Applied Econometrics with R

Karim Kilani

2019

Settings and appearance

```r
knitr::opts_chunk$set(echo=TRUE)
options(prompt="R>",digits=4)
1.32222222
```

```
## [1] 1.322
```

- Replaces R prompt $>$ by a new prompt R$>$.
- Reduces the number of digits shown when printing numbers from 7 (default) to 4 digits.

Introductory R Session

Example 1: The demand for economics journals

```
data("Journals",package="AER")
dim(Journals)
```

```
## [1] 180  10
```

```
names(Journals)[1:4]
```

```
## [1] "title"       "publisher" "society"    "price"
```

```
names(Journals)[5:7]
```

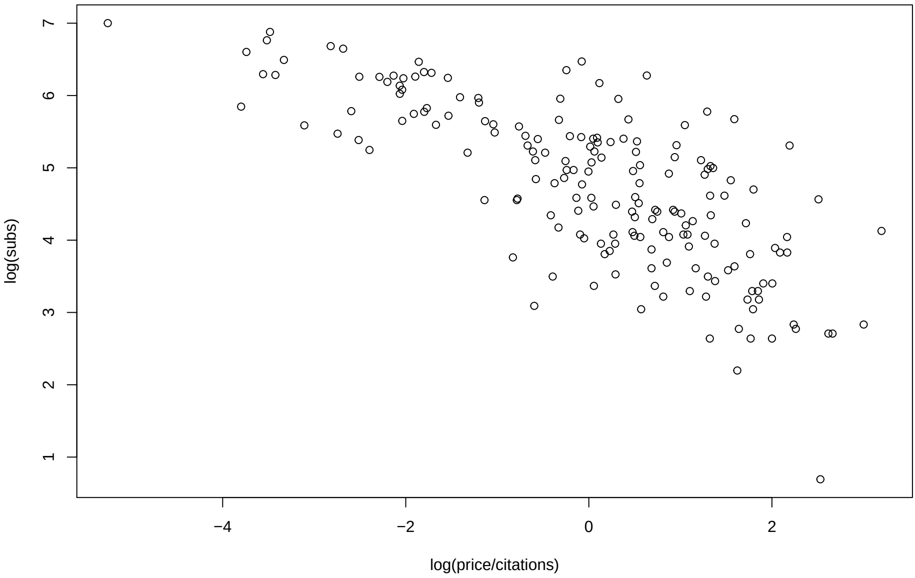
```
## [1] "pages"       "charpp"     "citations"
```

```
names(Journals)[8:10]
```

```
## [1] "foundingyear" "subs"           "field"
```

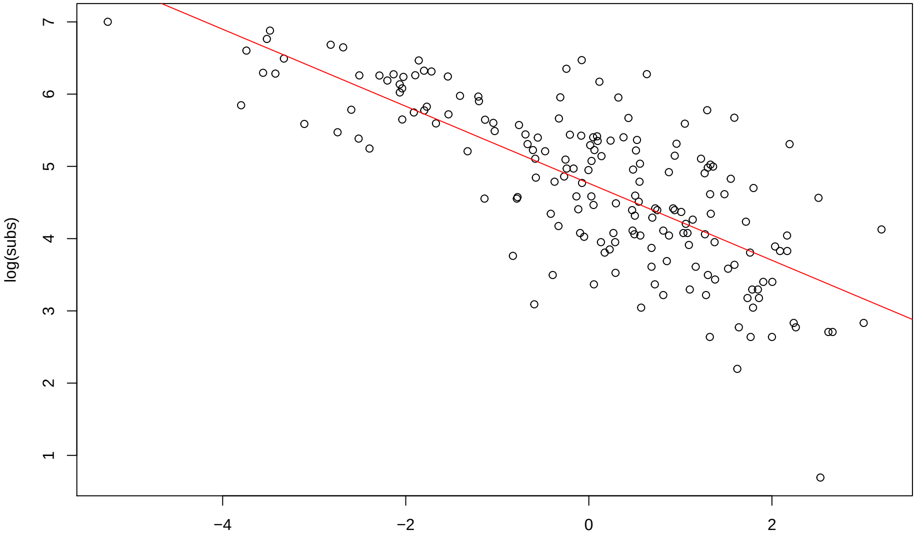
- We study the relation between the demand for economics journals and their price.
- Demand is measured by the number of library subscriptions (subs).
- Price is measured by is the price per citation.
- A scatterplot of the logarithms of the variables can be constructed.

```r
plot(log(subs)~log(price/citations),data=Journals)
```



