Timing Various Optimization Routines for the Aster Package

Charles J. Geyer

October 30, 2005

1 Preliminaries

1.1 Library and Data

```
> library(aster)

Loading required package: trust

> packageDescription("aster")$Version

[1] "0.4"

> data(echinacea)
```

That's our package and the dataset for our examples.

We need to reshape the data.

```
> vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04", "hdct02",
+     "hdct03", "hdct04")
> redata <- reshape(echinacea, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"
```

Set up root data.

```
> redata <- data.frame(redata, root = 1)
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"
```

Set up aster model (graph structure and families)

```
> pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
> fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
> families()[fam]

[1] "bernoulli"        "bernoulli"        "bernoulli"        "bernoulli"
[5] "bernoulli"        "bernoulli"        "non.zero.poisson" "non.zero.poisson"
[9] "non.zero.poisson"
```

Add dummy variable that is "pseudo-covariate" that indicates which variables are head count variables.

```
> hdct <- grep("hdct", as.character(redata$varb))
> hdct <- is.element(seq(along = redata$varb), hdct)
> redata <- data.frame(redata, hdct = as.integer(hdct))
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"  "hdct"
```

1.2 Fit Model

Here's the model we use for our timing tests.

```
> aout4 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
+     pred, fam, varb, id, root, data = redata)
> summary(aout4, show.graph = TRUE)

Call:
aster.formula(formula = resp ~ varb + nsloc + ewloc + pop * hdct -
    pop, pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = redata)


Graphical Model:
 variable predecessor family
 ld02     root        bernoulli
 ld03     ld02        bernoulli
 ld04     ld03        bernoulli
 fl02     ld02        bernoulli
 fl03     ld03        bernoulli
 fl04     ld04        bernoulli
```

```
 hdct02   fl02        non.zero.poisson
 hdct03   fl03        non.zero.poisson
 hdct04   fl04        non.zero.poisson

                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.463638   0.178183  -8.214  < 2e-16 ***
varbfl03        -0.328488   0.265142  -1.239 0.215377
varbfl04        -0.349519   0.240775  -1.452 0.146601
varbhdct02       1.808447   0.258014   7.009 2.40e-12 ***
varbhdct03       1.825590   0.210824   8.659  < 2e-16 ***
varbhdct04       2.338955   0.197099  11.867  < 2e-16 ***
varbld02        -0.989146   0.311742  -3.173 0.001509 **
varbld03         0.776572   0.392815   1.977 0.048049 *
varbld04         3.918323   0.330729  11.848  < 2e-16 ***
nsloc            0.013600   0.001729   7.867 3.64e-15 ***
ewloc            0.006473   0.001725   3.753 0.000174 ***
popEriley:hdct  -0.173902   0.089753  -1.938 0.052676 .
popLf:hdct      -0.157706   0.096580  -1.633 0.102490
popNessman:hdct -0.315681   0.139822  -2.258 0.023962 *
popNWLF:hdct    -0.109268   0.083325  -1.311 0.189740
popSPP:hdct      0.023541   0.086552   0.272 0.785630
popStevens:hdct -0.127403   0.089518  -1.423 0.154677
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     hdct
```

and here's the ANOVA (analysis of deviance, log likelihood ratio test) table for these models

1.3 Prediction

Set the confidence level and critical value.

```
> conf.level <- 0.95
> crit <- qnorm((1 + conf.level)/2)
```

Construct new data for "typical" individuals (having zero-zero geometry) in each population.

```
> newdata <- data.frame(pop = levels(echinacea$pop))
> for (v in vars) newdata[[v]] <- 1
> newdata$root <- 1
> newdata$ewloc <- 0
> newdata$nsloc <- 0
```

And reshape it.

```
> renewdata <- reshape(newdata, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> hdct <- grep("hdct", as.character(renewdata$varb))
> hdct <- is.element(seq(along = renewdata$varb), hdct)
> renewdata <- data.frame(renewdata, hdct = as.integer(hdct))
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"  "hdct"

> names(renewdata)

[1] "pop"   "root"  "ewloc" "nsloc" "varb"  "resp"  "id"    "hdct"
```

Construct linear function to predict.

```
> nind <- nrow(newdata)
> nnode <- length(vars)
> amat <- array(0, c(nind, nnode, nind))
> for (i in 1:nind) amat[i, grep("hdct", vars), i] <- 1
```

