Regression in R

create some fake data

```
x1 <- 1:100

x2 <- -0.1*x1 + rnorm(100)

x3 <- 0.05*x2 + rnorm(100)

y <- 2*x1 + 10*rnorm(100) + 10*x2

dat <- data.frame( y, x1, x2, x3 )

head( dat )
```

```
##             y x1          x2         x3
## 1  -1.356088  1 -0.01531433 -0.9257205
## 2  -0.356096  2 -1.45542891  2.0086764
## 3  -8.805591  3 -1.58602538  0.1715653
## 4 -24.880864  4 -2.48621082 -0.8030154
## 5  -3.444461  5 -0.28043740 -2.2281208
## 6   8.913825  6 -0.81350619  0.4789454
```

descriptive statistics

```
library( pastecs )
```

```
## Loading required package: boot
```

```
print( t( stat.desc( dat ) ), digits=3 )
```

```
##    nbr.val nbr.null nbr.na    min     max  range    sum median   mean
## y      100        0      0 -24.88 118.193 143.07 4906.4 52.744 49.064
## x1     100        0      0   1.00 100.000  99.00 5050.0 50.500 50.500
## x2     100        0      0 -11.09   0.575  11.67 -512.3 -5.185 -5.123
## x3     100        0      0  -2.23   2.446   4.67  -25.9 -0.328 -0.259
##    SE.mean CI.mean.0.95     var std.dev coef.var
## y    3.216        6.381 1034.29   32.16    0.655
## x1   2.901        5.757  841.67   29.01    0.574
## x2   0.299        0.593    8.93    2.99   -0.583
## x3   0.103        0.204    1.06    1.03   -3.971
```

```
# To copy and paste into Excel:
#
# descriptives <- t( stat.desc(dat) )
#
# write.table( descriptives, "clipboard", sep="\t", row.names=TRUE )
```

```
# To create nicely formatted tables for markdown documents use the kable() function
```

```r
library( knitr )

kable( t( stat.desc( dat )[ c(1,4,5,8,9,13), ] ), format="markdown", digits=3 )
```

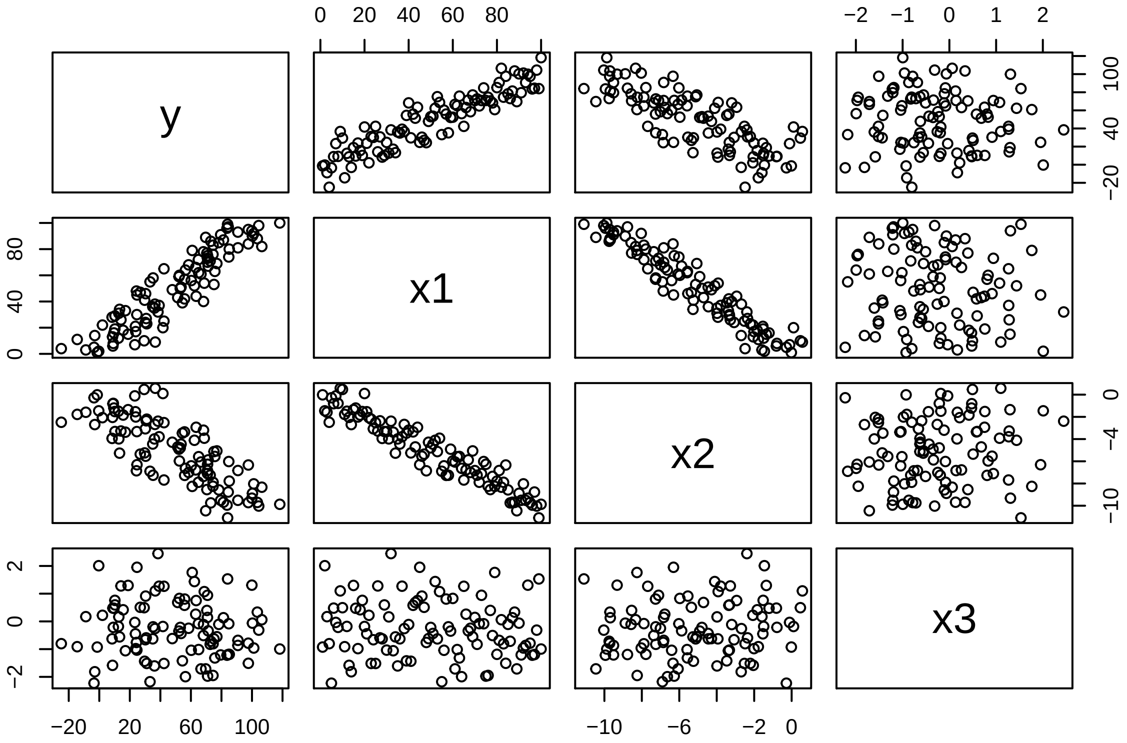
	nbr.val	min	max	median	mean	std.dev
y	100	-24.881	118.193	52.744	49.064	32.160
x1	100	1.000	100.000	50.500	50.500	29.011
x2	100	-11.094	0.575	-5.185	-5.123	2.989
x3	100	-2.228	2.446	-0.328	-0.259	1.027

