R Implementation of MUPE for General Nonlinear Models

Define function

```
# General nonlinear instantiation of the Minimum Unbiased Percent Error technique (MUPE)
# for multiplicative error models, which utilizes Iteratively Re-weighted Least Squares
# (IRLS) with weights equal to the squared inverse predictions from the prior iteration.
# Utilizes the MINPACK library implementation of the Levenberg-Marquardt algorithm. To
# install this, run 'install.packages("minpack.lm")' from the R console.
# Usage Example:
#   mlist = mupe_nonlinear(formula_str = "y ~ a * x1^b * c^x2", data = df,
#                          start = c(a=10, b=1, c=1))
#     - 'formula_str' must be a character string that resembles an R 'nls' formula object
#     - 'data' must be a dataframe containing the variables listed in formula_str
#     - 'start' is a named numeric vector of initial guesses (parameter names must
#       match those in formula_str). Whenever possible, provide values of the correct
#       sign and order of magnitude. For log-linear model forms, use the LOLS or PING
#       solution as the initial guess.
# Returns a list containing a standard R 'nls' object and accompanying details.
#
mupe_nonlinear = function(formula_str, data, start) {
  library(minpack.lm)                 # load Levenberg-Marquardt algorithm
  f = as.formula(formula_str)         # convert string to R formula object
  wt = rep(1, nrow(data))             # initialize weights
  pbeta = start;  conv = 1.0;  i = 0  # initialize other variables
  while (conv > 1e-5) {
    model = suppressWarnings(nlsLM(f, data, pbeta, weights=wt, control=list(maxiter=10)))
    wt = 1 / model$m$pred()^2           # calculate weights
    beta = model$m$getAllPars()         # solution of current iteration
    conv = max(abs((beta-pbeta)/beta))  # maximum fractional change in any parameter
    pbeta = beta                        # reset prior beta
    i = i + 1;  if (i == 200) break     # force stop, if necessary
  }
  return(list(model=model, start=start, mupe_iters=i))
}
```

Generate data to demonstrate equation of the form $y = b_0 * x_1^{b_1}$

```
# Simulate data
set.seed(0);  n = 20;  x1 = runif(n, 1000, 8000);  y = 90 * x1^0.8
# Apply multiplicative lognormal random error term with mean=1, cv=0.4
cv = 0.4;  loc = log(1 / sqrt(cv^2 + 1));  shape = sqrt(log(1 + cv^2))
y = y*rlnorm(n, loc, shape);  my_df = data.frame('y'=y, 'x1'=x1)
```

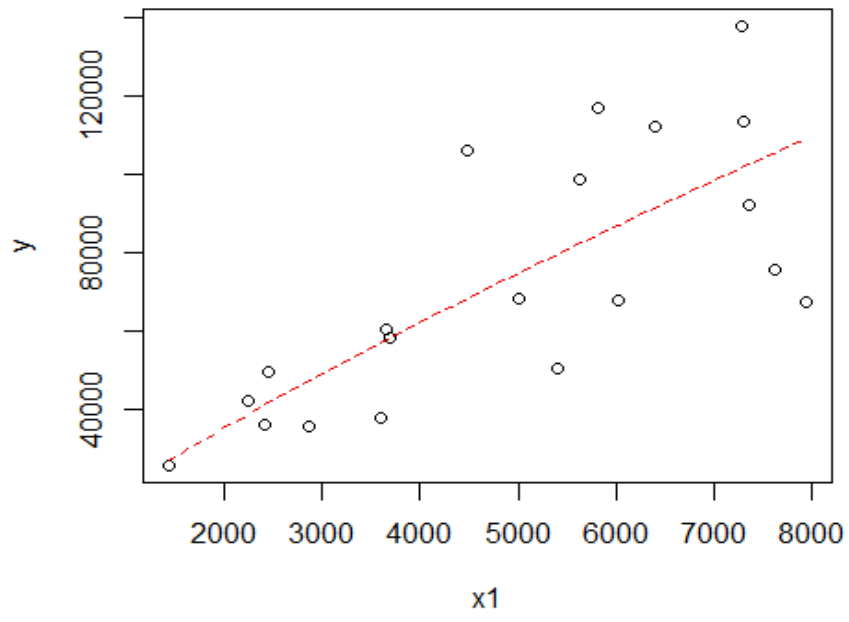
Apply method

```
my_mupe = mupe_nonlinear('y ~ b0 * x1^b1', my_df, c(b0=10, b1=1))

my_mupe  # output: final model, starting guess, and number of outer iterations

## $model
## Nonlinear regression model
##   model: y ~ b0 * x1^b1
##    data: data
##      b0      b1
## 68.3228  0.8214
##  weighted residual sum-of-squares: 1.359
##
## Number of iterations to convergence: 1
```

```
## Achieved convergence tolerance: 1.49e-08
##
## $start
## b0 b1
## 10  1
##
## $mupe_iters
## [1] 5

summary(my_mupe$model)  # mupe$model output is a standard R 'nls' object

##
## Formula: y ~ b0 * x1^b1
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## b0  68.3228    73.8887   0.925    0.367
## b1   0.8214     0.1285   6.391 5.11e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2747 on 18 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 1.49e-08

# The mean percent error of the MUPE solution is approximately zero
mean(my_mupe$model$m$resid() / my_mupe$model$m$fitted())

## [1] -3.138672e-08
```

Overlay fitted curve on scatterplot

```
par(mar=c(4.5,4.5,1,1))
plot(x1, y, xlab='x1', ylab='y')
xvec = seq(min(x1), max(x1), length.out=100)
lines(xvec, my_mupe$model$m$predict(data.frame('x1'=xvec)), col='red2', lty=2)
```



Generate data to demonstrate equation of the form $y = b_0 + b_1 x_1^{b_2}$

```
# Simulate data
set.seed(3);  n = 40;  x1 = runif(n, 30, 160);  y = 100 + 4*x1^1.2
# Apply multiplicative gamma random error term with mean=1, cv=0.2
cv = 0.3;  y = y*rgamma(n, 1/cv^2, 1/cv^2);  my_df = data.frame('y'=y, 'x1'=x1)
```

Apply method

```
my_mupe = mupe_nonlinear('y ~ b0 + b1*x1^b2', my_df, c(b0=10, b1=1, b2=1))
summary(my_mupe$model)

##
## Formula: y ~ b0 + b1 * x1^b2
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## b0 206.9082   198.6982   1.041  0.30448
## b1   0.9030     2.4677   0.366  0.71649
## b2   1.4870     0.5382   2.763  0.00887 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2793 on 37 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 1.49e-08

# The mean percent error of the MUPE solution is approximately zero
mean(my_mupe$model$m$resid() / my_mupe$model$m$fitted())

## [1] -2.952886e-12
```

Overlay fitted curve on scatterplot

```
par(mar=c(4.5,4.5,1,1))
plot(x1, y, xlab='x1', ylab='y')
xvec = seq(min(x1), max(x1), length.out=100)
lines(xvec, my_mupe$model$m$predict(data.frame('x1'=xvec)), col='red2', lty=2)
```