# Inference with Linear Equality and Inequality Constraints Using R: The Package ic.infer

**Ulrike Grömping**
BHT Berlin – University of Applied Sciences

**Abstract**

In linear models and multivariate normal situations, prior information in linear inequality form may be encountered, or linear inequality hypotheses may be subjected to statistical tests. R package **ic.infer** has been developed to support inequality-constrained estimation and testing for such situations. This article gives an overview of the principles underlying inequality-constrained inference that are far less well-known than methods for unconstrained or equality-constrained models, and describes their implementation in the package.

*Keywords*: inequality constraints, quadratic programming, order-restricted linear model, **ic.infer**, **mvtnorm**.

## 1. What is this?

This is not the source but a dummy source to make the pdf available as vignette and to have the code checked during R CMD check.

### 1.1. Example data

Two data examples are used in this section. The first example, taken from Table 1.3.1 in Robertson, Wright, and Dykstra (1988), concerns first-year grade point averages from 2397 Iowa university first-years (available as data frame `grades` in package **ic.infer**) as a function of two ordinal variables with 9 categories each, High-School-Ranking percentiles and ACT Classification[1]. Suppose that an admission policy is to be developed based on these figures. Of course, in order to appear just, an admission policy should be monotone in the sense that admission of a particular person implies that all persons who are better on one criterion and not worse on the other are also admitted. Thus, the predicted function must be monotone in both variables. Using this motivation, Robertson *et al.* (1988) demonstrate isotonic regression on these data. In this article, a two-way analysis of variance without interaction is fit to the data. The unrestricted linear model (cf. below) does contain reversals w.r.t. HSR, where applicants with HSR 41% to 50% would be assessed better than those with HSR 51% to 60%, and similarly applicants with HSR $< 20\%$ better than those with HSR 21% to 40%. Note

[1] ACT is an organization that offers – among other things – college entrance exams in the US; up to 1996, ACT stood for "American College Testing".

that estimates for the categories of HSR for which unrestricted estimates are reversed are not significantly different from 0. The restricted analyses in Sections 1.3 and 1.6 will restrict parameters for the factor HSR to be monotone. Note that this is an example of a model with a known diagonal (but not identity) $V_0$: assuming an unknown positive variance $\sigma^2$ of the grade points of each student, the variances of the grade means are proportional to the inverse number of students in each class. This can be easily accomodated in function `lm` by using the number of students `n` in the `weights` option (cf. the code below).

```
R> limo.grades <- lm(meanGPA ~ HSR + ACTC, grades, weights = n)
R> summary(limo.grades)

Call:
lm(formula = meanGPA ~ HSR + ACTC, data = grades, weights = n)

Weighted Residuals:
   Min     1Q Median     3Q    Max 
-2.224 -0.494 -0.149  0.433  1.776 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.5009     0.2367    6.34  6.4e-08 ***
HSR21-30     -0.1251     0.2540   -0.49  0.62456    
HSR31-40     -0.0272     0.2279   -0.12  0.90533    
HSR41-50      0.1489     0.2131    0.70  0.48796    
HSR51-60      0.0947     0.2077    0.46  0.65059    
HSR61-70      0.3129     0.2055    1.52  0.13419    
HSR71-80      0.4290     0.2044    2.10  0.04092 *  
HSR81-90      0.5612     0.2045    2.74  0.00839 ** 
HSR>=91       0.9703     0.2043    4.75  1.8e-05 ***
ACTC13-15     0.2937     0.1625    1.81  0.07662 .  
ACTC16-18     0.4565     0.1455    3.14  0.00286 ** 
ACTC19-21     0.5332     0.1402    3.80  0.00039 ***
ACTC22-24     0.6193     0.1391    4.45  4.7e-05 ***
ACTC25-27     0.6698     0.1396    4.80  1.5e-05 ***
ACTC28-30     0.8223     0.1454    5.66  7.5e-07 ***
ACTC31-33     0.9214     0.1846    4.99  7.7e-06 ***
ACTC34-36     1.0389     0.4959    2.09  0.04127 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.951 on 50 degrees of freedom
Multiple R-squared:  0.908,	Adjusted R-squared:  0.878 
F-statistic: 30.7 on 16 and 50 DF,  p-value: <2e-16
```

The second example uses a data set from Kutner, Nachtsheim, and Neter (2004) (online also at http://www.ats.ucla.edu/stat/sas/examples/alsm/alsmsasch7.htm) that contains observations on 20 females with body fat as the target variable and three explanatory variables all of which can be expected to be associated with an increase in body fat:

- triceps skinfold thickness
- thigh circumference
- mid arm circumference.