	nbr.val	min	max	median	mean	std.dev
y	100	-11.210	118.497	50.591	52.395	31.342
x1	100	1.000	100.000	50.500	50.500	29.011
x2	100	-10.851	1.071	-5.097	-4.964	3.135
x3	100	-2.548	1.857	-0.409	-0.380	1.060

pretty pairs plot

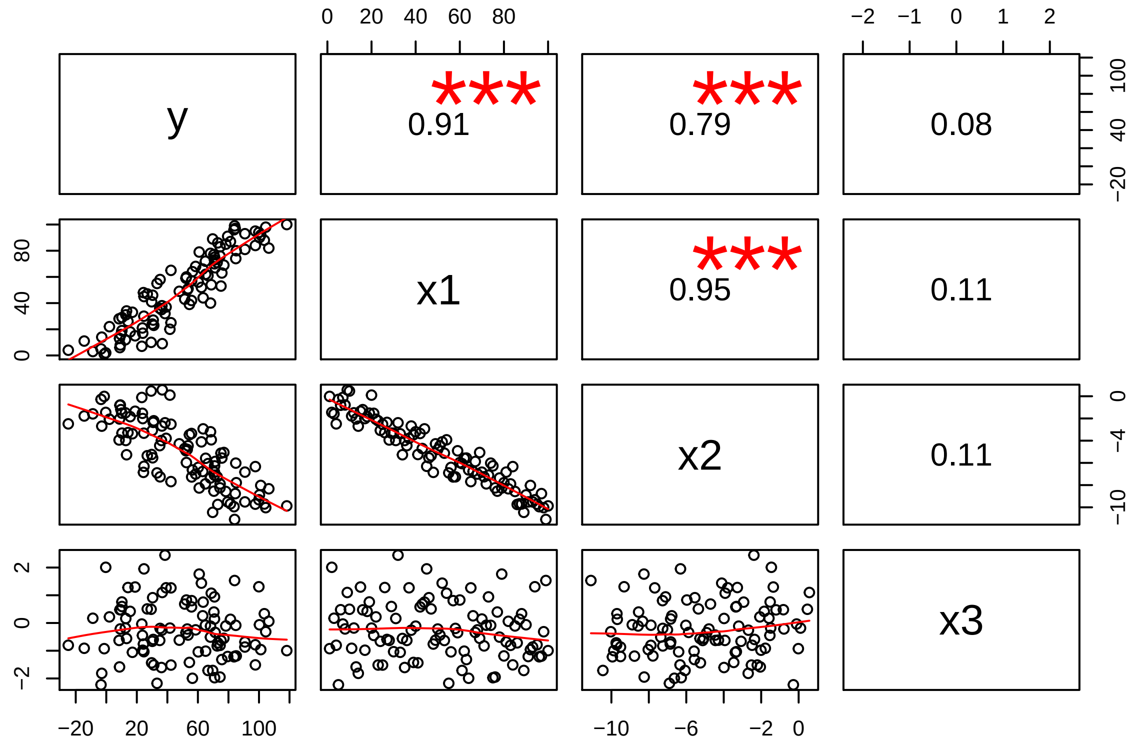
Convenient visual descriptives:

```r
pairs( dat )
```