- The number of subscriptions is decreasing with price.
- The corresponding linear regression model can be easily fitted by ordinary least squares (OLS).

```
j_lm<-lm(log(subs)~log(price/citations),data=Journals)
plot(log(subs)~log(price/citations),data=Journals)
abline(j_lm,col="red")
```



- A detailed summary of the fitted model can be obtained.

```
summary(j_lm)
```

```
##
## Call:
## lm(formula = log(subs) ~ log(price/citations), data = Jourr
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7248 -0.5361  0.0372  0.4662  1.8481
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.7662     0.0559    85.2   <2e-16 *
## log(price/citations)  -0.5331     0.0356   -15.0   <2e-16 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
##
## Residual standard error: 0.75 on 178 degrees of freedom
## Multiple R-squared:  0.557,  Adjusted R-squared:  0.555
```

- The estimated elasticity of the demand with respect to the price per citation is -0.5331, which is significantly different from 0.
- The $R^2 = 0.557$ of the model is quite satisfactory.

Example 2: Determinants of wages

- The application is the estimation of a wage equation in semi-logarithmic form based on data taken from Berndt (1991).
- They represent a random subsample (533 observations) of cross-section data from the May 1985 Current Population Survey.

```r
data("CPS1985",package="AER")
cps<-CPS1985
head(cps)
```

```
##       wage education experience age ethnicity region gender
## 1     5.10         8         21  35  hispanic  other female
## 1100  4.95         9         42  57      cauc  other female
## 2     6.67        12          1  19      cauc  other   male
## 3     4.00        12          4  22      cauc  other   male
## 4     7.50        12         17  35      cauc  other   male
## 5    13.07        13          9  28      cauc  other   male
##             sector union married
## 1    manufacturing    no     yes
## 1100 manufacturing    no     yes
## 2    manufacturing    no      no
## 3            other    no      no
## 4            other    no     yes
## 5            other   yes      no
```

- A wage equation is estimated with log(wage) as the dependent variable and education and experience (in number of years) as regressors.
- We estimate a multiple linear regression model by OLS (again via lm()).
- Quantile regressions (a refinement of the regression model) are fitted using the function rq() from the package quantreg.
- We need to specify τ, the quantiles that are to be modeled.
- We set (0.2, 0.35, 0.5, 0.65, 0.8).

```r
library("quantreg")
```

```
## Loading required package: SparseM

##
## Attaching package: 'SparseM'

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

```r
cps_lm <- lm(log(wage) ~ experience + I(experience^2) +
+ education, data = cps)
cps_rq <- rq(log(wage) ~ experience + I(experience^2) +
+ education, data = cps, tau = seq(0.2, 0.8, by = 0.15))
```

```r
cps2 <- data.frame(education = mean(cps$education),
 experience = min(cps$experience):max(cps$experience))
head(cps2)
```

```
##   education experience
## 1     13.02          0
## 2     13.02          1
## 3     13.02          2
## 4     13.02          3
## 5     13.02          4
## 6     13.02          5
```

```r
cps2 <- cbind(cps2, predict(cps_lm, newdata = cps2,
interval = "prediction"))
head(cps2)
```