These data are analysed as a regression model with all coefficients restricted to be non-negative. This example is similar in spirit to the customer satisfaction applications that instigated development of **ic.infer**, but much smaller, publicly available and included in the package. It also permits to demonstrate application of the simple $R^2$ decomposition function that is offered within **ic.infer**. The unrestricted linear model estimates for two of the three variables are negative, and in spite of high $R^2$ and rejection of the overall null hypothesis, no individual coefficient is statistically significant:

```
R> limo.bodyfat <- lm(BodyFat ~ ., bodyfat)
R> summary(limo.bodyfat)

Call:
lm(formula = BodyFat ~ ., data = bodyfat)

Residuals:
   Min     1Q Median     3Q    Max
-3.726 -1.611  0.392  1.466  4.128

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   117.08      99.78    1.17     0.26
Triceps         4.33       3.02    1.44     0.17
Thigh          -2.86       2.58   -1.11     0.28
Midarm         -2.19       1.60   -1.37     0.19

Residual standard error: 2.48 on 16 degrees of freedom
Multiple R-squared:  0.801,       Adjusted R-squared:  0.764
F-statistic: 21.5 on 3 and 16 DF,  p-value: 7.34e-06
```

## 1.2. Utilities for monotonicity situations

One of the most important special cases of inequality-related setups is the investigation of monotonic behavior of the expectation for a factor with ordered categories. In this subsection, package **ic.infer**'s support for this situation is described.

### *Difference contrasts*

The interpretation of coefficients for factors always depends on the factor coding. In R, default coding for conventional factors (as opposed to ordered factors) is a reference coding with the first factor level being the base category (called `contr.treatment`). For factors declared to be ordered, the default contrasts are polynomial. Alternative contrast codings are, among others, `contr.SAS`, `contr.helmert` and `contr.sum`. Among these, the polynomial and the Helmert coding do not allow simple assessment of monotonicity based on the estimated coefficients, while the others do.

There is one particular factor coding that is not routinely available in R but particularly suitable for assessing monotonicity for factors with ordered levels: each coefficient corresponds to the difference in expectation to the next lower category, implying that monotonicity corresponds to the same sign for all coefficients. The corresponding contrast function `contr.diff` has been implemented in package **ic.infer**.

