Biostatistics I: Linear Regression

Linear Regression in R

Nicole S. Erler
Department of Biostatistics, Erasmus Medical Center
n.erler@erasmusmc.nl
@N_Erler

Linear Regression in R

To fit a **linear regression model** in R we use the function `lm()`.

```
lm(formula, data, subset, weights, na.action,
   model = TRUE, x = FALSE, y = FALSE,
   contrasts = NULL, offset, ...)
```

The most important arguments of `lm()` are

- **`formula`**:
  a `formula` object
- **`data`**:
  `data.frame` (optional, but usually needed)

Model Formula

A **`formula`** object has the form

```
response ~ terms
```

for example

```
y ~ x1 + x2 + x3
```

Model Formula

A **`formula`** object has the form

```
response ~ terms
```

for example

```
y ~ x1 + x2 + x3
```

- Variables are separated by "**+**" signs.
- An **intercept** is automatically included.
- **One-sided formulas** (omitting the outcome) are possible.

Model Formula

The **intercept** can be removed from a formula by using **"-1"** or **"+0"**, i.e.,

```
y ~ x1 + x2 - 1
y ~ x1 + x2 + 0
```

Model Formula

The **intercept** can be removed from a formula by using **"-1"** or **"+0"**, i.e.,

```
y ~ x1 + x2 - 1
y ~ x1 + x2 + 0
```

Include all (other) variables as covariates using "**.**":

```
y ~ .
```

Model Formula

The **intercept** can be removed from a formula by using **"-1"** or **"+0"**, i.e.,

```
y ~ x1 + x2 - 1
y ~ x1 + x2 + 0
```

Include all (other) variables as covariates using "**.**":

```
y ~ .
```

Remove terms from the model using "**-**", e.g.:

```
y ~ . - x4
```

Model Formula

Example

If we had a `data.frame` like this:

y	x1	x2	x3	x4	x5
-1.01	-1.54	0.96	0.60	0.25	0.94
0.11	0.70	0.39	0.22	1.24	0.20
0.93	0.42	0.69	-0.59	0.28	-1.79
0.77	1.01	-0.19	-0.95	0.82	-0.53
-1.20	-0.66	-1.32	0.02	-0.21	0.08

Within `lm()`, the formula specified by

```
y ~ .
```

is identical to the formula

```
y ~ x1 + x2 + x3 + x4 + x5
```

Model Formula

Example

If we had a `data.frame` like this:

y	x1	x2	x3	x4	x5
-1.01	-1.54	0.96	0.60	0.25	0.94
0.11	0.70	0.39	0.22	1.24	0.20
0.93	0.42	0.69	-0.59	0.28	-1.79
0.77	1.01	-0.19	-0.95	0.82	-0.53
-1.20	-0.66	-1.32	0.02	-0.21	0.08

Within `lm()`, the formula specified by

```
y ~ .
```

is identical to the formula

```
y ~ x1 + x2 + x3 + x4 + x5
```

and the formula specified by

```
y ~ . - x3
```

is identical to the formula

```
y ~ x1 + x2 + x4 + x5
```

Algebraic Operations

Example: We would like to have the sum of **`x2`** and **`x3`** as a covariate.

Instead of pre-calculating this new variable

```
z <- x2 + x3
y ~ x1 + z
```

Algebraic Operations

Example: We would like to have the sum of **`x2`** and **`x3`** as a covariate.

Instead of pre-calculating this new variable

```
z <- x2 + x3
y ~ x1 + z
```

we can include this calculation directly in the model formula:

```
y ~ x1 + I(x2 + x3)
```

The function **`I()`** indicates **algebraic operations**,

Algebraic Operations

Example: We would like to have the sum of **`x2`** and **`x3`** as a covariate.

