Regression in R

load useful packages for formatting output

```r
library( pander ) # translate output to HTML / latex

library(  magrittr )  # use the pipe operator %>%

library( knitr )  # kable function formats tables
```

create a toy dataset

```r
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   6.489172  1  0.6132310  0.53043802
## 2 -35.774792  2 -3.1351324  0.82019800
## 3  29.058439  3  0.6588230 -1.19311475
## 4  11.477368  4 -0.3853383 -0.04353684
## 5 -17.965633  5 -2.0930484 -1.22033111
## 6  31.777646  6  1.3981708  1.19573860
```

descriptive statistics

```
summary( dat ) %>% kable
```

y	x1	x2	x3
Min. :-35.77	Min. : 1.00	Min. :-11.271	Min. :-2.4455
1st Qu.: 32.45	1st Qu.: 25.75	1st Qu.: -7.751	1st Qu.:-1.0294
Median : 49.87	Median : 50.50	Median : -4.949	Median :-0.3405
Mean : 52.55	Mean : 50.50	Mean : -4.958	Mean :-0.2958
3rd Qu.: 74.95	3rd Qu.: 75.25	3rd Qu.: -2.665	3rd Qu.: 0.5513
Max. :120.34	Max. :100.00	Max. : 1.398	Max. : 2.0431

```
library( pastecs ) # convenient descriptives function

stat.desc( dat ) %>% t %>% round(2) %>% pander
```

Table 2: Table continues below

	nbr.val	nbr.null	nbr.na	min	max	range	sum
y	100	0	0	-35.77	120.3	156.1	5255
x1	100	0	0	1	100	99	5050
x2	100	0	0	-11.27	1.4	12.67	-495.8
x3	100	0	0	-2.45	2.04	4.49	-29.58

	median	mean	SE.mean	CI.mean.0.95	var	std.dev	coef.var
y	49.87	52.55	3.19	6.33	1018	31.91	0.61
x1	50.5	50.5	2.9	5.76	841.7	29.01	0.57
x2	-4.95	-4.96	0.32	0.63	9.95	3.15	-0.64
x3	-0.34	-0.3	0.1	0.2	0.97	0.99	-3.34

```
# grab only the desired descriptives

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

	nbr.val	min	max	median	mean	std.dev
y	100	-35.775	120.339	49.872	52.548	31.909
x1	100	1.000	100.000	50.500	50.500	29.011
x2	100	-11.271	1.398	-4.949	-4.958	3.155
x3	100	-2.445	2.043	-0.340	-0.296	0.987

```
stat.desc( dat )[ c(1,4,5,8,9,13), ] %>% t %>% pander( digits=3 )
```

	nbr.val	min	max	median	mean	std.dev
y	100	-35.8	120	49.9	52.5	31.9
x1	100	1	100	50.5	50.5	29
x2	100	-11.3	1.4	-4.95	-4.96	3.15

	nbr.val	min	max	median	mean	std.dev
x3	100	-2.45	2.04	-0.34	-0.296	0.987

copy and paste a table to excel

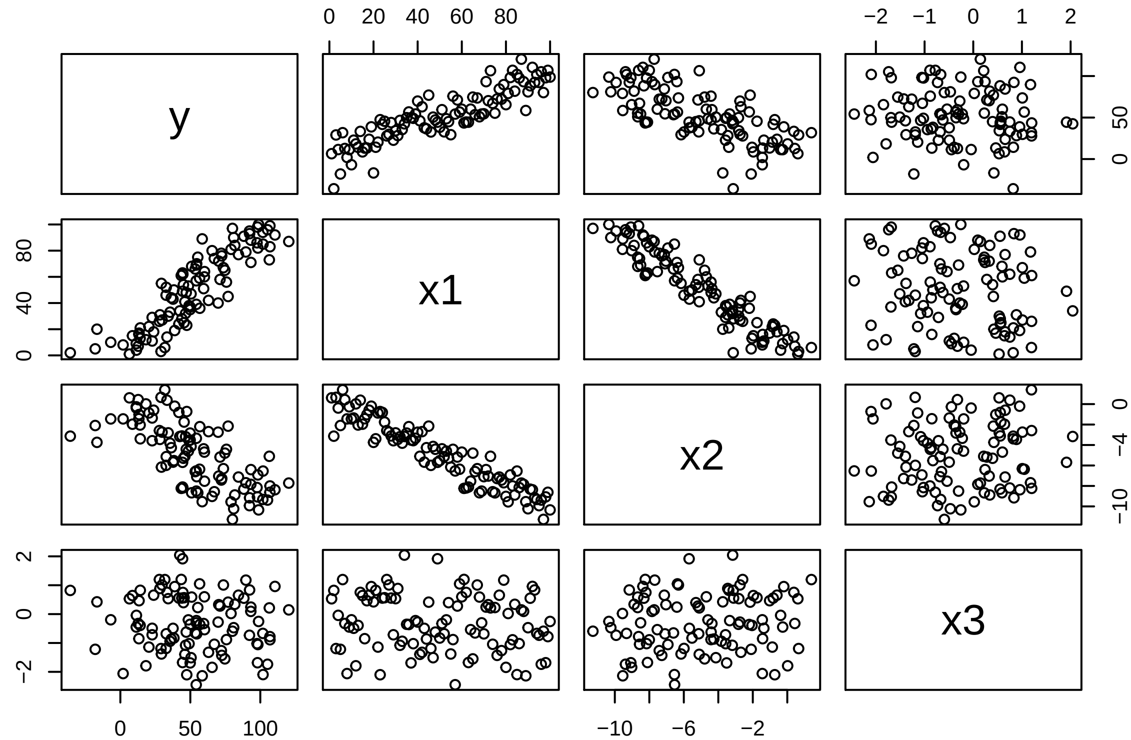
```
descriptives <- t( stat.desc(dat) )

