06236

LA-UR-09-*****

Approved for public release; distribution is unlimited.

Title: Illustrating the Practice of Statistics

Author(s): Christina A. Hamada
Michael S. Hamada

Intended for: Submission to The American Statistician

Los Alamos
NATIONAL LABORATORY
EST.1943

Illustrating the Practice of Statistics

C. A. Hamada
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1

M. S. Hamada
Statistical Sciences Group
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

09.27.09 1230

Abstract

Key Words: Bayesian, experimental design, frequentist, interval-censored data, modeling, optimization, regression, simulation.

1. Introduction

The practice of statistics involves analyzing data and planning data collection schemes to answer scientific questions. Issues often arise with the data that must be dealt with and can lead to new procedures. In analyzing data, these issues can sometimes be addressed through the statistical models that are developed. Simulation can also be helpful in evaluating a new procedure. Moreover, simulation coupled with optimization can be used to plan a data collection scheme.

The practice of statistics as just described is much more than just using a statistical package. In analyzing the data, it involves understanding the scientific problem and incorporating the scientist's knowledge. In modeling the data, it involves understanding how the data were collected and accounting for limitations of the data where possible. Moreover, the modeling is likely to be iterative by considering a series of models and evaluating the fit of these models. Designing a data collection scheme involves understanding the scientist's goal and staying within his/her budget in terms of time and the available resources. Consequently, a practicing statistician is faced with such tasks and requires skills and tools to do them quickly.

We have written this article for students to provide a glimpse of the practice of statistics. To illustrate the practice of statistics, we consider a problem motivated by some precipitation data that our relative, Masaru Hamada, collected some years ago. We describe his rain gauge observational study in Section 2. We describe modeling and an initial analysis of the precipitation data in Section 3. In Section 4, we consider alternative analyses that address potential issues with the precipitation data. In Section 5, we consider the impact of incorporating additional information. We design a data collection scheme to illustrate the use of simulation and optimization in Section 6. We conclude this article in Section 7 with a discussion.

2. The Rain Gauge Observational Data

In the metropolitan Washington, D.C. area, there is a network of volunteers that collects precipitation data year round. From the spring to the fall, the precipitation is collected in a plastic rain gauge. In the winter, a plastic rain gauge cannot be used because it will freeze and crack. As a way to record snowfall, the volunteers were instructed to collect the snow in a standard 2.5 metal can, let the snow melt indoors, pour the melted snow in the can into a plastic rain gauge, record the measurement, and estimate the snowfall by multiplying the can measurement by 0.44. Our relative thought that the 0.44 factor for estimating snowfall was too small and that 0.45 might be a better factor. So over two summers when it rained, he recorded the precipitation collected in the rain gauge and in a standard 2.5 metal can; both were mounted next to each other at the same height. The amount in the can was measured by first dumping the rain gauge and pouring the rain in the can into the rain gauge. Our relative collected the data the next morning after it rained at around 7 A.M. The data are displayed in Table 1, where the columns labeled x are the precipitation (in inches) collected in the standard 2.5 metal can and the columns labeled y are the precipitation (in inches) collected in the rain gauge. In summary, our relative collected these data to assess whether the 0.44 factor was too small.

Table 1: Rain Gauge Observational Data (precipitation in can (x) and in rain gauge (y) in inches)