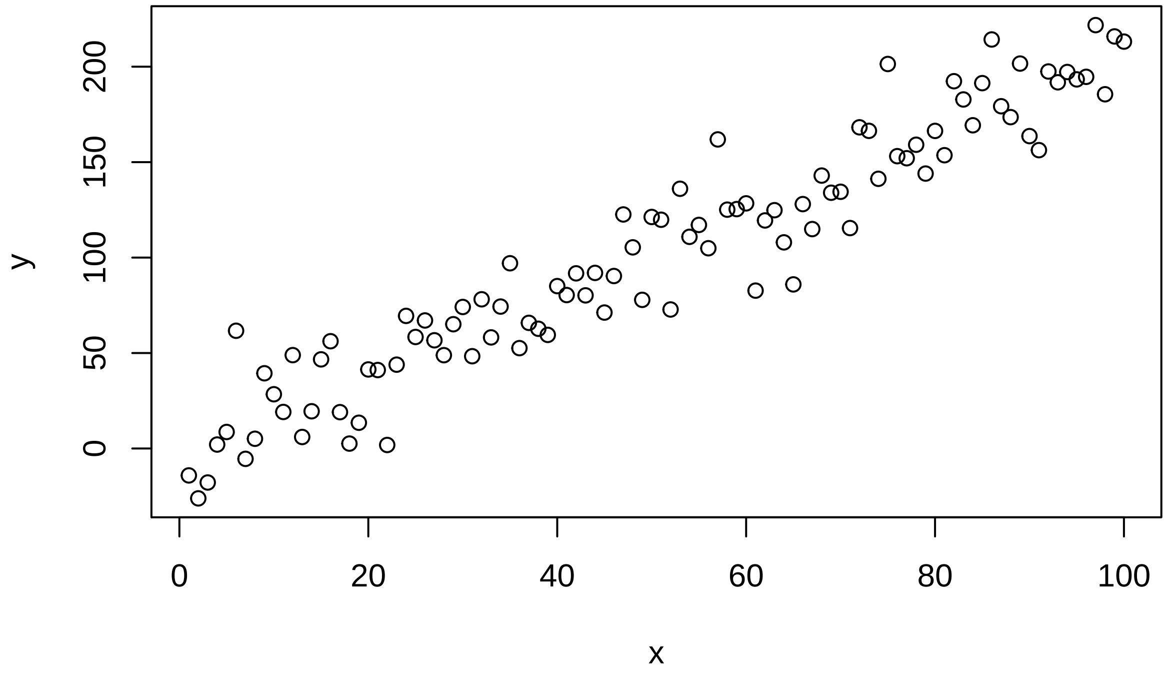
This one is better:

```
panel.cor <- function(x, y, digits=2, prefix="", cex.cor)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex <- 0.8/strwidth(txt)

    test <- cor.test(x,y)
    # borrowed from printCoefmat
    Signif <- symnum(test$p.value, corr = FALSE, na = FALSE,
                cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
                symbols = c("***", "**", "*", ".", " "))

    text(0.5, 0.5, txt, cex = 1.5 )
    text(.7, .8, Signif, cex=cex, col=2)
}


pairs( dat, lower.panel=panel.smooth, upper.panel=panel.cor)
```



create some fake regression data

```
x <- 1:100
y <- 2*x + rnorm(100,0,20)

plot( x, y )
```



```
dum <- sample( c("NJ","NY","MA","PA"), 100, replace=T )
```