For illustration, the unconstrained linear model for the grades data is re-calculated with this coding below. The contrast matrix shows that the expectation for the lowest level does not contain any of the coefficients, the expectation for the second level contains the first coefficient, the expectation for the third level the first two coefficients and so forth, until all the eight coefficients are contained in the expectation model for the highest level. The coefficients thus measure the average increase from each level to the next higher one.

```
R> grades.diff <- grades
R> ## change contrasts to contr.diff
R> contrasts(grades.diff$HSR) <- "contr.diff"
R> contrasts(grades.diff$ACTC) <- "contr.diff"
R> ## display contrasts
R> contrasts(grades.diff$HSR)

      21-30-<=20 31-40-21-30 41-50-31-40 51-60-41-50 61-70-51-60 71-80-61-70
<=20           0           0           0           0           0           0
21-30          1           0           0           0           0           0
31-40          1           1           0           0           0           0
41-50          1           1           1           0           0           0
51-60          1           1           1           1           0           0
61-70          1           1           1           1           1           0
71-80          1           1           1           1           1           1
81-90          1           1           1           1           1           1
>=91           1           1           1           1           1           1
      81-90-71-80 >=91-81-90
<=20            0          0
21-30           0          0
31-40           0          0
41-50           0          0
51-60           0          0
61-70           0          0
71-80           0          0
81-90           1          0
>=91            1          1

R> limo.grades.diff <- lm(meanGPA ~ HSR + ACTC, grades.diff, weights = n)
R> summary(limo.grades.diff)

Call:
lm(formula = meanGPA ~ HSR + ACTC, data = grades.diff, weights = n)

Weighted Residuals:
   Min     1Q Median     3Q    Max 
-2.224 -0.494 -0.149  0.433  1.776 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       1.5009     0.2367    6.34  6.4e-08 ***
HSR21-30-<=20    -0.1251     0.2540   -0.49   0.6246    
HSR31-40-21-30    0.0978     0.1942    0.50   0.6165    
HSR41-50-31-40    0.1761     0.1363    1.29   0.2021    
HSR51-60-41-50   -0.0542     0.0990   -0.55   0.5860    
HSR61-70-51-60    0.2182     0.0814    2.68   0.0099 ** 
HSR71-80-61-70    0.1161     0.0719    1.62   0.1123    
HSR81-90-71-80    0.1322     0.0655    2.02   0.0489 *  
HSR>=91-81-90     0.4091     0.0593    6.90  8.5e-09 ***
ACTC13-15-1-12    0.2937     0.1625    1.81   0.0766 .  
ACTC16-18-13-15   0.1628     0.1114    1.46   0.1503    
ACTC19-21-16-18   0.0767     0.0759    1.01   0.3168    
ACTC22-24-19-21   0.0861     0.0622    1.38   0.1727    
ACTC25-27-22-24   0.0505     0.0570    0.89   0.3801    
ACTC28-30-25-27   0.1524     0.0653    2.34   0.0236 *  
```

```
ACTC31-33-28-30   0.0991     0.1329    0.75   0.4591
ACTC34-36-31-33   0.1174     0.4913    0.24   0.8121
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.951 on 50 degrees of freedom
Multiple R-squared:  0.908,       Adjusted R-squared:  0.878
F-statistic: 30.7 on 16 and 50 DF,  p-value: <2e-16
```

### *Utility function for creating a monotonicity restriction matrix*

Generally, the restriction matrix *ui* has to be tailored to the situation at hand. Depending on the coding of a factor, it can be quite tedious to define the appropriate *ui* for hypotheses related to the relation of expectations between factor levels.

For the frequent situation, where monotonicity of factors with several ordered levels is of interest, package **ic.infer** provides the convenience function `make.mon.ui` for creating the appropriate restriction matrix *ui*. The function can be used whenever the current coding permits assessment of monotonicity in a simple way, i.e., for contrasts `contr.treatment` (currently with first category as baseline only), `contr.SAS`, `contr.diff` and `contr.sum`). The output below shows the matrix *ui* for two different factor codings: The matrix *ui* for the treatment contrasts calculates the first coefficient (=difference of second category to the first (=reference) category) and all differences between coefficients for next higher to next lower level. The matrix *ui* for the difference contrasts simply calculates each coefficient. For both codings, monotonicity constraints are of the form $ui\beta \geq 0$ (or $-ui\beta \geq 0$ for monotone decrease).

```
R> ## originally, treatment contrasts
R> ui.treat <- make.mon.ui(grades$HSR)
R> ui.treat

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    0    0    0    0    0    0
[2,]   -1    1    0    0    0    0    0    0
[3,]    0   -1    1    0    0    0    0    0
[4,]    0    0   -1    1    0    0    0    0
[5,]    0    0    0   -1    1    0    0    0
[6,]    0    0    0    0   -1    1    0    0
[7,]    0    0    0    0    0   -1    1    0
[8,]    0    0    0    0    0    0   -1    1

R> ui.diff <- make.mon.ui(grades.diff$HSR)
R> ui.diff

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

Function `make.mon.ui` can also be used for creating the matrix *ui* for investigating the monotonicity of a multivariate normal mean without using a linear model based on a factor. In this case, the first argument to `make.mon.ui` is the dimension of the multivariate normal distribution, and `type = "mean"` must be specified. The resulting matrix $ui$ then calculates differences of neighbouring means:

```
R> ## originally, treatment contrasts
R> make.mon.ui(5, type = "mean")

