

cor(R : Copyright 2003, The R Development Core Team
Version 1.6.2 (2003-01-10)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type `license()` or `licence()` for distribution details.

R is a collaborative project with many contributors.
Type `contributors()` for more information.

Type `demo()` for some demos, `help()` for on-line help, or
`help.start()` for a HTML browser interface to help.

Type `q()` to quit R.

```
> # R is a program for statistical graphics and statistical
> # analysis. The web site http://www.r-project.org/ describes
> # R as
> # "a language and environment for statistical computing and
> # graphics. It is a GNU project which is similar to the S
> # language and environment which was developed at Bell
> # Laboratories (formerly AT&T, now Lucent Technologies) by
> # John Chambers and colleagues. R can be considered as a
> # different implementation of S. There are some important
> # differences, but much code written for S runs unaltered
> # under R."
> # There is a commercial version of S, S-plus, which is
> # distributed by Insightful Software.
> #
> # Part 1. Scalars
> #
> # A scalar is a single number, string, or logical value.
>
> tu.pi <- 3.14159
> tu.e <- 2.71828
> tu.name <- "Steve Simon"
> tu.cats.are.better.than.dogs <- T
>
> # You can do all the standard calculations with numbers
>
> tu.a <- 3
> tu.b <- 4
> tu.hyp <- sqrt(tu.a^2+tu.b^2)
>
> # and logical values
>
> x <- T
> y <- F
> x&y
[1] FALSE
>
```



```

[32] 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
[63] 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
[94] 1 0 1 0 1 0 1
>
> # use square brackets to select one or more elements from
> # a vector.
>
> tu.third.prime <- tu.primes[3]
> tu.smallest.five.primes <- tu.primes[1:5]
> tu.third.prime
[1] 5
> tu.smallest.five.primes
[1] 2 3 5 7 11
>
> # A negative index means exclude just this value.
>
> tu.odd.primes <- tu.primes[-1]
> tu.odd.primes
[1] 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79
[22] 83 89 93 97
>
> # You can also use logic values to select elements from
> # a vector.
>
> tu.large.primes <- tu.primes[tu.primes>90]
>
> # Many statistical functions operate on vectors
>
> mean(tu.primes)
[1] 44.34615
> quantile(tu.primes)
 0%  25%  50%  75% 100%
2.0 17.5 42.0 70.0 97.0
> summary(tu.primes)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 2.00  17.50   42.00   44.35   70.00   97.00
>
> #
> # Part 3. Factors
> #
> # A factor is a vector with class "factor". You use this for
> # categorical data and the R system will recognize a factor
> # and treat it differently in many data analyses.
>
> tu.group <- factor(tu.alternate,levels=c(0,1),
+ labels=c("Con","Trt"))
>
> # There are small changes when you use print or summary on a
> # vector with class "factor". These functions are replaced
> # with print.factor and summary.factor.

```

```

>
> print(tu.group)
 [1] Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
[16] Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
[31] Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con
[46] Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
[61] Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
[76] Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
[91] Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt Con Trt
Levels: Con Trt
> summary(tu.group)
Con Trt
 50  50
>
> #
> # Part 4. Lists
> #
> # A list is an ordered collection of objects. Objects in a
> # list can be vectors, scalars, or even other lists. There
> # are additional objects, such as matrices that can also be
> # in a list.
>
> tu.growth.lis <- list(time=c(0,3.6,24,41,99),
+ wt=c(0.64,1.9,9.5,11.4,25.1))
> tu.ci <- list(limits=c(12.1,13.4),level=0.95)
>
> # Use the names() function to what objects are in the list.
>
> names(tu.ci)
[1] "limits" "level"
>
> # Use [[ ]] or $ to select individual objects in a list.
>
> tu.ci[[2]]
[1] 0.95
> tu.ci$level
[1] 0.95
>
> # You can shorten the object name.
>
> tu.ci$li
[1] 12.1 13.4
> tu.ci$le
[1] 0.95
>
> #
> # Part 5. Data frames
> #
> # A data frame is a special list with class "data.frame".
> # Typically, the components of a data frame are vectors

