Econ Models Using R Packages

Emmanuel S. Tsyawo

10/16/2021

Contents

Installing an R package **2**

Ordinary Least Squares **2**
An example from the mroz data . . . 2
An example from the AER package . . . 2

Instrumental Variable (IV) Regression **4**

Binary response models **6**
Probit model . . . 6
Logit model . . . 7

Count data models **8**
Poisson regression: . . . 8
Negative binomial regression . . . 9

Quantile Regression **10**

Installing an R package

R packages collect functions which are accessible for download and installation. This is a very convenient means of making programmes available to an entire community of users.

Let us install the package AER. You need internet connection for this step. Skip this step if you already have the package.

```
#install.packages("AER")
```

Ordinary Least Squares

This is just a revision of 1.1. Let us reproduce the code here and take a closer look at our results. Recall, the essence lies in the interpretation/meaning of our results.

A prototypical call of a linear regression looks like `fm <- lm(formula, data, ...)`.

An example from the mroz data

Outcome variable - nonwife income Exogenous variables - age, age^2, education, experience We allow for a quadratic term to allow for decreasing/increasing returns to age.

Let us load the data set. Ensure the working directory is set to the location of the data set.

```
dat<- read.csv("dat.csv",header = T,sep = " ") # load data set
```

Run OLS

```
reg1<- lm(dat$nonwife~dat$age + I(dat$age^2)+dat$education+dat$experience)
summary(reg1)
```

```
##
## Call:
## lm(formula = dat$nonwife ~ dat$age + I(dat$age^2) + dat$education +
##     dat$experience)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.494  -6.528  -1.903   3.651  67.241
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -31.373284  11.529632  -2.721  0.00666 **
## dat$age           1.434680   0.533934   2.687  0.00737 **
## I(dat$age^2)     -0.013608   0.006163  -2.208  0.02755 *
## dat$education     1.614511   0.174633   9.245  < 2e-16 ***
## dat$experience   -0.362535   0.051998  -6.972 6.87e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 748 degrees of freedom
## Multiple R-squared:  0.1476, Adjusted R-squared:  0.1431
## F-statistic: 32.39 on 4 and 748 DF,  p-value: < 2.2e-16
```

An example from the AER package

Let us load the package into memory

```r
library("AER")
```

```
## Loading required package: car

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.5

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: survival
```

Let us load data from the package

```r
data("Journals",package = "AER")
names(Journals) #view variable names
```

```
##  [1] "title"        "publisher"    "society"      "price"        "pages"
##  [6] "charpp"       "citations"    "foundingyear" "subs"         "field"
```

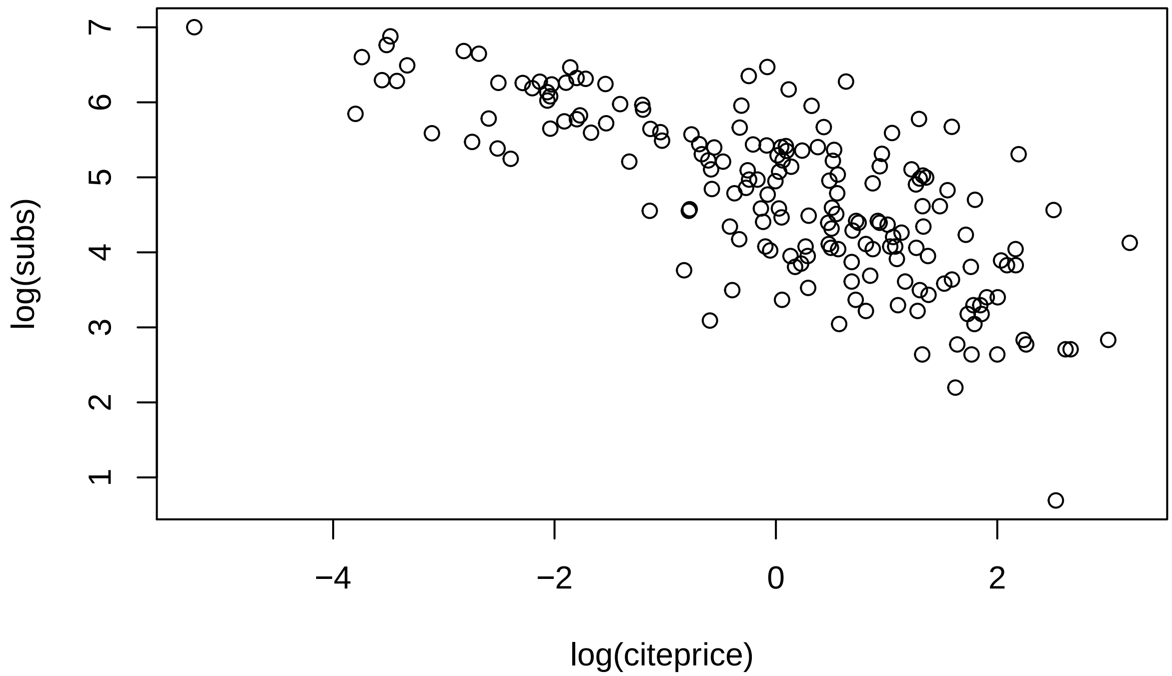
Explore the data

```r
journals <- Journals[, c("subs", "price")] # create a data frame of two
# variables subs, price
journals$citeprice <- Journals$price/Journals$citations # generate new
# variable citeprice and add it to the data frame. Notice the $ sign
summary(journals) # a summary of the data frame
```

```
##       subs             price          citeprice
##  Min.   :   2.0   Min.   :  20.0   Min.   : 0.005223
##  1st Qu.:  52.0   1st Qu.: 134.5   1st Qu.: 0.464495
##  Median : 122.5   Median : 282.0   Median : 1.320513
##  Mean   : 196.9   Mean   : 417.7   Mean   : 2.548455
##  3rd Qu.: 268.2   3rd Qu.: 540.8   3rd Qu.: 3.440171
##  Max.   :1098.0   Max.   :2120.0   Max.   :24.459459
```