Instead of pre-calculating this new variable

```
z <- x2 + x3
y ~ x1 + z
```

we can include this calculation directly in the model formula:

```
y ~ x1 + I(x2 + x3)
```

The function **`I()`** indicates **algebraic operations**,

e.g., to distinguish the sum `x2 + x3` from the formula including the separate variables:

```
y ~ x1 + x2 + x3
```

A First Linear Regression

```
model1 <- lm(SBP ~ age + gender, data = nhanes)
```

```
model1
```

```
##
## Call:
## lm(formula = SBP ~ age + gender, data = nhanes)
##
## Coefficients:
##  (Intercept)            age  genderfemale
##      106.097          0.365        -5.763
```

Model Summary

```
summary(model1)
```

```
##
## Call:
## lm(formula = SBP ~ age + gender, data = nhanes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.37  -9.08  -1.95   7.85  53.30
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  106.0969     3.2456   32.69  < 2e-16 ***
## age            0.3646     0.0715    5.10  8.4e-07 ***
## genderfemale  -5.7627     2.1531   -2.68   0.0081 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.3 on 183 degrees of freedom
## Multiple R-squared:  0.136,    Adjusted R-squared:  0.127
## F-statistic: 14.5 on 2 and 183 DF,  p-value: 1.49e-06
```

Elements of a Fitted Model

The fitted object `model1` is an object of class `"lm"`:

```
class(model1)
```

```
## [1] "lm"
```

Elements of a Fitted Model

The fitted object `model1` is an object of class `"lm"`:

```
class(model1)
```

```
## [1] "lm"
```

Such an object has the underlying structure of a `list` object:

```
str(model1)
```

```
## List of 13
##  $ coefficients : Named num [1:3] 106.097 0.365 -5.763
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "age" "genderfemale"
##  $ residuals    : Named num [1:186] -10.86 -14.62 -18.77 -8.48 -6.01 ...
##   ..- attr(*, "names")= chr [1:186] "10" "14" "41" "77" ...
##  $ effects      : Named num [1:186] -1626.98 66.9 38.4 -7.19 -6.27 ...
##   ..- attr(*, "names")= chr [1:186] "(Intercept)" "age" "genderfemale" "" ...
##  $ rank         : int 3
##  $ fitted.values: Named num [1:186] 119 120 129 114 121 ...
##   ..- attr(*, "names")= chr [1:186] "10" "14" "41" "77" ...
## [...]
```

Elements of a Fitted Model

```
names(model1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "xlevels"       "call"          "terms"
## [13] "model"
```

Elements of a Fitted Model

```
names(model1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "xlevels"       "call"          "terms"
## [13] "model"
```

We can access the separate elements either with the `$` operator, or using `[[]]`.

```
model1$coefficients
```

```
##  (Intercept)          age genderfemale
##     106.0969       0.3646      -5.7627
```

Elements of a Fitted Model

`coefficients`	a named vector of coefficients	$\hat{\beta}$
`residuals`	the residuals	$\hat{\varepsilon}$
`fitted.values`	the fitted mean values	$\hat{\mathbf{y}}$
`rank`	the numeric rank of the fitted linear model	usually: $p+1$
`weights`	the specified weights (only for weighted fits)	
`df.residual`	the residual degrees of freedom	$n-p-1$
`call`	the matched call	
`terms`	the terms object used	
`contrasts`	the contrasts used (only where relevant)	
`xlevels`	levels of the factor covariates (only where relevant)	
`offset`	the offset used (missing if none were used)	
`y`	the response used (if requested)	$\mathbf{y}$
`x`	the model matrix used (if requested)	design matrix $\mathbf{X}$
`model`	if requested (the default), the model frame used	
`na.action`	information returned by `model.frame` on the handling of `NA`s (where relevant)	

Model Rank

The column **rank** is number of **linearly independent** columns in a matrix.

The new variable `age_half` is a linear combination of `age`:

```
nhanes$age_half <- nhanes$age/2
```

⇨ **Rank-deficient model**:

```
lm_rank_def <- lm(SBP ~ age + age_half + gender, data = nhanes, x = TRUE)
```

Model Rank

The column **rank** is number of **linearly independent** columns in a matrix.

The new variable `age_half` is a linear combination of `age`:

```
nhanes$age_half <- nhanes$age/2
```

⇨ **Rank-deficient model**:

```
lm_rank_def <- lm(SBP ~ age + age_half + gender, data = nhanes, x = TRUE)
```

```
head(lm_rank_def$x, 3)  # design matrix
```

```
##    (Intercept) age age_half genderfemale
## 10           1  35     17.5            0
## 14           1  38     19.0            0
## 41           1  78     39.0            1
```

```
ncol(lm_rank_def$x)  # number of columns
```

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

```
lm_rank_def$rank      # rank
```

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

Model Rank

The least squares estimator **requires full rank**.