write.table( descriptives, "clipboard", sep="\t", row.names=TRUE )
```

Scatterplots

pretty pairs plot

Convenient visual descriptives:

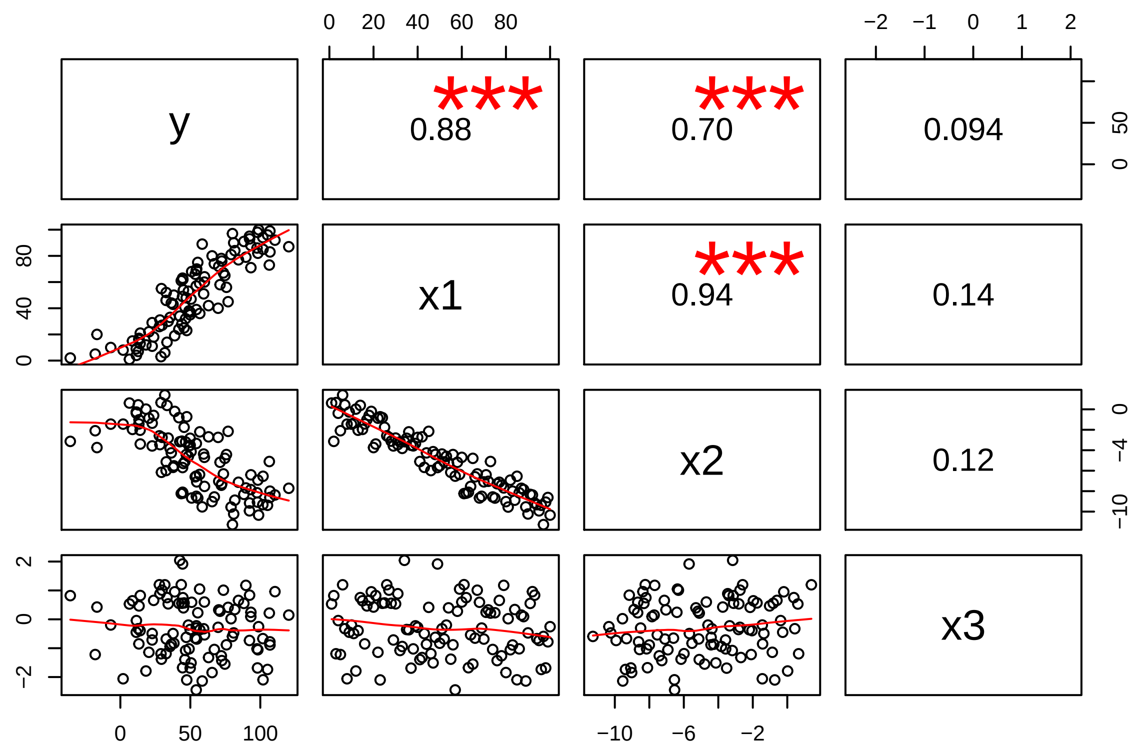
```
pairs( dat )
```



We can improve it:

```
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)
```

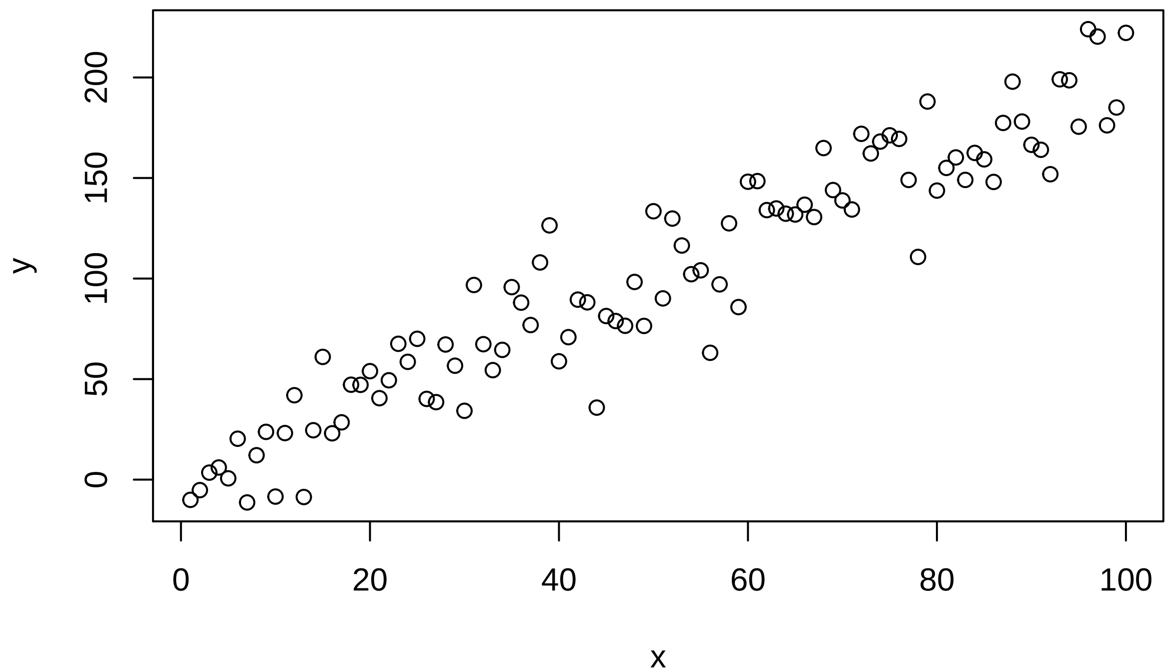
```
}

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



Regression Syntax

create some 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

The regression is run using the "linear model" command. The basic model will print the minimum output:

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

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
```

```
## (Intercept)           x
##     -0.2333      2.0171
```

To generate nicely-formatted regression tables save the results from the regression as an object, and format the output for inclusion in a markdown document using the `pander` package.

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

summary( m.01 ) %>% pander  # add pander to format for markdown docs
```

