object, list("x" = x)) - params[3],
          lower = 0, upper = 25, tol = 1e-10)$root
}

# Variance-covariance matrix of (theta1.hat, theta2.hat, Y0)'
covmat <- diag(3)
covmat[1:2, 1:2] <- vcov(mod)
covmat[3, 3] <- summary(mod)$sigma^2

# Numerically evaluate gradient
params <- c(coef(mod), y0 = 22)
```

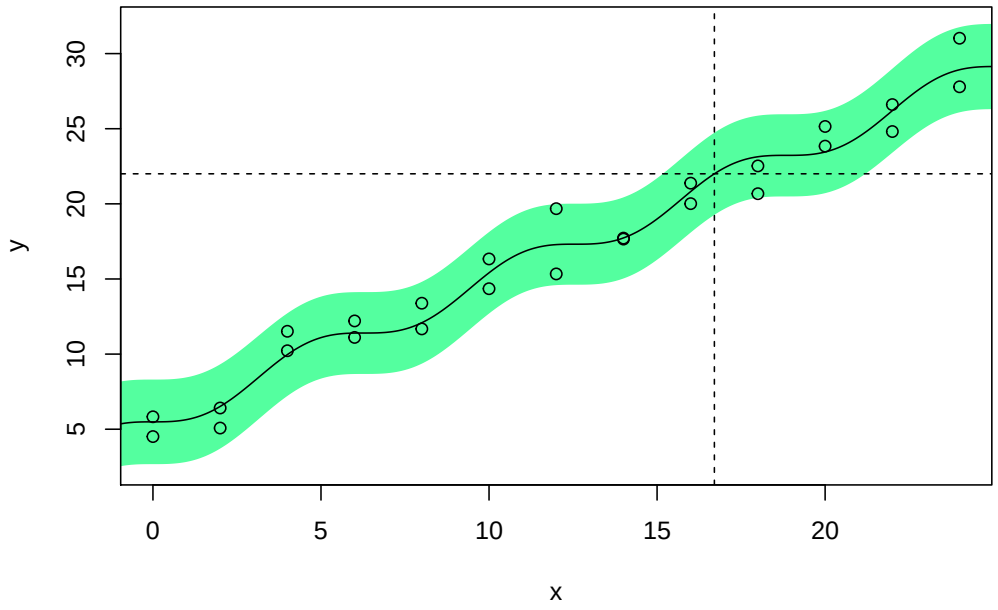

Figure 4: Scatterplot of simulated data with fitted calibration line and pointwise prediction band. A horizontal reference line is drawn through the observed value $y_0 = 22$. The vertical line identifies the position of the point estimate $\widehat{x}_0$.

```
grad <- attr(numericDeriv(quote(x0.fun(params)), "params"), "gradient")

# Calculate standard error
(se <- sqrt(grad %*% covmat %*% t(grad)))

##           [,1]
## [1,] 0.8909255
```

The following example uses the nasturtium data from the **drc** package. These data were analyzed in Racine-Poon (1988) using an approximate Bayesian approach. Bioassays were performed on a type of garden cress called nasturtium. Six replicates of the response (plant weight in mg after the third week of growth) were measured at seven preselected concentrations of an agrochemical. The weights corresponding to three new soil samples, all sharing the same (unknown) concentration x_0, were observed to be 309, 296, and 419 mg. The block of code below fits a log-logistic model

$$f(x;\boldsymbol{\theta}) = \begin{cases} \theta_1, & x = 0 \\ \theta_1 / \left[1 + \exp\left\{\theta_2 + \theta_3 \ln(x)\right\}\right], & x > 0, \end{cases}$$

and computes an approximate 95% inversion interval for x_0.

```
# Load package containing nasturtium data
library(drc)

# Fit log-logistic model
mod <- nls(weight ~ theta1/(1 + exp(theta2 + theta3*log(conc))),
           start = list(theta1 = 1000, theta2 = -1, theta3 = 1),
           data = nasturtium)
plotFit(mod, interval = "prediction") # figure not shown
# Compute approximate 95% inversion interval
invest(mod, y0 = c(309, 296, 419), interval = "inversion")

## estimate    lower    upper
## 2.263854 1.772244 2.969355
```

The interval obtained is (1.7722, 2.9694) with a point estimate of 2.2639. We can check the point estimate manually. Some algebra gives

$$\widehat{x}_0 = \exp\left\{\frac{\log\left(\widehat{\theta}_1/\bar{y}_0 - 1\right) - \widehat{\theta}_2}{\widehat{\theta}_3}\right\} = 2.2639.$$

A Wald interval can also be obtained by running the following line of code:

```
invest(mod, y0 = c(309, 296, 419), interval = "Wald")

##  estimate     lower     upper        se
## 2.2638535 1.6888855 2.8388214 0.2847023
```

the output for which is

```
  estimate    lower    upper       se
    2.2639   1.6889   2.8388   0.2847
```

In certain cases (i.e., when $\widehat{x}_0$ can be written in closed-form), one can use the very useful `deltaMethod` function from the **car** package (Fox and Weisberg, 2011) to compute the approximate standard error used in the Wald interval. For the nasturtium example, the minimal code to obtain $\widehat{\text{se}}\left(\widehat{x}_0\right)$ is

```
# Using the deltaMethod function in the car package
library(car)
covmat <- diag(4)
covmat[1:3, 1:3] <- vcov(mod)
covmat[4, 4] <- summary(mod)$sigma^2/3 # since length(y0) = 3
(se <- deltaMethod(c(coef(mod), y0.bar = mean(c(309, 296, 419))),
                   g = "exp((log(theta1/y0.bar - 1) - theta2) / theta3)",
                   vcov. = covmat)$SE)

## [1] 0.2847019
```

which produces an estimated standard error of 0.2847019, the same as that obtained automatically by `invest`.

As indicated by the bootstrap distribution obtained in the next section, the symmetric Wald interval obtained here seems unrealistic for this problem. In the next section, we show how to obtain a *bias-corrected and accelerated* (BC_a) bootstrap confidence interval for x_0 using the **boot** package (Canty and Ripley, 2013).

Bootstrap calibration intervals

The bootstrap (Efron, 1979) provides an alternative means for computing calibration intervals. This is useful in nonlinear calibration problems where inference traditionally relies on large samples, approximate normality, and linear approximations. A practical guide to the bootstrap is provided by Davison and Hinkley (1997) and the companion R package **boot**, and bootstrap resampling for controlled calibration is discussed in Jones and Rocke (1999). Although it is likely for a "bootstrap" option to appear in a future release of **investr**, it is quite simple to set up using the recommended **boot** package. First, we discuss a naive approach to calculating bootstrap calibration intervals.

A simple, but ultimately wrong approach to resampling in controlled calibration is demonstrated in the following example for the nasturtium data:

```
library(boot)

# Function to compute estimate of x0
x0.fun <- function(object, y) {
  theta <- unname(coef(object))
  exp((log(theta[1]/mean(y) - 1) - theta[2]) / theta[3])
}

# Bootstrap setup
```

```
y0 <- c(309, 296, 419)
res <- resid(mod) - mean(resid(mod)) # center the residuals
n <- length(res)
boot.data <- data.frame(nasturtium, res = res, fit = fitted(mod))
boot.fun <- function(data, i) {
boot.mod <- nls(fit + res[i] ~ theta1/(1 + exp(theta2 + theta3*log(conc))),
                start = list(theta1 = 1000, theta2 = -1, theta3 = 1),
                data = data)

  # Make sure the original estimate also gets returned
  if (all(i == 1:n)) x0.fun(mod, y0) else x0.fun(boot.mod, y0)
}

# Run bootstrap simulation (takes about 50s on a standard laptop)
set.seed(123) # for reproducibility
res <- boot(boot.data, boot.fun, R = 9999) # collect 9,999 bootstrap samples
boot.ci(res, type = "bca") # obtain BCa confidence interval for x0

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 9999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 2.04,  2.52 )
## Calculations and Intervals on Original Scale
```

This produces an approximate 95% BC_a calibration interval of (2.04, 2.52), quite shorter than the ones produced by the inversion and Wald methods in the previous section. Did the bootstrap really do that well? The answer here is no, but it is not the bootstrap's fault. Recall that $\bar{y}_0$ is an observed value of the random variable $\overline{Y}_0$ which has variance σ^2/m. This source of variability is ignored in the bootstrap Monte Carlo simulation above, which treats $\bar{y}_0$ as a fixed constant. Compare the interval just obtained with the output from

```
invest(mod, y0 = mean(c(309, 296, 419)), mean.response = T)