We are finally ready to make a prediction.

```
> pout4 <- predict(aout4, varvar = varb, idvar = id, root = root,
+     newdata = renewdata, se.fit = TRUE, amat = amat)
```

2 Timing a Simulation

Create parameter for simulation.

```
> theta.hat <- predict(aout4, model.type = "cond", parm.type = "canon")
> theta.hat <- matrix(theta.hat, nrow = nrow(aout4$x), ncol = ncol(aout4$x))
> fit.hat <- pout4$fit
> beta.hat <- aout4$coefficients
```

We also need root data, and it will be simpler if we actually don't use the forms of the `aster` and `predict.aster` functions that take formulas (because then we don't have to cram the simulated data in a data frame and we avoid a lot of repetitive parsing of the same formulas)

```
> root <- aout4$root
> modmat <- aout4$modmat
> modmat.pred <- pout4$modmat
> x.pred <- matrix(1, nrow = dim(modmat.pred)[1], ncol = dim(modmat.pred)[2])
> root.pred <- x.pred
```

2.1 Using NLM

Now we're ready for a simulation

```
> save.time <- proc.time()
> set.seed(42)
> nboot <- 100
> cover <- matrix(0, nboot, length(fit.hat))
> for (iboot in 1:nboot) {
+     xstar <- raster(theta.hat, pred, fam, root)
+     aout4star <- aster(xstar, root, pred, fam, modmat, beta.hat,
+         method = "nlm", check.analyticals = FALSE)
+     pout4star <- predict(aout4star, x.pred, root.pred, modmat.pred,
+         amat, se.fit = TRUE)
+     upper <- pout4star$fit + crit * pout4star$se.fit
+     lower <- pout4star$fit - crit * pout4star$se.fit
+     cover[iboot, ] <- as.numeric(lower <= fit.hat & fit.hat <=
+         upper)
+ }
> pboot <- apply(cover, 2, mean)
> pboot.se <- sqrt(pboot * (1 - pboot)/nboot)
> cbind(pboot, pboot.se)
```

```
     pboot   pboot.se
[1,]  0.91 0.02861818
[2,]  0.98 0.01400000
[3,]  0.94 0.02374868
[4,]  0.91 0.02861818
[5,]  0.94 0.02374868
[6,]  0.96 0.01959592
[7,]  0.94 0.02374868
```

```
> elapsed.time.nlm <- proc.time() - save.time
> elapsed.time.nlm

[1] 269.97  36.57 320.34   0.00   0.00
```

2.2 Using Trust

Repeat the simulation. Use `trust` the new default.

```
> save.time <- proc.time()
> set.seed(42)
> cover <- matrix(0, nboot, length(fit.hat))
> for (iboot in 1:nboot) {
+     xstar <- raster(theta.hat, pred, fam, root)
+     aout4star <- aster(xstar, root, pred, fam, modmat, beta.hat)
+     pout4star <- predict(aout4star, x.pred, root.pred, modmat.pred,
+         amat, se.fit = TRUE)
+     upper <- pout4star$fit + crit * pout4star$se.fit
+     lower <- pout4star$fit - crit * pout4star$se.fit
+     cover[iboot, ] <- as.numeric(lower <= fit.hat & fit.hat <=
+         upper)
+ }
> pboot <- apply(cover, 2, mean)
> pboot.se <- sqrt(pboot * (1 - pboot)/nboot)
> cbind(pboot, pboot.se)

     pboot   pboot.se
[1,]  0.91 0.02861818
[2,]  0.98 0.01400000
[3,]  0.94 0.02374868
[4,]  0.91 0.02861818
[5,]  0.94 0.02374868
[6,]  0.96 0.01959592
[7,]  0.94 0.02374868

> elapsed.time.trust <- proc.time() - save.time
> elapsed.time.trust

[1] 79.12  2.11 86.47  0.00  0.00
```