The goal is to estimate the effect of the price per citation on the number of library subscriptions.

```r
plot(log(subs) ~ log(citeprice), data = journals) # visualise relationship between variables.
```



Run a linear regression

```
jour_lm <- lm(log(subs) ~ log(citeprice), data = journals) # create regression object
summary(jour_lm)
```

```
##
## Call:
## lm(formula = log(subs) ~ log(citeprice), data = journals)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72478 -0.53609  0.03721  0.46619  1.84808
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.76621    0.05591   85.25   <2e-16 ***
## log(citeprice) -0.53305    0.03561  -14.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7497 on 178 degrees of freedom
## Multiple R-squared:  0.5573, Adjusted R-squared:  0.5548
## F-statistic:   224 on 1 and 178 DF,  p-value: < 2.2e-16
```

Instrumental Variable (IV) Regression

The example used is taken from the documentation in the `ivreg()` function of the `AER` package. To view the documentation of this function, type `?ivreg` into the console.

Compute additional variables needed in the regression.

```
data("CigarettesSW", package = "AER")
CigarettesSW$rprice <- with(CigarettesSW, price/cpi)
CigarettesSW$rincome <- with(CigarettesSW, income/population/cpi)
```

```r
CigarettesSW$tdiff <- with(CigarettesSW, (taxs - tax)/cpi)
```

The following runs a linear IV regression where `tdiff` and `tax/cpi` are used as excluded instruments for `log(rprice)`.

```r
## model
fm <- ivreg(log(packs) ~ log(rprice) + log(rincome) | log(rincome) + tdiff + I(tax/cpi), data = Cigaret
summary(fm)
```

```
##
## Call:
## ivreg(formula = log(packs) ~ log(rprice) + log(rincome) | log(rincome) +
##     tdiff + I(tax/cpi), data = CigarettesSW, subset = year ==
##     "1995")
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.6006931 -0.0862222 -0.0009999  0.1164699  0.3734227
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    9.8950     1.0586   9.348 4.12e-12 ***
## log(rprice)   -1.2774     0.2632  -4.853 1.50e-05 ***
## log(rincome)   0.2804     0.2386   1.175    0.246
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1879 on 45 degrees of freedom
## Multiple R-Squared: 0.4294,  Adjusted R-squared: 0.4041
## Wald test: 13.28 on 2 and 45 DF,  p-value: 2.931e-05
```

```r
summary(fm, vcov = sandwich, df = Inf, diagnostics = TRUE) #use robust standard errors with model diagn
```

```
##
## Call:
## ivreg(formula = log(packs) ~ log(rprice) + log(rincome) | log(rincome) +
##     tdiff + I(tax/cpi), data = CigarettesSW, subset = year ==
##     "1995")
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.6006931 -0.0862222 -0.0009999  0.1164699  0.3734227
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    9.8950     0.9288  10.654  < 2e-16 ***
## log(rprice)   -1.2774     0.2417  -5.286 1.25e-07 ***
## log(rincome)   0.2804     0.2458   1.141    0.254
##
## Diagnostic tests:
##                  df1 df2 statistic p-value
## Weak instruments   2  44   228.738  <2e-16 ***
## Wu-Hausman         1  44     3.823  0.0569 .
## Sargan             1  NA     0.333  0.5641
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1879 on Inf degrees of freedom
## Multiple R-Squared: 0.4294,  Adjusted R-squared: 0.4041
## Wald test: 34.51 on 2 DF,  p-value: 3.214e-08
```