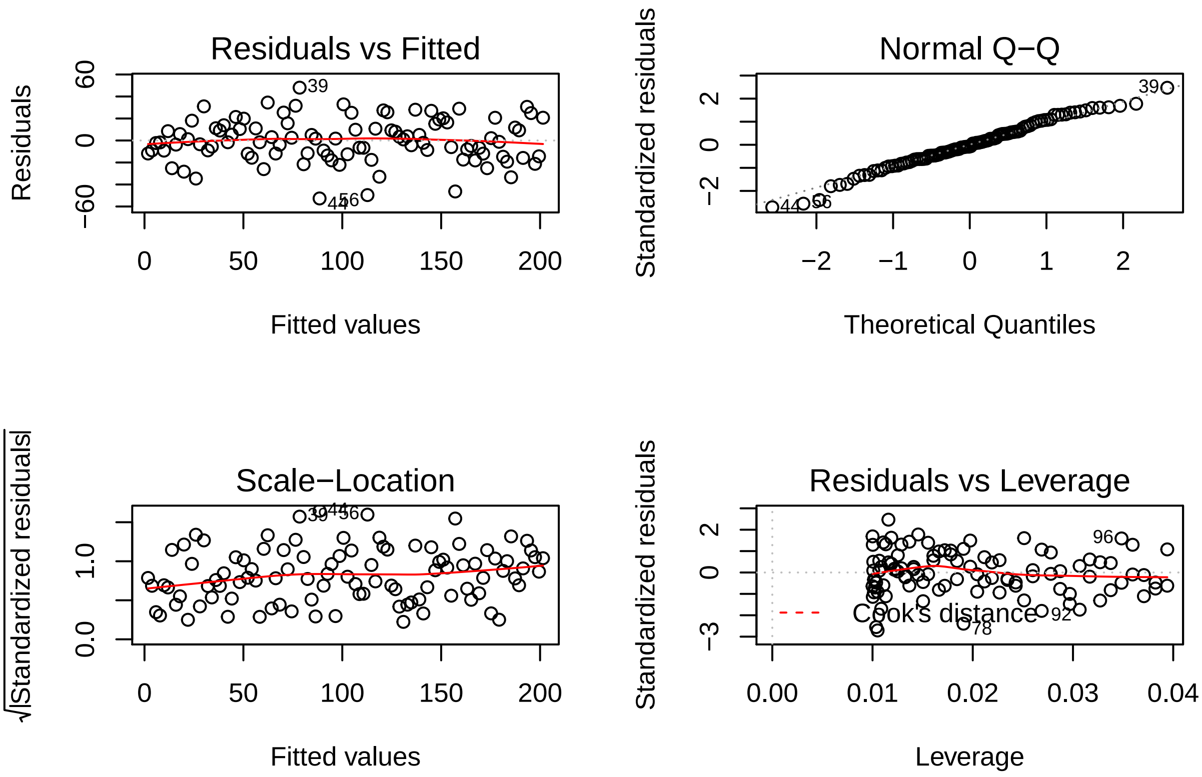
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.2333	3.939	-0.05922	0.9529
x	2.017	0.06772	29.78	6.548e-51

Table 7: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.55	0.9005	0.8995

nice visual diagnostics of model fit

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



useful model fit functions

```
coefficients( m.01 ) %>% pander # model coefficients
```

(Intercept)	x
-0.2333	2.017

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

	2.5 %	97.5 %
(Intercept)	-8.051	7.584
x	1.883	2.151

```
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
```

