Maximizando la funcion de verosimilitud perfil

Repitamos el ejemplo guiado de los rieles usando un enfoque de maxima verosimilitud.

En este caso:

$\Sigma_\theta = I\sigma_b^2,$

$\Sigma_\phi = I\sigma^2,$

por lo que los parametros $(\theta, \phi) = (\log \sigma, \log\ \sigma_b)$.

La siguiente funcion toma como entrada los siguientes parametros:

- (θ, ϕ)
- X
- Z
- y

y obteniene como salida la funcion de verosimilitud (negativa) con los atributos $(\hat{\beta}, \hat{b})$

In [7]:
```
llm <- function(parameters, X,Z,y) {
    sigma.b <- exp(parameters[1])
    sigma <- exp(parameters[2])
    ## Dimensiones
    n <- length(y); pr <- ncol(Z); pf <- ncol(X)
    ## Matriz de diseño X|Z
    X1 <- cbind(X,Z)
    #Matriz de covarianza para efectos aleatorios
    ipsi <- c(rep(0,pf),rep(1/sigma.b^2,pr))
    #Parametros optimos (beta,b) calculando inversa
    b1 <- solve(crossprod(X1)/sigma^2+diag(ipsi),t(X1)%*%y/sigma^2)
    #Calculando el 5to termino de la ec. final del MLM para el MLM
    ldet <- sum(log(diag(chol(crossprod(Z)/sigma^2 + diag(ipsi[-(1:pf)])))))
    #Calculando ec. final del MLM para el MLM
    l <- (-sum((y-X1%*%b1)^2)/sigma^2 - sum(b1^2*ipsi) - n*log(sigma^2) - pr
    attr(l,"b") <- as.numeric(b1) ## return \hat beta and \hat b
    return(-l)
}
```

In [17]:
```
Rail$Rail
```

1 · 1 · 1 · 2 · 2 · 2 · 3 · 3 · 3 · 4 · 4 · 4 · 5 · 5 · 5 · 6 · 6 · 6

▶ **Levels**:

In [15]:
```
Z <- model.matrix(~ Rail$Rail-1)
Z
```

A matrix: 18 × 6 of type dbl

	Rail$Rail2	Rail$Rail5	Rail$Rail1	Rail$Rail6	Rail$Rail3	Rail$Rail4
1	0	0	1	0	0	0
2	0	0	1	0	0	0
3	0	0	1	0	0	0
4	1	0	0	0	0	0
5	1	0	0	0	0	0
6	1	0	0	0	0	0
7	0	0	0	0	1	0
8	0	0	0	0	1	0
9	0	0	0	0	1	0
10	0	0	0	0	0	1
11	0	0	0	0	0	1
12	0	0	0	0	0	1
13	0	1	0	0	0	0
14	0	1	0	0	0	0
15	0	1	0	0	0	0
16	0	0	0	1	0	0
17	0	0	0	1	0	0
18	0	0	0	1	0	0

In [18]:
```
help(model.matrix)
```

Construct Design Matrices

Description

`model.matrix` creates a design (or model) matrix, e.g., by expanding factors to a set of dummy variables (depending on the contrasts) and expanding interactions similarly.

Usage

```
model.matrix(object, ...)

## Default S3 method:
model.matrix(object, data = environment(object),
             contrasts.arg = NULL, xlev = NULL, ...)
## S3 method for class 'lm'
model.matrix(object, ...)
```

Arguments

`object`	an object of an appropriate class. For the default method, a model formula or a `terms` object.
`data`	a data frame created with `model.frame`. If another sort of object, `model.frame` is called first.
`contrasts.arg`	a list, whose entries are values (numeric matrices, `function`s or character strings naming functions) to be used as replacement values for the `contrasts` replacement function and whose names are the names of columns of `data` containing `factor`s.
`xlev`	to be used as argument of `model.frame` if `data` is such that `model.frame` is called.
`...`	further arguments passed to or from other methods.

Details

`model.matrix` creates a design matrix from the description given in `terms(object)`, using the data in `data` which must supply variables with the same names as would be created by a call to `model.frame(object)` or, more precisely, by evaluating `attr(terms(object), "variables")`. If `data` is a data frame, there may be other columns and the order of columns is not important. Any character variables are coerced to

factors. After coercion, all the variables used on the right-hand side of the formula must be logical, integer, numeric or factor.

If `contrasts.arg` is specified for a factor it overrides the default factor coding for that variable and any `"contrasts"` attribute set by `C` or `contrasts`. Whereas invalid `contrasts.arg`s have been ignored always, they are warned about since **R** version 3.6.0.

In an interaction term, the variable whose levels vary fastest is the first one to appear in the formula (and not in the term), so in `~ a + b + b:a` the interaction will have `a` varying fastest.

By convention, if the response variable also appears on the right-hand side of the formula it is dropped (with a warning), although interactions involving the term are retained.

Value

The design matrix for a regression-like model with the specified formula and data.

There is an attribute `"assign"`, an integer vector with an entry for each column in the matrix giving the term in the formula which gave rise to the column. Value `0` corresponds to the intercept (if any), and positive values to terms in the order given by the `term.labels` attribute of the `terms` structure corresponding to `object`.

If there are any factors in terms in the model, there is an attribute `"contrasts"`, a named list with an entry for each factor. This specifies the contrasts that would be used in terms in which the factor is coded by contrasts (in some terms dummy coding may be used), either as a character vector naming a function or as a numeric matrix.

References

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

`model.frame`, `model.extract`, `terms`

`sparse.model.matrix` from package Matrix for creating *sparse* model matrices, which may be more efficient in large dimensions.

Examples

```
ff <- log(Volume) ~ log(Height) + log(Girth)
utils::str(m <- model.frame(ff, trees))
mat <- model.matrix(ff, m)
```

```
dd <- data.frame(a = gl(3,4), b = gl(4,1,12)) # balanced 2-way
options("contrasts") # typically 'treatment' (for unordered factors)
model.matrix(~ a + b, dd)
model.matrix(~ a + b, dd, contrasts.arg = list(a = "contr.sum"))
model.matrix(~ a + b, dd, contrasts.arg = list(a = "contr.sum", b = co
ntr.poly))
m.orth <- model.matrix(~a+b, dd, contrasts.arg = list(a = "contr.helme
rt"))
crossprod(m.orth) # m.orth is  ALMOST  orthogonal
# invalid contrasts.. ignored with a warning:
stopifnot(identical(
   model.matrix(~ a + b, dd),
   model.matrix(~ a + b, dd, contrasts.arg = "contr.FOO")))
```

[Package *stats* version 4.3.1]

In [10]:
```
X <- matrix(1,18,1)
```

Definir y usar funcion `optim`

In [11]:
```
rail.mod <- optim(c(0,0),llm,X=X,Z=Z,y=Rail$travel)
```

In [12]:
```
exp(rail.mod$par)
```

22.6291657130585 · 4.02407242434335