Basic usage of R

Deepayan Sarkar

R basics

Basics of using R

R is more flexible than a regular calculator

- In fact, R is a full programming language
- Most standard data analysis methods are already implemented
- Can be extended by writing add-on packages
- Thousands of add-on packages are available

Major concepts

- Variables (in the context of programming)
- Data structures needed for data analyis
- Functions (set of instructions for performing a procedure)

Variables

- Variables are symbols that may be associated with different values
- Computations involving variables are done using their current value

```
x <- 10 # assignment
sqrt(x)
```

```
[1] 3.162278
```

```
x <- -1
sqrt(x)
```

```
Warning in sqrt(x): NaNs produced
```

```
[1] NaN
```

```
x <- -1+0i
sqrt(x)
```

```
[1] 0+1i
```

Data structures for data analysis

- Vectors
- Matrices
- Data frames (a spreadsheet-like data set)

- Lists (general collection of objects)

Atomic vectors

- Indexed collection of homogeneous scalars, can be
  - Numeric / Integer
  - Categorical (factor)
  - Character
  - Logical (`TRUE` / `FALSE`)
- Missing values are allowed, indicated as `NA`
- Elements are indexed starting from 1
- ith element of vector `x` can be extracted using `x[i]`
- There are also more sophisticated forms of (vector) indexing

Atomic vectors: examples

```
month.name # built-in
```

```
 [1] "January"   "February"  "March"     "April"     "May"       "June"      "July"      "August"    "Se
[10] "October"   "November"  "December"
```

```
x <- rnorm(10)
x
```

```
 [1] -0.8894800 -1.2710620  0.8958026  1.2368336  1.6313476  0.1790313 -0.2952426 -0.6430979  1.4552944
```

```
str(x) # useful function
```

```
 num [1:10] -0.889 -1.271 0.896 1.237 1.631 ...
```

```
str(month.name)
```

```
 chr [1:12] "January" "February" "March" "April" "May" "June" "July" "August" "September" "October" "No
```

```
m <- sample(1:12, 30, rep = TRUE)
m
```

```
 [1]  4  4 11  4  6 10  3  8 12 11  7 12  1  8 12 12  3  6  8 12 10  7  9  4 12  5 11  2 10  7
```

```
mf <- factor(m, levels = 1:12, labels = month.name)
mf
```

```
 [1] April     April     November  April     June      October   March     August    December  November
[12] December  January   August    December  December  March     June      August    December  October
[23] September April     December  May       November  February  October   July
Levels: January February March April May June July August September October November December
```

```
str(m)
```

```
 int [1:30] 4 4 11 4 6 10 3 8 12 11 ...
```

```
str(mf)
```

```
 Factor w/ 12 levels "January","February",..: 4 4 11 4 6 10 3 8 12 11 ...
```

Atomic vectors

- "Scalars" are just vectors of length 1

```
str(numeric(2))
```

```
 num [1:2] 0 0
```

```
str(numeric(1))
```

```
 num 0
```

```
str(0)
```

```
 num 0
```

- Vectors can have length zero

```
numeric(0)
```

```
numeric(0)
```

```
logical(0)
```

```
logical(0)
```

Types of indexing

- Indexing refers to extracting subsets of vectors (or other kinds of data)
- R supports several kinds of indexing:
  - Indexing by a vector of positive integers
  - Indexing by a vector of negative integers
  - Indexing by a logical vector
  - Indexing by a vector of names
- The "standard" C-like indexing with a scalar (vector of length 1):

```
month.name[2] # the first index is 1, not 0
```

```
[1] "February"
```

- The "index" can also be an integer vector

```
month.name[c(2, 4, 6, 9, 11)]
```

```
[1] "February"  "April"     "June"      "September" "November"
```

- Elements can be repeated

```
month.name[c(2, 2, 6, 4, 6, 11)]
```

```
[1] "February" "February" "June"     "April"    "June"     "November"
```

- "Out-of-bounds" indexing give `NA` (missing)

```
month.name[13]
```

```
[1] NA
```

```
month.name[seq(1, by = 2, length.out = 8)]
```

```
[1] "January"   "March"     "May"       "July"      "September" "November"  NA          NA
```

- Negative integers omit the specified entries

```r
month.name[-2]
```

```
 [1] "January"   "March"     "April"     "May"       "June"      "July"      "August"    "September" "O
[10] "November"  "December"
```

```r
month.name[-c(2, 4, 6, 9, 11)]
```

```
[1] "January"  "March"    "May"      "July"     "August"   "October"  "December"
```

- Cannot be mixed with positive integers

```r
month.name[c(2, -3)]
```

```
Error in month.name[c(2, -3)]: only 0's may be mixed with negative subscripts
```

- Zero has a special meaning - doesn't select anything

```r
month.name[0]
```

```
character(0)
```

```r
month.name[integer(0)] ## same as empty index
```

```
character(0)
```

```r
month.name[c(1, 2, 0, 11, 12)]
```

```
[1] "January"  "February" "November" "December"
```

```r
month.name[-c(1, 2, 0, 11, 12)]
```

```
[1] "March"     "April"     "May"       "June"      "July"      "August"    "September" "October"
```

- Indexing by logical vector: select `TRUE` elements

```r
month.name[c(TRUE, FALSE, FALSE)] # index replicated
```

```
[1] "January" "April"   "July"    "October"
```

```r
print(i <- substring(month.name, 1, 1) == "J")
```

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

```r
month.name[i]
```

```
[1] "January" "June"    "July"
```

- Common use: extract subset satisfying a certain condition (also called "filtering")

```r
(x <- rnorm(20))
```

```
 [1] -0.84295920 -0.35924343  1.09843469  0.79839075 -0.76003950  0.17105579 -1.04201749  0.34942506  0
[10] -1.27118809 -0.06690657  0.62230315 -1.15121320 -1.01296190 -0.76648511  0.25785313  0.54744156 -0
[19] -1.71862042  1.24414823
```

```r
x[x > 0]
```

```
[1] 1.0984347 0.7983908 0.1710558 0.3494251 0.4473083 0.6223031 0.2578531 0.5474416 1.2441482
```

```r
mean(x[x > 0])
```

```
[1] 0.6151512
```

- Sometimes logical indexing can be replaced by integer indexing using `which()`

```r
i
```

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

```r
which(i)
```

```
[1] 1 6 7
```

```r
month.name[ which(i) ]
```

```
[1] "January" "June"    "July"
```

```r
month.name[ -which(i) ] # same as month.name[ !i ]
```

```
[1] "February"  "March"     "April"     "May"       "August"    "September" "October"   "November"  "De
```

- But be careful about zero-length indices

```r
which( substring(month.name, 1, 1) == "B")
```

```
integer(0)
```

```r
-which( substring(month.name, 1, 1) == "B")
```

```
integer(0)
```

Lists

- Lists are vectors with arbitrary types of components
- May or may not have names
- Usual vector indexing by `[` works in the usual way
- Individual elements can be extracted either by name (`$`) or by position `x[[i]]`

Lists: examples

```r
ml <- list(imonth = m, fmonth = mf)
str(ml)
```

```
List of 2
 $ imonth: int [1:30] 4 4 11 4 6 10 3 8 12 11 ...
 $ fmonth: Factor w/ 12 levels "January","February",..: 4 4 11 4 6 10 3 8 12 11 ...
```

```r
ml$imonth
```

```
 [1]  4  4 11  4  6 10  3  8 12 11  7 12  1  8 12 12  3  6  8 12 10  7  9  4 12  5 11  2 10  7
```

```r
ml[[2]]
```

```
 [1] April     April     November  April     June      October   March     August    December  November
[12] December  January   August    December  December  March     June      August    December  October
[23] September April     December  May       November  February  October   July
Levels: January February March April May June July August September October November December
```

```r
ml[["fmonth"]]
```

```
 [1] April     April     November  April     June      October   March     August    December  November
[12] December  January   August    December  December  March     June      August    December  October
[23] September April     December  May       November  February  October   July
Levels: January February March April May June July August September October November December
```

Most common structures for statistical data

- Vectors, matrices / arrays: vectors with dimension

```r
VADeaths
```

```
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
```

```r
dim(VADeaths)
```

```
[1] 5 4
```

- Indexing works in same way, but in two dimensions

```r
VADeaths[1:2, c(2, 3)]
```

```
      Rural Female Urban Male
50-54          8.7       15.4
55-59         11.7       24.3
```

- Example: Indexing by "empty" index (selects all) and name

```r
VADeaths[, c("Rural Male", "Rural Female")]
```

```
      Rural Male Rural Female
50-54       11.7          8.7
55-59       18.1         11.7
60-64       26.9         20.3
65-69       41.0         30.9
70-74       66.0         54.3
```

- Data frames: lists that also behave like a matrix

```r
str(iris)
```

```
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
dim(iris)
```

```
[1] 150   5
```

```r
iris
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
```

14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa
28	5.2	3.5	1.5	0.2	setosa
29	5.2	3.4	1.4	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
31	4.8	3.1	1.6	0.2	setosa
32	5.4	3.4	1.5	0.4	setosa
33	5.2	4.1	1.5	0.1	setosa
34	5.5	4.2	1.4	0.2	setosa
35	4.9	3.1	1.5	0.2	setosa
36	5.0	3.2	1.2	0.2	setosa
37	5.5	3.5	1.3	0.2	setosa
38	4.9	3.6	1.4	0.1	setosa
39	4.4	3.0	1.3	0.2	setosa
40	5.1	3.4	1.5	0.2	setosa
41	5.0	3.5	1.3	0.3	setosa
42	4.5	2.3	1.3	0.3	setosa
43	4.4	3.2	1.3	0.2	setosa
44	5.0	3.5	1.6	0.6	setosa
45	5.1	3.8	1.9	0.4	setosa
46	4.8	3.0	1.4	0.3	setosa
47	5.1	3.8	1.6	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
49	5.3	3.7	1.5	0.2	setosa
50	5.0	3.3	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
56	5.7	2.8	4.5	1.3	versicolor
57	6.3	3.3	4.7	1.6	versicolor
58	4.9	2.4	3.3	1.0	versicolor
59	6.6	2.9	4.6	1.3	versicolor
60	5.2	2.7	3.9	1.4	versicolor
61	5.0	2.0	3.5	1.0	versicolor
62	5.9	3.0	4.2	1.5	versicolor
63	6.0	2.2	4.0	1.0	versicolor
64	6.1	2.9	4.7	1.4	versicolor
65	5.6	2.9	3.6	1.3	versicolor
66	6.7	3.1	4.4	1.4	versicolor
67	5.6	3.0	4.5	1.5	versicolor

