APTS Statistical Computing: Assessment 2016/17

The work provided here is intended to take up to half a week to complete. Students should talk to their supervisors to find out whether or not their department requires this work as part of any formal accreditation process. It is anticipated that departments will decide on the appropriate *level* of assessment locally, and may choose to drop some (or indeed all) of the parts, accordingly. So make sure that your supervisor or local organizer of APTS assessment has looked at the assignment before you start, and has told you which parts of it to do. In order to avoid undermining institutions' local assessment procedures the module lecturer will not respond to enquiries from students about this assignment.

1. The theoretical definitions of many probability and statistics statements are replaced by alternative expressions that have more robust computational floating point arithmetic properties, e.g. asymptotic series for tail probabilities in cumulative distribution functions. Here, you will investigate a computational robustification of the Box-Cox transformation.

   The Box-Cox transformation $y_i^{(\lambda)} = G(y_i; \lambda)$ of positive data values y_i is defined via

   $$G(y;\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \lambda \neq 0, \\ \ln(y), & \lambda = 0. \end{cases}$$

   In mathematical theory, the $\lambda = 0$ case is the limit of $G(y;\lambda)$ as $\lambda \to 0$, but in computational practice, cancellation error is introduced for small λ and $\ln(y)$ values. One way of reducing the error is to replace the expression with a few terms of the Taylor expansion around $\lambda \ln(y) = 0$. For simplicity we can assume that the error is caused by y^λ being evaluated as $\texttt{comp}\{y^\lambda\} = y^\lambda(1+\epsilon)$ for some $|\epsilon| \approx \epsilon_0$, greater than or equal to the machine precision.

   Note: If we were to use the results in practice, the appropriate value for ϵ_0 needs to be determined; doing that is not part of the assignment.

   Hint: $y^\lambda = e^{\lambda \ln(y)}$ and $e^x = \sum_{k=0}^\infty \frac{x^k}{k!}$

   (a) Find an expression for an approximate error bound for $\texttt{comp}\left\{\frac{y^\lambda - 1}{\lambda}\right\}$, valid for small $\lambda \ln(y)$ values.

   (b) Derive a Taylor expansion of $G(y;\lambda)$ valid for small values of $\lambda \ln(y)$.

   (c) For $|\lambda \ln(y)| \leq \delta < 1$, the Taylor series of order one has an error that can be bounded by $C\delta^2 |ln(y)|$ for some constant C. Find C.

   (d) Find a value $\delta > 0$ such that the first degree Taylor approximation error bound is smaller than the error bound for $\texttt{comp}\left\{\frac{y^\lambda - 1}{\lambda}\right\}$ for all $|\lambda \ln(y)| \leq \delta$

   (e) Take the `boxcox` template function below and modify it to implement the robustified Box-Cox transformation you have now derived.

```
## y: vector of values > 0
## lambda: arbitrary scalar
boxcox <- function(y, lambda) {
    eps <- 1e6 * .Machine$double.eps
    delta <- NA ### What is \delta?
    use.taylor <- abs(lambda * log(y)) <= delta
    result <- numeric(length(y))
    if (any(!use.taylor)) {
      result[!use.taylor] <- (y[!use.taylor]^lambda - 1) / lambda
    }
    if (any(use.taylor)) {
      logy <- log(y[use.taylor])
      result[use.taylor] <- NA ### What is the Taylor approximation?
    }
    result
}
```

(f) The approximation derived above may not be appropriate for numerical derivatives of the Box-Cox transformation. Why? Suggest a method for fixing that problem.

2. A traditional setup for measuring the *gravitational constant* is to have two spheres suspended in an arrangement that leads to an dampened oscillating movement, where the oscillation frequency is related to the gravitational constant. Taking measurement error into account, a reasonable model for the measurements $(y_1, y_2, \ldots, y_n)$ of the positions at times $(t_1, t_2, \ldots, t_n)$ is given by

$$y_i = \alpha e^{-\beta t_i} \cos(\omega t_i + \gamma) + \epsilon_i,$$

where α, β, ω are positive parameters, $\gamma \in [0, 2\pi)$ is a phase shift parameter, and ϵ_i are iid $\mathrm{N}(0, \sigma^2)$.

(a) Run the following code to generate a synthetic data set:

```
model.mean <- function(time, theta) {
  (theta[1] * exp(-theta[2] * time) *
   cos(theta[3] * time + theta[4]))
}
set.seed(12345L)
theta00 <- c(10, 0.02, 0.1, 0.2, log(0.1))
data <- data.frame(time=0:200,
                   y=(model.mean(0:200, theta00) +
                      rnorm(201, sd=exp(theta00[5]))))
plot(data$time, data$y)
```

(b) Complete the function below for evaluating the negated log-likelihood for $\theta = (\alpha, \beta, \omega, \gamma, \log(\sigma))$:

```
## theta = c(\alpha, \beta, \omega, \gamma, \sigma)
negloglike <- function(theta, data) {
  ## Use model.mean(...) in combination with -sum(dnorm(..., log=TRUE))
  ## to evaluate the negated log-likelihood
}
```

(c) In the following, you may treat all the parameters as having no constraints (except σ, as shown in the code above). This leads to a non-identifiable model. Explain some of the difficulties this may lead to when using numerical optimisation to estimate ω and γ in particular. You will use `optim` to find the maximum likelihood estimate of θ, starting from `theta.start <- c(5, 0.05, 0.05, 0.05, 0)`. The following code shows the shape of the negative log-likelihood function around the starting point:

```
## Draw the local shape of the target function:
par(mfrow=c(2,2))
curve(vapply(x, function(x) negloglike(
        theta.start*c(0,1,1,1,1)+c(x,0,0,0,0), data), 1.0),
      -10, 10, n=10000)
curve(vapply(x, function(x) negloglike(
        theta.start*c(1,0,1,1,1)+c(0,x,0,0,0), data), 1.0),
      -0.1, 0.1, n=10000)
curve(vapply(x, function(x) negloglike(
        theta.start*c(1,1,0,1,1)+c(0,0,x,0,0), data), 1.0),
      -1, 1, n=10000)
curve(vapply(x, function(x) negloglike(
        theta.start*c(1,1,1,0,1)+c(0,0,0,x,0), data), 1.0),
      -10, 10, n=10000)
par(mfrow=c(1,1))
```

(d) Try to optimise `negloglike` and inspect the output:

```
opt <- optim(theta.start, negloglike, data=data)
rbind(theta00, opt$par)
opt$convergence
opt$counts
```

Is the result close to the true θ values? If no, what went wrong?

(e) Fix the optimisation problem by setting a suitable `control` option in the call to `optim`.

(f) Use the observed Fisher information to construct an approximate 95% confidence interval for ω. Hint: `optimHess` can be used to obtain a numerical evaluation of the Hessian of the negative log-likelihood.