basic regression syntax

```
lm( y ~ x )
```

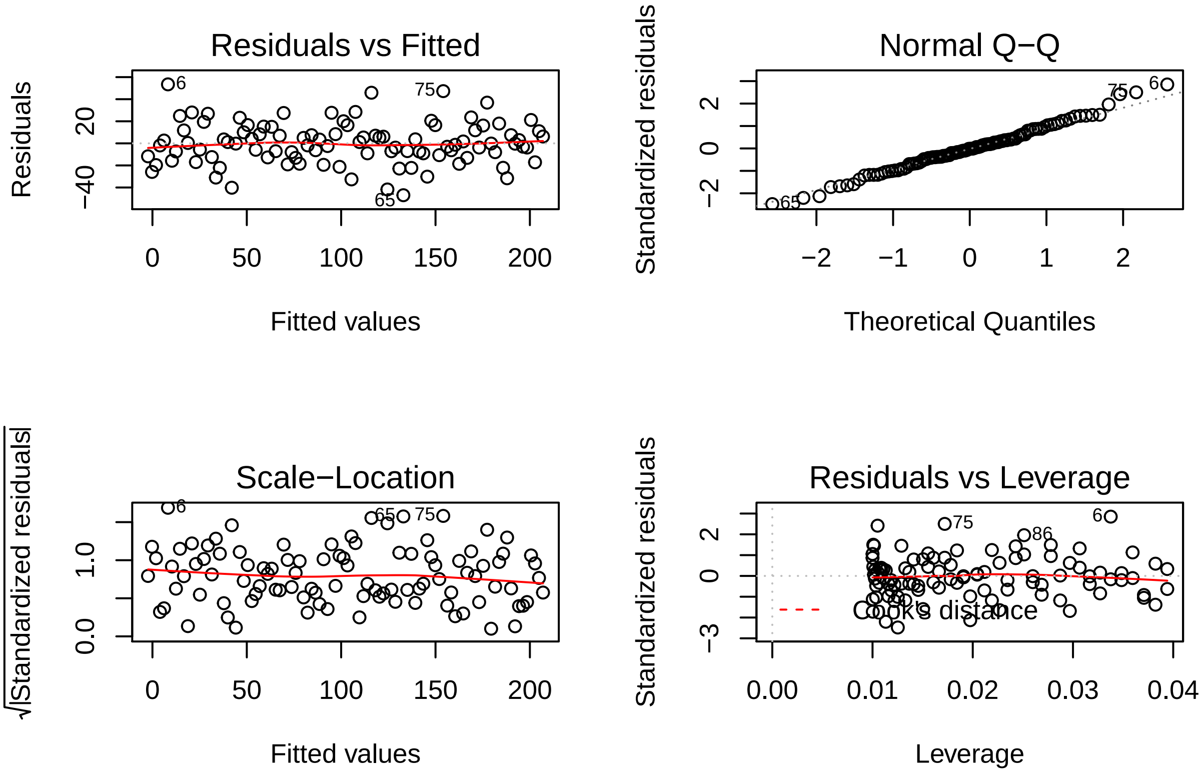
```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      -4.460        2.114
```

```
m.01 <- lm( y ~ x )

summary( m.01 )
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.000 -11.932  -0.287  11.197  53.447
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.46027    3.84369   -1.16    0.249
## x            2.11414    0.06608   31.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.07 on 98 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9117
## F-statistic:  1024 on 1 and 98 DF,  p-value: < 2.2e-16
```

nice visual diagnostics

```r
par( mfrow=c(2,2) )
plot( m.01 )
```



useful model fit functions

```
coefficients( m.01 ) # model coefficients
```

```
## (Intercept)           x
##   -4.460267    2.114140
```

```
confint( m.01, level=0.95) # CIs for model parameters
```