68	5.8	2.7	4.1	1.0	versicolor
69	6.2	2.2	4.5	1.5	versicolor
70	5.6	2.5	3.9	1.1	versicolor
71	5.9	3.2	4.8	1.8	versicolor
72	6.1	2.8	4.0	1.3	versicolor
73	6.3	2.5	4.9	1.5	versicolor
74	6.1	2.8	4.7	1.2	versicolor
75	6.4	2.9	4.3	1.3	versicolor
76	6.6	3.0	4.4	1.4	versicolor
77	6.8	2.8	4.8	1.4	versicolor
78	6.7	3.0	5.0	1.7	versicolor
79	6.0	2.9	4.5	1.5	versicolor
80	5.7	2.6	3.5	1.0	versicolor
81	5.5	2.4	3.8	1.1	versicolor
82	5.5	2.4	3.7	1.0	versicolor
83	5.8	2.7	3.9	1.2	versicolor
84	6.0	2.7	5.1	1.6	versicolor
85	5.4	3.0	4.5	1.5	versicolor
86	6.0	3.4	4.5	1.6	versicolor
87	6.7	3.1	4.7	1.5	versicolor
88	6.3	2.3	4.4	1.3	versicolor
89	5.6	3.0	4.1	1.3	versicolor
90	5.5	2.5	4.0	1.3	versicolor
91	5.5	2.6	4.4	1.2	versicolor
92	6.1	3.0	4.6	1.4	versicolor
93	5.8	2.6	4.0	1.2	versicolor
94	5.0	2.3	3.3	1.0	versicolor
95	5.6	2.7	4.2	1.3	versicolor
96	5.7	3.0	4.2	1.2	versicolor
97	5.7	2.9	4.2	1.3	versicolor
98	6.2	2.9	4.3	1.3	versicolor
99	5.1	2.5	3.0	1.1	versicolor
100	5.7	2.8	4.1	1.3	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
103	7.1	3.0	5.9	2.1	virginica
104	6.3	2.9	5.6	1.8	virginica
105	6.5	3.0	5.8	2.2	virginica
106	7.6	3.0	6.6	2.1	virginica
107	4.9	2.5	4.5	1.7	virginica
108	7.3	2.9	6.3	1.8	virginica
109	6.7	2.5	5.8	1.8	virginica
110	7.2	3.6	6.1	2.5	virginica
111	6.5	3.2	5.1	2.0	virginica
112	6.4	2.7	5.3	1.9	virginica
113	6.8	3.0	5.5	2.1	virginica
114	5.7	2.5	5.0	2.0	virginica
115	5.8	2.8	5.1	2.4	virginica
116	6.4	3.2	5.3	2.3	virginica
117	6.5	3.0	5.5	1.8	virginica
118	7.7	3.8	6.7	2.2	virginica
119	7.7	2.6	6.9	2.3	virginica
120	6.0	2.2	5.0	1.5	virginica
121	6.9	3.2	5.7	2.3	virginica

```
122          5.6          2.8           4.9          2.0  virginica
123          7.7          2.8           6.7          2.0  virginica
124          6.3          2.7           4.9          1.8  virginica
125          6.7          3.3           5.7          2.1  virginica
126          7.2          3.2           6.0          1.8  virginica
127          6.2          2.8           4.8          1.8  virginica
128          6.1          3.0           4.9          1.8  virginica
129          6.4          2.8           5.6          2.1  virginica
130          7.2          3.0           5.8          1.6  virginica
131          7.4          2.8           6.1          1.9  virginica
132          7.9          3.8           6.4          2.0  virginica
133          6.4          2.8           5.6          2.2  virginica
134          6.3          2.8           5.1          1.5  virginica
135          6.1          2.6           5.6          1.4  virginica
136          7.7          3.0           6.1          2.3  virginica
137          6.3          3.4           5.6          2.4  virginica
138          6.4          3.1           5.5          1.8  virginica
139          6.0          3.0           4.8          1.8  virginica
140          6.9          3.1           5.4          2.1  virginica
141          6.7          3.1           5.6          2.4  virginica
142          6.9          3.1           5.1          2.3  virginica
143          5.8          2.7           5.1          1.9  virginica
144          6.8          3.2           5.9          2.3  virginica
145          6.7          3.3           5.7          2.5  virginica
146          6.7          3.0           5.2          2.3  virginica
147          6.3          2.5           5.0          1.9  virginica
148          6.5          3.0           5.2          2.0  virginica
149          6.2          3.4           5.4          2.3  virginica
150          5.9          3.0           5.1          1.8  virginica
```

Data Frames

- Rectangular (matrix-like) structure
- Each column is a vector
- Different columns can have different types
- Every column must have a name
- Most built-in data sets in R are data frames

Data input / output

- Statistical data are usually structured like a spreadsheet (e.g., Excel)
- Typical approach: read data from spreadsheet file into data frame
- Easiest route:
  - R itself cannot read Excel files directly
  - Save as CSV file from Excel
  - Read with `read.csv()` or `read.table()` (more flexible)
- Alternative option:
  - Use "Import Dataset" menu item in R Studio (requires add-on package)

- Data frames can be exported as a spreadsheet file using `write.csv()` or `write.table()`

```
data(Cars93, package = "MASS")
write.csv(Cars93, file = "cars93.csv") # export
cars <- read.csv("cars93.csv") # import (path relative to working directory)
```

- Another example: Average global temperature

```
globalTemp <- read.csv("data/annual.csv")
str(globalTemp)
```

```
'data.frame':	151 obs. of  8 variables:
 $ Year      : int  1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 ...
 $ Temp      : num  -0.411 -0.518 -0.315 -0.491 -0.296 -0.295 -0.315 -0.268 -0.287 -0.282 ...
 $ CO2       : num  286 287 287 287 287 ...
 $ CH4       : num  838 840 841 842 844 ...
 $ NO2       : num  289 289 289 289 289 ...
 $ Irradiance: num  1361 1361 1361 1361 1361 ...
 $ Nino_SST  : num  26.7 26.4 26.2 26.3 26.3 ...
 $ Volcano   : num  0.00281 0.00859 0.01318 0.00707 0.00302 ...
```

Functions

- Most useful things in R happen by calling functions
- Functions have one or more arguments
  - All arguments have names
  - Arguments may be compulsory or optional
  - Optional arguments have "default" values
  - Arguments may or may no be named
  - Optional arguments are usually named
- Functions normally also have a useful "return" value
- For example, linear models are fit using the function `lm()`

Other important features

- Getting help
- Generic functions and methods
- How matrices and arrays are implemented in R
- Homework — read the following tutorials
  - Introduction and Language Overview I
  - Language Overview II
  - Supplementary files: Gcsemv.txt, iris.xls,

Basic statistical problems

Steps in a typical data analysis problem

- Formulate purpose of the analysis, e.g.,

- prediction
- testing / identifying important variables

- Build model
- Check and refine model
- Use model for further insight

Types of data

- Categorical
- Numeric (continuous)
- Also discrete numeric (e.g., count data)
- Data roles
  - Response or outcome variable
  - Predictors or explanatory variable

Simplest case: one predictor, one response

Predictor	Response	Problem type
Categorical	Numeric	*t*-test, ANOVA (testing)
Numeric	Numeric	Regression (prediction, testing)
Categorical	Categerical	Test of independence (testing)
Either	Categerical	Classification (prediction)

(the general class of problems with a continuous response is often called "regression")

Examples

Example: sleep data

Amount of extra sleep (in hours) after taking three sleep-inducing drugs

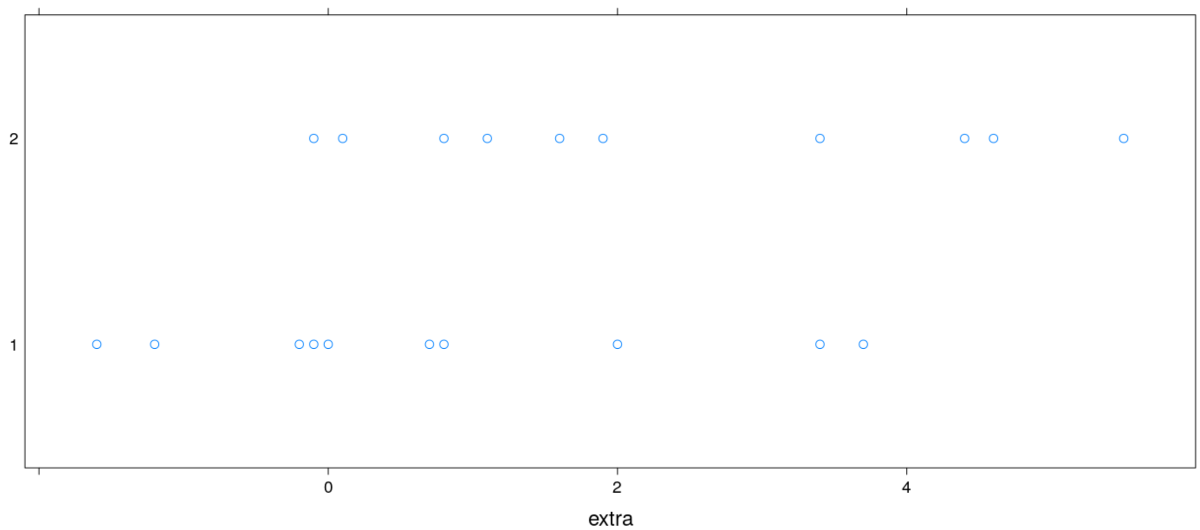
```
sleep
```

```
   extra group ID
1    0.7     1  1
2   -1.6     1  2
3   -0.2     1  3
4   -1.2     1  4
5   -0.1     1  5
6    3.4     1  6
7    3.7     1  7
8    0.8     1  8
9    0.0     1  9
10   2.0     1 10
11   1.9     2  1
12   0.8     2  2
13   1.1     2  3
14   0.1     2  4
```

```
15  -0.1     2  5
16   4.4     2  6
17   5.5     2  7
18   1.6     2  8
19   4.6     2  9
20   3.4     2 10
```

First step: plot the data

```r
library(lattice)
stripplot(group ~ extra, data = sleep)
```



Possible questions:

- Do the drugs work?
- Is one of the drugs more effective than the other?

Example: 1993 passenger car models

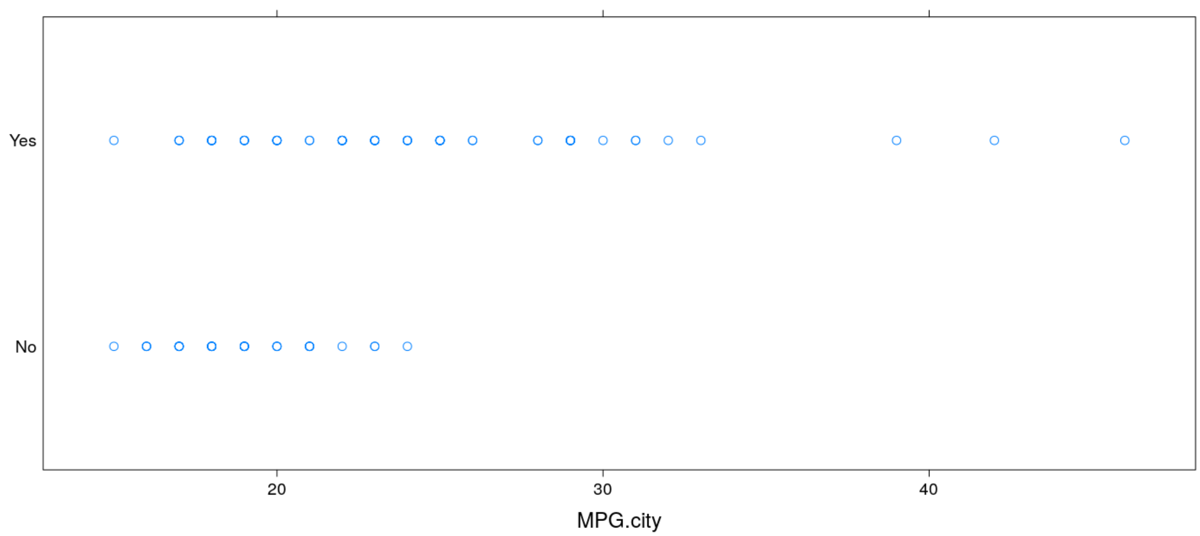
```r
data(Cars93, package = "MASS")
str(Cars93)
```

```
'data.frame':   93 obs. of  27 variables:
 $ Manufacturer      : Factor w/ 32 levels "Acura","Audi",..: 1 1 2 2 3 4 4 4 4 5 ...
 $ Model             : Factor w/ 93 levels "100","190E","240",..: 49 56 9 1 6 24 54 74 73 35 ...
 $ Type              : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3 3 3 2 2 3 2 ...
 $ Min.Price         : num  12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...
 $ Price             : num  15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...
 $ Max.Price         : num  18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...
 $ MPG.city          : int  25 18 20 19 22 22 19 16 19 16 ...
 $ MPG.highway       : int  31 25 26 26 30 31 28 25 27 25 ...
 $ AirBags           : Factor w/ 3 levels "Driver & Passenger",..: 3 1 2 1 2 2 2 2 2 2 ...
 $ DriveTrain        : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3 2 2 3 2 2 ...
 $ Cylinders         : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2 4 4 4 5 ...
 $ EngineSize        : num  1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
 $ Horsepower        : int  140 200 172 172 208 110 170 180 170 200 ...
```

```
 $ RPM               : int  6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...
 $ Rev.per.mile      : int  2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...
 $ Man.trans.avail   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
 $ Fuel.tank.capacity: num  13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...
 $ Passengers        : int  5 5 5 6 4 6 6 6 5 6 ...
 $ Length            : int  177 195 180 193 186 189 200 216 198 206 ...
 $ Wheelbase         : int  102 115 102 106 109 105 111 116 108 114 ...
 $ Width             : int  68 71 67 70 69 69 74 78 73 73 ...
 $ Turn.circle       : int  37 38 37 37 39 41 42 45 41 43 ...
 $ Rear.seat.room    : num  26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...
 $ Luggage.room      : int  11 15 14 17 13 16 17 21 14 18 ...
 $ Weight            : int  2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...
 $ Origin            : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...
 $ Make              : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5 6 7 9 8 10 ...
```

