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
```

```
## Warning: package 'knitr' was built under R version 3.3.1
```

create some fake data

```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 18.8823390  1  0.7162691 0.5829701
## 2 30.5999952  2  1.7686181 0.1454880
## 3 10.8490256  3 -0.4485324 0.7103859
## 4  0.2311398  4  0.4365521 2.5353503
## 5 -7.4533215  5 -2.1878183 0.9760055
## 6 15.3782118  6  0.1705823 0.8274089
```

descriptive statistics

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

y	x1	x2	x3
Min. :-12.47	Min. : 1.00	Min. :-11.555	Min. :-2.203479
1st Qu.: 20.86	1st Qu.: 25.75	1st Qu.: -7.478	1st Qu.:-0.761729
Median : 51.02	Median : 50.50	Median : -4.943	Median : 0.006735
Mean : 49.70	Mean : 50.50	Mean : -5.055	Mean :-0.002688
3rd Qu.: 73.54	3rd Qu.: 75.25	3rd Qu.: -2.608	3rd Qu.: 0.714227
Max. :111.22	Max. :100.00	Max. : 1.769	Max. : 2.635754

```r
library( pastecs ) # convenient descriptives function
```

```
## Loading required package: boot

##
## Attaching package: 'pastecs'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```r
stat.desc( dat ) %>% t %>% pander
```

Table 2: Table continues below

	nbr.val	nbr.null	nbr.na	min	max	range	sum
y	100	0	0	-12.47	111.2	123.7	4970
x1	100	0	0	1	100	99	5050
x2	100	0	0	-11.56	1.769	13.32	-505.5