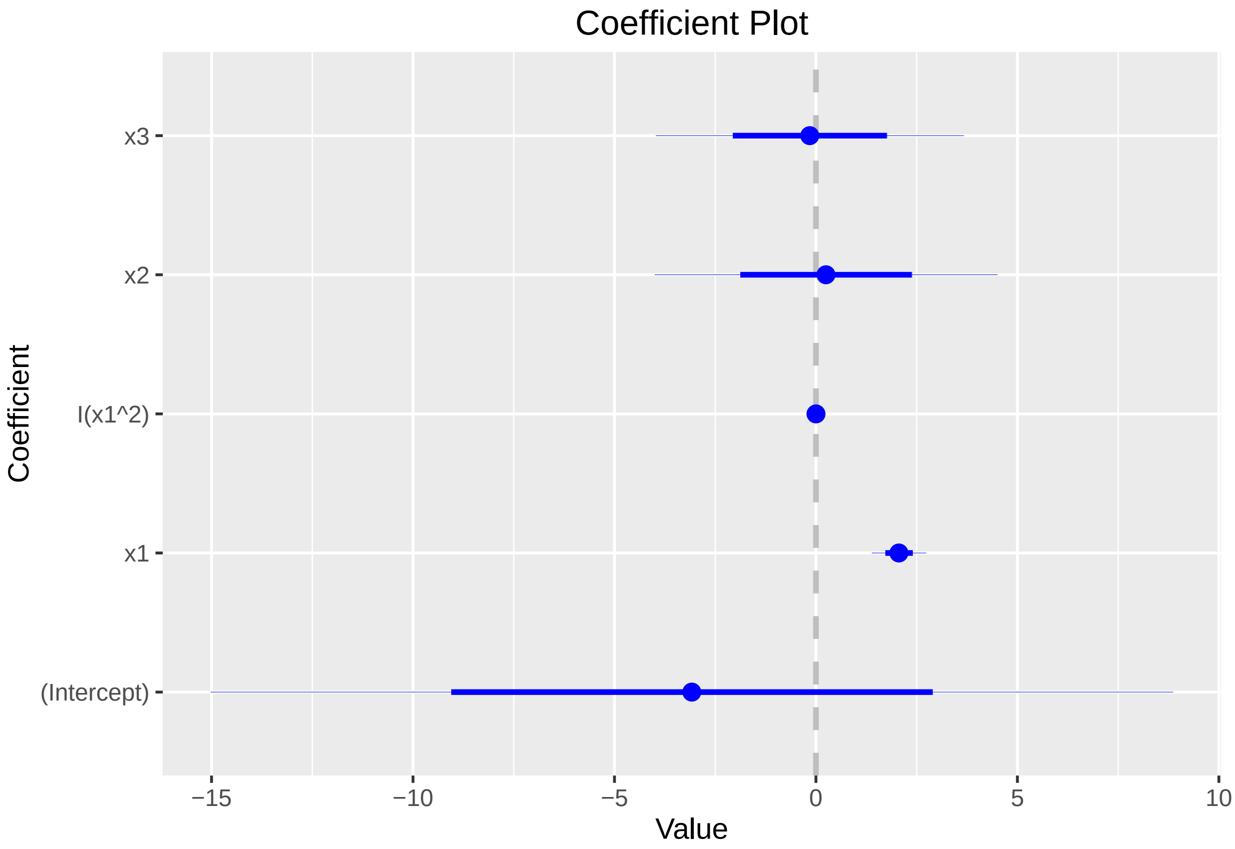
```
##                   2.5 %   97.5 %
## (Intercept) -12.087946 3.167412
## x             1.983008 2.245272
```

```
# not run because of long output

# anova( m.01 ) # anova table
# fitted( m.01 ) # predicted values
# residuals( m.01 ) # residuals
# influence( m.01 ) # regression diagnostics
```

```
library( coefplot )
```

```
## Loading required package: ggplot2
```

```
m.02 <-  lm( y ~ x1 + I(x1^2) + x2 + x3 )

coefplot(m.02)
```



pretty output

```r
# install.packages( "memisc" )

library( memisc )
```

```
## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma

## Loading required package: MASS

##
## Attaching package: 'memisc'

## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts

## The following object is masked from 'package:base':
##
##     as.array
```

```r
m.02 <- lm( y ~ x + I(x^2) )  # quadratic term
m.03 <- lm( y ~ x - 1 )       # no intercept term


pretty.table <- mtable("Model 1"=m.01,"Model 2"=m.02,"Model 3"=m.03,
                  summary.stats=c("R-squared","F","p","N"))

pretty.table
```

```
##
## Calls:
## Model 1: lm(formula = y ~ x)
## Model 2: lm(formula = y ~ x + I(x^2))
## Model 3: lm(formula = y ~ x - 1)
##
## ================================================
##                  Model 1    Model 2    Model 3
## ------------------------------------------------
##   (Intercept)  -4.460     -3.122
##                (3.844)    (5.866)
##   x             2.114***   2.035***   2.048***
##                (0.066)    (0.268)    (0.033)
##   I(x^2)                   0.001
##                           (0.003)
## ------------------------------------------------
##   R-squared        0.9        0.9        1.0
##   F             1023.6      507.1     3885.3
##   p                  0.0        0.0        0.0
##   N                 100        100        100
## ================================================
```