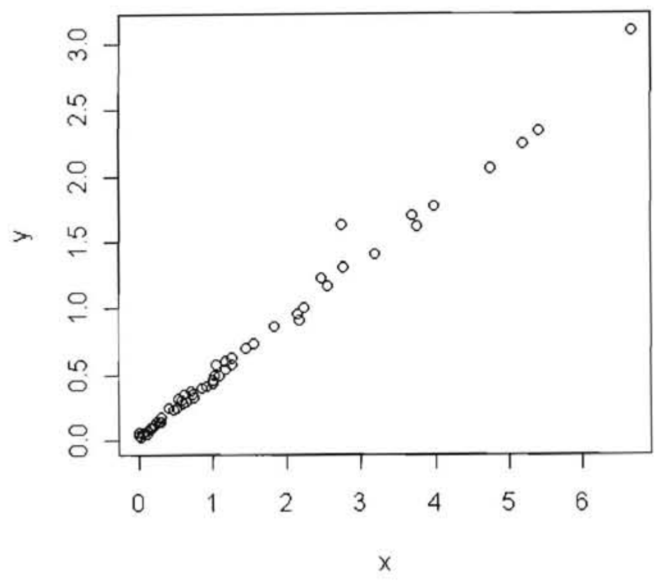
x	y	Interval x	Interval y	x	y	Interval x	Interval y
0.11	0.05	0.105 0.115	0.045 0.055	0.41	0.25	0.40 0.42	0.24 0.26
1.08	0.50	1.05 1.10	0.49 0.51	1.45	0.70	1.425 1.475	0.69 0.71
1.16	0.54	1.15 1.20	0.53 0.55	0.22	0.12	0.21 0.23	0.115 0.125
2.75	1.31	2.725 2.775	1.30 1.35	2.22	1.00	2.20 2.25	0.99 1.025
0.12	0.07	0.115 0.125	0.065 0.075	0.70	0.38	0.69 0.71	0.37 0.39
0.60	0.28	0.59 0.61	0.27 0.29	**2.73**	**1.63**	**2.70 2.75**	**1.60 1.65**

1.55	0.73	1.525 1.575	0.72 0.74	0.02	0.02	0.015 0.025	0.015 0.025
1.00	0.46	0.99 1.025	0.45 0.47	0.18	0.09	0.175 0.185	0.085 0.095
0.61	0.35	0.60 0.62	0.34 0.36	0.27	0.14	0.26 0.28	0.135 0.145
3.18	1.40	3.15 3.20	1.375 1.425	1.25	0.62	1.225 1.275	0.61 0.63
2.16	0.91	2.15 2.20	0.90 0.92	0.46	0.23	0.45 0.47	0.22 0.24
1.82	0.86	1.80 1.85	0.85 0.87	0.31	0.17	0.30 0.32	0.165 0.175
4.75	2.05	4.725 4.775	2.025 2.075	0.75	0.33	0.74 0.76	0.32 0.34
1.05	0.58	1.025 1.075	0.57 0.59	2.55	1.17	2.525 2.575	1.15 1.20
0.92	0.41	0.91 0.93	0.40 0.42	1.00	0.43	0.99 1.025	0.42 0.44
0.86	0.40	0.85 0.87	0.39 0.41	3.98	1.77	3.95 4.00	1.75 1.80
0.24	0.14	0.23 0.25	0.135 0.145	1.26	0.58	1.25 1.30	0.57 0.59
0.01	0.03	0.005 0.015	0.025 0.035	5.40	2.34	5.375 5.425	2.30 2.35
0.51	0.25	0.50 0.52	0.24 0.26	1.02	0.50	1.00 1.05	0.49 0.51
2.15	0.96	2.125 2.175	0.95 0.97	3.75	1.62	3.725 3.775	1.60 1.65
0.53	0.32	0.52 0.54	0.31 0.33	3.70	1.70	3.675 3.725	1.675 1.725
5.20	2.25	5.175 5.225	2.225 2.275	0.30	0.14	0.29 0.31	0.135 0.145
0.00	0.06	0.00 0.01	0.055 0.065	0.07	0.06	0.065 0.075	0.055 0.065
1.17	0.60	1.15 1.20	0.59 0.61	0.58	0.31	0.57 0.59	0.30 0.32
6.67	3.10	6.65 6.70	3.075 3.125	0.72	0.35	0.71 0.73	0.34 0.36
0.04	0.04	0.035 0.045	0.035 0.045	0.63	0.29	0.62 0.64	0.28 0.30
2.22	1.00	2.20 2.25	0.99 1.025	1.55	0.73	1.525 1.575	0.72 0.74
0.05	0.05	0.045 0.055	0.045 0.055	2.47	1.23	2.45 2.50	1.20 1.25
0.15	0.09	0.145 0.155	0.085 0.095				

3. Modeling and an Initial Analysis

We begin considering a model for the rain gauge observational data by plotting the data in Figure 1. We note that the point (2.73, 1.63) is apparently different from the other 56 data points and discard it in subsequent analyses. Too much has passed since the data were collected to find out more why this point appears to be quite different from the rest; we assume that something like a gross recording error occurred.