Binary response models

Binary response models are suitable for binary outcomes. Two popular choices include the Probit and Logit models. Like other non-linear models in general, parameter estimates are not directly interpretable or of interest to the researcher.

Probit model

We begin with a probit regression:

```
data("SwissLabor") # load data from the AER package
swiss_probit <- glm(participation ~ . + I(age^2),
                   data = SwissLabor, family = binomial(link = "probit"))
```

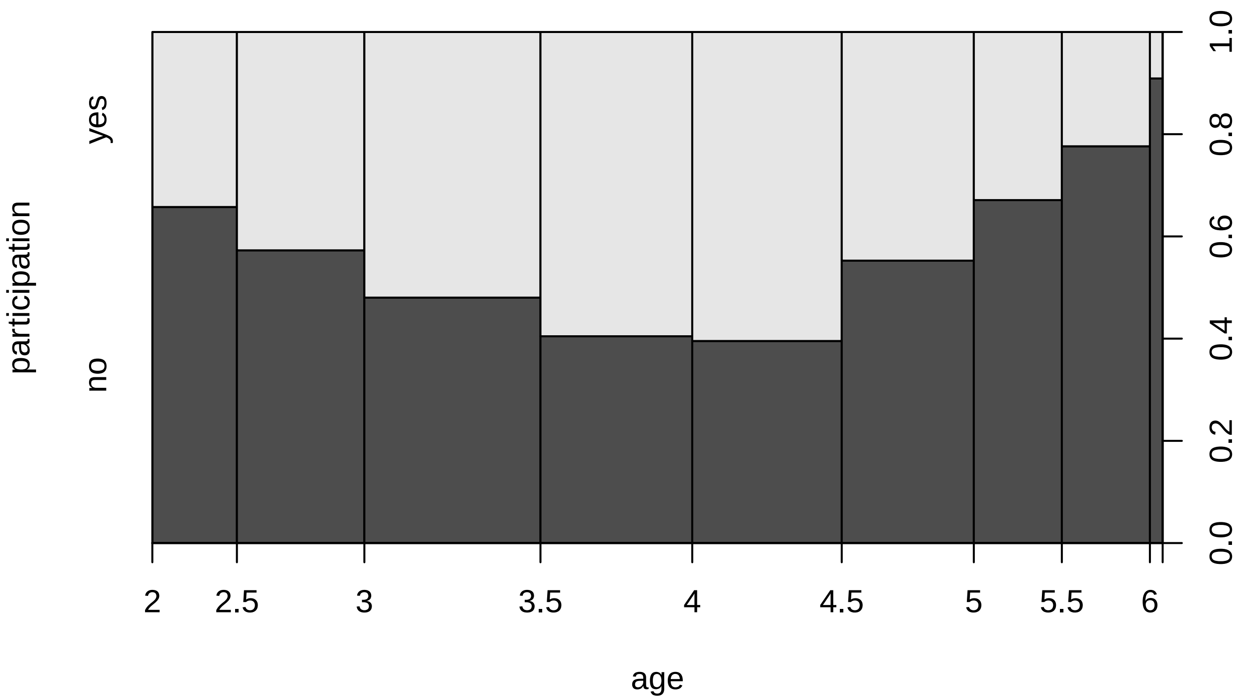
The use of `~.` implies regress participation on all other variables in the data set and a quadratic term of age.

```
summary(swiss_probit) # summary of results
```

```
##
## Call:
## glm(formula = participation ~ . + I(age^2), family = binomial(link = "probit"),
##     data = SwissLabor)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9191  -0.9695  -0.4792   1.0209   2.4803
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.74909    1.40695   2.665  0.00771 **
## income      -0.66694    0.13196  -5.054 4.33e-07 ***
## age          2.07530    0.40544   5.119 3.08e-07 ***
## education    0.01920    0.01793   1.071  0.28428
## youngkids   -0.71449    0.10039  -7.117 1.10e-12 ***
## oldkids     -0.14698    0.05089  -2.888  0.00387 **
## foreignyes   0.71437    0.12133   5.888 3.92e-09 ***
## I(age^2)    -0.29434    0.04995  -5.893 3.79e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1203.2  on 871  degrees of freedom
## Residual deviance: 1017.2  on 864  degrees of freedom
## AIC: 1033.2
##
## Number of Fisher Scoring iterations: 4
```

Visualisation (using a spinogram)

```
plot(participation ~ age, data = SwissLabor, ylevels = 2:1)
```



By setting ylevels = 2:1, the order of participation levels is reversed, highlighting participation (rather than non-participation).