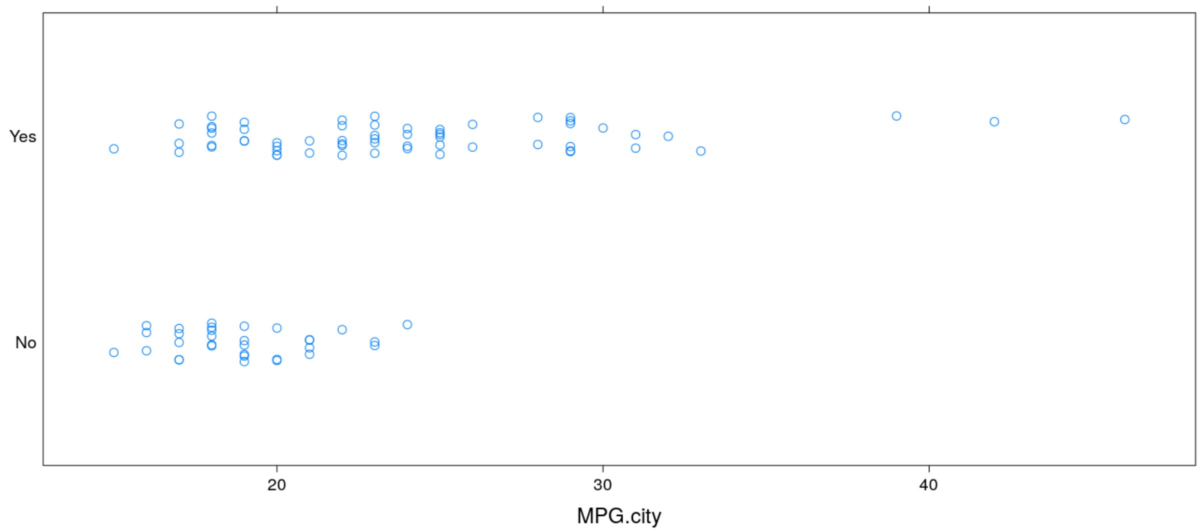
Exploratory plots

```
stripplot(Man.trans.avail ~ MPG.city, data = Cars93)
```



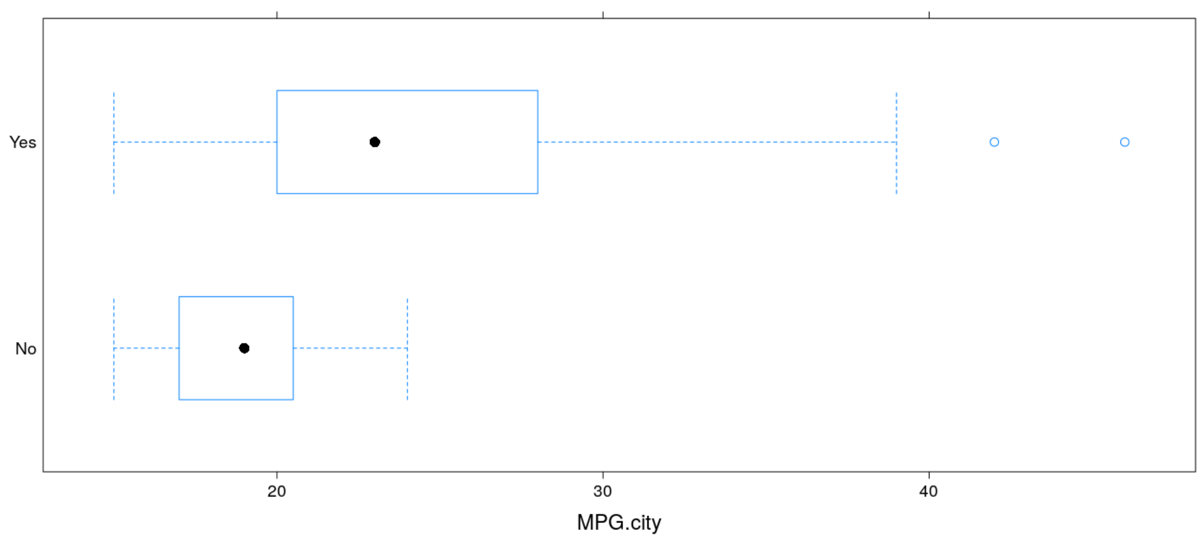
- Are manual transmission cars more fuel efficient?

```
stripplot(Man.trans.avail ~ MPG.city, data = Cars93, jitter = TRUE)
```



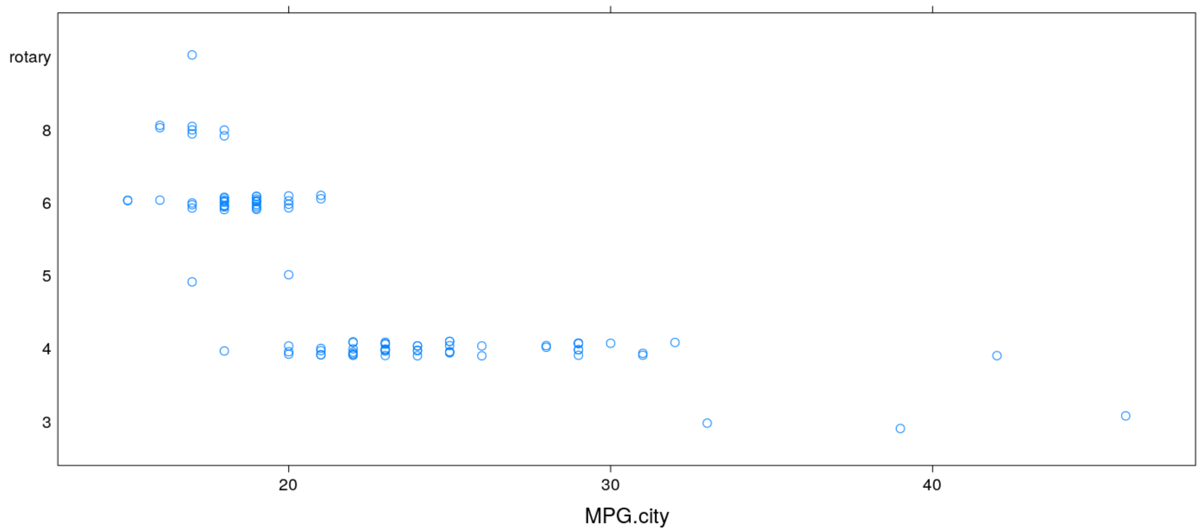
- Are manual transmission cars more fuel efficient?

```
bwplot(Man.trans.avail ~ MPG.city, data = Cars93)
```



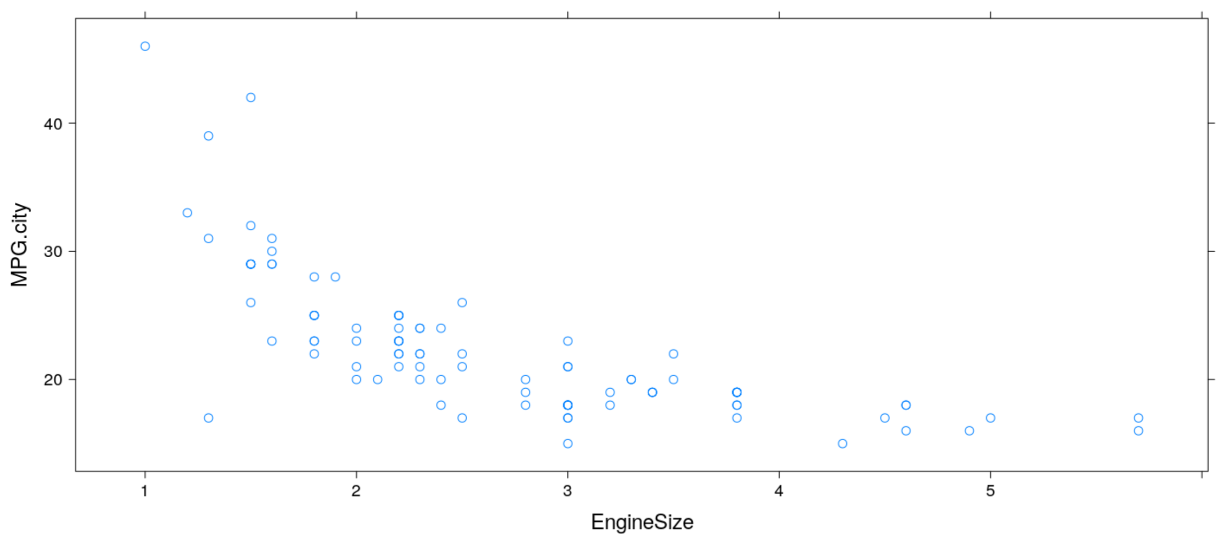
- Are manual transmission cars more fuel efficient?

```
stripplot(Cylinders ~ MPG.city, data = Cars93, jitter = TRUE)
```



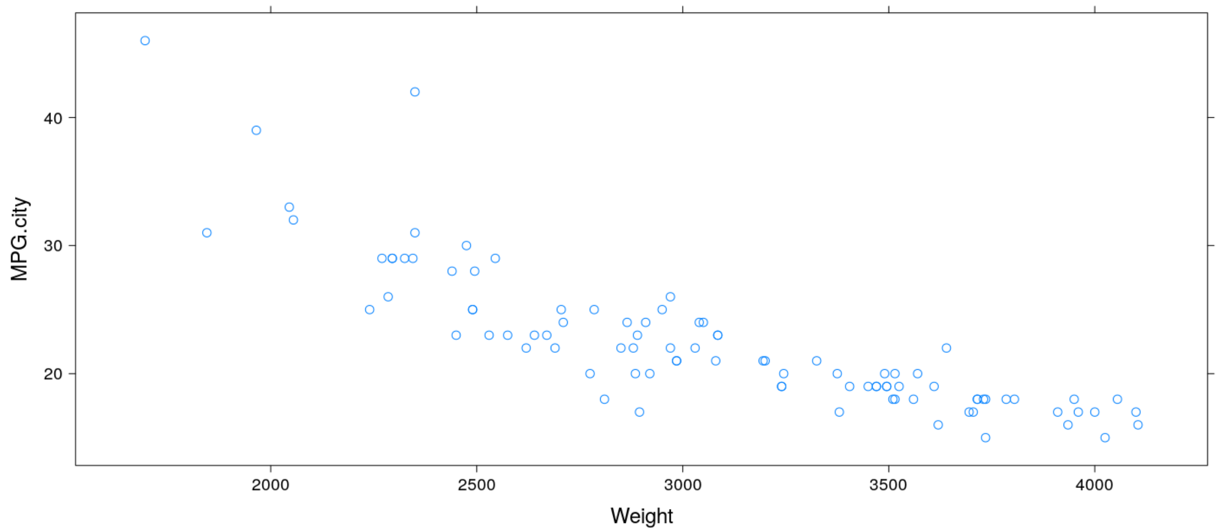
- Does fuel efficiency depend on number of cylinders?

```
xyplot(MPG.city ~ EngineSize, data = Cars93)
```