Figure 1: Plot of Precipitation in Can (x) and Rain Gauge (y)



Regarding the form of the model, the question about the factor applied to the can measurement to obtain the equivalent rain gauge measurement suggests a linear model. Figure 1 also suggests a linear model with the form $y = \beta_1 x + \varepsilon$. At first thought, a model without an intercept makes sense because if it does not rain, then there should be no water in the can or the rain gauge; that is, β_1 is the factor our relative has a question about. A residual plot (not shown here) shows too many positive residuals, so that next we consider the model with an intercept,

$$y = \beta_0 + \beta_1 x + \varepsilon \ , \tag{1}$$

where error $\varepsilon \sim N(0,\sigma^2)$. See the results from fitting this model in R (R Core Development Team, 2004) using frequentist inference in Table 2, which indeed show that both the intercept and slope are significant and that the estimate for β_1 is close to 0.44. It is interesting that without the intercept the estimate for β_1 is 0.4511, which would have suggested changing the factor.

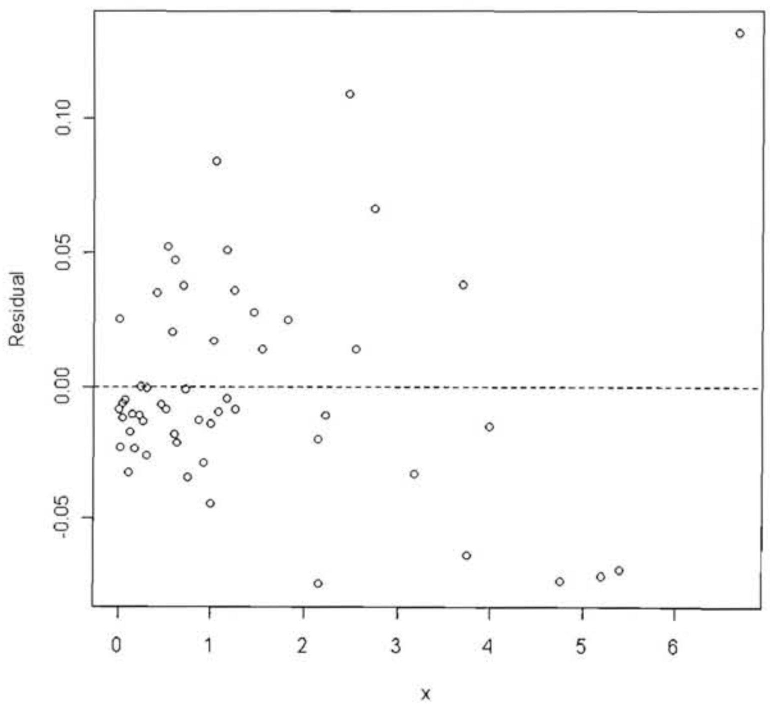
As an aside, we wondered why there was a non-zero intercept. Recall that the data were collected over the summer and the measurements were taken the next morning. Consequently, one explanation is evaporation and likely there is more in the metal can than the plastic rain gauge. Another reason might be that there is some loss in pouring the rain in the can into the rain gauge because beads of water tend to remain on the metal can.

Table 2: Results from Fitting (1) to the Rain Gauge Observational Data

Parameter	Estimate	Std. Err.	t	p value
β_0	0.0344	0.0076	4.5	$< 10^{-04}$
β_1	0.4398	0.0037	120.5	$< 10^{-15}$

Before drawing such conclusions, we should check how well the model fits by looking at the residual plot in Figure 2. The plot suggests that the spread of the residuals increases as the can measurements increase. More importantly, we wonder if we fit such a model, would the results suggest a different factor than 0.44.