```

```

> # which are all the same length.
>
> tu.growth.dat <-
data.frame(time=c(0,3.6,24,41,99),wt=c(0.64,1.9,9.5,11.4,25.1))
> class(tu.growth.dat)
[1] "data.frame"
>
> # Notice how a data frame prints differently than a list.
>
> tu.growth.lis
$time
[1] 0.0 3.6 24.0 41.0 99.0

$wt
[1] 0.64 1.90 9.50 11.40 25.10

> tu.growth.dat
  time    wt
1  0.0  0.64
2  3.6  1.90
3 24.0  9.50
4 41.0 11.40
5 99.0 25.10
>
> # You can refer to individual vectors in a data frame, just
> # like you would with a list. But sometimes this requires too
> # much typing. You can use the attach() function to tell R
> # that you want to refer to the variables in the data frame
> # directly. Be sure to use the detach() function when you
> # are done.
>
> cor(tu.growth.dat$time,tu.growth.dat$wt)
[1] 0.9923357
> attach(tu.growth.dat)
> cor(time,wt)
[1] 0.9923357
> detach()
>
> #
> # Part 6. Linear regression
> #
> # The lm() function computes a linear regression model.
> # You need to specify a model with the dependent variable
> # on the left side of the tilde (~) and the independent
> # variables on the right side of the tilde.
> #
> # The lm() function creates a list with class "lm".
>
> tu.lin.mod <- lm(wt~time,data=tu.growth.dat)
> class(tu.lin.mod)

```

```

[1] "lm"
> names(tu.lin.mod)
 [1] "coefficients" "residuals"      "effects"
 [4] "rank"         "fitted.values" "assign"
 [7] "qr"          "df.residual"   "xlevels"
[10] "call"        "terms"         "model"
>
> # When you print or plot a list of class "lm" it calls up
> # special functions print.lm and plot.lm.
>
> print(tu.lin.mod)

```

Call:

```
lm(formula = wt ~ time, data = tu.growth.dat)
```

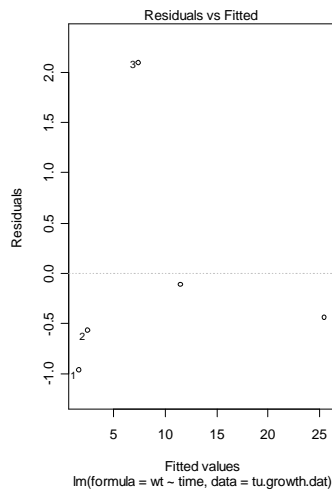
Coefficients:

```

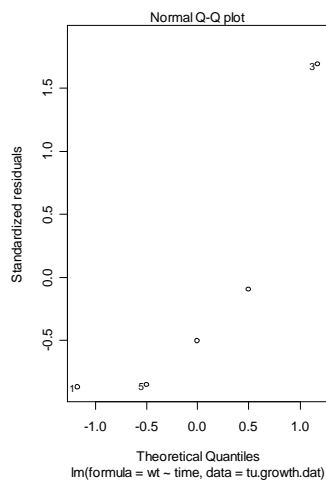
(Intercept)      time
    1.6042      0.2418

```

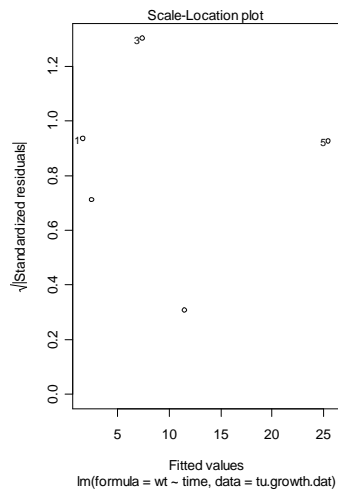
```
> plot(tu.lin.mod)
```



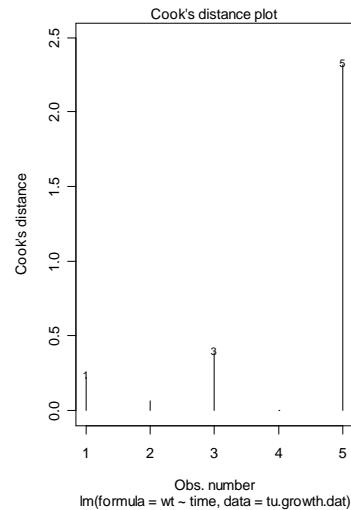
Hit <Return> to see next plot:



Hit <Return> to see next plot:



Hit <Return> to see next plot:



Hit <Return> to see next plot:

```
> #  
> # Part 7. Nonlinear regression  
> #  
> # The nls() function computes a nonlinear regression model.  
> # You need to specify a model with the dependent variable  
> # on the left side of the tilde, and a formula incorporating  
> # one or more parameters and one or more independent  
> # variables on the right side of the tilde. You also need to  
> # specify starting values for your parameters.  
>  
> tu.nls.mod <- nls(wt~a+c*(1-exp(-b*time)),  
+ data=tu.growth.dat,start=list(c=27,b=0.015,a=1))  
>  
> # The nls() function produces a list with class "nls".  
> # This class inherits from the class "lm". You get nicely  
> # formatted output just by printing this list, or you can  
> # get more details by using the summary() function. Both
```

```

> # print() and summary() will actually use the print.nls()
> # and the summary.nls() functions.
>
> class(tu.nls.mod)
[1] "nls"
> names(tu.nls.mod)
[1] "m"      "data" "call"
> print(tu.nls.mod)
Nonlinear regression model
  model: wt ~ a + c * (1 - exp(-b * time))
  data:  tu.growth.dat
           c           b           a
55.718853265  0.005706368  0.920806003
residual sum-of-squares:  3.524924
> summary(tu.nls.mod)

Formula: wt ~ a + c * (1 - exp(-b * time))

Parameters:
      Estimate Std. Error t value Pr(>|t|)
c 55.718853   34.999564   1.592   0.252
b  0.005706   0.004783   1.193   0.355
a  0.920806   1.004257   0.917   0.456

Residual standard error: 1.328 on 2 degrees of freedom

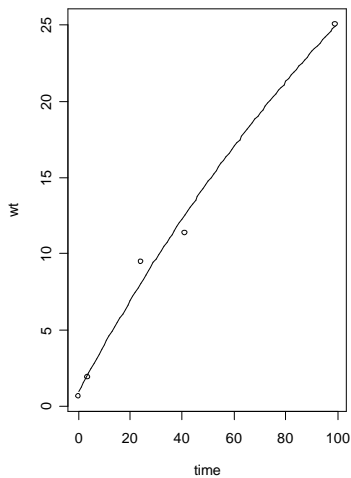
Correlation of Parameter Estimates:
           c           b
b -0.9941
a  0.4752 -0.5431

>
> # Most regression models will allow you to get predicted
> # values using the predict() function. You can get
> # predictions at new values by including a data frame
> # with new values for all of your independent variables.
>
> tu.pred.nls <- predict(tu.nls.mod,newdata=data.frame(time=0:99))
> tu.pred.nls
 [1]  0.920806  1.237853  1.553096  1.866545  2.178210  2.488102
 [7]  2.796231  3.102606  3.407238  3.710137  4.011312  4.310774
[13]  4.608531  4.904595  5.198973  5.491677  5.782715  6.072097
[19]  6.359832  6.645930  6.930400  7.213252  7.494494  7.774136
[25]  8.052186  8.328655  8.603550  8.876881  9.148657  9.418886
[31]  9.687578  9.954741 10.220384 10.484515 10.747143 11.008277
[37] 11.267925 11.526095 11.782797 12.038037 12.291826 12.544170
[43] 12.795079 13.044560 13.292621 13.539271 13.784517 14.028368
[49] 14.270831 14.511914 14.751626 14.989974 15.226966 15.462609
[55] 15.696911 15.929880 16.161523 16.391849 16.620864 16.848575
[61] 17.074991 17.300119 17.523965 17.746538 17.967845 18.187892

```

```
[67] 18.406687 18.624237 18.840549 19.055631 19.269488 19.482129
[73] 19.693560 19.903787 20.112819 20.320661 20.527320 20.732803
[79] 20.937118 21.140269 21.342265 21.543111 21.742815 21.941382
[85] 22.138819 22.335133 22.530330 22.724416 22.917398 23.109281
[91] 23.300073 23.489779 23.678406 23.865959 24.052446 24.237871
[97] 24.422241 24.605562 24.787839 24.969080
```

```
> attach(tu.growth.dat)
> plot(time,wt)
```



```
> lines(0:99,tu.pred.nls)
> detach()
>
> #
> # Part 8. Getting data into R.
> #
> # I have not had time to develop this section. You are probably
> # interested in using ODBC to get data into R. You can download
> # a special library, RODBC, from the Internet for this purpose.
> #
> # Part 9. Further reading.
> #
> # Modern Applied Statistics with S-PLUS. Second Edition. W.N.
> # Venables, B.D. Ripley (1994) New York: Springer-Verlag.
> # [There is a third and fourth edition of this book.]
> #
> # Mixed-Effects Models in S and S-PLUS. J.C. Pinheiro, D.M.
> # Bates (2000) New York: Springer-Verlag.
> #
> # An Introduction to R. W.N. Venables, D.M. Smith, and the R
> # Development Core Team. (2001) Network Theory Limited.
> # Bristol, UK. [This entire document is included with the
> # R help files.]
```