R Implementation of MUPE for Linear Models

Define function

```
# Linear 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.
# Usage Example:
#   mupe = mupe_linear(formula_str = "y ~ 0 + x", data = df)
#     - 'formula_str' must be a character string that resembles an R 'lm' formula object
#       (default is no intercept, i.e. a simple factor model)
#     - 'data' must be a dataframe containing the variables listed in formula_str
# Returns a list containing a standard R 'lm' object and the number of iterations.
#
mupe_linear = function(formula_str = "y ~ 0 + x", data) {
  f = as.formula(formula_str)           # convert string to R formula object
  model = lm(f, data)                   # 1st iteration (Ordinary Least Squares)
  conv = 1.0;  i = 1                    # initialize convergence and counter
  while (conv > 1e-5) {
    wt = 1 / model$fitted^2             # calculate weights
    pbeta = model$coef                  # solution of prior iteration
    model = lm(f, data, weights=wt)     # weighted least squares
    beta = model$coef                   # solution of current iteration
    conv = max(abs((beta-pbeta)/beta))  # maximum fractional change in any parameter
    i = i + 1;  if (i == 200) break     # force stop, if necessary
  }
  return(list(model=model, mupe_iters=i))
}
```

Generate data to demonstrate equation of the form $y = b_0 + b_1 x_1$

```
# Simulate data
set.seed(0);  n = 20;  x1 = runif(n, 20, 150);  y = 180 + 6*x1
# Apply multiplicative lognormal random error term with mean=1, cv=0.3
cv = 0.3;  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_linear('y ~ x1', my_df)

my_mupe  # output: final model and number of iterations

## $model
##
## Call:
## lm(formula = f, data = data, weights = wt)
##
## Coefficients:
## (Intercept)           x1
##     158.708        5.739
##
##
## $mupe_iters
## [1] 5
```

```
summary(my_mupe$model)  # mupe$model is a standard R 'lm' object
```

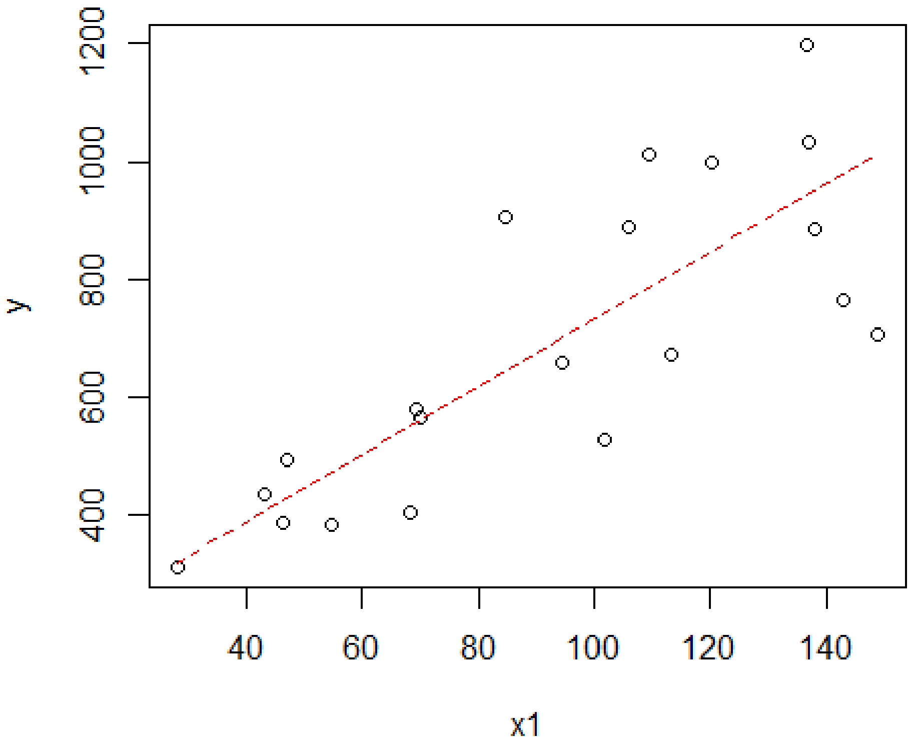
```
## 
## Call:
## lm(formula = f, data = data, weights = wt)
## 
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.30411 -0.17301 -0.00728  0.15494  0.40601 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 158.7081    60.4194   2.627   0.0171 *  
## x1            5.7392     0.7947   7.222 1.02e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.2102 on 18 degrees of freedom
## Multiple R-squared:  0.7434, Adjusted R-squared:  0.7292 
## F-statistic: 52.15 on 1 and 18 DF,  p-value: 1.02e-06
```

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

```
## [1] -2.440756e-15
```

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, predict(my_mupe$model, data.frame('x1'=xvec)), col='red2', lty=2)
```


y
1200
1000
800
600
400
40
60
80
100
120
140
x1