Figure 2: Residual Plot from Fitting (1) to the Rain Gauge Observational Data



Because we often need to address such issues quickly, we may not even have the time to figure out what is in the literature nor to obtain specialized software to address them. We are aware of models with non-constant error variance so we write a standard one down as

$$\log[\sigma(x)] = \alpha_0 + \alpha_0 x. \qquad (2)$$

To avoid having to develop methodology to fit (1) with $\varepsilon \sim N(0,\sigma^2(x))$ and $\sigma^2(x)$ given in (2), we take a Bayesian approach using WinBUGS (Spiegelharter et al. (2004)), where we only need to specify a model and rely on Bayes' Theorem (De Groot, 1970) to do the inference. WinBUGS employs the advanced Bayesian computing as described in Gelman et al. (2004). See the WinBUGS code, which used diffuse prior distributions for the model parameters, in Appendix A that produced the results in Table 3. The scaled residual plot (i.e., the residuals scaled by their standard deviations using the posterior medians of the parameters in (1) and (2)) in Figure 3 appears to be better. Also a normal plot of the scaled residuals (not shown here) is much straighter than the residuals for (1) alone. From Table 3, see that a 95% credible interval for β_1 is (0.4341, 0.4563) which rules out neither 0.44 nor 0.45.

Figure 3: Scaled Residual Plot from Fitting (1) and (2) to the Rain Gauge Observational Data

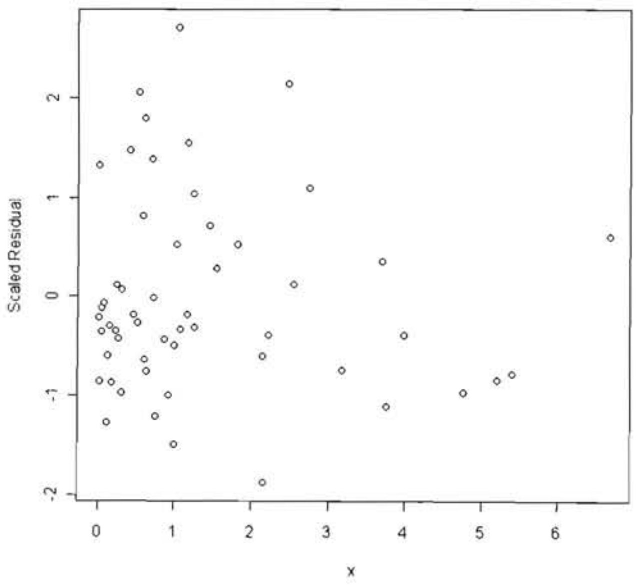


Table 3: Results from Fitting (1) and (2) to the Rain Gauge Observational Data Using WinBUGS (0.025, 0.50, and 0.975 posterior quantiles)

Parameter	0.025	0.50	0.975
α_0	-4.091	-3.806	-3.520
α_1	0.1629	0.3015	0.4771
β_0	0.01983	0.03041	0.04165
β_1	0.4341	0.4444	0.4563

By now, the reader has likely figured out that the factor is the ratio of the area of the top of the rain gauge and the area of the top of the can. The rain gauge's dimensions are 64mm by 59 mm and the can has a radius 52 mm. Consequently, the factor is 3776/8494.867 = 0.4445. We see that the median of β_1 is surprisingly close to the truth.

4. Addressing Additional Data Issues

Assessing the quality of the measurements is always a good thing to do in any analysis. For the rain gauge observational study, we noticed that the gradations on the rain gauge are larger for larger amounts of rain: 0.01 inches for 0 to 0.20 inches, 0.02 inches for 0.20 to 1.00 inches, and 0.05 inches for amounts larger than 1.00 inch. Our relative had recorded his best estimate of the amount of rain in the can and rain gauge as displayed in the *x* and *y* columns of Table 1. Based on the size of the gradations, we can be sure that the actual amounts in the can and rain gauge are in intervals as shown in the columns labeled "Interval *x*" and "Interval *y*" in Table 1.