```
##   education experience   fit    lwr   upr
## 1     13.02          0 1.689 0.7756 2.602
## 2     13.02          1 1.723 0.8110 2.635
## 3     13.02          2 1.757 0.8451 2.668
## 4     13.02          3 1.789 0.8781 2.700
## 5     13.02          4 1.820 0.9098 2.730
## 6     13.02          5 1.850 0.9404 2.760
```

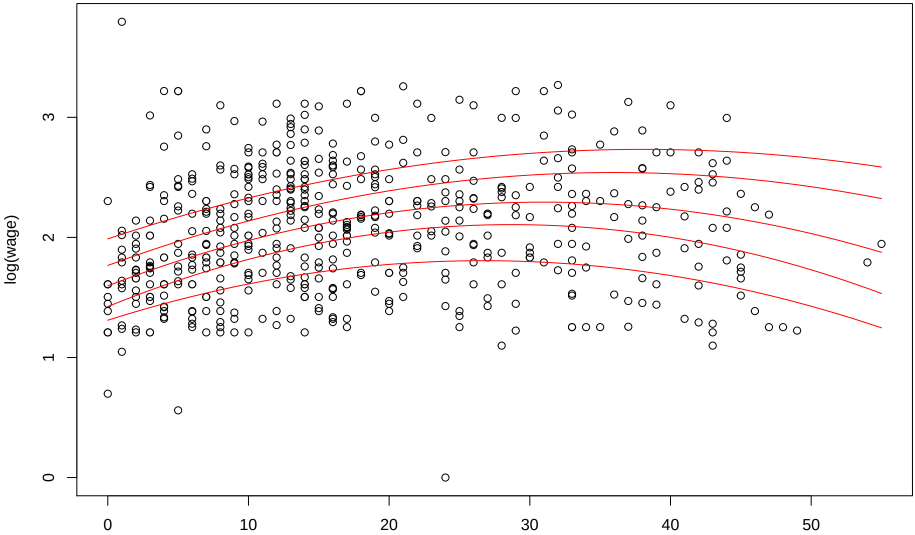
```r
cps2 <- cbind(cps2,predict(cps_rq, newdata = cps2, type = ""))
cps2[1:4,1:5]
```

```
##   education experience   fit    lwr   upr
## 1     13.02          0 1.689 0.7756 2.602
## 2     13.02          1 1.723 0.8110 2.635
## 3     13.02          2 1.757 0.8451 2.668
## 4     13.02          3 1.789 0.8781 2.700
```

```r
cps2[1:4,6:10]
```

```
##   tau= 0.20 tau= 0.35 tau= 0.50 tau= 0.65 tau= 0.80
## 1     1.310     1.424     1.597     1.766     1.988
## 2     1.347     1.471     1.641     1.809     2.026
## 3     1.382     1.516     1.684     1.850     2.064
## 4     1.415     1.559     1.725     1.890     2.101
```

```r
plot(log(wage) ~ experience, data = cps)
for(i in 6:10) lines(cps2[,i]~experience,
data = cps2,col="red")
```



Getting Started

- http://www.R-project.org/

Packages

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

```
## Loading required package: car
## Loading required package: carData
## 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
##
## Attaching package: 'survival'
## The following object is masked from 'package:quantreg':
```

Working with R

Handling objects

```
objects()
```

```
## [1] "cps"       "cps_lm"    "cps_rq"    "CPS1985"   "cps2"
## [7] "j_lm"      "Journals"
```

- There are currently eight objects, resulting from the introductory session
- This is not the complete list of available objects, given that some objects must already exist prior to the execution of any commands.

```r
search()
```

```
##  [1] ".GlobalEnv"         "package:AER"         "package:survi
##  [4] "package:sandwich"   "package:lmtest"      "package:zoo"
##  [7] "package:car"        "package:carData"     "package:quant
## [10] "package:SparseM"    "package:stats"       "package:graph
## [13] "package:grDevices"  "package:utils"       "package:datas
## [16] "package:methods"    "Autoloads"           "package:base'
```

```r
x <- 2
objects()
```

```
## [1] "cps"        "cps_lm"     "cps_rq"     "CPS1985"    "cps2"
## [7] "j_lm"       "Journals"   "x"
```

```r
remove(x)
objects()
```

```
## [1] "cps"        "cps_lm"     "cps_rq"     "CPS1985"    "cps2"
## [7] "j_lm"       "Journals"
```

Calling functions

```r
log(16, 2)
```

```
## [1] 4
```

```r
log(x = 16, 2)
```

```
## [1] 4
```

```r
log(16, base = 2)
```

```
## [1] 4
```

```r
log(base = 2, x = 16)
```

```
## [1] 4
```

Classes and generic functions

```
class(CPS1985)
```

```
## [1] "data.frame"
```

##File management

```r
getwd()
```

```
## [1] "/cloud/project"
```

Getting Help

- ?options or help("options").
- example("lm").

R as a Calculator

```r
1+1
```

```
## [1] 2
```

```r
2^3
```

```
## [1] 8
```

```r
log(exp(sin(pi/4)^2)*exp(cos(pi/4)^2))
```

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

Vector arithmetic