x3	100	0	0	-2.203	2.636	4.839	-0.2688

	median	mean	SE.mean	CI.mean.0.95	var	std.dev	coef.var
y	51.02	49.7	3.214	6.377	1033	32.14	0.6466
x1	50.5	50.5	2.901	5.757	841.7	29.01	0.5745
x2	-4.943	-5.055	0.3049	0.6049	9.295	3.049	-0.6031
x3	0.006735	-0.002688	0.1011	0.2006	1.022	1.011	-376.2

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

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

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

library( knitr )
```

```r
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	-12.469	111.221	51.017	49.705	32.140
x1	100	1.000	100.000	50.500	50.500	29.011
x2	100	-11.555	1.769	-4.943	-5.055	3.049
x3	100	-2.203	2.636	0.007	-0.003	1.011

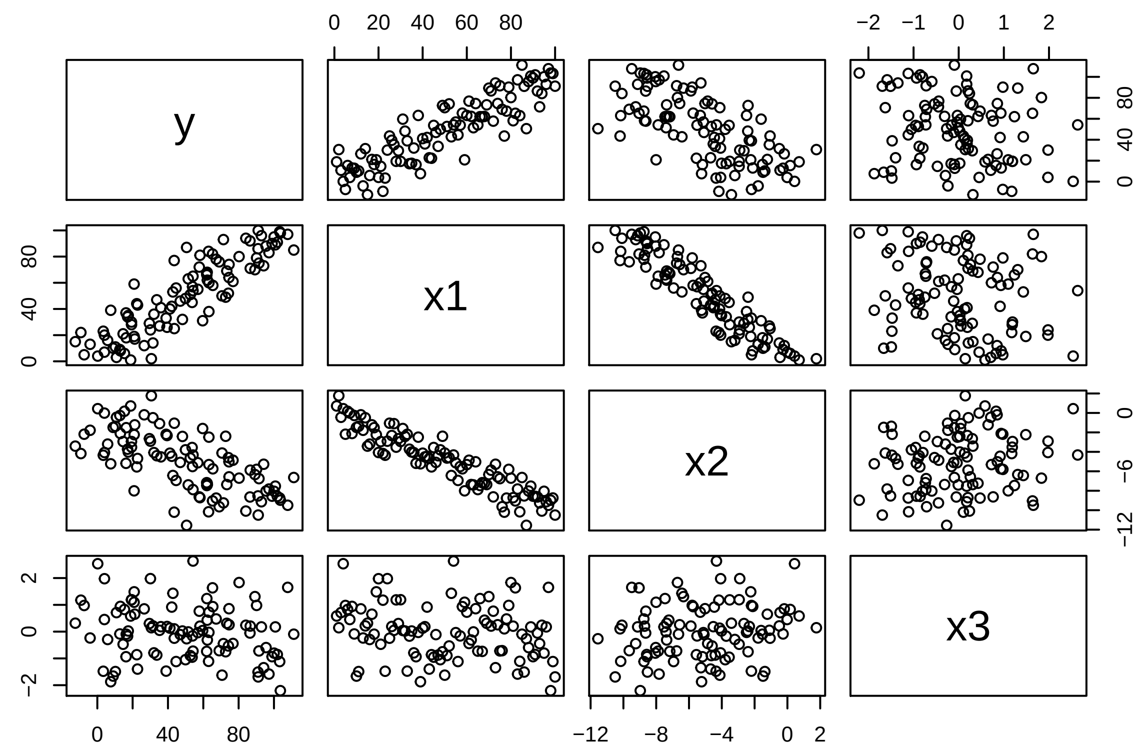
```r
t( stat.desc( dat )[ c(1,4,5,8,9,13), ] ) %>% pander
```

	nbr.val	min	max	median	mean	std.dev
y	100	-12.47	111.2	51.02	49.7	32.14
x1	100	1	100	50.5	50.5	29.01
x2	100	-11.56	1.769	-4.943	-5.055	3.049
x3	100	-2.203	2.636	0.006735	-0.002688	1.011

pretty pairs plot

Convenient visual descriptives:

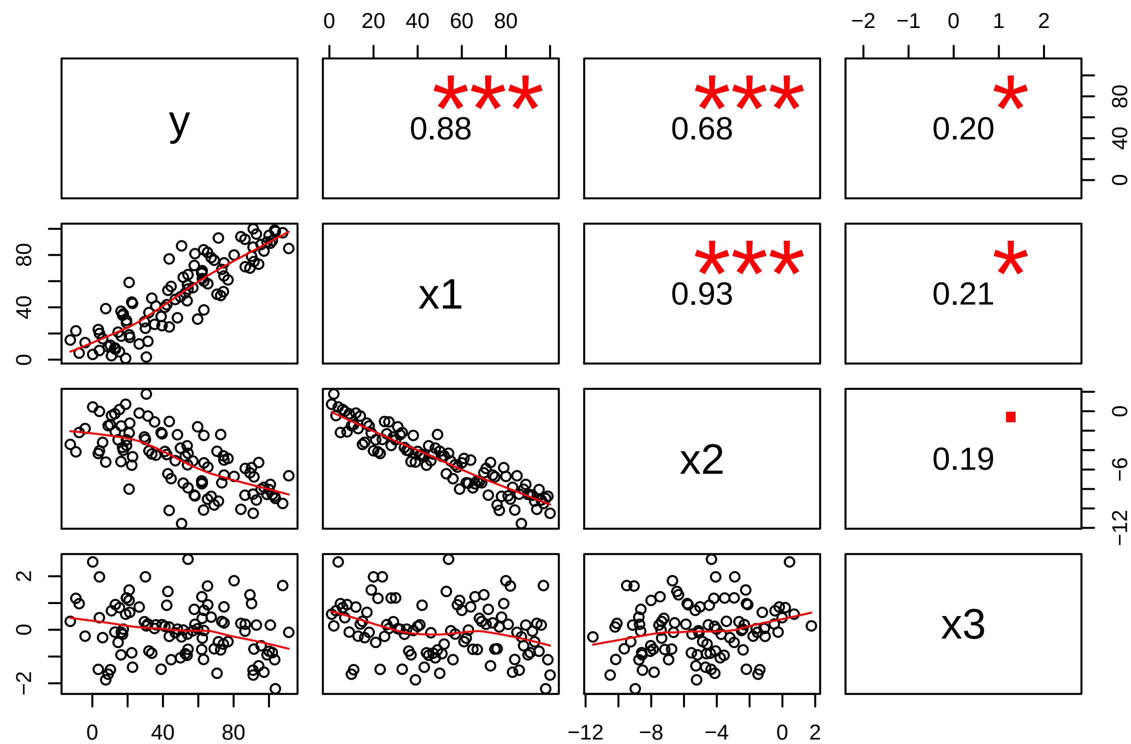
```
pairs( dat )
```



We can improve it:

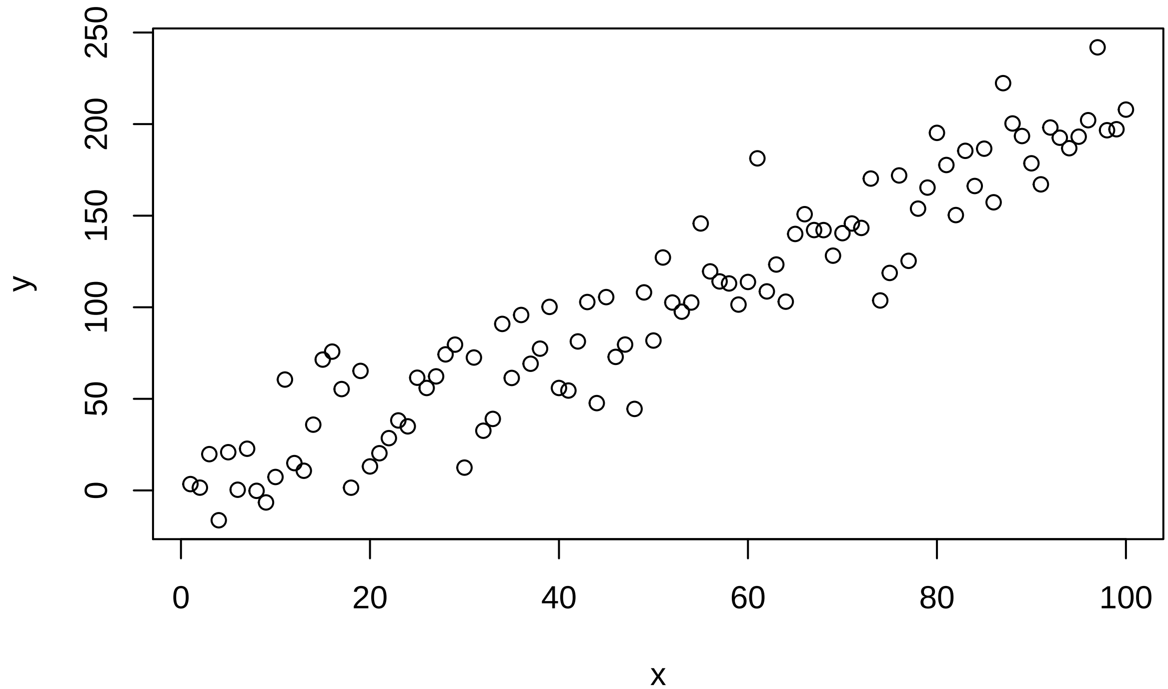
```
panel.cor <- function(x, y, digits=2, prefix="", cex.cor)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex <- 0.8/strwidth(txt)