     [,1] [,2] [,3] [,4] [,5]
[1,]   -1    1    0    0    0
[2,]    0   -1    1    0    0
[3,]    0    0   -1    1    0
[4,]    0    0    0   -1    1
```

## 1.3. Estimation

Function `ic.est` for inequality-constrained estimation of normal means uses the routine `solve.QP` from R-package **quadprog** to determine the constrained estimate. It is possible to declare the first few rows of the restrictions $ui\mu \geq ci$ to be equality restrictions (via the parameter `meq`). It has been pointed out above that estimation of $\beta$ in the restricted linear model is equivalent to estimation of $\beta$ based on the multivariate normal distribution of the unrestricted estimate $\hat{\beta}$. Thus, we can illustrate function `ic.est` using the estimates from one of the linear models above. For example, one can estimate the coefficients of the factor `HSR` with treatment contrasts under the restriction of non-decreasing behavior, i.e., $\beta_1 \geq 0, \beta_2 - \beta_1 \geq 0, \ldots, \beta_9 - \beta_8 \geq 0$ (contrast matrix `ui.treat` defined in 1.2.2):

```
R> HSRmon <- ic.est(coef(limo.grades)[2:9],
+         ui = ui.treat,
+         Sigma = vcov(limo.grades)[2:9, 2:9])
R> HSRmon

Constrained estimate:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
  0.0000    0.0492    0.1918    0.1918    0.3892    0.5055    0.6377    1.0469
```

It is also possible to indicate that the first few restrictions (number given by option `meq`) are equality restrictions. For example, the code below declares that the first three restrictions are equality instead of inequality restrictions:

```
R> HSReq <- ic.est(coef(limo.grades)[2:9],
+         ui = ui.treat,
+         Sigma = vcov(limo.grades)[2:9, 2:9], meq = 3)
R> HSReq

Constrained estimate:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
   0.000     0.000     0.000     0.038     0.256     0.372     0.505     0.914
```

A summary-function on objects of class `orest` – as generated by function `ic.est` – gives more detailed information, showing also the restrictions, which of them are active, and indicating which estimates are subject to a restriction (regardless whether active or not). For the monotonicity-restricted estimate, we get

```
R> summary(HSRmon)
```

```
Order-restricted estimated mean with restrictions of coefficients of
HSR21-30 HSR31-40 HSR41-50 HSR51-60 HSR61-70 HSR71-80 HSR81-90 HSR>=91


 Inequality restrictions:
    HSR21-30 HSR31-40 HSR41-50 HSR51-60 HSR61-70 HSR71-80 HSR81-90 HSR>=91
1: A 1        0        0        0        0        0        0        0        %*%colnames  >=  0
2:  -1        1        0        0        0        0        0        0        %*%colnames  >=  0
3:   0       -1        1        0        0        0        0        0        %*%colnames  >=  0
4: A 0        0       -1        1        0        0        0        0        %*%colnames  >=  0
5:   0        0        0       -1        1        0        0        0        %*%colnames  >=  0
6:   0        0        0        0       -1        1        0        0        %*%colnames  >=  0
7:   0        0        0        0        0       -1        1        0        %*%colnames  >=  0
8:   0        0        0        0        0        0       -1        1        %*%colnames  >=  0

 Note: Restrictions marked with A are active.

Restricted estimate:
R HSR21-30 R HSR31-40 R HSR41-50 R HSR51-60 R HSR61-70 R HSR71-80 R HSR81-90  R HSR>=91
   0.00000    0.04917    0.19181    0.19181    0.38920    0.50551    0.63772    1.04688