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

coefplot(m.02)
```

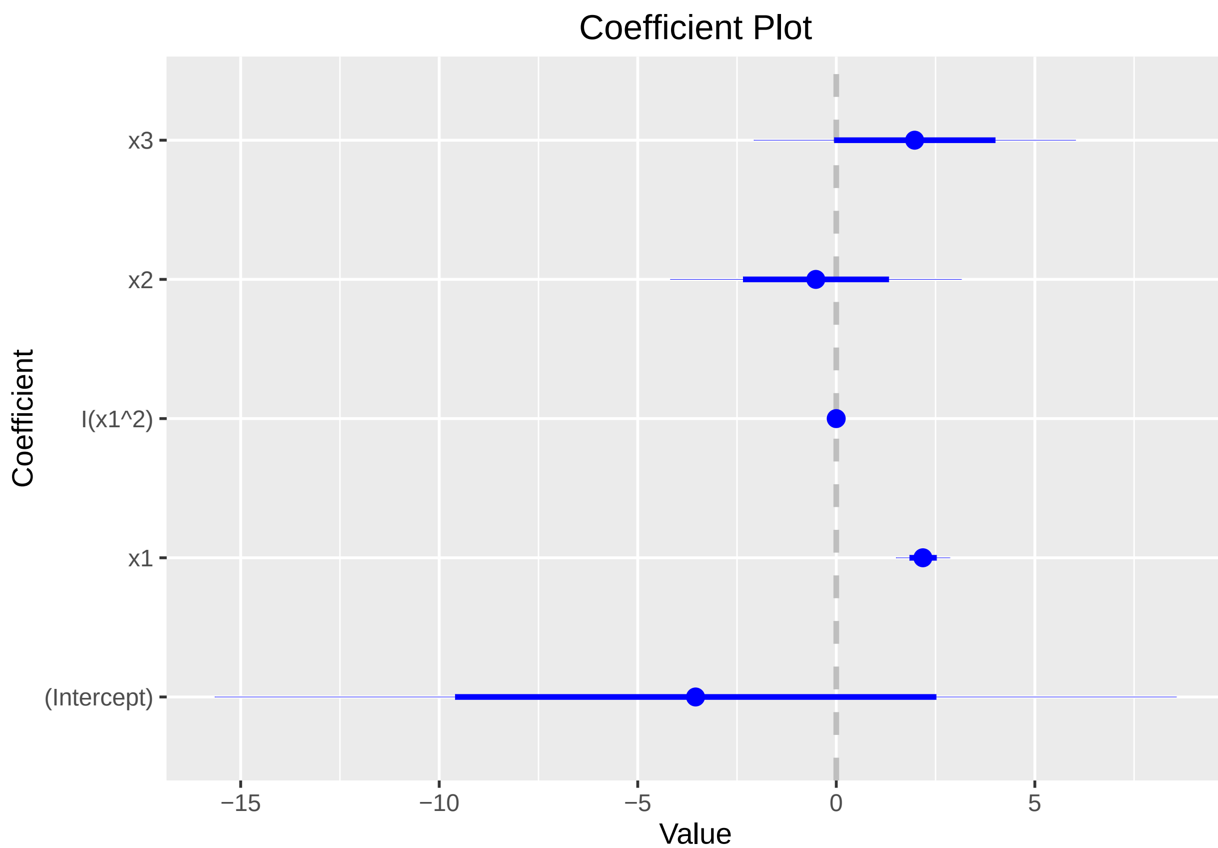


table with multiple regression models

```r
library( memisc )

x_sqr <- x * x

m.01 <- lm( y ~ x )
m.02 <- lm( y ~ x + x_sqr )  # 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 %>% pander
```

	Model 1	Model 2	Model 3
(Intercept)	-0.233 (3.939)	-3.987 (5.993)	

	Model 1	Model 2	Model 3
x	2.017***	2.238***	2.014***
	(0.068)	(0.274)	(0.033)
x_sqr		-0.002	
		(0.003)	
R-squared	0.9	0.9	1.0
F	887.1	442.5	3626.4
p	0.0	0.0	0.0
N	100	100	100

specification

```
summary( lm( y ~ x1 + x2 + x3 ) ) %>% pander
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0002184	3.976	5.493e-05	1
x1	1.969	0.1991	9.886	2.611e-16
x2	-0.5661	1.828	-0.3097	0.7574
x3	2.035	2.019	1.008	0.316

Table 12: Fitting linear model: y ~ x1 + x2 + x3

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.64	0.9017	0.8986

```
# add different functional forms

# square x1

summary( lm( y ~ x1 + x1^2 + x2 + x3 ) )  %>% pander      # incorrect
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0002184	3.976	5.493e-05	1
x1	1.969	0.1991	9.886	2.611e-16
x2	-0.5661	1.828	-0.3097	0.7574
x3	2.035	2.019	1.008	0.316

Table 14: Fitting linear model: y ~ x1 + x1^2 + x2 + x3

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.64	0.9017	0.8986

```
summary( lm( y ~ x1 + I(x1^2) + x2 + x3 ) )  %>% pander   # correct - enclose with I()
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-3.544	6.054	-0.5855	0.5596
x1	2.181	0.3384	6.445	4.754e-09
I(x1^2)	-0.002056	0.002644	-0.7776	0.4387
x2	-0.5157	1.833	-0.2814	0.779
x3	1.973	2.025	0.9746	0.3322

Table 16: Fitting linear model: y ~ x1 + I(x1^2) + x2 + x3

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.68	0.9023	0.8982

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-49.12	12.34	-3.98	0.000134
log(x1)	26.45	4.739	5.581	2.213e-07
x2	-11.06	1.395	-7.931	3.983e-12
x3	1.077	2.488	0.4331	0.6659