```r
x <- c(1.8, 3.14, 4, 88.169, 13)
length(x)
```

```
## [1] 5
```

```r
2*x+3
```

```
## [1]   6.60   9.28  11.00 179.34  29.00
```

```r
5:1*x+1:5
```

```
## [1]  10.00  14.56  15.00 180.34  18.00
```

Subsetting vectors

```r
x
```

```
## [1]  1.80  3.14  4.00 88.17 13.00
```

```r
x[c(1,4)]
```

```
## [1]  1.80 88.17
```

```r
x[-c(2,3,5)]
```

```
## [1]  1.80 88.17
```

Patterned vectors

```r
ones <- rep(1,10)
even <- seq(from = 2, to = 20, by = 2)
trend <- 1981:2005
c(ones,even)
```

```
##  [1]  1  1  1  1  1  1  1  1  1  1  2  4  6  8 10 12 14 16
```

Matrix Operations

```
A <- matrix(1:6, nrow = 2)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
matrix(1:6, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Transpose

```r
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
t(A)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

Dimensions

```
dim(A)
```

```
## [1] 2 3
```

```
nrow(A)
```

```
## [1] 2
```

```
ncol(A)
```

```
## [1] 3
```

Inverting of a Matrix

```r
A1 <- A[1:2, c(1, 3)]
solve(A1)
```

```
##       [,1]  [,2]
## [1,] -1.5  1.25
## [2,]  0.5 -0.25
```

```r
A1 %*% solve(A1)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Patterned matrices

```
diag(4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
diag(1,4,4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
diag(rep(c(1,2),c(10, 10)))
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,
##  [1,]    1    0    0    0    0    0    0    0    0     0
```

```r
diag(rep(c(1,2),c(5,5)))
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    0    0    0    0    0    0    0    0     0
##  [2,]    0    1    0    0    0    0    0    0    0     0
##  [3,]    0    0    1    0    0    0    0    0    0     0
##  [4,]    0    0    0    1    0    0    0    0    0     0
##  [5,]    0    0    0    0    1    0    0    0    0     0
##  [6,]    0    0    0    0    0    2    0    0    0     0
##  [7,]    0    0    0    0    0    0    2    0    0     0
##  [8,]    0    0    0    0    0    0    0    2    0     0
##  [9,]    0    0    0    0    0    0    0    0    2     0
## [10,]    0    0    0    0    0    0    0    0    0     2
```

- Yields a diagonal matrix of size 10×10 whose first 5 diagonal elements are 1, while the remaining ones are 2.

```r
A1
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
```

```r
diag(A1)
```

```
## [1] 1 6
```

- Extract the diagonal from an existing matrix.

```
A1
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
```

```
upper.tri(A1)
```

```
##       [,1]  [,2]
## [1,] FALSE  TRUE
## [2,] FALSE FALSE
```

```
lower.tri(A1)
```

```
##       [,1]  [,2]
## [1,] FALSE FALSE
## [2,]  TRUE FALSE
```

---

* Positions

- Add a column of ones to $A1$.

Combining matrices

```r
rbind(A1,diag(4, 2))
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    4    0
## [4,]    0    4
```

- Combines $A1$ and diag(4, 2) by rows.

R as a Programming Language

The mode of a vector

```r
x <- c(1.8, 3.14, 4, 88.169, 13)
```

Logical and character vectors

```
x > 3.5
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE
```

```
names(x)<- c("a", "b", "c", "d", "e")
x
```

```
##     a     b     c     d     e
##  1.80  3.14  4.00 88.17 13.00
```

```
names(x) <- letters[1:5]
```

- Assign labels to objects.

```r
names(x) <- letters[1:5]
x
```

```
##     a     b     c     d     e
##  1.80  3.14  4.00 88.17 13.00
```

More on subsetting

```r
x[3:5]
```

```
##     c     d     e
##  4.00 88.17 13.00
```

```r
x[c("c", "d", "e")]
```

```
##     c     d     e
##  4.00 88.17 13.00
```

```r
x[x > 3.5]
```

```
##     c     d     e
##  4.00 88.17 13.00
```