In the practice of statistics, we may wonder if the results from analyzing the interval data will differ substantially from what we have already obtained, but we need to balance our concern that they may differ substantially against the deadline when the analysis is needed. Often as in this case, we can handle the measurement limitations by modeling. That is, we treat the interval *y* data as interval-censored data and the interval *x* data as having a uniform distribution. The WinBUGS code in Appendix A is easily modified as displayed in Appendix B and the results are shown in Table 4. Note that the Table 4 results are nearly the same as those in Table 3. Consequently, we would likely present the simpler analysis of the *x* and *y* data in the analysis report, but note that the measurement uncertainty had been addressed and not found to have an impact on the results.

Table 4: Results from Fitting (1) and (2) to the Interval Censored Rain Gauge Observational Data Using WinBUGS (0.025, 0.50, and 0.975 posterior quantiles)

Parameter	0.025	0.50	0.975
α_0	-4.127	-3.836	-3.520
α_1	0.1622	0.3043	0.4761
β_0	0.01922	0.02980	0.04111
β_1	0.4341	0.4445	0.4562

5. Evaluating the Impact of Additional Information

We wondered if knowing some additional information about the factor would reduce the uncertainty of estimating the factor. For example, suppose that we know the area of rain gauge is 3776 mm^2, the area of the can is proportional to the squared radius (2704 mm^2), and the factor is the ratio of the two areas. In other words, we need to estimate π. Consequently, we adapted the WinBUGS code in Appendix A by replacing β_1 by $3776/(\pi\times2704)$ and using a diffuse prior on π (i.e., N(3,100)). The results for the factor by evaluating the posterior of $3776/(\pi\times2704)$ was virtually the same as for β_1 in Table 3. Interestingly, the 95% credible interval for π obtained from this analysis was (3.058, 3.216), with a posterior median of 3.14. We also considered the two other variations where we know π but we do not know the area of the rain gauge or where we do not know the radius of the can; both gave virtually the same posterior for β_1. Apparently, no real information was added in any of these scenarios; only the form of the problem was changed.

6. Designing a Data Collection Scheme

While the practicing statistician often analyzes data that have already been collected, opportunities arise when the statistician is asked to help develop a data collection scheme or design. Suppose that our relative did not want to wait to collect additional years of rain data, but instead wanted to conduct an experiment to find out whether a factor of 0.45 was warranted in converting the can to rain gauge measurement. Such an experiment might use a uniform delivery system such as a sprinkler with a rotating head. By positioning the can and rain gauge together at the same height and running the sprinkler for different lengths of time, measurements from the can and rain gauge could be taken. Next, we consider the planning of such an experiment.

First, we consider a criterion for the design. Suppose that we want a design that with high probability, say 0.90, will produce a 95% confidence interval for β_1 in (1) does not contain 0.45. But we also want to minimize the time that it will take to perform the experiment. Suppose that we do not have much time to produce a design so that we will assume only (1) holds, i.e., with constant error variance. We will also assume that $\beta_0 = 0$, $\beta_1 = 0.4445$, and $\sigma = 0.01$. We might have picked $\beta_1 = 0.445$, halfway between 0.44 and 0.45 but decided to use the true value since this is only an illustration. The $\sigma = 0.01$ value we might obtain by doing some preliminary runs for a given amount of time. In setting up the experiment, we would also need to choose a sprinkler and confirm that it is providing uniform delivery of water, an experiment in itself.

Next, we consider a class of data collection schemes. We will restrict our search to two point designs whose minimum amount is 0.2 inches and maximum amount is 1.5 inches; that is, we believe the linear model (1) holds. We also consider designs with a maximum of 200 samples per point. So how do we find a design within this class that minimizes time while meeting the criterion that with 0.90 probability the design will produce a 95% confidence interval for β_1 that does not contain 0.45? Note that we will estimate the probability by generating 1000 data sets and use the observed proportion.

We know from the experimental design literature that the D-optimal design spreads the two points as far apart with equal sample sizes. Consequently, we evaluate this design and find that the criterion has an observed proportion of 1.000 but it requires nearly 378 hours to run; assuming that the sprinkler delivers 0.90 inches per hour in the can, then the amounts of 0.2 and 1.5 inches correspond to about 13 and 100 minutes. Note that the evaluation is simple to develop in R by repeatedly generating data, fitting (1), and computing a 95% confidence interval for β_1. We used the R function evalSolution() presented in Appendix C.

