Supplement S1

Keil P. (2014) Limits of uncertainty about estimates of probability of ecological events. PeerJ preprint.

Description: This supplementary material provides R codes for probability density function, cumulative distribution function, quantile function and random number generator of the truncated exponential distribution on the 0 to 1 interval. The same codes as below are also provided in Supplement S2 in the form of a raw text file.

```r
# CALCULATING MU FROM ALPHA
# -------------------------
# Arguments: aphas - numeric vector (or scalar) of alpha values
# Value: numeric vector of means mu

  alpha.to.mu <- function(alphas)
  {
    mus <- numeric(length(alphas))
    for(i in 1:length(alphas))
    {
      if(alphas[i]==0) mus[i] <- 0.5
      else if(alphas[i]>700) mus[i] <- 0.99857 # approximation for extremes
      else if(alphas[i]<(-700)) mus[i] <- 0.00143 # approximation for extremes
      else mus[i] <- exp(alphas[i]) / (exp(alphas[i])-1) - 1/alphas[i]
    }
    return(mus)
  }

# CALCULATING ALPHA  FROM MU
# -------------------------------
# Arguments: mus - numeric vector (or scalar) of means mu
# Value: numeric vector of alpha values

  mu.to.alpha <- function(mus)
  {
    f <- function(alpha, mu)
    {
      # square root of the difference is the criterion
      # to be minimized by optimize:
      criterion <- (mu - alpha.to.mu(alpha))^2
      return(criterion)
    }

    alphas <- numeric(length(mus))
    for(i in 1:length(mus))
    {
      # the one-dimensional optimization from the stats pacakge:
      optim.alpha <- optimize(f, interval=c(-700,700), mu=mus[i])$minimum
      alphas[i] <- optim.alpha
    }
    return(alphas)
  }
```

```
# PROBABILITY DENSITY FUNCTION
# ----------------------------
# Arguments:
#   P - numeric vector or scalar of P values
#   alpha - a scalar
# Value: numeric vector of probability densities

  PDF <- function(P, alpha)
  {
    # the limiting case where the PDF is uniform:
    if(alpha==0) pd <- rep(1, times=length(P))
    # else, proceed according to the formula
    else  pd <- (alpha*exp(alpha*P) ) / ( exp(alpha) - 1)
    return(pd)
  }

# CUMULATIVE DISTRIBUTION FUNCTION
# --------------------------------
# Arguments:
#   P - numeric vector or scalar of P values
#   alpha - a scalar
# Value: numeric vector of cumulative densities

  CDF <- function(P, alpha)
  {
    alpha*(exp(alpha*P)/alpha - 1/alpha)/(exp(alpha) - 1)
    (exp(alpha*P) - 1)/(exp(alpha) - 1)
  }

# QUANTILE FUNCTION
# -----------------
# Arguments:
#   Q - numeric vector or quantiles
#   alpha - a scalar
# Value: numeric vector of P values

  QF <- function(Q, alpha)
  {
    # function to be solved
    f <- function(P, fixed)
    {
      alpha <- fixed$alpha
      Q <- fixed$Q
      criterion <- Q - CDF(P, alpha)
      return(criterion)
    }

    P <- numeric(length(Q))
    for(i in 1:length(Q))
    {
      fixed <- list(alpha=alpha, Q=Q[i])
      # solving the f numerically by uniroot()
      root.p <- uniroot(f, lower=0, upper=1, fixed=fixed)
      P[i] <-root.p$root
    }
    return(P)
  }
```

```r
# RANDOM NUMBER GENERATOR (the inverse transform sampling)
# ---------------------------------------------------------
# Arguments:
#   N - number of random numbers to be drawn from the PDF
#   alpha - a scalar
# Value: numeric vector of the random draws

  RG <- function(N, alpha)
  {
    U <- runif(N, min=0, max=1)
    rnd.draws <- QF(U, alpha)
    return(rnd.draws)
  }
```