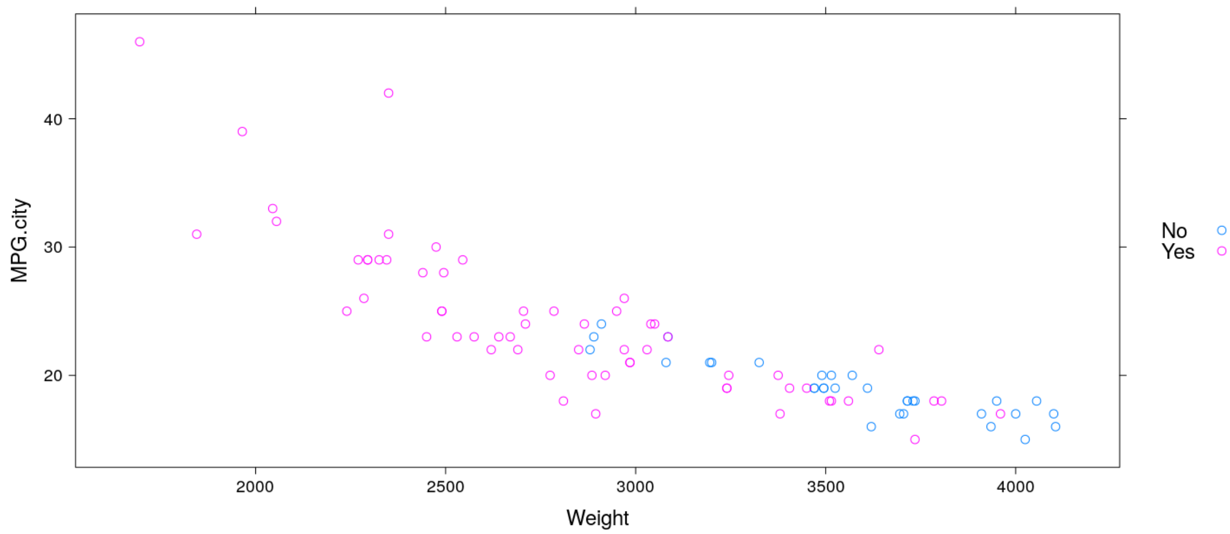
- Does fuel efficiency depend on engine size?

```
xyplot(MPG.city ~ Weight, data = Cars93)
```



- Does fuel efficiency depend on weight?

```
xyplot(MPG.city ~ Weight, data = Cars93, groups = Man.trans.avail, auto.key = list(space = "right"))
```



- How does dependence on weight vary with manual transmission?

Fitting models in R : demo

- Two-sample comparison (categorical vs categorical)
  - Test of independence
  - χ^2-test
  - Permutation test
- Two-sample comparisons:

    - Nonparametric (rank-sum test)
    - Two-sample *t*-test
    - Permutation test?
- Multi-sample comparisons: ANOVA
- Regression

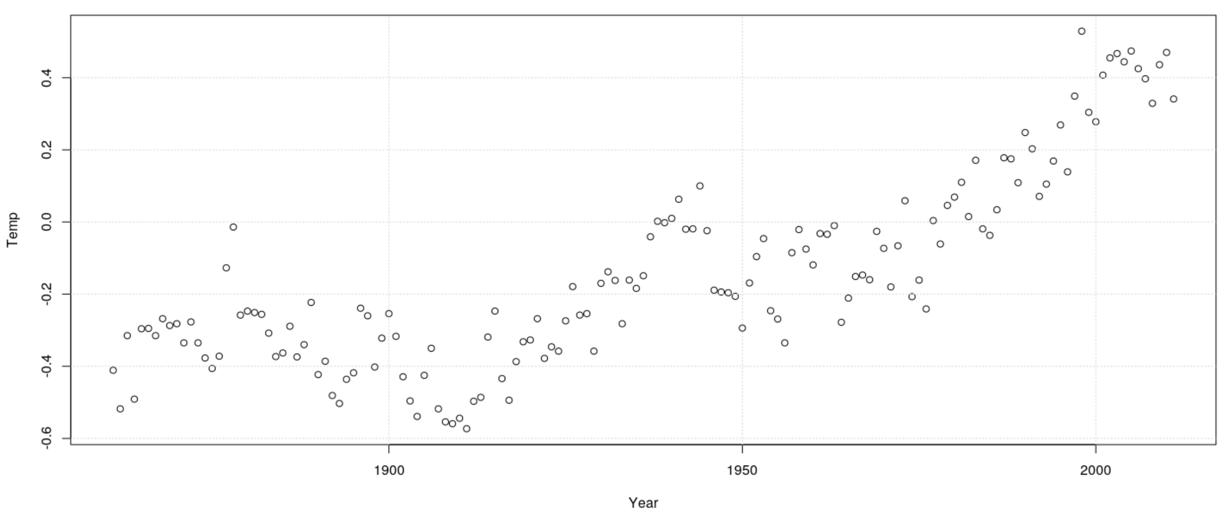
The formula-data interface: some details

The formula interface

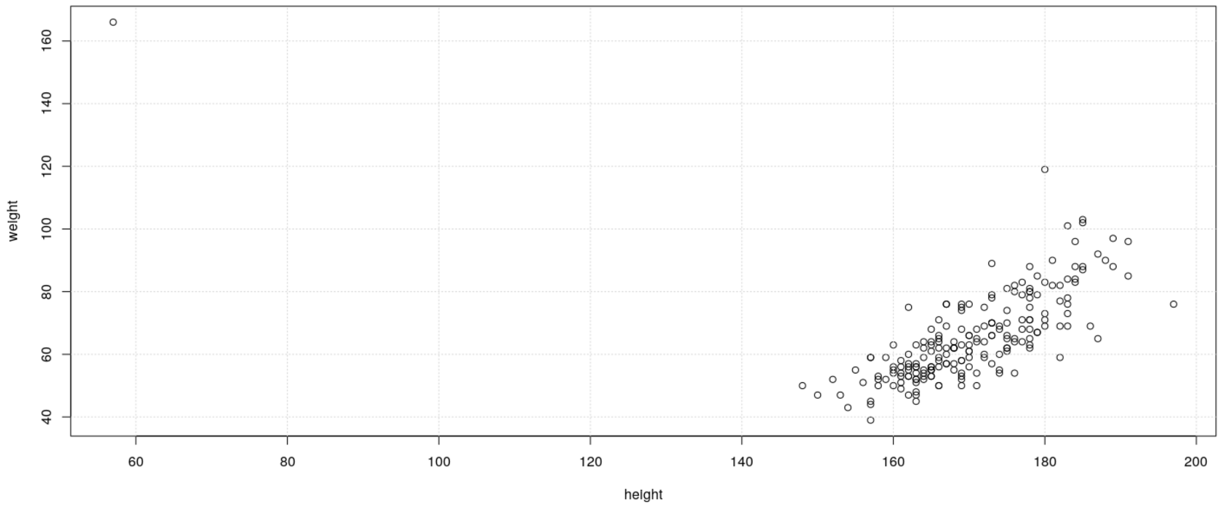
- The same task can be performed in many different ways in R
- This is especially true for graphics (more details later)
    - base R graphics (*graphics* package)
    - *lattice* package
    - *ggplot2* package
- One generally useful approach is the "formula interface"
- Functions using the formula interface have two main arguments
    - A formula specifying the roles of variables (e.g., `y ~ x`)
    - A dataset where the variables in the formula are defined

The formula interface in base graphics

```
plot(Temp ~ Year, data = globalTemp)
grid()
```

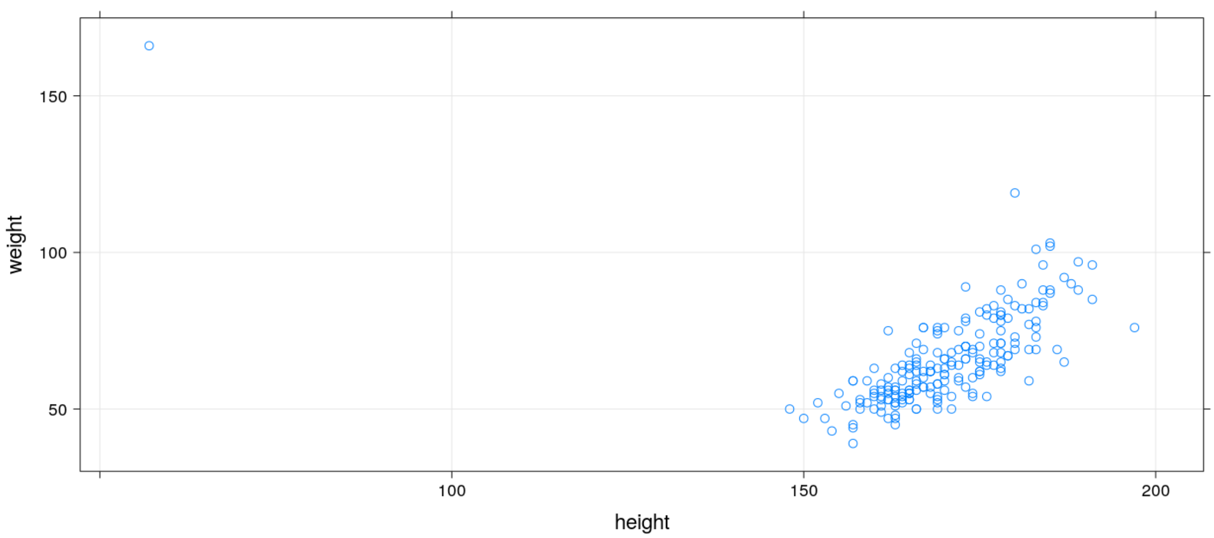



```
data(Davis, package = "carData")
plot(weight ~ height, Davis)
grid()
```



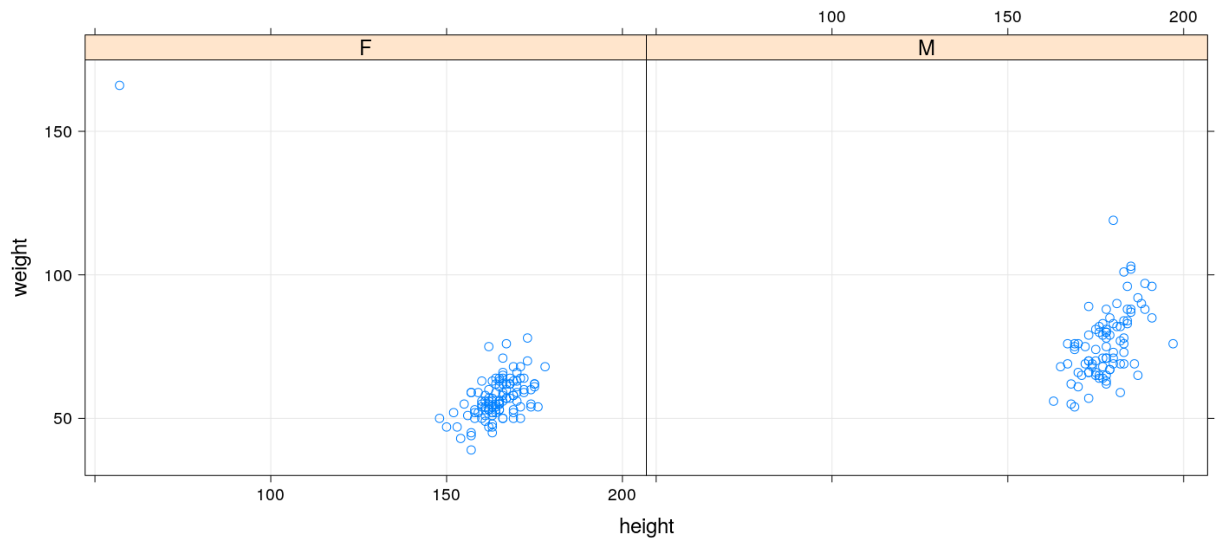
The formula interface in *lattice*

```
xyplot(weight ~ height, Davis, grid = TRUE)
```