How do we find a design that takes less time? We doubt that there is either literature or software for our problem because of the specialized criterion and the time minimization aspect. Since we can evaluate a design, it is pretty easy to write code in R for a genetic algorithm (GA) that is presented in

Appendix C. A genetic algorithm is a stochastic optimization method based on evolutionary principles (Goldberg (1989), Michalewicz (1992)) and has been used for design; Hamada et al. (2001) is one such example. Please see these references for further details. For the design problem we are considering, we used a population of 50 designs and ran the GA for 25 generations in which 100 new designs through crossover and mutation are generated from the previous designs and evaluated each generation. The best 50 designs among these 100 new designs and the 50 previous designs become the new generation of 50 designs. The GA found the following design: 89 samples at 0.22 inches and 47 samples at 1.31 inches with an observed probability of meeting the criterion of 0.92 and a time of nearly 90 hours. Consequently, we have found a design requiring less than a quarter of time that the D-optimal design requires!

If we had more time, we might see how the two point design performs under (1) and (2). But since we fit (1) and (2) using WinBUGS, an evaluation of this design is more involved. It may be possible to use the R interface for WinBUGS, but we have sometimes experienced WinBUGS crashing on a particular data set for admittedly more complicated models. However, we have found YADAS (Graves, 2003) to be robust in evaluating many data sets, although its use requires a steep learning curve. And if we had even more time, we might use a GA whose evalSolution() calls YADAS through a system command to analyze generated data sets in evaluating a design; that is, evaluating a single design may it self be time consuming, let alone 100's or 1000's of designs.

7. Discussion

This article has introduced students to the practice of statistics, which involves analyzing data and designing data collection schemes. The analysis of data involves developing models whose form provides a way to answer the scientists's questions. The models should be faithful to the data in the way they are collected and account for their limitations. For example, preliminary modeling of the rain gauge observational data suggested an increasing error variance with the amount in the can. Moreover, the rain gauge measures less precisely for larger amounts. In this case, both of these features could be handled in the modeling. We also saw that additional information for the rain gauge observational data did not improve the inference. However, often in practice improved inferences result from scientific information provided by the scientist; see Annis (2005) that used engineering knowledge to improve a helicopter design (Box, 1992) and required fewer resources to do it. In considering the design of a data collection scheme, we saw that a candidate design could be evaluated through simulation and that we could find a better design by optimization coupled with simulation.

This article has shown that the practicing statistician cannot rely on canned packages alone. The practicing statistician needs to have a toolkit to explore different analyses and to design data collection schemes. A package like R is useful because we can combine analyses with its graphical capabilities to assess how well models fits. The package WinBUGS frees the practicing statistician to concentrate on developing models that are faithful to the data and not have to worry about developing specialized inference methodology. In designing data collection schemes, the practicing statistician can use R's programming capabilities by evaluating schemes through simulation and by finding better schemes through implementing an optimization algorithm such as a genetic algorithm.

With these things in mind, students can prepare for the practice of statistics by learning to develop models that address a scientist's questions and that are faithful to the data and their limitations. Students can also be learning packages that have modeling and programming capabilities so that they can analyze data with models that they develop, assess their fit through graphics, assess data collection schemes through simulation, and design data collection schemes through optimization. Students can make small attempts even in analyzing data sets assigned for homework or take on small projects that require more effort but have the potential for learning even more about the practice of statistics.

References

D. Annis, D. (2005). "Rethinking the paper helicopter: combining statistical and engineering knowledge," *The American Statistician*, 59, 320-326.

Box, G.E.P. (1992). "Teaching engineers experimental design with a paper helicopter," *Quality Engineering*, 4, 453-459.

DeGroot, M.H. (1970). *Optimal Statistical Decisions*. New York: McGraw-Hill.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2004). *Bayesian Data Analysis, Second Edition*, Boca Raton: Chapman & Hall/CRC.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading: Addison-Wesley.

Graves, T.L. (2003). "An Introduction to YADAS". (yadas.lanl.gov).