⇨ It is not possible to estimate parameters of linearly dependent variables:

```
lm_rank_def
```

```
##
## Call:
## lm(formula = SBP ~ age + age_half + gender, data = nhanes, x = TRUE)
##
## Coefficients:
##  (Intercept)          age     age_half  genderfemale
##      106.097        0.365           NA        -5.763
```

`df.residual`, `call` & `xlevels`

Residual degrees of freedom: $n - p - 1$

```
model1$df.residual
```

```
## [1] 183
```

```
nrow(model1$model) - length(model1$coef)
```

```
## [1] 183
```

df.residual, call & xlevels

Residual degrees of freedom: $n - p - 1$

```
model1$df.residual
```

```
## [1] 183
```

```
nrow(model1$model) - length(model1$coef)
```

```
## [1] 183
```

Function call:

```
model1$call
```

```
## lm(formula = SBP ~ age + gender, data = nhanes)
```

Levels of factor covariates:

```
model1$xlevels
```

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

Additional Arguments

`lm()` has several more (optional) arguments:

- **`subset`**:
  a `logical` vector specifying which observations should be used
- **`weights`**:
  a `numeric` vector of weights for weighted least squares
- **`na.action`**:
  a character string indicating how missing values should be handled
- **`contrasts`**:
  a list specifying contrasts for categorical covariates
- **`offset`**:
  a `numeric` vector or matrix

Subsets

Fit a model on a subset of the data:

```
lm_sub <- lm(SBP ~ age + gender, data = nhanes, subset = educ == "low")
```

Subsets

Fit a model on a subset of the data:

```
lm_sub <- lm(SBP ~ age + gender, data = nhanes, subset = educ == "low")
```

original data:

```
nrow(nhanes)
```

```
## [1] 186
```

design matrix:

```
nrow(lm_sub$model)
```

```
## [1] 70
```

same as with `subset()`:

```
nrow(subset(nhanes, educ == "low"))
```

```
## [1] 70
```

Subsets

Fit a model on a subset of the data:

```
lm_sub <- lm(SBP ~ age + gender, data = nhanes, subset = educ == "low")
```

original data:

```
nrow(nhanes)
```

```
## [1] 186
```

design matrix:

```
nrow(lm_sub$model)
```

```
## [1] 70
```

same as with `subset()`:

```
nrow(subset(nhanes, educ == "low"))
```

```
## [1] 70
```

Use of multiple criteria:

```
lm_sub2 <- lm(SBP ~ age + gender, data = nhanes,
              subset = educ == "low" & age < 60)
```

Weights: Weighted Least Squares

Assumption of the **OLS**: $\mathrm{var}(\varepsilon_i) = \sigma^2$

⇨ **Weighted least squares:** $\sum_{i=1}^{n} w_i \hat{\varepsilon}_i^2 \to \min_{\boldsymbol{\beta}},$

with weights w_i inversely proportional to the variances $\mathrm{var}(\varepsilon_i)$.

Weights: Weighted Least Squares

Assumption of the **OLS**: $\text{var}(\varepsilon_i) = \sigma^2$

⇨ **Weighted least squares:** $\sum_{i=1}^{n} w_i \hat{\varepsilon}_i^2 \rightarrow \min_{\beta}$,

with weights w_i inversely proportional to the variances $\text{var}(\varepsilon_i)$.

(here: assumption variance associated with age)

```
lm_WLS <- lm(SBP ~ age + gender, weights = 1/age, data = nhanes)

lm_WLS$weights
```

```
##   [1] 0.02857 0.02632 0.01282 0.04348 0.02500 0.01852 0.03226 0.03704 0.02703
##  [10] 0.02000 0.01587 0.03846 0.02857 0.02273 0.02941 0.01667 0.04167 0.02083
##  [...]
```

Missing Values

```
model2 <- lm(SBP ~ age + gender + alc, data = nhanes)
summary(model2)
```

```
## [...]
##
## Residual standard error: 13.8 on 148 degrees of freedom
##   (34 observations deleted due to missingness)
## Multiple R-squared:  0.182,    Adjusted R-squared:  0.165
## F-statistic: 10.9 on 3 and 148 DF,  p-value: 1.56e-06
```