    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)
}
```

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



create some fake regression data

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

plot( x, y )
```




```r
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:

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

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      -4.241        2.102
```

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.

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

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-4.241	4.346	-0.976	0.3315
x	2.102	0.07471	28.14	9.604e-49

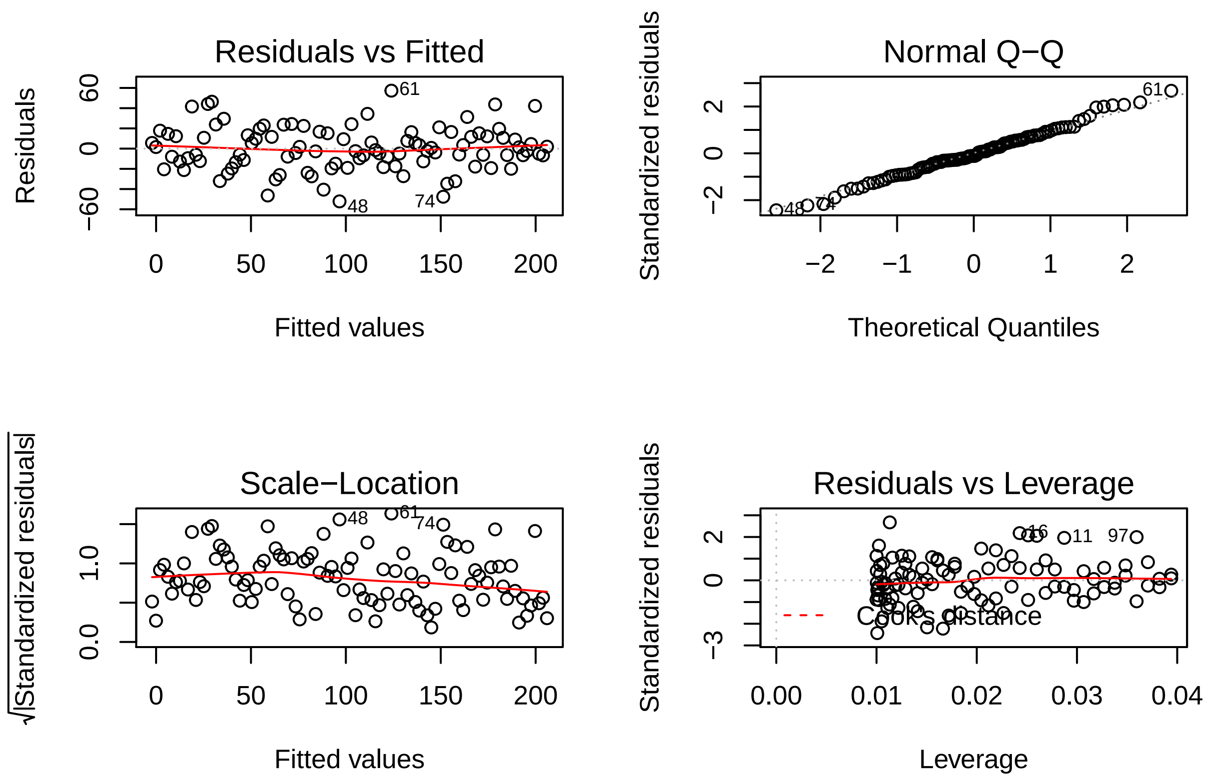
	Estimate	Std. Error	t value	Pr(>|t|)