Hamada, M., Martz, H.F., Reese, C.S., and Wilson, A.G. (2001). "Finding Near-Optimal Bayesian Experimental Designs Via Genetic Algorithms," *The American Statistician*, 55, 175-181.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, New York: Springer-Verlag.

R Development Core Team (2004). R: A Language and Environment for Statistical Computing. Vienna: R Foundation for Statistical Computing. http://www.R-project.org.

Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2004). WinBUGS Version 1.4 User Manual.

Appendix A: WinBUGS Code for Fitting Model (1) and (2)

```
# error std dev increases in x

        model
        {
                for( i in 1 : N ) {
                        y[i] ~ dnorm(mu[i],tau[i]) # tau is precision or 1/var
                        mu[i] <- beta0 + beta1 * x[i]
                        tau[i] <-1/var[i]
                        var[i]<-sigma[i]*sigma[i]
                        sigma[i]<-exp(alpha0+alpha1*x[i])
                }

                # priors
                beta0 ~ dnorm(0.0,1.0E-6)
                beta1 ~ dnorm(0.0,1.0E-6)
                alpha0 ~ dnorm(0.0,1.0E-6)
                alpha1 ~ dnorm(0.0,1.0E-6)
        }
```

Appendix B: WinBUGS Code for Fitting Model (1) and (2) with Interval Censored Data

```
#y[i], x[i] - interval censored data

        model
        {
                for( i in 1 : N ) {
                        y[i] ~ dnorm(mu[i],tau[i])I(yl[i],yh[i])
                        x[i] ~ dunif(xl[i],xh[i])
                        mu[i]<-beta0+beta1*x[i]+0*xx[i]+0*yy[i] #xx[] and yy[] are unused variables
                                                                # in the data set that are zeroed out
                        tau[i] <-1/var[i]
                        var[i]<-sigma[i]*sigma[i]
                        sigma[i]<-exp(alpha0+alpha1*x[i])
                }
```

```
            beta0 ~ dnorm(0.0,1.0)
            beta1 ~ dnorm(0.0,1.0)
            alpha0 ~ dnorm(0.0,1.0E-1)
            alpha1 ~ dnorm(0.0,1.0)
}
```