Missing Values

```r
model2 <- lm(SBP ~ age + gender + alc, data = nhanes)
summary(model2)
```

```
## [...]
##
## Residual standard error: 13.8 on 148 degrees of freedom
##   (34 observations deleted due to missingness)
## Multiple R-squared:  0.182,	Adjusted R-squared:  0.165
## F-statistic: 10.9 on 3 and 148 DF,  p-value: 1.56e-06
```

Global option:

```r
getOption("na.action")
```

```
## [1] "na.omit"
```

Missing Values

Trying to fit a model with **missing values** results in an **error**:

```
lm(SBP ~ age + gender + alc, data = nhanes, na.action = NULL)
```

```
## Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...): NA/NaN/Inf
in 'x'
```

```
lm(SBP ~ age + gender + alc, data = nhanes, na.action = "na.pass")
```

```
## Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...): NA/NaN/Inf
in 'x'
```

Missing Values

Trying to fit a model with **missing values** results in an **error**:

```
lm(SBP ~ age + gender + alc, data = nhanes, na.action = NULL)
```

```
## Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...): NA/NaN/Inf
in 'x'
```

```
lm(SBP ~ age + gender + alc, data = nhanes, na.action = "na.pass")
```

```
## Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...): NA/NaN/Inf
in 'x'
```

`na.fail` gives a similar error:

```
lm(SBP ~ age + gender + alc, data = nhanes, na.action = "na.fail")
```

```
## Error in na.fail.default(structure(list(SBP = c(108, 105.333333333333, : missing
values in object
```

Missing Values

The options **`na.omit`** and **`na.exclude`** will remove rows with missing values:

```
model2_omit <- lm(SBP ~ age + gender + alc, data = nhanes, na.action = "na.omit")
model2_exclude <- lm(SBP ~ age + gender + alc, data = nhanes,
                     na.action = "na.exclude")
```

There is a relevant difference when we use functions like

- `residuals()`
- `predict()`

```
head(residuals(model2_omit))
```

```
##       14      41      77      91     105     149
## -12.006 -17.001 -12.860  -3.394   9.295  -9.242
```

```
head(residuals(model2_exclude))
```

```
##       10      14      41      77      91     105
##       NA -12.006 -17.001 -12.860  -3.394   9.295
```

Contrasts

When categorical variables are coded as **`factor`**, R applies contrasts.

```
lm(SBP ~ age + gender + smoke, data = nhanes)
```

```
##
## Call:
## lm(formula = SBP ~ age + gender + smoke, data = nhanes)
##
## Coefficients:
##  (Intercept)           age  genderfemale       smoke.L       smoke.Q
##      102.661         0.446        -8.011        -1.752         4.723
```

Contrasts

When categorical variables are coded as **`factor`**, R applies contrasts.

```
lm(SBP ~ age + gender + smoke, data = nhanes)
```

```
##
## Call:
## lm(formula = SBP ~ age + gender + smoke, data = nhanes)
##
## Coefficients:
##  (Intercept)            age   genderfemale        smoke.L        smoke.Q
##      102.661          0.446         -8.011         -1.752          4.723
```