specification

```r
summary( lm( y ~ x1 + x2 + x3 ) )
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.466 -12.108  -0.452  11.528  53.562
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.4359     3.9019  -1.137    0.258
## x1            2.1345     0.2175   9.812 3.77e-16 ***
## x2            0.2145     2.1115   0.102    0.919
## x3           -0.1826     1.8964  -0.096    0.923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.27 on 96 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9099
## F-statistic: 334.3 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
# add different functional forms

# square x1

summary( lm( y ~ x1 + x1^2 + x2 + x3 ) )  # not right
```

```
##
## Call:
## lm(formula = y ~ x1 + x1^2 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.466 -12.108  -0.452  11.528  53.562
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.4359     3.9019  -1.137    0.258
## x1            2.1345     0.2175   9.812 3.77e-16 ***
## x2            0.2145     2.1115   0.102    0.919
## x3           -0.1826     1.8964  -0.096    0.923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.27 on 96 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9099
## F-statistic: 334.3 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
summary( lm( y ~ x1 + I(x1^2) + x2 + x3 ) )  # like this
```

```
##
## Call:
## lm(formula = y ~ x1 + I(x1^2) + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.985 -12.295  -0.777  10.520  52.647
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.080226   5.968037  -0.516    0.607
## x1           2.058652   0.333357   6.176 1.62e-08 ***
## I(x1^2)      0.000785   0.002606   0.301    0.764
## x2           0.248586   2.124575   0.117    0.907
## x3          -0.153319   1.907933  -0.080    0.936
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.36 on 95 degrees of freedom
## Multiple R-squared:  0.9127, Adjusted R-squared:  0.9091
## F-statistic: 248.4 on 4 and 95 DF,  p-value: < 2.2e-16
```

```r
summary( lm( y ~ log(x1) + x2 + x3 ) )  # log of x1 in formula works fine
```

```
##
## Call:
## lm(formula = y ~ log(x1) + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.137 -18.258  -0.396  17.223  66.623
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -51.9951    12.2681  -4.238 5.18e-05 ***
## log(x1)      23.6072     4.8932   4.824 5.28e-06 ***
## x2          -13.3240     1.5229  -8.749 7.21e-14 ***
## x3           -0.6644     2.4076  -0.276    0.783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.47 on 96 degrees of freedom
## Multiple R-squared:  0.8592, Adjusted R-squared:  0.8548
## F-statistic: 195.2 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
# interactions

summary( lm( y ~ x1 + x2 ) )
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.759 -11.963  -0.287  11.304  53.458
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.4241     3.8800  -1.140    0.257
## x1            2.1349     0.2164   9.866 2.61e-16 ***
## x2            0.2112     2.1004   0.101    0.920
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.17 on 97 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9108
## F-statistic: 506.6 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
summary( lm( y ~ x1 + x2 + I(x1*x2) ) )
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + I(x1 * x2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.987 -11.869  -0.624  10.218  51.667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.54690    5.89977  -0.262    0.794
## x1           2.05630    0.24852   8.274 7.45e-13 ***
## x2           1.06246    2.48179   0.428    0.670
## I(x1 * x2)  -0.01601    0.02468  -0.649    0.518
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.23 on 96 degrees of freedom
## Multiple R-squared:  0.913,  Adjusted R-squared:  0.9103
## F-statistic: 335.9 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
summary( lm( y ~ x1*x2 ) ) # shortcut
```

```
##
## Call:
## lm(formula = y ~ x1 * x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.987 -11.869  -0.624  10.218  51.667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.54690    5.89977  -0.262    0.794
## x1           2.05630    0.24852   8.274 7.45e-13 ***
## x2           1.06246    2.48179   0.428    0.670
## x1:x2       -0.01601    0.02468  -0.649    0.518
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.23 on 96 degrees of freedom
## Multiple R-squared:  0.913,  Adjusted R-squared:  0.9103
## F-statistic: 335.9 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
# dummy variables

summary( lm( y ~ x1 + x2 + x3 + dum ) ) # drop one level
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + dum)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.163 -12.245  -0.965  10.027  51.709
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.6334     4.7769  -0.551    0.583
## x1            2.1338     0.2191   9.741 7.24e-16 ***
## x2            0.1342     2.1294   0.063    0.950
## x3           -0.2050     1.9431  -0.105    0.916
## dumNJ        -0.7546     5.3030  -0.142    0.887
## dumNY        -2.3549     5.4269  -0.434    0.665
## dumPA        -6.6512     5.4422  -1.222    0.225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.4 on 93 degrees of freedom
## Multiple R-squared:  0.9142, Adjusted R-squared:  0.9087
## F-statistic: 165.1 on 6 and 93 DF,  p-value: < 2.2e-16
```

```
summary( lm( y ~ x1 + x2 + x3 + dum - 1) ) # keep all, drop intercept
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + dum - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.163 -12.245  -0.965  10.027  51.709
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## x1      2.1338     0.2191   9.741 7.24e-16 ***
## x2      0.1342     2.1294   0.063    0.950
## x3     -0.2050     1.9431  -0.105    0.916
## dumMA  -2.6334     4.7769  -0.551    0.583
## dumNJ  -3.3880     5.3696  -0.631    0.530
## dumNY  -4.9882     5.1799  -0.963    0.338
## dumPA  -9.2846     5.6346  -1.648    0.103
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.4 on 93 degrees of freedom
## Multiple R-squared:  0.9759, Adjusted R-squared:  0.9741
## F-statistic: 538.6 on 7 and 93 DF,  p-value: < 2.2e-16
```

