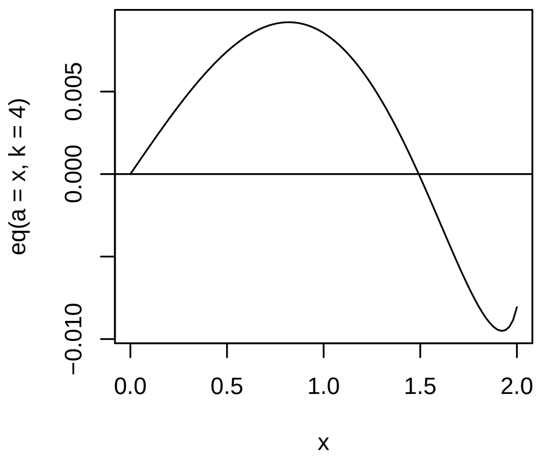
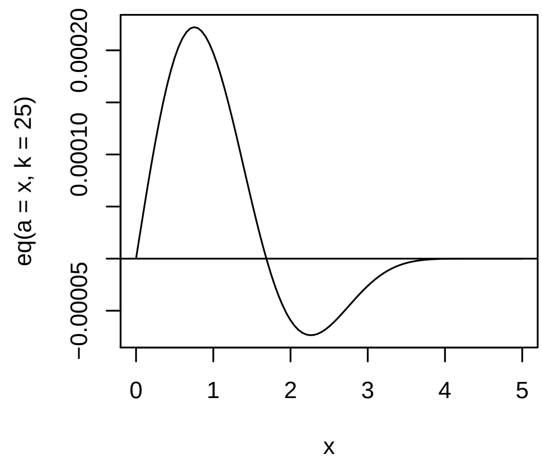
11.1 The equality $log(exp(x)) = exp(log(x))$ does not hold in computer computations because of insufficient precision.

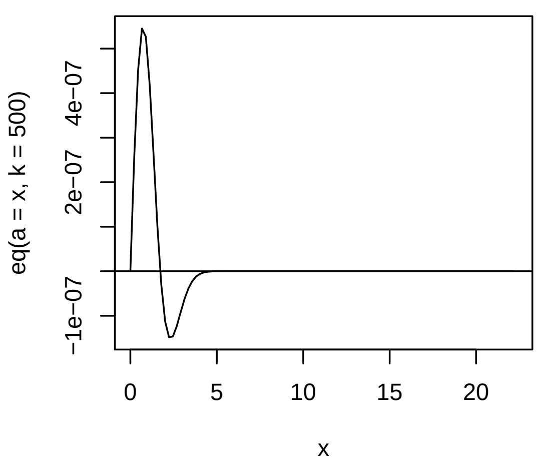
```
x = 10
log(exp(x)) == exp(log(x))

## [1] FALSE

# Using all.equal we can check if the identity holds with near equality
isTRUE(all.equal(log(exp(x)), exp(log(x))))

## [1] TRUE


# interestingly R can habdle the case where x=0 or 1 setting both sides to
# precisely 1 or 0 respectively
x = 1
log(exp(x)) == exp(log(x))

## [1] TRUE

x = 0
log(exp(x)) == exp(log(x))

## [1] TRUE
```

11.4 Intersection points of two curves in a bounded region (o,$sqrtk$).

```
K <- c(4:25, 100, 500, 1000)
eq <- function(a, k) {
    # since we are looking for p(t>x) we can use pt with the option
    # lower.tail=FALSE
    sk1 <- pt(sqrt((a^2 * (k - 1))/(k - a^2)), df = k - 1, lower.tail = F)
    sk <- pt(sqrt(((a^2) * k)/(k + 1 - a^2)), df = k, lower.tail = F)
    return(sk1 - sk)
}
par(mfrow = c(2, 2))
curve(eq(a = x, k = 4), xlim = c(0, 2))
abline(a = 0, b = 0)
curve(eq(a = x, k = 25), xlim = c(0, 5))
abline(a = 0, b = 0)
curve(eq(a = x, k = 500), xlim = c(0, sqrt(500)))

## Warning:  NaNs produced

abline(a = 0, b = 0)
par(mfrow = c(1, 1))
```




```
# it appears we can bound the search by 0.1 and 2 instead of 0 and sqrt(k)
roots <- numeric(length(K))
for (i in 1:length(K)) {
    roots[i] <- uniroot(f = eq, interval = c(0.1, 2), k = K[i])$root
}
cbind(k, roots)

## Error:  object 'k' not found
```