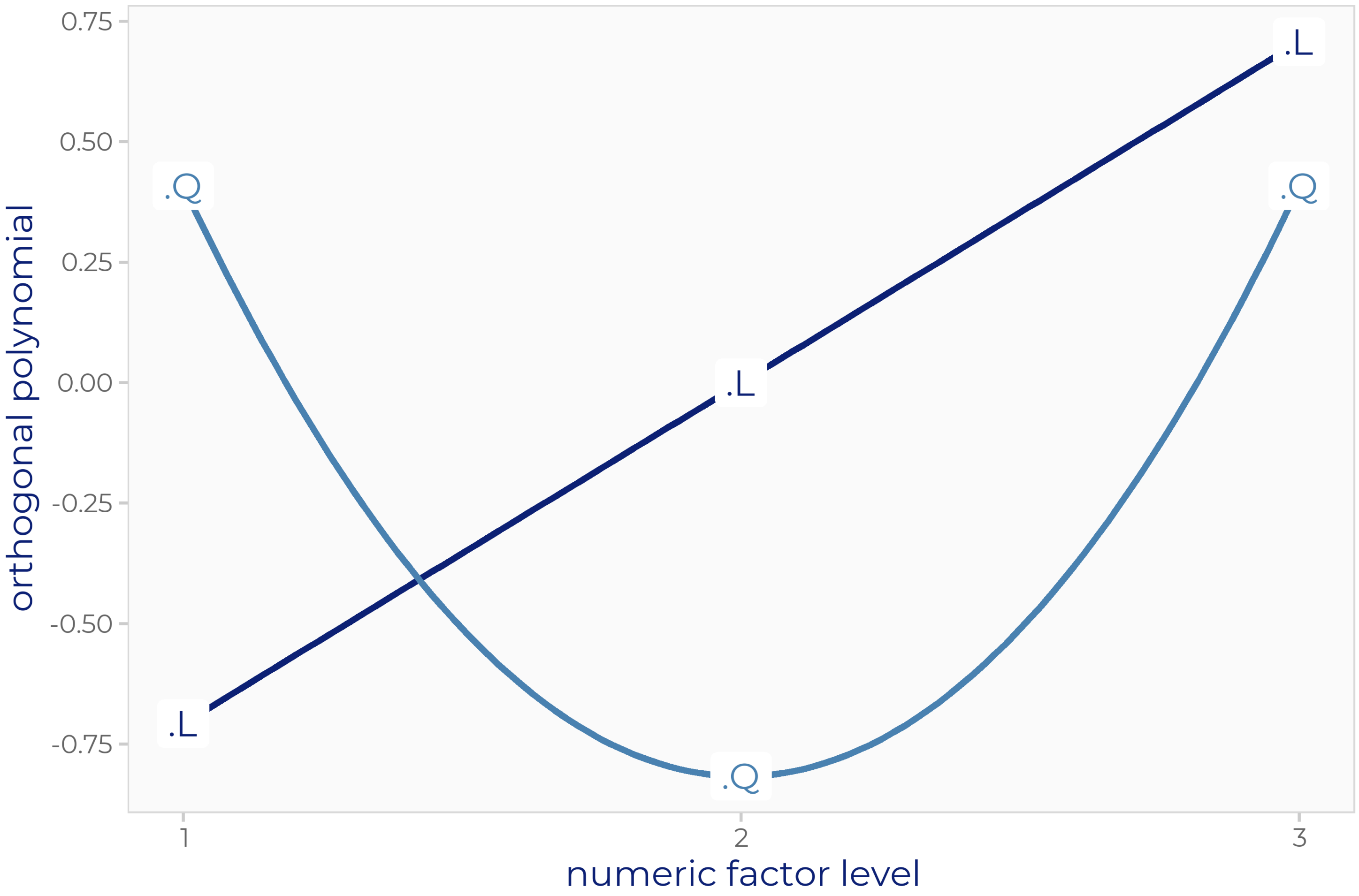
```
contrasts(nhanes$smoke)
```

```
##                .L      .Q
## [1,] -7.071e-01  0.4082
## [2,] -7.850e-17 -0.8165
## [3,]  7.071e-01  0.4082
```