Lists

```
mylist <- list(sample=rnorm(5),family="normal distribution",
parameters=list(mean=0,sd=1))
```

```
mylist
```

```
## $sample
## [1]  1.1241 -0.3772 -0.4719 -0.3263 -0.9550
##
## $family
## [1] "normal distribution"
##
## $parameters
## $parameters$mean
## [1] 0
##
## $parameters$sd
## [1] 1
```

```r
mylist[[1]]
```

```
## [1]  1.1241 -0.3772 -0.4719 -0.3263 -0.9550
```

```r
mylist[["sample"]]
```

```
## [1]  1.1241 -0.3772 -0.4719 -0.3263 -0.9550
```

```r
mylist$sample
```

```
## [1]  1.1241 -0.3772 -0.4719 -0.3263 -0.9550
```

```r
mylist[[3]]$mean
```

```
## [1] 0
```

```r
mylist[[3]]$sd
```

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

Logical comparisons

```r
x <- c(1.8, 3.14, 4, 88.169, 13)
x > 3 & x <= 4
```

```
## [1] FALSE  TRUE  TRUE FALSE FALSE
```

```r
which(x > 3 & x <= 4)
```

```
## [1] 2 3
```

```r
all(x > 3)
```

```
## [1] FALSE
```

```r
all(x > 3)
```

```
## [1] FALSE
```

```r
any(x > 3)
```

```
## [1] TRUE
```

```r
(1.5 - 0.5) == 1
```

```
## [1] TRUE
```

```r
(1.9 - 0.9) == 1
```

```
## [1] FALSE
```

```r
all.equal(1.9 - 0.9, 1)
```

```
## [1] TRUE
```

```r
identical(1.5 - 0.5,1)
```

```
## [1] TRUE
```

```r
identical(1.9 - 0.9,1)
```

```
## [1] FALSE
```

```r
7 + TRUE
```

```
## [1] 8
```

```r
7+FALSE
```

```
## [1] 7
```

Coercion

```
x
```

```
## [1]  1.80  3.14  4.00 88.17 13.00
```

```
is.numeric(x)
```

```
## [1] TRUE
```

```
is.character(x)
```

```
## [1] FALSE
```

```r
as.character(x)
```

```
## [1] "1.8"     "3.14"    "4"       "88.169" "13"
```

```r
c(1, "a")
```

```
## [1] "1" "a"
```

Random number generation

```
set.seed(123)
rnorm(2)
```

```
## [1] -0.5605 -0.2302
```

```
rnorm(2)
```

```
## [1] 1.55871 0.07051
```

```
set.seed(123)
rnorm(2)
```

```
## [1] -0.5605 -0.2302
```

- Setting the seed to a specific value, simulations can be made exactly reproducible.

```r
sample(1:5)
```

```
## [1] 5 1 2 3 4
```

```r
sample(c("male","female"),size=5,replace=TRUE,
prob = c(0.2, 0.8))
```

```
## [1] "female" "male"   "female" "female" "female"
```

Flow control

```
x <- c(1.8, 3.14, 4, 88.169, 13)
if (rnorm(1) > 0) sum(x) else mean(x)
```

```
## [1] 22.02
```

```
ifelse(x > 4,sqrt(x),x^2)
```

```
## [1]  3.240  9.860 16.000  9.390  3.606
```

```r
x
```

```
## [1]  1.80  3.14  4.00 88.17 13.00
```

```r
for(i in 2:5) {x[i]<-x[i]-x[i-1]}
x
```

```
## [1]   1.80   1.34   2.66  85.51 -72.51
```

```r
x
```

```
## [1]   1.80   1.34   2.66  85.51 -72.51
```

```r
while(sum(x)<100) {x<-2*x}
x
```

```
## [1]   14.40   10.72   21.28  684.07 -580.07
```

Writing functions

```
cmeans <- function(X)
{
rval <- rep(0, ncol(X))
for(j in 1:ncol(X))
{
mysum<-0
for(i in 1:nrow(X)) mysum<-mysum+X[i,j]
rval[j] <- mysum/nrow(X)
}
return(rval)
}
```

```r
X <- matrix(1:20, ncol = 2)
X
```

```
##       [,1] [,2]
##  [1,]    1   11
##  [2,]    2   12
##  [3,]    3   13
##  [4,]    4   14
##  [5,]    5   15
##  [6,]    6   16
##  [7,]    7   17
##  [8,]    8   18
##  [9,]    9   19
## [10,]   10   20
```

```r
cmeans(X)
```

```
## [1]  5.5 15.5
```

```r
colMeans(X)
```

```
## [1]  5.5 15.5
```

```r
X <- matrix(rnorm(2*10^6), ncol = 2)
system.time(colMeans(X))
```

```
##    user  system elapsed
##   0.000   0.000   0.002
```

```r
system.time(cmeans(X))
```

```
##    user  system elapsed
##   0.136   0.000   0.134
```