2.3 Using Optim, Method L-BFGS-B

Repeat the simulation. Use `method = "L-BFGS-B"`.

```
> save.time <- proc.time()
> set.seed(42)
> cover <- matrix(0, nboot, length(fit.hat))
> for (iboot in 1:nboot) {
+     xstar <- raster(theta.hat, pred, fam, root)
+     aout4star <- aster(xstar, root, pred, fam, modmat, beta.hat,
+         method = "L-BFGS-B")
+     pout4star <- predict(aout4star, x.pred, root.pred, modmat.pred,
+         amat, se.fit = TRUE)
+     upper <- pout4star$fit + crit * pout4star$se.fit
+     lower <- pout4star$fit - crit * pout4star$se.fit
+     cover[iboot, ] <- as.numeric(lower <= fit.hat & fit.hat <=
+         upper)
+ }
> pboot <- apply(cover, 2, mean)
> pboot.se <- sqrt(pboot * (1 - pboot)/nboot)
> cbind(pboot, pboot.se)

     pboot   pboot.se
[1,]  0.91 0.02861818
[2,]  0.98 0.01400000
[3,]  0.94 0.02374868
[4,]  0.91 0.02861818
[5,]  0.94 0.02374868
[6,]  0.96 0.01959592
[7,]  0.94 0.02374868

> elapsed.time.lbfgsb <- proc.time() - save.time
> elapsed.time.lbfgsb

[1] 1226.62  192.89 1479.54    0.00    0.00
```

2.4 Using Optim, Method CG

Repeat the simulation. Use `method = "CG"`.

```
> save.time <- proc.time()
> set.seed(42)
```

```
> cover <- matrix(0, nboot, length(fit.hat))
> for (iboot in 1:nboot) {
+     xstar <- raster(theta.hat, pred, fam, root)
+     aout4star <- aster(xstar, root, pred, fam, modmat, beta.hat,
+         method = "L-BFGS-B")
+     pout4star <- predict(aout4star, x.pred, root.pred, modmat.pred,
+         amat, se.fit = TRUE)
+     upper <- pout4star$fit + crit * pout4star$se.fit
+     lower <- pout4star$fit - crit * pout4star$se.fit
+     cover[iboot, ] <- as.numeric(lower <= fit.hat & fit.hat <=
+         upper)
+ }
> pboot <- apply(cover, 2, mean)
> pboot.se <- sqrt(pboot * (1 - pboot)/nboot)
> cbind(pboot, pboot.se)

     pboot   pboot.se
[1,]  0.91 0.02861818
[2,]  0.98 0.01400000
[3,]  0.94 0.02374868
[4,]  0.91 0.02861818
[5,]  0.94 0.02374868
[6,]  0.96 0.01959592
[7,]  0.94 0.02374868

> elapsed.time.cg <- proc.time() - save.time
> elapsed.time.cg

[1] 1224.24  194.50 1480.99    0.00    0.00
```

2.5 Summary

```
> tvec <- c(elapsed.time.trust[1], elapsed.time.nlm[1], elapsed.time.lbfgsb[1],
+     elapsed.time.cg[1])
> foo <- tvec
> hvec <- floor(foo/60^2)
> foo <- foo - hvec * 60^2
> mvec <- floor(foo/60)
> svec <- foo - mvec * 60
> foo <- hvec
> foo <- cbind(foo, mvec)
```

```
> foo <- cbind(foo, svec)
> foo <- cbind(foo, tvec/tvec[1])
> foo <- round(foo, 1)
> dimnames(foo) <- list(c("trust", "nlm", "optim L-BFGS-B", "optim CG"),
+     c("hours", "minutes", "seconds", "ratio"))
> if (all(foo[, 1] == 0)) foo <- foo[, -1]
> foo

               minutes seconds ratio
trust                1    19.1   1.0
nlm                  4    30.0   3.4
optim L-BFGS-B      20    26.6  15.5
optim CG            20    24.2  15.5
```

And info about the computer (if linux box)

```
> fred <- try(system("cat /proc/cpuinfo", intern = TRUE))
> sally <- try(system("hostname -f", intern = TRUE))
> if (!inherits(fred, "try-error")) {
+     cpuname <- fred[grep("model name", fred)]
+     cpusped <- fred[grep("cpu MHz", fred)]
+     cpuname <- cpuname[1]
+     cpusped <- cpusped[1]
+     cpuname <- sub("^[^:]*: *", "", cpuname)
+     cpusped <- sub("^[^:]*: *", "", cpusped)
+     cat("cpu:", cpuname, "\n")
+     cat("cpu speed:", cpusped, "MHz\n")
+ }

cpu: AMD Athlon(tm) XP 3000+
cpu speed: 2162.844 MHz

> if (!inherits(sally, "try-error")) {
+     cat("My name is", sally, "\n")
+ }

My name is oak.stat.umn.edu
```