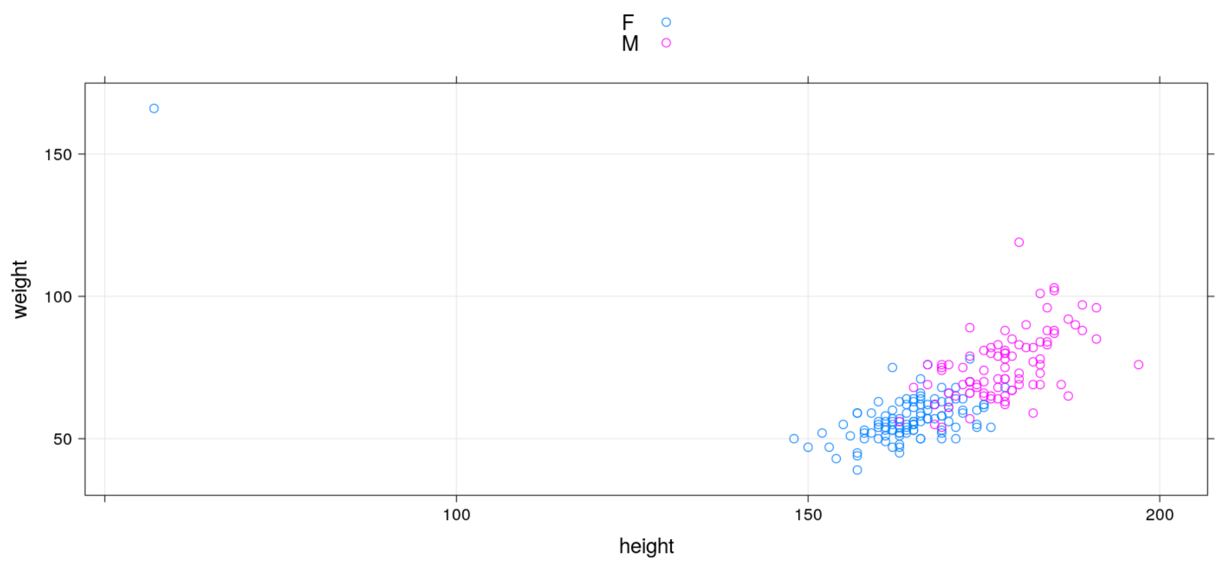
The formula interface in *lattice*: conditioning

```
xyplot(weight ~ height | sex, Davis, grid = TRUE)
```



The formula interface in *lattice*: grouping

```
xyplot(weight ~ height, Davis, grid = TRUE, groups = sex, auto.key = TRUE)
```



The formula interface for linear models

- Basic form: `y ~ model`, where `y` is the name of the response variable
- `model` consists of a series of *terms* separated by `+`
- Each term is usually a predictor by itself (main effect), or an interaction
- Terms can be functions of variables, possibly a matrix (e.g., `log(x)`, `poly(x, 2)`)
- Interactions are defined using a `:` (e.g., `a:b`)

- The intercept term is represented by `1`
- In R, each term is a symbolic representation, not the actual columns of X
- Each term gets expanded into one or more columns in X (using `model.matrix()`)
- Some shortcuts:
  - `y ~ a * b` is equivalent to `y ~ a + b + a:b`
  - `y ~ a * b * c` is equivalent to `y ~ a + b + c + a:b + b:c + c:a + a:b:c`
  - `y ~ (a + b + c)^2` is equivalent to `y ~ a + b + c + a:b + b:c + c:a`
- `y ~ a * b * c - a:b:c` is also equivalent to `y ~ a + b + c + a:b + b:c + c:a`
- `y ~ x` is equivalent to `y ~ 1 + x` (intercept term is implied)
- `y ~ x - 1` and `y ~ 0 + x` explicitly removes the intercept term
- See `help(formula)` for more details

Obtaining the model matrix

- R converts each model specification into a model matrix X
- Numeric predictors are retained as they are
- Categorical predictors are converted using a full rank re-parameterization (which can be customized)
- By default, the first column of the dummy variable matrix is omitted
- Remaining coefficients represent "effect compared to first (omitted) level"

Example: Simple Linear Regression

```r
model.matrix( ~ 1 + height, data = Davis)
```

```
   (Intercept) height
1            1    182
2            1    161
3            1    161
4            1    177
5            1    157
6            1    170
7            1    167
8            1    186
9            1    178
10           1    171
11           1    175
12           1     57
13           1    161
14           1    168
15           1    163
16           1    166
17           1    187
18           1    168
19           1    197
20           1    175
21           1    180
22           1    170
23           1    175
```

24	1	173
25	1	171
26	1	166
27	1	169
28	1	166
29	1	157
30	1	183
31	1	166
32	1	178
33	1	173
34	1	164
35	1	169
36	1	176
37	1	166
38	1	174
39	1	178
40	1	187
41	1	164
42	1	178
43	1	163
44	1	183
45	1	179
46	1	160
47	1	180
48	1	161
49	1	174
50	1	162
51	1	182
52	1	165
53	1	169
54	1	185
55	1	177
56	1	176
57	1	170
58	1	183
59	1	172
60	1	173
61	1	165
62	1	177
63	1	180
64	1	173
65	1	189
66	1	162
67	1	165
68	1	164
69	1	158
70	1	178
71	1	175
72	1	173
73	1	165
74	1	163
75	1	166
76	1	171
77	1	160

78	1	160
79	1	182
80	1	183
81	1	165
82	1	168
83	1	169
84	1	167
85	1	170
86	1	182
87	1	178
88	1	165
89	1	163
90	1	162
91	1	173
92	1	161
93	1	184
94	1	180
95	1	189
96	1	165
97	1	185
98	1	169
99	1	159
100	1	155
101	1	164
102	1	178
103	1	163
104	1	163
105	1	175
106	1	164
107	1	152
108	1	167
109	1	166
110	1	166
111	1	183
112	1	179
113	1	174
114	1	179
115	1	167
116	1	168
117	1	184
118	1	184
119	1	169
120	1	178
121	1	178
122	1	167
123	1	178
124	1	165
125	1	179
126	1	169
127	1	153
128	1	157
129	1	171
130	1	157
131	1	166

132	1	185
133	1	160
134	1	148
135	1	177
136	1	162
137	1	172
138	1	167
139	1	188
140	1	191
141	1	175
142	1	163
143	1	165
144	1	176
145	1	171
146	1	160
147	1	165
148	1	157
149	1	173
150	1	184
151	1	168
152	1	162
153	1	150
154	1	162
155	1	163
156	1	169
157	1	172
158	1	170
159	1	169
160	1	167
161	1	163
162	1	161
163	1	162
164	1	172
165	1	163
166	1	159
167	1	170
168	1	166
169	1	191
170	1	158
171	1	169
172	1	163
173	1	170
174	1	176
175	1	168
176	1	178
177	1	174
178	1	170
179	1	178
180	1	174
181	1	176
182	1	154
183	1	181
184	1	165
185	1	173

```
186           1    162
187           1    172
188           1    169
189           1    183
190           1    158
191           1    185
192           1    173
193           1    164
194           1    156
195           1    164
196           1    175
197           1    180
198           1    175
199           1    181
200           1    177
attr(,"assign")
[1] 0 1
```

Example: Additive effect of group

```r
model.matrix( ~ 1 + height + sex, data = Davis)
```

```
    (Intercept) height sexM
1             1    182    1
2             1    161    0
3             1    161    0
4             1    177    1
5             1    157    0
6             1    170    1
7             1    167    1
8             1    186    1
9             1    178    1
10            1    171    1
11            1    175    1
12            1     57    0
13            1    161    0
14            1    168    0
15            1    163    0
16            1    166    0
17            1    187    1
18            1    168    0
19            1    197    1
20            1    175    0
21            1    180    1
22            1    170    0
23            1    175    1
24            1    173    1
25            1    171    0
26            1    166    0
27            1    169    0
28            1    166    0
29            1    157    0
30            1    183    1
31            1    166    0
32            1    178    1
```

33	1	173	1
34	1	164	0
35	1	169	0
36	1	176	1
37	1	166	0
38	1	174	1
39	1	178	1
40	1	187	1
41	1	164	0
42	1	178	1
43	1	163	0
44	1	183	1
45	1	179	1
46	1	160	0
47	1	180	1
48	1	161	0
49	1	174	0
50	1	162	0
51	1	182	1
52	1	165	0
53	1	169	1
54	1	185	1
55	1	177	1
56	1	176	1
57	1	170	0
58	1	183	1
59	1	172	1
60	1	173	1
61	1	165	1
62	1	177	1
63	1	180	1
64	1	173	0
65	1	189	1
66	1	162	0
67	1	165	0
68	1	164	0
69	1	158	0
70	1	178	1
71	1	175	0
72	1	173	1
73	1	165	0
74	1	163	0
75	1	166	0
76	1	171	0
77	1	160	0
78	1	160	0
79	1	182	1
80	1	183	1
81	1	165	0
82	1	168	1
83	1	169	0
84	1	167	0
85	1	170	0
86	1	182	1

87	1	178	1
88	1	165	0
89	1	163	0
90	1	162	0
91	1	173	1
92	1	161	0
93	1	184	1
94	1	180	1
95	1	189	1
96	1	165	0
97	1	185	1
98	1	169	0
99	1	159	0
100	1	155	0
101	1	164	0
102	1	178	1
103	1	163	0
104	1	163	0
105	1	175	0
106	1	164	0
107	1	152	0
108	1	167	0
109	1	166	0
110	1	166	0
111	1	183	1
112	1	179	1
113	1	174	0
114	1	179	1
115	1	167	0
116	1	168	0
117	1	184	1
118	1	184	1
119	1	169	1
120	1	178	1
121	1	178	1
122	1	167	1
123	1	178	0
124	1	165	0
125	1	179	1
126	1	169	0
127	1	153	0
128	1	157	0
129	1	171	0
130	1	157	0
131	1	166	0
132	1	185	1
133	1	160	0
134	1	148	0
135	1	177	1
136	1	162	0
137	1	172	0
138	1	167	0
139	1	188	1
140	1	191	1

141	1	175	1
142	1	163	0
143	1	165	0
144	1	176	0
145	1	171	0
146	1	160	0
147	1	165	0
148	1	157	0
149	1	173	0
150	1	184	1
151	1	168	0
152	1	162	0
153	1	150	0
154	1	162	0
155	1	163	0
156	1	169	1
157	1	172	1
158	1	170	0
159	1	169	0
160	1	167	0
161	1	163	0
162	1	161	0
163	1	162	0
164	1	172	0
165	1	163	1
166	1	159	0
167	1	170	0
168	1	166	0
169	1	191	1
170	1	158	0
171	1	169	1
172	1	163	0
173	1	170	1
174	1	176	1
175	1	168	1
176	1	178	1
177	1	174	0
178	1	170	1
179	1	178	1
180	1	174	1
181	1	176	1
182	1	154	0
183	1	181	1
184	1	165	0
185	1	173	1
186	1	162	0
187	1	172	0
188	1	169	0
189	1	183	1
190	1	158	0
191	1	185	1
192	1	173	1
193	1	164	0
194	1	156	0

```
195            1    164    0
196            1    175    1
197            1    180    1
198            1    175    1
199            1    181    1
200            1    177    1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$sex
[1] "contr.treatment"
```

Example: Interaction

```
model.matrix( ~ 1 + height * sex, data = Davis)
```

```
    (Intercept) height sexM height:sexM
1             1    182    1         182
2             1    161    0           0
3             1    161    0           0
4             1    177    1         177
5             1    157    0           0
6             1    170    1         170
7             1    167    1         167
8             1    186    1         186
9             1    178    1         178
10            1    171    1         171
11            1    175    1         175
12            1     57    0           0
13            1    161    0           0
14            1    168    0           0
15            1    163    0           0
16            1    166    0           0
17            1    187    1         187
18            1    168    0           0
19            1    197    1         197
20            1    175    0           0
21            1    180    1         180
22            1    170    0           0
23            1    175    1         175
24            1    173    1         173
25            1    171    0           0
26            1    166    0           0
27            1    169    0           0
28            1    166    0           0
29            1    157    0           0
30            1    183    1         183
31            1    166    0           0
32            1    178    1         178
33            1    173    1         173
34            1    164    0           0
35            1    169    0           0
36            1    176    1         176
37            1    166    0           0
38            1    174    1         174
```