Interpretation of results using the average partial effects. The j'th partial effect of the i'th observation is given by $PE_{ij} = \phi(X_i\beta) \times \beta_j$ where $\phi(\cdot)$ denotes the PDF of the standard normal distribution.

```
favp <- mean(dnorm(predict(swiss_probit, type = "link")))
ape_p = favp * coef(swiss_probit) # these give the average partial effects
ape_p
```

```
##  (Intercept)       income          age    education    youngkids      oldkids
##  1.241929965 -0.220931858  0.687466185  0.006358743 -0.236682273 -0.048690170
##   foreignyes     I(age^2)
##  0.236644422 -0.097504844
```

Exercise: How would you calculate their standard errors and/or asymptotic distributions?

Logit model

Run the regression in the preceding section using the logit model.

```
swiss_logit <- glm(participation ~ . + I(age^2),
                   data = SwissLabor, family = binomial(link = "logit"))
summary(swiss_logit)
```

```
##
## Call:
## glm(formula = participation ~ . + I(age^2), family = binomial(link = "logit"),
##     data = SwissLabor)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9061  -0.9627  -0.4924   1.0171   2.3915
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.19639    2.38309   2.600  0.00932 **
## income      -1.10409    0.22571  -4.892 1.00e-06 ***
## age          3.43661    0.68789   4.996 5.86e-07 ***
## education    0.03266    0.02999   1.089  0.27611
## youngkids   -1.18575    0.17202  -6.893 5.46e-12 ***
## oldkids     -0.24094    0.08446  -2.853  0.00433 **
## foreignyes   1.16834    0.20384   5.732 9.94e-09 ***
## I(age^2)    -0.48764    0.08519  -5.724 1.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1203.2  on 871  degrees of freedom
## Residual deviance: 1017.6  on 864  degrees of freedom
## AIC: 1033.6
##
## Number of Fisher Scoring iterations: 4
```

Compute the average partial effect. The j'th partial effect of the i'th observation is given by $PE_j = \lambda(X_i\beta)\times\beta_j$ where $\lambda(\cdot)$ denotes the PDF of the logistic distribution.

```r
favl <- mean(dlogis(predict(swiss_logit, type = "link")))
ape_l = favl * coef(swiss_logit) # these give the average partial effects
ape_l
```

```
##  (Intercept)       income          age    education    youngkids      oldkids
##  1.236910940 -0.220397098  0.686009624  0.006520208 -0.236696710 -0.048095386
##   foreignyes     I(age^2)
##  0.233222695 -0.097342199
```

Compare average partial effects:

```r
rbind(ape_p,ape_l)
```

```
##       (Intercept)     income       age   education  youngkids     oldkids
## ape_p    1.241930 -0.2209319 0.6874662 0.006358743 -0.2366823 -0.04869017
## ape_l    1.236911 -0.2203971 0.6860096 0.006520208 -0.2366967 -0.04809539
##       foreignyes    I(age^2)
## ape_p  0.2366444 -0.09750484
## ape_l  0.2332227 -0.09734220
```

What do you say?

Count data models

These models have count outcome variables, e.g., the number of items sold per day or the number of trips per month. Popular count data models include the Poisson and Negative-Binomial.

Poisson regression:

We begin with the standard model for count data, a Poisson regression.

First, let us load the data set.

```r
data("RecreationDemand") # load data from AER package
```

Fitting the model is as simple as

```r
rd_pois <- glm(trips ~ ., data = RecreationDemand,
                   family = poisson)
summary(rd_pois)
```

```
##
## Call:
## glm(formula = trips ~ ., family = poisson, data = RecreationDemand)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -11.8465   -1.1411   -0.8896   -0.4780   18.6071
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.264993   0.093722   2.827  0.00469 **
## quality      0.471726   0.017091  27.602  < 2e-16 ***
## skiyes       0.418214   0.057190   7.313 2.62e-13 ***
## income      -0.111323   0.019588  -5.683 1.32e-08 ***
## userfeeyes   0.898165   0.078985  11.371  < 2e-16 ***
## costC       -0.003430   0.003118  -1.100  0.27131
## costS       -0.042536   0.001670 -25.467  < 2e-16 ***
## costH        0.036134   0.002710  13.335  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4849.7  on 658  degrees of freedom
## Residual deviance: 2305.8  on 651  degrees of freedom
## AIC: 3074.9
##
## Number of Fisher Scoring iterations: 7
```

Can you interprete the above results?