Orthogonal Polynomials

```r
round(contr.poly(3), 3)
```

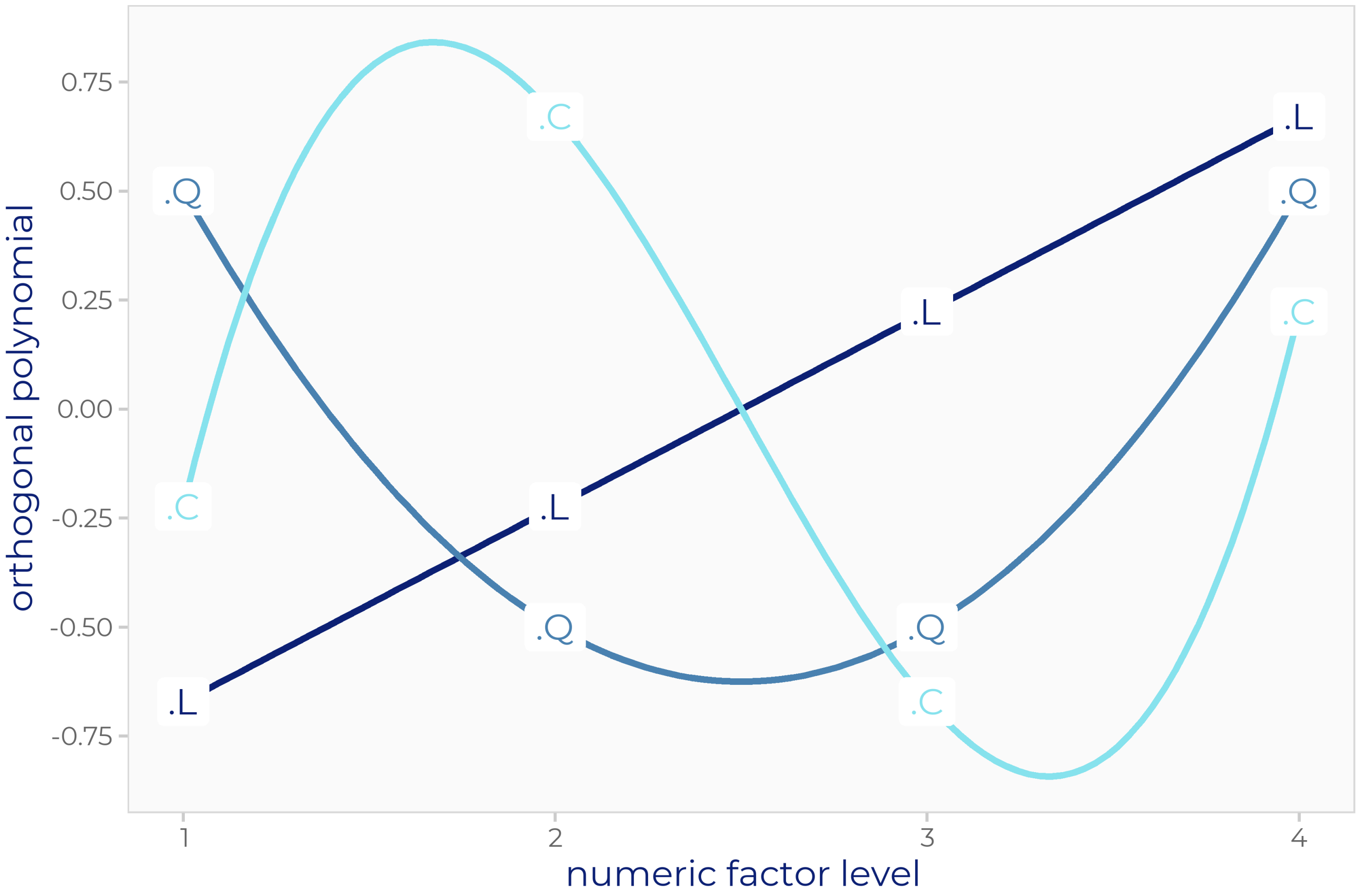
```
##              .L     .Q
## [1,] -0.707  0.408
## [2,]  0.000 -0.816
## [3,]  0.707  0.408
```



Orthogonal Polynomials

```
round(contr.poly(4), 3)
```

```
##             .L   .Q     .C
## [1,] -0.671  0.5 -0.224
## [2,] -0.224 -0.5  0.671
## [3,]  0.224 -0.5 -0.671
## [4,]  0.671  0.5  0.224
```



Contrasts

Global Option:

```
getOption("contrasts")
```

```
##           unordered             ordered
## "contr.treatment"        "contr.poly"
```

Contrasts

Global Option:

```r
getOption("contrasts")
```

```
##           unordered           ordered
## "contr.treatment"      "contr.poly"
```

Dummy Coding:

```r
contr.treatment(c("A", "B", "C"))
```

```
##   B C
## A 0 0
## B 1 0
## C 0 1
```

Effect Coding:

```r
contr.sum(c("A", "B", "C"))
```

```
##   [,1] [,2]
## A    1    0
## B    0    1
## C   -1   -1
```

Contrasts

```r
lm(SBP ~ age + gender + smoke + occup, data = nhanes,
   contrasts = list(smoke = "contr.treatment",
                    occup = "contr.sum"))
```

```
##
## Call:
## lm(formula = SBP ~ age + gender + smoke + occup, data = nhanes,
##     contrasts = list(smoke = "contr.treatment", occup = "contr.sum"))
##
## Coefficients:
##  (Intercept)           age  genderfemale   smokeformer  smokecurrent
##      108.645         0.398        -8.175        -7.590        -1.992
##       occup1        occup2
##       -0.942         2.581
```