39	1	178	1	178
40	1	187	1	187
41	1	164	0	0
42	1	178	1	178
43	1	163	0	0
44	1	183	1	183
45	1	179	1	179
46	1	160	0	0
47	1	180	1	180
48	1	161	0	0
49	1	174	0	0
50	1	162	0	0
51	1	182	1	182
52	1	165	0	0
53	1	169	1	169
54	1	185	1	185
55	1	177	1	177
56	1	176	1	176
57	1	170	0	0
58	1	183	1	183
59	1	172	1	172
60	1	173	1	173
61	1	165	1	165
62	1	177	1	177
63	1	180	1	180
64	1	173	0	0
65	1	189	1	189
66	1	162	0	0
67	1	165	0	0
68	1	164	0	0
69	1	158	0	0
70	1	178	1	178
71	1	175	0	0
72	1	173	1	173
73	1	165	0	0
74	1	163	0	0
75	1	166	0	0
76	1	171	0	0
77	1	160	0	0
78	1	160	0	0
79	1	182	1	182
80	1	183	1	183
81	1	165	0	0
82	1	168	1	168
83	1	169	0	0
84	1	167	0	0
85	1	170	0	0
86	1	182	1	182
87	1	178	1	178
88	1	165	0	0
89	1	163	0	0
90	1	162	0	0
91	1	173	1	173
92	1	161	0	0

93	1	184	1	184
94	1	180	1	180
95	1	189	1	189
96	1	165	0	0
97	1	185	1	185
98	1	169	0	0
99	1	159	0	0
100	1	155	0	0
101	1	164	0	0
102	1	178	1	178
103	1	163	0	0
104	1	163	0	0
105	1	175	0	0
106	1	164	0	0
107	1	152	0	0
108	1	167	0	0
109	1	166	0	0
110	1	166	0	0
111	1	183	1	183
112	1	179	1	179
113	1	174	0	0
114	1	179	1	179
115	1	167	0	0
116	1	168	0	0
117	1	184	1	184
118	1	184	1	184
119	1	169	1	169
120	1	178	1	178
121	1	178	1	178
122	1	167	1	167
123	1	178	0	0
124	1	165	0	0
125	1	179	1	179
126	1	169	0	0
127	1	153	0	0
128	1	157	0	0
129	1	171	0	0
130	1	157	0	0
131	1	166	0	0
132	1	185	1	185
133	1	160	0	0
134	1	148	0	0
135	1	177	1	177
136	1	162	0	0
137	1	172	0	0
138	1	167	0	0
139	1	188	1	188
140	1	191	1	191
141	1	175	1	175
142	1	163	0	0
143	1	165	0	0
144	1	176	0	0
145	1	171	0	0
146	1	160	0	0

147	1	165	0	0
148	1	157	0	0
149	1	173	0	0
150	1	184	1	184
151	1	168	0	0
152	1	162	0	0
153	1	150	0	0
154	1	162	0	0
155	1	163	0	0
156	1	169	1	169
157	1	172	1	172
158	1	170	0	0
159	1	169	0	0
160	1	167	0	0
161	1	163	0	0
162	1	161	0	0
163	1	162	0	0
164	1	172	0	0
165	1	163	1	163
166	1	159	0	0
167	1	170	0	0
168	1	166	0	0
169	1	191	1	191
170	1	158	0	0
171	1	169	1	169
172	1	163	0	0
173	1	170	1	170
174	1	176	1	176
175	1	168	1	168
176	1	178	1	178
177	1	174	0	0
178	1	170	1	170
179	1	178	1	178
180	1	174	1	174
181	1	176	1	176
182	1	154	0	0
183	1	181	1	181
184	1	165	0	0
185	1	173	1	173
186	1	162	0	0
187	1	172	0	0
188	1	169	0	0
189	1	183	1	183
190	1	158	0	0
191	1	185	1	185
192	1	173	1	173
193	1	164	0	0
194	1	156	0	0
195	1	164	0	0
196	1	175	1	175
197	1	180	1	180
198	1	175	1	175
199	1	181	1	181
200	1	177	1	177

```
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$sex
[1] "contr.treatment"
```

Example: Transformed predictor

```
data(UN, package = "carData")
model.matrix( ~ 1 + log(ppgdp), data = UN)
```

```
                                 (Intercept) log(ppgdp)
Afghanistan                                1   6.212606
Albania                                    1   8.209907
Algeria                                    1   8.405815
Angola                                     1   8.371450
Anguilla                                   1   9.528801
Argentina                                  1   9.122831
Armenia                                    1   8.016549
Aruba                                      1  10.036772
Australia                                  1  10.952890
Austria                                    1  10.717940
Azerbaijan                                 1   8.637214
Bahamas                                    1  10.019562
Bahrain                                    1   9.808303
Bangladesh                                 1   6.507875
Barbados                                   1   9.581718
Belarus                                    1   8.648572
Belgium                                    1  10.687727
Belize                                     1   8.410899
Benin                                      1   6.608136
Bermuda                                    1  11.436311
Bhutan                                     1   7.624228
Bolivia                                    1   7.589791
Bosnia and Herzegovina                     1   8.406865
Botswana                                   1   8.909627
Brazil                                     1   9.279456
Brunei Darussalam                          1  10.393527
Bulgaria                                   1   8.758585
Burkina Faso                               1   6.253252
Burundi                                    1   5.173887
Cambodia                                   1   6.681106
Cameroon                                   1   7.095562
Canada                                     1  10.744212
Cape Verde                                 1   8.084562
Cayman Islands                             1  10.951647
Central African Republic                   1   6.111024
Chad                                       1   6.589477
Chile                                      1   9.383260
China                                      1   8.378850
Colombia                                   1   8.735975
Comoros                                    1   6.602045
Congo                                      1   7.887997
Cook Islands                               1   9.410183
Costa Rica                                 1   8.949469
```

Cote dIvoire	1	7.051076
Croatia	1	9.533836
Cuba	1	8.648993
Cyprus	1	10.252887
Czech Republic	1	9.843674
Democratic Republic of the Congo	1	5.301313
Denmark	1	10.930070
Djibouti	1	7.156645
Dominica	1	8.856632
Dominican Republic	1	8.555529
Timor Leste	1	6.559757
Ecuador	1	8.312037
Egypt	1	7.883710
El Salvador	1	8.139032
Equatorial Guinea	1	9.732248
Eritrea	1	6.061690
Estonia	1	9.556438
Ethiopia	1	5.782594
Fiji	1	8.173491
Finland	1	10.703283
France	1	10.585217
French Polynesia	1	10.113303
Gabon	1	9.430985
Gambia	1	6.361475
Georgia	1	7.893684
Germany	1	10.593056
Ghana	1	7.195337
Greece	1	10.185043
Greenland	1	10.471431
Grenada	1	8.913147
Guatemala	1	7.966344
Guinea	1	6.057954
Guinea-Bissau	1	6.290457
Guyana	1	8.005033
Haiti	1	6.417875
Honduras	1	7.613917
Hong Kong	1	10.367967
Hungary	1	9.463742
Iceland	1	10.578420
India	1	7.248789
Indonesia	1	7.989323
Iran	1	8.561612
Iraq	1	6.789535
Ireland	1	10.741174
Israel	1	10.285739
Italy	1	10.430495
Jamaica	1	8.496786
Japan	1	10.672227
Jordan	1	8.399603
Kazakhstan	1	9.123333
Kenya	1	6.686859
Kiribati	1	7.291792
Kuwait	1	10.723937
Kyrgyzstan	1	6.763192

Laos	1	6.954257
Latvia	1	9.274535
Lebanon	1	9.136015
Lesotho	1	6.888267
Liberia	1	5.387244
Libya	1	9.334397
Lithuania	1	9.303421
Luxembourg	1	11.562624
Macao	1	10.819582
Madagascar	1	6.044768
Malawi	1	5.878856
Malaysia	1	9.032744
Maldives	1	8.452014
Mali	1	6.394928
Malta	1	9.883244
Marshall Islands	1	8.029237
Mauritania	1	7.030946
Mauritius	1	8.921097
Mexico	1	9.116107
Micronesia	1	7.892900
Moldova	1	7.393755
Mongolia	1	7.717218
Montenegro	1	8.781064
Morocco	1	7.960324
Mozambique	1	6.010041
Myanmar	1	6.775594
Namibia	1	8.541827
Nauru	1	8.730707
Nepal	1	6.281706
Neth Antilles	1	9.919415
Netherlands	1	10.755980
New Caledonia	1	10.472190
New Zealand	1	10.385052
Nicaragua	1	7.031653
Niger	1	5.879695
Nigeria	1	7.122705
North Korea	1	6.222576
Norway	1	11.345556
Oman	1	9.942275
Pakistan	1	6.910950
Palau	1	9.289318
Palestinian Territory	1	7.506317
Panama	1	8.937744
Papua New Guinea	1	7.264310
Paraguay	1	7.927000
Peru	1	8.596134
Philippines	1	7.668608
Poland	1	9.414358
Portugal	1	9.972902
Puerto Rico	1	10.183427
Qatar	1	11.189933
Republic of Korea	1	9.954760
Romania	1	8.925641
Russian Federation	1	9.244877

```
Rwanda                              1   6.277207
Saint Lucia                         1   8.806439
Samoa                               1   8.114714
Sao Tome and Principe               1   7.157190
Saudi Arabia                        1   9.670035
Senegal                             1   6.939932
Serbia                              1   8.541535
Seychelles                          1   9.345797
Sierra Leone                        1   5.862779
Singapore                           1  10.687003
Slovakia                            1   9.678843
Slovenia                            1  10.048012
Solomon Islands                     1   7.084645
Somalia                             1   4.743191
South Africa                        1   8.889419
Spain                               1  10.326884
Sri Lanka                           1   7.772879
St Vincent and Grenadines           1   8.727730
Sudan                               1   7.509280
Suriname                            1   8.856234
Swaziland                           1   8.105066
Sweden                              1  10.797659
Switzerland                         1  11.140124
Syria                               1   7.983270
Tajikistan                          1   6.704414
Tanzania                            1   6.246107
TFYR Macedonia                      1   8.397170
Thailand                            1   8.436590
Togo                                1   6.262636
Tonga                               1   8.172757
Trinidad and Tobago                 1   9.629386
Tunisia                             1   8.348088
Turkey                              1   9.219805
Turkmenistan                        1   8.431090
Tuvalu                              1   8.066898
Uganda                              1   6.232448
Ukraine                             1   8.017967
United Arab Emirates                1  10.587208
United Kingdom                      1  10.500311
United States                       1  10.748194
Uruguay                             1   9.388687
Uzbekistan                          1   7.263540
Vanuatu                             1   7.994126
Venezuela                           1   9.510645
Viet Nam                            1   7.075555
Yemen                               1   7.270452
Zambia                              1   7.121091
Zimbabwe                            1   6.351060
attr(,"assign")
[1] 0 1
```