standardized regression coefficients (beta)

```
# install.packages( "lm.beta" )

library( lm.beta )

m.01.beta <- lm.beta( m.01 )

summary( m.01.beta )
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.000 -11.932  -0.287  11.197  53.447
##
## Coefficients:
##             Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) -4.46027      0.00000    3.84369   -1.16    0.249
## x            2.11414      0.95531    0.06608   31.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.07 on 98 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9117
## F-statistic:  1024 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
# coef( m.01.beta )
```

```
# note the standard error is not standardized - describes regular coefficients not standardized

summary( m.01 )
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.000 -11.932  -0.287  11.197  53.447
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.46027    3.84369   -1.16    0.249
## x            2.11414    0.06608   31.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.07 on 98 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9117
## F-statistic:  1024 on 1 and 98 DF,  p-value: < 2.2e-16
```

or just use the formula:

```
lm.beta <- function( my.mod )
{
    b <- summary(my.mod)$coef[-1, 1]
    sx <- sd( my.mod$model[,-1] )
    sy <- sd( my.mod$model[,1] )
    beta <- b * sx/sy
    return(beta)
}


coefficients( m.01 )
```

```
## (Intercept)           x
##   -4.460267    2.114140
```

```
lm.beta( m.01 )
```

```
## [1] 0.9553146
```

robust standard errors

```
# install.packages( "sandwhich" )
# install.packages( "lmtest" )

library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
m.01 <- lm( y ~ x )


# REGULAR STANDARD ERRORS
summary( m.01 ) # not-robust
```

```
##
## Call:
```

```
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.000 -11.932  -0.287  11.197  53.447
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.46027    3.84369   -1.16    0.249
## x            2.11414    0.06608   31.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.07 on 98 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9117
## F-statistic:  1024 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
# ROBUST STANDARD ERRORS

# reproduce the Stata default
coeftest( m.01, vcov=vcovHC(m.01,"HC1") )      # robust; HC1 (Stata default)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.460267   3.945755 -1.1304   0.2611
## x            2.114140   0.065438 32.3076   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ROBUST STANDARD ERRORS

# check that "sandwich" returns HC0
coeftest(m.01, vcov = sandwich)                # robust; sandwich
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.46027    3.90610 -1.1419   0.2563
## x            2.11414    0.06478 32.6356   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(m.01, vcov = vcovHC(m.01, "HC0"))     # robust; HC0
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.46027    3.90610 -1.1419   0.2563
## x            2.11414    0.06478 32.6356   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ROBUST STANDARD ERRORS

# check that the default robust var-cov matrix is HC3
coeftest(m.01, vcov = vcovHC(m.01))            # robust; HC3
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.460267   4.017243 -1.1103   0.2696
## x            2.114140   0.066632 31.7288   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(m.01, vcov = vcovHC(m.01, "HC3"))     # robust; HC3 (default)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.460267   4.017243 -1.1103   0.2696
## x            2.114140   0.066632 31.7288   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```