Negative binomial regression

In R, the function for the negative binomial model is provided in the `MASS` package (Venables and Ripley 2002). Let us re-run the count data regression using the negative binomial model.

```r
library("MASS") # this package is in-built. you don't need to install it
rd_nb <- glm.nb(trips ~ ., data = RecreationDemand)
summary(rd_nb)
```

```
##
## Call:
## glm.nb(formula = trips ~ ., data = RecreationDemand, init.theta = 0.7292568331,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9727  -0.6256  -0.4619  -0.2897   5.0494
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.121936   0.214303  -5.235 1.65e-07 ***
## quality      0.721999   0.040117  17.998  < 2e-16 ***
## skiyes       0.612139   0.150303   4.073 4.65e-05 ***
## income      -0.026059   0.042453  -0.614    0.539
## userfeeyes   0.669168   0.353021   1.896    0.058 .
## costC        0.048009   0.009185   5.227 1.72e-07 ***
## costS       -0.092691   0.006653 -13.931  < 2e-16 ***
## costH        0.038836   0.007751   5.011 5.42e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.7293) family taken to be 1)
##
##     Null deviance: 1244.61  on 658  degrees of freedom
## Residual deviance:  425.42  on 651  degrees of freedom
## AIC: 1669.1
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.7293
##           Std. Err.:  0.0747
##
##  2 x log-likelihood:  -1651.1150
```

Compare the Poisson and Negative-Binomial results:

```
rbind(rd_pois$coefficients,rd_nb$coefficients)
```

```
##      (Intercept)   quality    skiyes      income userfeeyes        costC
## [1,]   0.2649934 0.4717259 0.4182137 -0.11132317  0.8981653 -0.003429706
## [2,]  -1.1219363 0.7219990 0.6121388 -0.02605884  0.6691676  0.048008668
##            costS      costH
## [1,] -0.04253641 0.03613362
## [2,] -0.09269101 0.03883569
```

Any observations ?

Quantile Regression

Quantile regression generalises the Least Absolute Deviations (LAD) to arbitrary quantiles. This is particularly useful if the researcher is only interested in a sub-population characterised by its location (quantile) on the outcome distribution. E.g., what is the impact of a policy on the household consumption of households around the poverty line.

The package to use is `quantreg`.

```
#install.packages("quantreg") #install this package if it is not already available.
require("quantreg") # the command require() plays the same role as library()
```

```
## Loading required package: quantreg
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve

## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "numericVector" of class "Mnumeric"; definition not updated

##
## Attaching package: 'quantreg'

## The following object is masked from 'package:survival':
##
##     untangle.specials
```

To get documentation on a function `fun` in a package that is installed, type `?fun` into the console. Look up the function `rq` in `quantreg` package.

```
?rq
```

Skim through the documentation and run the examples.

```
data(stackloss)
summary(rq(stack.loss ~ stack.x,tau=.5))  # LAD is a special case of quantreg
```

```
##
## Call: rq(formula = stack.loss ~ stack.x, tau = 0.5)
##
## tau: [1] 0.5
##
## Coefficients:
##                    coefficients lower bd  upper bd
## (Intercept)        -39.68986    -41.61973 -29.67754
## stack.xAir.Flow      0.83188      0.51278   1.14117
## stack.xWater.Temp    0.57391      0.32182   1.41090
## stack.xAcid.Conc.   -0.06087     -0.21348  -0.02891
```

```
summary(rq(stack.loss ~ stack.x,tau=.25))
```

```
## Warning in rq.fit.br(x, y, tau = tau, ci = TRUE, ...): Solution may be nonunique

##
## Call: rq(formula = stack.loss ~ stack.x, tau = 0.25)
##
## tau: [1] 0.25
##
## Coefficients:
##                    coefficients lower bd  upper bd
## (Intercept)        -36.00000    -53.84270 -36.00000
## stack.xAir.Flow      0.50000      0.24823   0.96678
## stack.xWater.Temp    1.00000      0.31679   2.19067
## stack.xAcid.Conc.    0.00000     -0.57947   0.00000
```

What is the interpretation of your results?