Example: Quadratic model

```
data(Prestige, package = "carData")
Prestige$income.sq <- Prestige$income^2
model.matrix( ~ 1 + income + income.sq, data = Prestige)
```

```
                          (Intercept) income income.sq
gov.administrators                  1  12351 152547201
general.managers                    1  25879 669722641
accountants                         1   9271  85951441
purchasing.officers                 1   8865  78588225
chemists                            1   8403  70610409
physicists                          1  11030 121660900
biologists                          1   8258  68194564
architects                          1  14163 200590569
civil.engineers                     1  11377 129436129
mining.engineers                    1  11023 121506529
surveyors                           1   5902  34833604
draughtsmen                         1   7059  49829481
computer.programers                 1   8425  70980625
economists                          1   8049  64786401
psychologists                       1   7405  54834025
social.workers                      1   6336  40144896
lawyers                             1  19263 371063169
librarians                          1   6112  37356544
vocational.counsellors              1   9593  92025649
ministers                           1   4686  21958596
university.teachers                 1  12480 155750400
primary.school.teachers             1   5648  31899904
secondary.school.teachers           1   8034  64545156
physicians                          1  25308 640494864
veterinarians                       1  14558 211935364
osteopaths.chiropractors            1  17498 306180004
nurses                              1   4614  21288996
nursing.aides                       1   3485  12145225
physio.therapsts                    1   5092  25928464
pharmacists                         1  10432 108826624
medical.technicians                 1   5180  26832400
commercial.artists                  1   6197  38402809
radio.tv.announcers                 1   7562  57183844
athletes                            1   8206  67338436
secretaries                         1   4036  16289296
typists                             1   3148   9909904
bookkeepers                         1   4348  18905104
tellers.cashiers                    1   2448   5992704
computer.operators                  1   4330  18748900
shipping.clerks                     1   4761  22667121
file.clerks                         1   3016   9096256
receptionsts                        1   2901   8415801
mail.carriers                       1   5511  30371121
postal.clerks                       1   3739  13980121
telephone.operators                 1   3161   9991921
collectors                          1   4741  22477081
claim.adjustors                     1   5052  25522704
travel.clerks                       1   6259  39175081
```

office.clerks	1	4075	16605625
sales.supervisors	1	7482	55980324
commercial.travellers	1	8780	77088400
sales.clerks	1	2594	6728836
newsboys	1	918	842724
service.station.attendant	1	2370	5616900
insurance.agents	1	8131	66113161
real.estate.salesmen	1	6992	48888064
buyers	1	7956	63297936
firefighters	1	8895	79121025
policemen	1	8891	79049881
cooks	1	3116	9709456
bartenders	1	3930	15444900
funeral.directors	1	7869	61921161
babysitters	1	611	373321
launderers	1	3000	9000000
janitors	1	3472	12054784
elevator.operators	1	3582	12830724
farmers	1	3643	13271449
farm.workers	1	1656	2742336
rotary.well.drillers	1	6860	47059600
bakers	1	4199	17631601
slaughterers.1	1	5134	26357956
slaughterers.2	1	5134	26357956
canners	1	1890	3572100
textile.weavers	1	4443	19740249
textile.labourers	1	3485	12145225
tool.die.makers	1	8043	64689849
machinists	1	6686	44702596
sheet.metal.workers	1	6565	43099225
welders	1	6477	41951529
auto.workers	1	5811	33767721
aircraft.workers	1	6573	43204329
electronic.workers	1	3942	15539364
radio.tv.repairmen	1	5449	29691601
sewing.mach.operators	1	2847	8105409
auto.repairmen	1	5795	33582025
aircraft.repairmen	1	7716	59536656
railway.sectionmen	1	4696	22052416
electrical.linemen	1	8316	69155856
electricians	1	7147	51079609
construction.foremen	1	8880	78854400
carpenters	1	5299	28079401
masons	1	5959	35509681
house.painters	1	4549	20693401
plumbers	1	6928	47997184
construction.labourers	1	3910	15288100
pilots	1	14032	196897024
train.engineers	1	8845	78234025
bus.drivers	1	5562	30935844
taxi.drivers	1	4224	17842176
longshoremen	1	4753	22591009
typesetters	1	6462	41757444
bookbinders	1	3617	13082689

```
attr(,"assign")
[1] 0 1 2
```

```r
## model.matrix( ~ 1 + income + I(income^2), data = Prestige) # equivalent
```

Example: Better way to fit polynomial terms

```r
data(Prestige, package = "carData")
model.matrix( ~ 1 + poly(income, 2), data = Prestige)
```

```
                           (Intercept) poly(income, 2)1 poly(income, 2)2
gov.administrators                   1      0.130137548    -0.1072432276
general.managers                     1      0.447167922     0.4892810362
accountants                          1      0.057957362    -0.0975820423
purchasing.officers                  1      0.048442701    -0.0922834385
chemists                             1      0.037615674    -0.0851135246
physicists                           1      0.099179747    -0.1097078893
biologists                           1      0.034217580    -0.0826129170
architects                           1      0.172601995    -0.0877174041
civil.engineers                      1      0.107311736    -0.1100216640
mining.engineers                     1      0.099015702    -0.1096945122
surveyors                            1     -0.020995575    -0.0252248050
draughtsmen                          1      0.006118865    -0.0573525828
computer.programers                  1      0.038131246    -0.0854824785
economists                           1      0.029319639    -0.0787981805
psychologists                        1      0.014227419    -0.0654814106
social.workers                       1     -0.010824730    -0.0381685721
lawyers                              1      0.292121133     0.0674919322
librarians                           1     -0.016074198    -0.0316216974
vocational.counsellors               1      0.065503473    -0.1011177165
ministers                            1     -0.049492687     0.0167476978
university.teachers                  1      0.133160679    -0.1064705850
primary.school.teachers              1     -0.026948096    -0.0171524351
secondary.school.teachers            1      0.028968112    -0.0785148400
physicians                           1      0.433786465     0.4430616994
veterinarians                        1      0.181858869    -0.0809816935
osteopaths.chiropractors             1      0.250758137    -0.0029629262
nurses                               1     -0.051180016     0.0194966481
nursing.aides                        1     -0.077638272     0.0664579058
physio.therapsts                     1     -0.039978026     0.0017985919
pharmacists                          1      0.085165543    -0.1075600048
medical.technicians                  1     -0.037915735    -0.0013179764
commercial.artists                   1     -0.014082213    -0.0341396038
radio.tv.announcers                  1      0.017906733    -0.0689453356
athletes                             1      0.032998954    -0.0816870139
secretaries                          1     -0.064725518     0.0426330809
typists                              1     -0.085535909     0.0818806002
bookkeepers                          1     -0.057413759     0.0299081857
tellers.cashiers                     1     -0.101940497     0.1159802916
computer.operators                   1     -0.057835591     0.0306272641
shipping.clerks                      1     -0.047735052     0.0139155625
file.clerks                          1     -0.088629346     0.0880975999
receptionsts                         1     -0.091324385     0.0935947077
mail.carriers                        1     -0.030158709    -0.0126460986
postal.clerks                        1     -0.071685750     0.0552605954
```

telephone.operators	1	-0.085231253	0.0812736807
collectors	1	-0.048203755	0.0146676702
claim.adjustors	1	-0.040915431	0.0032297748
travel.clerks	1	-0.012629235	-0.0359502746
office.clerks	1	-0.063811548	0.0410121895
sales.supervisors	1	0.016031923	-0.0671977969
commercial.travellers	1	0.046450716	-0.0910554328
sales.clerks	1	-0.098518969	0.1086380404
newsboys	1	-0.137796239	0.2002157373
service.station.attendant	1	-0.103768437	0.1199525535
insurance.agents	1	0.031241319	-0.0803244883
real.estate.salesmen	1	0.004548712	-0.0556998086
buyers	1	0.027140173	-0.0770208390
firefighters	1	0.049145755	-0.0927070407
policemen	1	0.049052015	-0.0926508562
cooks	1	-0.086285833	0.0833786510
bartenders	1	-0.067209642	0.0470822920
funeral.directors	1	0.025101317	-0.0753136284
babysitters	1	-0.144990822	0.2187217519
launderers	1	-0.089004308	0.0888579101
janitors	1	-0.077942929	0.0670408678
elevator.operators	1	-0.075365065	0.0621384571
farmers	1	-0.073935522	0.0594495128
farm.workers	1	-0.120501116	0.1579222466
rotary.well.drillers	1	0.001455275	-0.0523688918
bakers	1	-0.060905593	0.0359160679
slaughterers.1	1	-0.038993751	0.0003056443
slaughterers.2	1	-0.038993751	0.0003056443
canners	1	-0.115017297	0.1451589655
textile.weavers	1	-0.055187422	0.0261435795
textile.labourers	1	-0.077638272	0.0664579058
tool.die.makers	1	0.029179028	-0.0786849979
machinists	1	-0.002622436	-0.0478267149
sheet.metal.workers	1	-0.005458087	-0.0445665599
welders	1	-0.007520378	-0.0421432322
auto.workers	1	-0.023128171	-0.0223749198
aircraft.workers	1	-0.005270606	-0.0447846782
electronic.workers	1	-0.066928420	0.0465753998
radio.tv.repairmen	1	-0.031611686	-0.0105716461
sewing.mach.operators	1	-0.092589882	0.0962019123
auto.repairmen	1	-0.023503133	-0.0218689722
aircraft.repairmen	1	0.021515743	-0.0722068629
railway.sectionmen	1	-0.049258336	0.0163682312
electrical.linemen	1	0.035576818	-0.0836275106
electricians	1	0.008181156	-0.0594845993
construction.foremen	1	0.048794228	-0.0924958795
carpenters	1	-0.035126955	-0.0054623717
masons	1	-0.019659772	-0.0269859064
house.painters	1	-0.052703299	0.0220036647
plumbers	1	0.003048864	-0.0540971955
construction.labourers	1	-0.067678344	0.0479289326
pilots	1	0.169531993	-0.0897553077
train.engineers	1	0.047973999	-0.0919981932
bus.drivers	1	-0.028963517	-0.0143361135

```
taxi.drivers                        1     -0.060319715     0.0348992178
longshoremen                        1     -0.047922533     0.0142161326
typesetters                         1     -0.007871905    -0.0417257711
bookbinders                         1     -0.074544836     0.0605930322
attr(,"assign")
[1] 0 1 1
```

The poly() function

```
x <- 1:10
poly(x, 2, simple = TRUE)
```

```
                1           2
 [1,] -0.49543369  0.52223297
 [2,] -0.38533732  0.17407766
 [3,] -0.27524094 -0.08703883
 [4,] -0.16514456 -0.26111648
 [5,] -0.05504819 -0.34815531
 [6,]  0.05504819 -0.34815531
 [7,]  0.16514456 -0.26111648
 [8,]  0.27524094 -0.08703883
 [9,]  0.38533732  0.17407766
[10,]  0.49543369  0.52223297
```

```
crossprod(cbind(1, poly(x, 2))) # columns are orthogonal
```

```
                            1             2
  1.000000e+01  2.220446e-16  1.110223e-16
1 2.220446e-16  1.000000e+00 -1.773693e-17
2 1.110223e-16 -1.773693e-17  1.000000e+00
```

Fitting a linear model

```
fm <- lm(weight ~ 1 + height * sex, data = Davis)
```

- Linear models are fit using `lm()`
- The result is usually stored in a variable for further processing

The return value of lm()