## estimate    lower    upper
## 2.263854 2.026871 2.552866
```

We can get a hold on this extra source of variability by again sampling with replacement from the centered residuals. In particular, we resample the responses y_{0j} $(j = 1, 2, \ldots, m)$, by randomly selecting m additional residuals $e_j^\star$ from the centered residuals and calculating the bootstrap responses

$$y_{0j}^\star = \bar{y}_0 + e_j^\star, \quad j = 1, 2, \ldots, m.$$

Let $\bar{y}_0^\star = \sum_{j=1}^m y_{0j}^\star/m$. The correct bootstrap estimate of x_0 is given by $\widehat{x}_0^\star = f^{-1}\left(\bar{y}_0^\star; \widehat{\boldsymbol{\theta}}^\star\right)$. To implement this, all we need to do is make the following changes to `boot.fun`:

```
boot.fun <- function(data, i) {
  boot.mod <- nls(fit + res[i] ~ theta1/(1 + exp(theta2 + theta3*log(conc))),
                  start = list(theta1 = 1000, theta2 = -1, theta3 = 1),
                  data = data)

  # Simulate the correct variance
  Y0 <- y0 + sample(data$res, size = length(y0), replace = T)

  # Make sure the original estimate also gets returned
  if (all(i == 1:n)) x0.fun(mod, y0) else x0.fun(boot.mod, Y0)
}
```

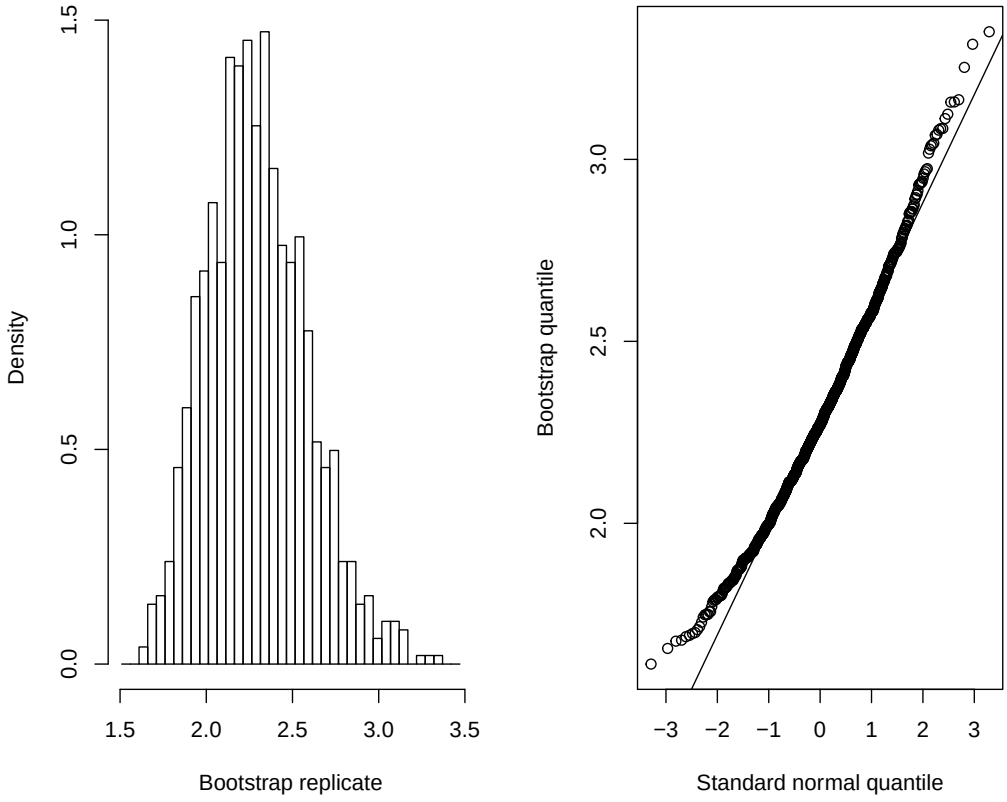

Figure 5: *Left* Histogram of bootstrap replicates, a vertical line indicates the position of the original estimate $\widehat{x}_0$. *Right* Normal Q-Q plot of the bootstrap replicates.

Rerunning the previous example with the modified `boot.fun` function produces a more realistic 95% confidence interval for x_0 of (1.818, 2.950) and an estimated standard error for $\widehat{x}_0$ of 0.2861 (similar to that obtained by the delta method earlier). Figure 5 shows a histogram and normal Q-Q plot of the 9,999 bootstrap replicates of $\widehat{x}_0$. The bootstrap distribution of $\widehat{x}_0$ is comparable to the approximate posterior density of x_0 given in Racine-Poon (1988, fig. 9). The bootstrap distribution is skewed to the right and clearly not normal, suggesting that the Wald interval may not be realistic for this problem.