Vectorized calculations

```
cmeans2 <- function(X) {
rval <- rep(0, ncol(X))
for(j in 1:ncol(X)) rval[j] <- mean(X[,j])
return(rval)
}
```

```r
system.time(cmeans2(X))
```

```
##    user  system elapsed
##   0.016   0.004   0.017
```

```r
apply(X, 2, mean)
```

```
## [1] -0.0005206 -0.0015578
```

```r
system.time(apply(X, 2, mean))
```

```
##    user  system elapsed
##   0.020   0.004   0.027
```

1. Element-wise computations should be avoided if vectorized computations are available.
2. Optimized solutions (if available) typically perform better than the generic for or apply() solution.
3. Loops can be written more compactly using the apply() function.

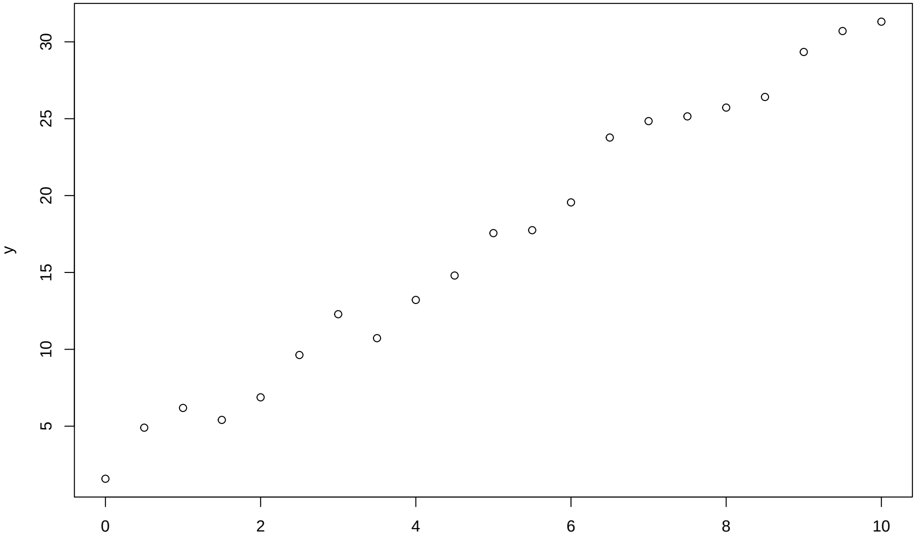
Formulas

```r
f<-y~x
class(f)
```

```
## [1] "formula"
```

- y is explained by x.

```
x<-seq(from = 0, to = 10, by = 0.5)
y<-2+3*x+rnorm(21)
plot(f)
```



```r
lm(f)
```

```
##
## Call:
## lm(formula = f)
##
## Coefficients:
## (Intercept)            x
##        2.00         3.01
```

Data Management in R

Creation from scratch

```
mydata<-data.frame(one=1:10,two=11:20,three=21:30)
mydata
```

```
##    one two three
## 1    1  11    21
## 2    2  12    22
## 3    3  13    23
## 4    4  14    24
## 5    5  15    25
## 6    6  16    26
## 7    7  17    27
## 8    8  18    28
## 9    9  19    29
## 10  10  20    30
```

```r
mydata<-as.data.frame(matrix(1:30,ncol=3))
names(mydata)<-c("one","two","three")
mydata
```

```
##    one two three
## 1    1  11    21
## 2    2  12    22
## 3    3  13    23
## 4    4  14    24
## 5    5  15    25
## 6    6  16    26
## 7    7  17    27
## 8    8  18    28
## 9    9  19    29
## 10  10  20    30
```

Subset selection

```r
mydata$two
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
mydata[,"two"]
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
mydata[,2]
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
attach(mydata)
two
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
detach(mydata)
```

```r
with(mydata,two)
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
mydata.sub<-subset(mydata,two <= 16,select=-two)
mydata.sub
```

```
##   one three
## 1   1    21
## 2   2    22
## 3   3    23
## 4   4    24
## 5   5    25
## 6   6    26
```

Import and export