- `lm()` returns a list
- The meaning of most components are obvious, but see `help(lm)` for details
- The `$qr` component represents the QR decomposition used to solve the normal equations

```
str(fm)
```

```
List of 13
 $ coefficients : Named num [1:4] 160.497 -0.627 -261.828 1.622
  ..- attr(*, "names")= chr [1:4] "(Intercept)" "height" "sexM" "height:sexM"
 $ residuals    : Named num [1:200] -2.87 -1.58 -6.58 -6.89 -3.09 ...
  ..- attr(*, "names")= chr [1:200] "1" "2" "3" "4" ...
 $ effects      : Named num [1:200] -930.6 40.4 127.4 87.5 -2.8 ...
  ..- attr(*, "names")= chr [1:200] "(Intercept)" "height" "sexM" "height:sexM" ...
 $ rank         : int 4
 $ fitted.values: Named num [1:200] 79.9 59.6 59.6 74.9 62.1 ...
  ..- attr(*, "names")= chr [1:200] "1" "2" "3" "4" ...
 $ assign       : int [1:4] 0 1 2 3
```

```
$ qr           :List of 5
 ..$ qr   : num [1:200, 1:4] -14.1421 0.0707 0.0707 0.0707 0.0707 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr [1:200] "1" "2" "3" "4" ...
 .. .. ..$ : chr [1:4] "(Intercept)" "height" "sexM" "height:sexM"
 .. ..- attr(*, "assign")= int [1:4] 0 1 2 3
 .. ..- attr(*, "contrasts")=List of 1
 .. .. ..$ sex: chr "contr.treatment"
 ..$ qraux: num [1:4] 1.07 1.06 1.04 1.02
 ..$ pivot: int [1:4] 1 2 3 4
 ..$ tol  : num 1e-07
 ..$ rank : int 4
 ..- attr(*, "class")= chr "qr"
$ df.residual  : int 196
$ contrasts    :List of 1
 ..$ sex: chr "contr.treatment"
$ xlevels      :List of 1
 ..$ sex: chr [1:2] "F" "M"
$ call         : language lm(formula = weight ~ 1 + height * sex, data = Davis)
$ terms        :Classes 'terms', 'formula'  language weight ~ 1 + height * sex
 .. ..- attr(*, "variables")= language list(weight, height, sex)
 .. ..- attr(*, "factors")= int [1:3, 1:3] 0 1 0 0 0 1 0 1 1
 .. .. ..- attr(*, "dimnames")=List of 2
 .. .. .. ..$ : chr [1:3] "weight" "height" "sex"
 .. .. .. ..$ : chr [1:3] "height" "sex" "height:sex"
 .. ..- attr(*, "term.labels")= chr [1:3] "height" "sex" "height:sex"
 .. ..- attr(*, "order")= int [1:3] 1 1 2
 .. ..- attr(*, "intercept")= int 1
 .. ..- attr(*, "response")= int 1
 .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
 .. ..- attr(*, "predvars")= language list(weight, height, sex)
 .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "factor"
 .. .. ..- attr(*, "names")= chr [1:3] "weight" "height" "sex"
$ model        :'data.frame':  200 obs. of  3 variables:
 ..$ weight: int [1:200] 77 58 53 68 59 76 76 69 71 65 ...
 ..$ height: int [1:200] 182 161 161 177 157 170 167 186 178 171 ...
 ..$ sex   : Factor w/ 2 levels "F","M": 2 1 1 2 1 2 2 2 2 2 ...
 ..- attr(*, "terms")=Classes 'terms', 'formula'  language weight ~ 1 + height * sex
 .. .. ..- attr(*, "variables")= language list(weight, height, sex)
 .. .. ..- attr(*, "factors")= int [1:3, 1:3] 0 1 0 0 0 1 0 1 1
 .. .. .. ..- attr(*, "dimnames")=List of 2
 .. .. .. .. ..$ : chr [1:3] "weight" "height" "sex"
 .. .. .. .. ..$ : chr [1:3] "height" "sex" "height:sex"
 .. .. ..- attr(*, "term.labels")= chr [1:3] "height" "sex" "height:sex"
 .. .. ..- attr(*, "order")= int [1:3] 1 1 2
 .. .. ..- attr(*, "intercept")= int 1
 .. .. ..- attr(*, "response")= int 1
 .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
 .. .. ..- attr(*, "predvars")= language list(weight, height, sex)
 .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "factor"
 .. .. .. ..- attr(*, "names")= chr [1:3] "weight" "height" "sex"
- attr(*, "class")= chr "lm"
```

Using the fitted model: summary()

```
summary(fm)
```

```

Call:
lm(formula = weight ~ 1 + height * sex, data = Davis)

Residuals:
    Min      1Q  Median      3Q     Max 
-23.091  -6.331  -0.995   6.207  41.230 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  160.49748   13.45954  11.924  < 2e-16 ***
height        -0.62679    0.08199  -7.644 9.17e-13 ***
sexM        -261.82753   32.72161  -8.002 1.05e-13 ***
height:sexM    1.62239    0.18644   8.702 1.33e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.06 on 196 degrees of freedom
Multiple R-squared:  0.5626,	Adjusted R-squared:  0.556 
F-statistic: 84.05 on 3 and 196 DF,  p-value: < 2.2e-16
```

Using the fitted model: anova()

```
anova(fm)
```

```
Analysis of Variance Table

Response: weight
            Df  Sum Sq Mean Sq F value    Pr(>F)    
height       1  1630.9  1630.9  16.119 8.468e-05 ***
sex          1 16219.8 16219.8 160.306 < 2.2e-16 ***
height:sex   1  7662.0  7662.0  75.726 1.329e-15 ***
Residuals  196 19831.3   101.2                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fm.height <- lm(weight ~ 1 + height, data = Davis)
anova(fm.height, fm)
```

```
Analysis of Variance Table

Model 1: weight ~ 1 + height
Model 2: weight ~ 1 + height * sex
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)    
1    198 43713                                  
2    196 19831  2     23882 118.02 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the fitted model: coefficients()

```
coefficients(fm)
```

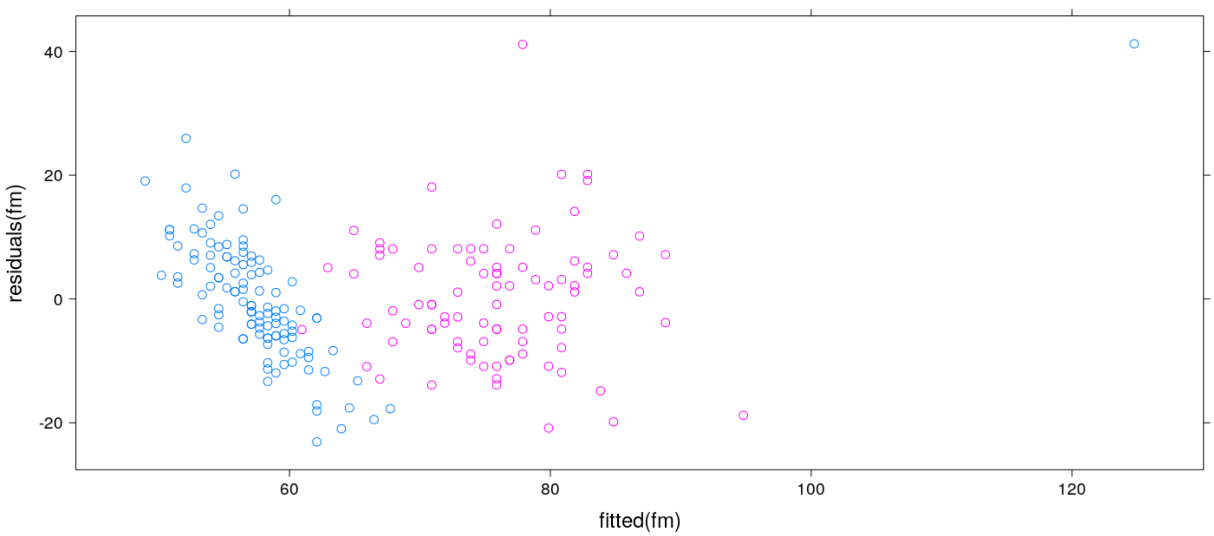
```
 (Intercept)       height         sexM  height:sexM 
 160.4974836   -0.6267909 -261.8275339    1.6223890 
```

```
coef(fm)
```

```
 (Intercept)       height         sexM  height:sexM 
 160.4974836   -0.6267909 -261.8275339    1.6223890 
```

Using the fitted model: Residuals and fitted values

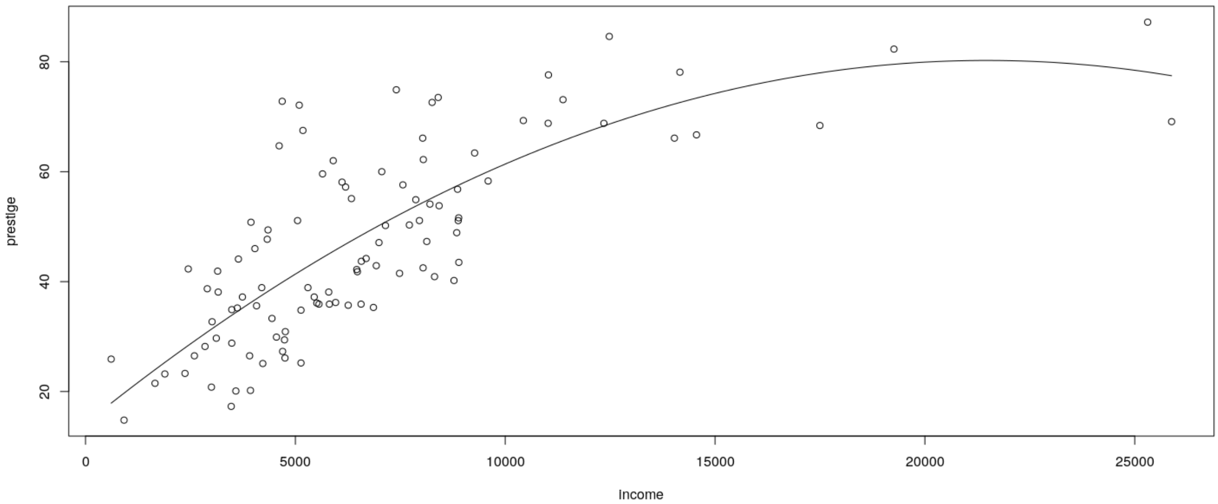
```
xyplot(residuals(fm) ~ fitted(fm), groups = fm$model$sex)
```



Using the fitted model: `predict(fm, newdata = )`

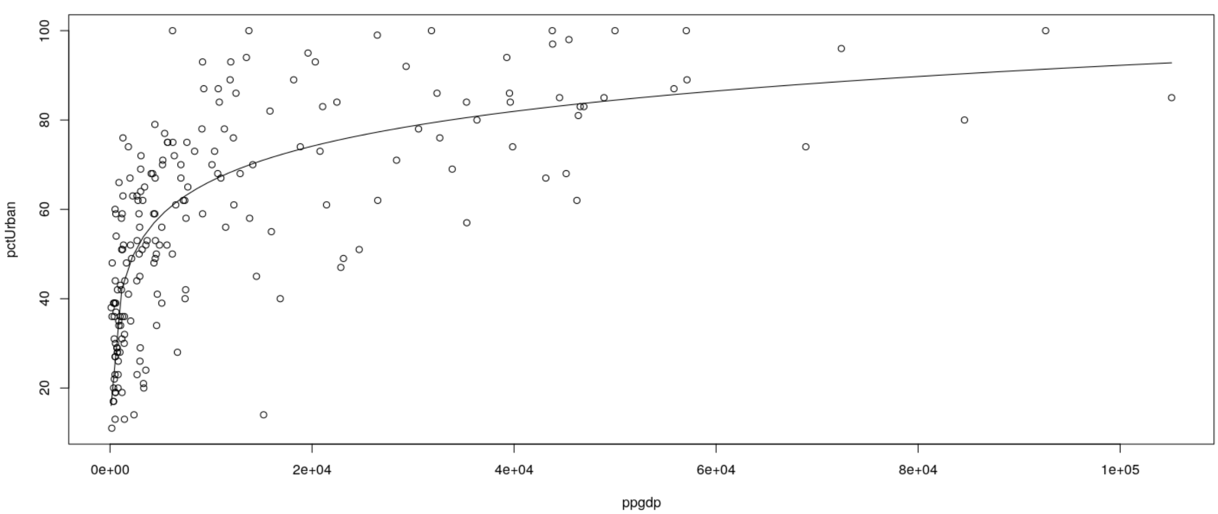
- `newdata` is a data frame containing values of predictors

```
fm.Prestige <- lm(prestige ~ 1 + poly(income, 2), Prestige)
fm.pred.income <- seq(min(Prestige$income), max(Prestige$income), length.out = 100)
fm.pred.prestige <- predict(fm.Prestige, newdata = data.frame(income = fm.pred.income))
plot(prestige ~ income, Prestige)
lines(fm.pred.prestige ~ fm.pred.income)
```



- Note that R takes care of any transformations needed

```
fm.UN <- lm(pctUrban ~ 1 + log(ppgdp), UN)
fm.pred.ppgdp <- seq(min(UN$ppgdp, na.rm = TRUE), max(UN$ppgdp, na.rm = TRUE), length.out = 100)
fm.pred.pctUrban <- predict(fm.UN, newdata = data.frame(ppgdp = fm.pred.ppgdp))
plot(pctUrban ~ ppgdp, UN)
lines(fm.pred.pctUrban ~ fm.pred.ppgdp)
```



Summary: common interface

- Most modeling functions in R support the formula interface
- Most also support some common methods:
  - `coef()` to obtain coefficients

- `fitted()` to get fitted values
- `residuals()` to get residuals
- `predict()` to get predicted values for new predictor values