Appendix C: R Code for Designing a Data Collection Scheme

```
# 2 pt data collection plan
# -- minimize time
# -- require probability of beta1 0.975 upper confidence bound < 0.45|beta1 = 0.4445
#   >= 0.90

# function getRandomSolution
getRandomSolution <- function(maxn,minx,maxx)
{
x <- rep(0,4) # x[1:2] are amounts, x[3:4] are sample sizes
x[3] <- sample(maxn,1)
x[4] <- sample(maxn,1)
a<-runif(2)
x[1] <- minx+a[1]*(maxx-minx)
x[2] <- minx+a[2]*(maxx-minx)
x[1:2] <- sort(x[1:2]) # order amounts
x #return
}
# end getRandomSolution

# function getCrossoverSolution
getCrossoverSolution <- function(popsize,curpop,cureval)
{
#pick parents
invo <- 1/order(cureval)
pind <- sample(seq(1:popsize),2,prob=invo)
new <- curpop[pind[1],]
for(i in 1:4)
{#for
if(rbinom(1,1,.5)==1)
{#if
new[i] <- curpop[pind[2],i]
}#if
}#for
new[1:2] <- sort(new[1:2]) # order amounts
new #return
}
# end getCrossoverSolution
# function getMutationSolution
getMutationSolution <- function(cursol,mr,ms,ig,minx,maxx,maxn)
{
prMut <- exp(-ms*ig)
new <- cursol
fact <- ms*exp(-mr*ig)
for(i in 3:4)
{#for
if(rbinom(1,1,prMut)==1)
{#if2
z <- (cursol[i]-1)/(maxn-1)
if(z==1) z=.999
if(z==0) z=.001
d <- log(z/(1-z))+rnorm(1)*fact
u <- exp(d)/(1+exp(d))
new[i] <- 1+floor(maxn*u)
}#if1
}#for
# mutate x1, x2
if(rbinom(1,1,prMut)==1)
{#if
z <- (cursol[1]-minx)/(maxx-minx)
if(z==1) z=.999
if(z==0) z=.001
d <- log(z/(1-z))+rnorm(1)*fact
u <- exp(d)/(1+exp(d))
new[1] <- minx+u*(maxx-minx)
}#if
if(rbinom(1,1,prMut)==1)
{#if
z <- (cursol[2]-minx)/(maxx-minx)
if(z==1) z=.999
if(z==0) z=.001
d <- log(z/(1-z))+rnorm(1)*fact
u <- exp(d)/(1+exp(d))
new[2] <- minx+u*(maxx-minx)
}#if
new[1:2] <- sort(new[1:2]) # order amounts
new # return
}
# end getMutationSolution

# function evalSolution
evalSolution <- function(sol,nsim)
{
# build data collection plan
x <- c(rep(sol[1],sol[3]),rep(sol[2],sol[4]))
n <- length(x)
factr <- sqrt(sum((x-mean(x))^2))
pr <- 0
for(i in 1:nsim){
#generate data
y <- 0 + 0.4445*x+.01*rnorm(n)
# fit model
fit <- lm(y~x)
beta1hat <- fit$coef[2]
# calculate confidence interval
sigmahat <- sqrt(sum((fit$res)^2)/(n-2))
# get 95% confidence interval
box <- qt(.975,n-2)*sigmahat/factr
high <- beta1hat+box
if(high < .45) pr<-pr+1
}
pr <- pr/nsim
# get time assume 0.15 inches per 10 min
time <- (sum(x)/(.15*6))
# penalize if pr < .9
{
if(pr < .9) crit <- 999999+time
else
crit <- time
}
c(crit,pr) # return
}
# end evalSolution

# data collection plan and evaluation parameters
nsim <- 1000 # number data sets per evaluation
maxx <- 1.5 # max amount
minx <- .2 # min amount
maxn <- 200 # max sample size per amount

# GA parameters
maxgen <- 25  # max number of generations
popsize <- 50 # population size
mrate <- .01 # used in mutation, relaxation
msigma <- 1 # used in mutation, relaxation
numvar<-4 # solution is dimension 4, 2 amounts, 2 sample sizes
pop <- matrix(rep(0,popsize*3*numvar),ncol=numvar)
   # solutions, top popsize solutions at end of generation is next population
eval <- matrix(rep(0,popsize*3*2),ncol=2) # eval - evaluate criterion
bSol <- matrix(rep(0,maxgen*numvar),ncol=numvar) # best solution per generation
```

```
bEval <- matrix(rep(0,maxgen*2),ncol=2) # criterion of best solution per generation

# START GA

#get initial population - randomly
for(i in 1:popsize){
pop[i,] <- getRandomSolution(maxn,minx,maxx)
}
#evaluate initial population
for(i in 1:popsize){
eval[i,] <- evalSolution(pop[i,],nsim)
}
#order  1:popsize
o <- order(eval[1:popsize])
pop[1:popsize,] <- pop[o,]
eval[1:popsize,] <- eval[o,]
pop0 <- pop[1,]
eval0 <- eval[1,]

# run GA for maxgen generations
for(igen in 1:maxgen)
{#igen

#apply crossover
offset <- popsize
for(i in 1:popsize){
pop[offset+i,] <- getCrossoverSolution (popsize,pop[1:popsize,],eval[1:popsize])
}
#evaluate crossover population
for(i in 1:popsize){
eval[i+offset,] <- evalSolution(pop[i+offset,],nsim)
}

#apply mutation
offset <- 2*popsize
for(i in 1:popsize){
pop[offset+i,] <- getMutationSolution(pop[i,],mrate,msigma,igen,minx,maxx,maxn)
}
#evaluate mutation population
for(i in 1:popsize){
eval[i+offset,] <- evalSolution(pop[i+offset,],nsim)
}

#order 3*popsize solutions
o <- order(eval[,1])
pop <- pop[o,]
eval <- eval[o,]
#save best
bSol[igen,] <- pop[1,]
bEval[igen,] <- eval[1,]
}#igen

c(eval0,pop0) # print criterion and best solution for generation 0
cbind(bEval,bSol) # print criterion and best solution for remaining generations
```