```
# Bootstrap calibration intervals. In general, nsim should be as large as
# reasonably possible (say, nsim = 9999).
boo <- invest(mod, y0 = c(309, 296, 419), interval = "percentile",
              nsim = 999, seed = 101)
boo  # print bootstrap summary

##  estimate     lower     upper        se      bias
## 2.2638535 1.8005534 2.9335622 0.2909187 0.0320422

plot(boo)  # plot results
```

Summary

We introduced the **investr** package for computing point estimates along with inversion and Wald-based confidence intervals for linear and nonlinear calibration problems with constant variance. We also showed how the **boot** package can be used for constructing approximate $100(1-\alpha)\%$ calibration intervals, which for convenience, may be incorporated into a future release of **investr**. The authors are currently working on extending the package to handle the case of heteroscedastic errors, random

coefficient models (e.g., objects of class `lme` from the recommended **nlme** package (Pinheiro et al., 2013)), and even multivariate calibration problems (e.g., objects of class `mlm`).

Acknowledgements

The authors would like to thank two anonymous reviewers and the Editor for their helpful comments and suggestions.

Bibliography

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. Wiley series in probability and statistics: Probability and statistics section. John Wiley & Sons, New York, 1988. [p1, 6]

P. Brown. *Measurement, Regression, and Calibration*. Oxford science publications. Clarendon Press, 1993. [p6]

A. Canty and B. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2013. URL http://cran.r-project.org/web/packages/boot/index.html. R package version 1.3-9. [p9]

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Applications*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1997. [p9]

R. Dorfman. A note on the δ-method for finding variance formulae. *The Biometric Bulletinn*, 1:129–137, 1938. [p3]

N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley series in probability and mathematical statistics: Applied Probability and Statistics. Wiley, New York, 1998. [p2]

B. Efron. Bootstrap methods: Another look at the jackknife. *Annals fo Statistics*, 7(1):1–26, 1979. [p9]

C. Eisenhart. The interpretation of certain regression methods and their use in biological and industrial research. *Annals of Mathematical Statistics*, 10(2):162–186, 1939. [p2]

E. C. Fieller. Some problems in interval estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 16:175–185, 1954. [p2]

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edition, 2011. URL http://socserv.socsci.mcmaster.ca/jfox/Books/Companion. [p9]

F. A. Graybill. *Theory and Application of the Linear Model*. Duxbury classic series. Duxbury, Pacific Grove, CA, 1976. [p1]

F. A. Graybill and H. K. Iyer. *Regression Analysis: Concepts and Applications*. Duxbury Press, Belmont, CA, 1994. [p2, 4, 5]

S. Huet. *Statistical Tools for Nonlinear Regression: A Practical Guide with S-PLUS and R Examples*. Springer Series in Statistics. Springer, 2004. [p6]

G. Jones and D. M. Rocke. Bootstrapping in controlled calibration experiments. *Technometrics*, 41(3): 224–233, 1999. [p9]

R. Miller. *Simultaneous Statistical Inference*. Springer Series in Statistics. Springer-Verlag, 1981. [p6]

C. Osborne. Calibration: A review. *International Statistical Review*, 59(3):309–336, 1991. [p1]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2013. R package version 3.1-110. [p12]

A. Racine-Poon. A bayesian approach to nonlinear calibration problems. *Journal of the American Statistical Association*, 83(403):650–656, 1988. [p8, 11]

C. Ritz and J. C. Streibig. Bioassay analysis using r. *Journal of Statistical Software*, 12, 2005. URL http://www.bioassay.dk. [p6]

J. R. Schwenke and G. A. Milliken. On the calibration problem extended to nonlinear models. *Biometrics*, 47(2):563–574, 1991. [p6]

G. Seber and A. Lee. *Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2003. [p6]

G. Seber and C. Wild. *Nonlinear Regression*. Wiley Series in Probability and Statistics. Wiley, 2003. [p6]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. [p6]

H. Wickham and W. Chang. *devtools: Tools to make developing R code easier*, 2013. URL http://cran.r-project.org/package=devtools. R package version 1.4.1. [p1]

Brandon M. Greenwell
Air Force Institute of Technology
Wright Ptrsn AFB, OH 45433
USA
brandon.greenwell@afit.edu

Christine M. Schubert Kabban
Air Force Institute of Technology
Wright Ptrsn AFB, OH 45433
USA
christine.schubertkabban@afit.edu