Table 7: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.56	0.8899	0.8888

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
-4.241	2.102

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

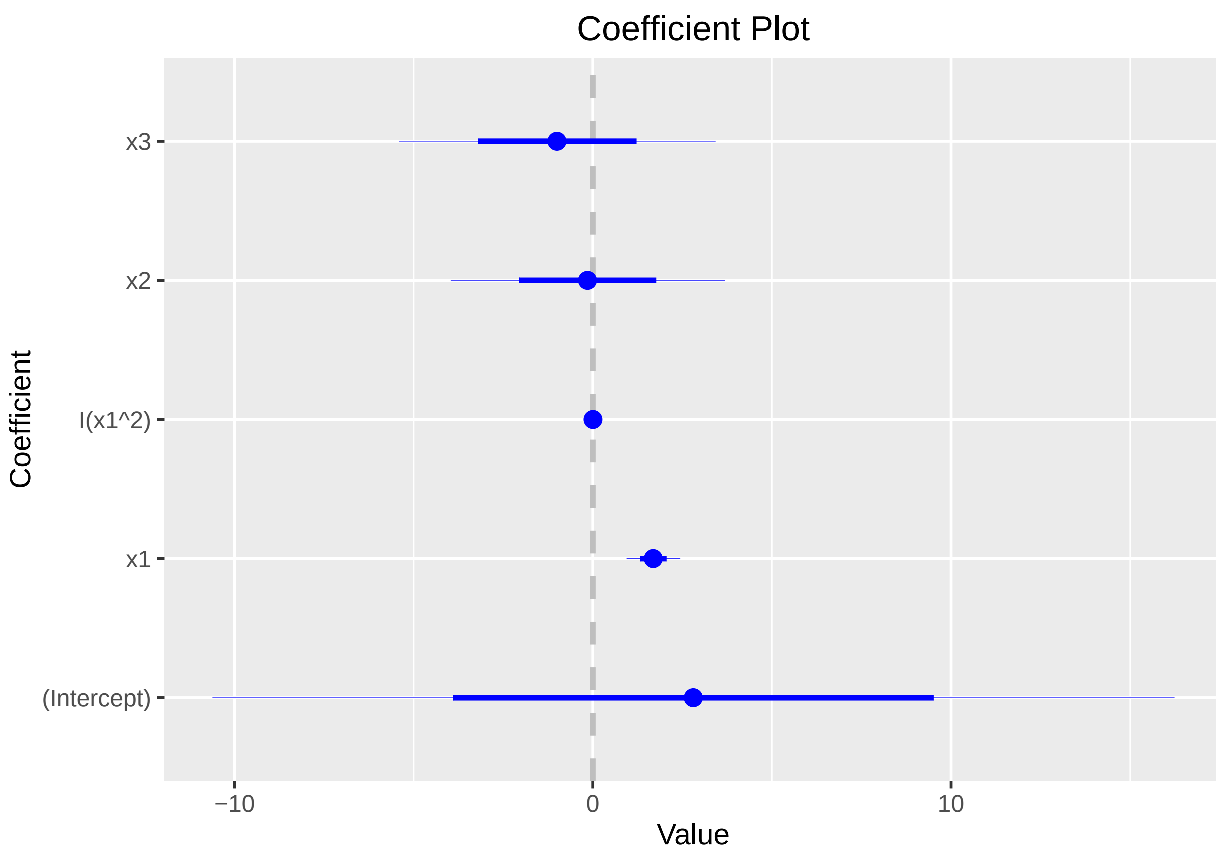
	2.5 %	97.5 %
(Intercept)	-12.86	4.382
x	1.954	2.251

```r
# 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
```

```r
library( coefplot )
```