Table 18: Fitting linear model: y ~ log(x1) + x2 + x3

Observations	Residual Std. Error	R^2	Adjusted R^2
100	24.24	0.8502	0.8455

```
# interactions

summary( lm( y ~ x1 + x2 ) ) %>% pander
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1118	3.974	-0.02813	0.9776
x1	1.955	0.1987	9.84	2.972e-16
x2	-0.6045	1.828	-0.3308	0.7415

Table 20: Fitting linear model: y ~ x1 + x2

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.64	0.9006	0.8986

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1593	5.752	-0.0277	0.978

	Estimate	Std. Error	t value	Pr(>|t|)
x1	1.957	0.2437	8.031	2.448e-12
x2	-0.6173	2.148	-0.2874	0.7744
I(x1 * x2)	0.0002887	0.02512	0.01149	0.9909

Table 22: Fitting linear model: y ~ x1 + x2 + I(x1 * x2)

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.74	0.9006	0.8975

```r
summary( lm( y ~ x1*x2 ) ) %>% pander    # shortcut
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1593	5.752	-0.0277	0.978
x1	1.957	0.2437	8.031	2.448e-12
x2	-0.6173	2.148	-0.2874	0.7744
x1:x2	0.0002887	0.02512	0.01149	0.9909

Table 24: Fitting linear model: y ~ x1 * x2

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.74	0.9006	0.8975

```r
# dummy variables

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.54	5.87	0.092	0.9269
x1	1.991	0.2038	9.772	6.205e-16
x2	-0.2866	1.866	-0.1536	0.8782
x3	2.172	2.069	1.05	0.2965
dumNJ	-4.117	5.752	-0.7158	0.4759
dumNY	2.915	5.993	0.4863	0.6279
dumPA	0.6642	5.543	0.1198	0.9049

Table 26: Fitting linear model: y ~ x1 + x2 + x3 + dum

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.78	0.9033	0.8971

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

	Estimate	Std. Error	t value	Pr(>|t|)
x1	1.991	0.2038	9.772	6.205e-16
x2	-0.2866	1.866	-0.1536	0.8782
x3	2.172	2.069	1.05	0.2965
dumMA	0.54	5.87	0.092	0.9269
dumNJ	-3.577	4.942	-0.7239	0.4709
dumNY	3.455	5.21	0.663	0.509
dumPA	1.204	5.591	0.2154	0.8299

Table 28: Fitting linear model: y ~ x1 + x2 + x3 + dum - 1

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.78	0.9742	0.9722

Standardized Coefficients and Robust Standard Errors

standardized regression coefficients (beta)

```r
library( lm.beta )

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

summary( m.01.beta ) # %>% pander
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.697 -11.984  -1.411  12.248  48.008
##
## Coefficients:
##             Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) -0.23329      0.00000    3.93926  -0.059    0.953
## x            2.01710      0.94896    0.06772  29.785   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.55 on 98 degrees of freedom
## Multiple R-squared:  0.9005, Adjusted R-squared:  0.8995
## F-statistic: 887.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

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

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

summary( m.01 ) %>% pander
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.2333	3.939	-0.05922	0.9529
x	2.017	0.06772	29.78	6.548e-51

Table 30: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.55	0.9005	0.8995

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 ) %>% pander
```

(Intercept)	x
-0.2333	2.017

```
lm.beta( m.01 ) %>% pander
```

0.949

robust standard errors

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

library(sandwich)
library(lmtest)

m.01 <- lm( y ~ x )


# REGULAR STANDARD ERRORS - not robust

summary( m.01 ) %>% pander
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.2333	3.939	-0.05922	0.9529
x	2.017	0.06772	29.78	6.548e-51

Table 33: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	19.55	0.9005	0.8995

```r
# 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) -0.233294   3.469377 -0.0672   0.9465
## x            2.017104   0.061829 32.6241   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# 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) -0.233294   3.434508 -0.0679    0.946
## x            2.017104   0.061207 32.9554   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.233294   3.434508 -0.0679    0.946
## x            2.017104   0.061207 32.9554   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# 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) -0.233294   3.517347 -0.0663   0.9473
## x            2.017104   0.062936 32.0502   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.233294   3.517347 -0.0663   0.9473
## x            2.017104   0.062936 32.0502   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```