11.6 Cauchy CDF.

```r
cauch.pdf <- function(x, eta, theta) {
    return(1/(theta * pi * (1 + ((x - eta)/theta)^2)))
}
p.cauch <- function(q, theta, eta, lower.tail = TRUE) {
    return(integrate(f = cauch.pdf, lower = -Inf, upper = q, theta = theta,
        eta = eta)$value)
}
quants <- matrix(1:5, 5, 1)
mine <- apply(quants, 1, p.cauch, theta = 1, eta = 0)
bases <- pcauchy(quants)
cbind(quants, mine, bases)
```

```
##          mine
## [1,] 1 0.7500 0.7500
## [2,] 2 0.8524 0.8524
## [3,] 3 0.8976 0.8976
## [4,] 4 0.9220 0.9220
## [5,] 5 0.9372 0.9372
```

```r
for (i in 1:5) {
    print(isTRUE(all.equal(mine[i], bases[i])))
}
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

```r
# the quantiles compare very favorably when using the standard cauchy
mine <- apply(quants, 1, p.cauch, theta = 2, eta = 2)
bases <- pcauchy(quants, location = 2, scale = 2)
cbind(quants, mine, bases)
```

```
##          mine
## [1,] 1 0.3524 0.3524
## [2,] 2 0.5000 0.5000
## [3,] 3 0.6476 0.6476
## [4,] 4 0.7500 0.7500
## [5,] 5 0.8128 0.8128
```

```r
for (i in 1:5) {
    print(isTRUE(all.equal(mine[i], bases[i])))
}
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

```r
# changing the scale and location parameters does not affect the precision
# outside of computer tolerance
```

11.7 Application of the simplex method.

```
library(boot)
A1 <- rbind(c(2, 1, 1), c(1, -1, 3))
b1 <- c(2, 3)
a <- c(4, 2, 9)
simplex(a = a, A1 = A1, b1 = b1, maxi = TRUE)

##
## Linear Programming Results
##
## Call : simplex(a = a, A1 = A1, b1 = b1, maxi = TRUE)
##
## Maximization Problem with Objective Function Coefficients
## x1 x2 x3
##  4  2  9
##
##
## Optimal solution has the following values
##   x1   x2   x3
## 0.00 0.75 1.25
## The optimal value of the objective  function is 12.75.
```

11.A The complete log likelihood is:

$$f_{x,x_{n+1}}(X, X_{n+1}|\theta) = f_{x_{n+1}|x}(X_{n+1}|X, \theta) f_x(X|\theta)$$

where $X = \sum_{i=1}^{n} x_i \sim Gamma(n, \theta)$ since $x_1 ... x_n$ iid Exp($\theta$). Using the independence of $X$ and $X_{n+1}$ the full likelihood then becomes,

$l(\theta|X_{i...n}, X_{n+1}) = log\left[\frac{\theta^n X^{n-1} e^{-\theta X}}{\Gamma(n)}\right] + log\theta - \theta X_{n+1}$

$= nlog\theta + (n-1)logX - \theta X - log\Gamma(n) + log\theta - \theta X_{n+1}$

$= (n+1)log\theta + (n-1)logX - log\Gamma(n) - \theta(X + X_{n+1})$

**The E-step is then:** $Q(\theta, \theta_{t-1}) = E[l(\theta|X_{i...n}, X_{n+1})|X, \theta_{t-1}]$

$= E[(n+1)log\theta + (n-1)logX - log\Gamma(n) - \theta(X + X_{n+1})|X, \theta_{t-1}]$

$= (n+1)log\theta + (n-1)logX - log\Gamma(n) - \theta(X + E(X_{n+1}|\theta_{t-1}))$

$= (n+1)log\theta + (n-1)logX - log\Gamma(n) - \theta\left(X + \frac{1}{\theta_{t-1}}\right)$

**The M-step is then to maximize Q**:
$\frac{dQ}{d\theta} - x + \frac{1}{\theta_{t-1}}$, set equal to 0 and solve for $\theta$.

$\theta_t = \frac{(n+1)\theta_{t-1}}{x\theta_{t-1}+1}$

```r
tol = sqrt(.Machine$double.eps)

update = function(theta, x1, n) {
    ((n + 1) * theta)/(1 + x1 * theta)
}
Q = function(th, th1, x1, n) {
    (n + 1) * log(th) + (n - 1) * log(x1) - lgamma(n) - th * (x1 + (1/th1))
}

x <- c(840, 157, 145, 44, 33, 121, 150, 280, 434, 736, 584, 887, 263, 1901,
    695, 294, 562, 721, 76, 710, 46, 402, 194, 759, 319, 460, 40, 1336, 335,
    1354, 454, 36, 667, 40, 556, 99, 304, 375, 567, 139, 780, 203, 436, 30,
    384, 1299, 9, 209, 599, 83, 832, 328, 126, 1617, 638, 937, 735, 38, 365,
    92, 82, 220)
x1 = sum(x)  # observed datum
n = 62
theta = 1  #theta0 initial guess
thetaold = theta

while (abs(Q(update(theta, x1, n), theta, x1, n) - Q(theta, thetaold, x1, n)) >
    tol) {
    thetaold = theta
    theta = update(theta, x1, n)
    print(theta)
}

## [1] 0.002237
## [1] 0.002202
## [1] 0.002202
## [1] 0.002202
## [1] 0.002202
## [1] 0.002202
```