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

```r
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
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)	-4.241	2.350	
	(4.346)	(6.575)	
x	2.102***	1.715***	2.039***
	(0.075)	(0.301)	(0.037)
x__sqr		0.004	
		(0.003)	
R-squared	0.9	0.9	1.0
F	791.9	400.0	3026.3
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)	-3.893	4.47	-0.8709	0.386
x1	2.103	0.2017	10.43	1.808e-17
x2	0.07565	1.909	0.03963	0.9685
x3	-0.906	2.216	-0.4089	0.6835

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.77	0.8901	0.8866

```
# add different functional forms

# square x1

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-3.893	4.47	-0.8709	0.386
x1	2.103	0.2017	10.43	1.808e-17
x2	0.07565	1.909	0.03963	0.9685
x3	-0.906	2.216	-0.4089	0.6835

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.77	0.8901	0.8866

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.805	6.711	0.418	0.6769
x1	1.687	0.3711	4.544	1.62e-05
I(x1^2)	0.003898	0.002922	1.334	0.1854
x2	-0.1489	1.909	-0.07802	0.938
x3	-1.002	2.208	-0.4536	0.6511

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.68	0.8921	0.8875

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-41.12	15.98	-2.573	0.01161
log(x1)	21.53	6.335	3.398	0.0009891
x2	-12.81	1.909	-6.708	1.365e-09
x3	-1.847	3.072	-0.6013	0.5491

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	30.03	0.7907	0.7842

```
# interactions

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-4.228	4.376	-0.9662	0.3363
x1	2.112	0.1997	10.57	7.776e-18
x2	0.09567	1.9	0.05034	0.96

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.68	0.8899	0.8876

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.2428	6.351	0.03823	0.9696
x1	1.97	0.2473	7.967	3.339e-12
x2	1.325	2.284	0.5803	0.5631
I(x1 * x2)	-0.02644	0.02722	-0.9715	0.3338

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.68	0.8909	0.8875

Observations	Residual Std. Error	R^2	Adjusted R^2

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.2428	6.351	0.03823	0.9696
x1	1.97	0.2473	7.967	3.339e-12
x2	1.325	2.284	0.5803	0.5631
x1:x2	-0.02644	0.02722	-0.9715	0.3338

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.68	0.8909	0.8875

```r
# dummy variables

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-10.18	5.601	-1.817	0.07245
x1	2.092	0.203	10.3	4.655e-17
x2	0.09409	1.913	0.04919	0.9609
x3	-0.4745	2.241	-0.2117	0.8328
dumNJ	4.434	6.298	0.7041	0.4831
dumNY	7.244	6.487	1.117	0.267
dumPA	14.04	5.874	2.389	0.0189

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.42	0.8969	0.8903

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

	Estimate	Std. Error	t value	Pr(>|t|)
x1	2.092	0.203	10.3	4.655e-17
x2	0.09409	1.913	0.04919	0.9609
x3	-0.4745	2.241	-0.2117	0.8328
dumMA	-10.18	5.601	-1.817	0.07245
dumNJ	-5.742	6.05	-0.949	0.3451
dumNY	-2.933	6.099	-0.4809	0.6317
dumPA	3.859	5.589	0.6904	0.4917

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

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.42	0.9706	0.9684

standardized regression coefficients (beta)

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

library( lm.beta )

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

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

	Estimate	Standardized	Std. Error	t value	Pr(>|t|)
(Intercept)	-4.241	0	4.346	-0.976	0.3315
x	2.102	0.9433	0.07471	28.14	9.604e-49

Table 30: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.56	0.8899	0.8888

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

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

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-4.241	4.346	-0.976	0.3315
x	2.102	0.07471	28.14	9.604e-49

Table 32: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.56	0.8899	0.8888

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
-4.241	2.102

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

0.9433

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 - not robust

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-4.241	4.346	-0.976	0.3315
x	2.102	0.07471	28.14	9.604e-49

Table 35: Fitting linear model: y ~ x

Observations	Residual Std. Error	R^2	Adjusted R^2
100	21.56	0.8899	0.8888

```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) -4.241025   4.302141 -0.9858   0.3267
## x            2.102283   0.069402 30.2915   <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) -4.241025   4.258903 -0.9958   0.3218
## x            2.102283   0.068704 30.5991   <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) -4.241025   4.258903 -0.9958   0.3218
## x            2.102283   0.068704 30.5991   <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) -4.241025   4.361841 -0.9723   0.3333
## x            2.102283   0.070508 29.8161   <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) -4.241025   4.361841 -0.9723   0.3333
## x            2.102283   0.070508 29.8161   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```