 Note: Estimates marked with R are involved in restrictions.
```

While it would be possible to determine the estimate even for linearly dependent rows of the constraint matrix $R$, this is not permitted in package **ic.infer** – if the package encounters linearly dependent rows in `ui` (the package notation for $R$), it aborts with an error message that suggests a subset of independent rows of `ui`.

## 1.4. Hypothesis testing

Package **ic.infer** implements the likelihood ratio tests for test problems **??**, **??**, and **??** in function `ic.test`. The principal argument to function `ic.test` is an object of class `orest` as output by function `ic.est`; an object of class `orlm` output by function `orlm` can also be processed, since it inherits from class `orest`. Among other things, the input object contains information on the restrictions that were used for estimation. The type of test problem is indicated to function `ic.test` via option `TP`. `TP = 1`, `TP = 2`, and `TP = 3` refer to the test problems introduced in Section **??**. Three extensions of these problems are additionally implemented:

- For `TP = 1` and `TP = 2`, the first few restrictions can be declared equality instead of inequality restrictions – this is implemented in function `ic.test` through access to the `meq`-element of the input object. This modification requires different calculation of the weights for the null distributions of the test statistics: these weights depend on the conditional covariance matrix given the equality constraints are true, cf. Shapiro (1988, formula (5.9)) and Section 1.5. The test statistics continue to be given by (**??**), (**??**) or their modification for unkown $\sigma^2$ (**??**), but with $\hat{\mu}^*$ observing equality- and inequality restrictions.
- Additional equality restrictions can be included in the null hypothesis of **??**. For these, the alternative hypothesis is not directional. This test problem is implemented in the package as `TP = 11`, and the additional restrictions are handed to function `ic.test` through arguments `ui0.11` and `ci0.11`. `TP = 11` is, for example, used in the summary

function for class `orlm`, when testing the null hypothesis that all coefficients except the intercept are 0 in the presence of constraints $R\beta \geq 0$ that do not affect all elements of $\beta$. Again, the test statistic for this test problem is already given as (**??**) or its modification (**??**) above by making $\hat{\mu}_=$ observe the additional equality restrictions as well. Here, the weights are the same as without equality restrictions, but the degrees of freedom of the distributions in the mixture need to be adjusted.

- Some equality restrictions can be maintained in the alternative hypothesis of **??**. This is implemented as `TP = 21` using option `meq.alt`, which indicates the number of the first few equality-restrictions that are to be maintained under the alternative hypothesis. `meq.alt` must not be larger than the `meq`-element of the input object of function `ic.test`. Here, the test statistic (**??**) (or (**??**)) has to use the restricted estimated under the maintained equality restrictions $\hat{\mu}_{=,alt}$ instead of $y$.

A few examples are shown below. First, the equality- and inequality-restricted estimate `HSReq` of the HSR coefficients is subjected to a test of type **??**. We see that equality of all restrictions is clearly rejected; note that option `brief=FALSE` requests detailed information on constraints that is not shown per default.

```
R> summary(ic.test(HSReq), brief = FALSE)

Order-related hypothesis test:

Type 1 Test:
 H0: all restrictions active(=)
     vs.
 H1: at least one restriction strictly true (>)
Test statistic        p-value
           215        <0.0001

Restrictions on  HSR21-30 HSR31-40 HSR41-50 HSR51-60 HSR61-70 HSR71-80 HSR81-90 HSR>=91
     HSR21-30 HSR31-40 HSR41-50 HSR51-60 HSR61-70 HSR71-80 HSR81-90 HSR>=91
1: A 1        0        0        0        0        0        0        0        %*%colnames  ==  0
2: A -1       1        0        0        0        0        0        0        %*%colnames  ==  0
3: A 0        -1       1        0        0        0        0        0        %*%colnames  ==  0
4:   0        0        -1       1        0        0        0        0        %*%colnames  >=  0
5:   0        0        0        -1       1        0        0        0        %*%colnames  >=  0
6:   0        0        0        0        -1       1        0        0        %*%colnames  >=  0
7:   0        0        0        0        0        -1       1        0        %*%colnames  >=  0
8:   0        0        0        0        0        0        -1       1        %*%colnames  >=  0

Restricted estimate under H0:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
       0         0         0         0         0         0         0         0

Restricted estimate under union of H0 and H1 :
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
   0.000     0.000     0.000     0.038     0.256     0.372     0.505     0.914
```

Now we test the null hypothesis that restrictions hold vs. the alternative that they are violated. We see that this null hypothesis is not rejected, i.e., the data do not provide proof that these restrictions are not all true.

```
R> summary(ic.test(HSReq, TP = 2))

Order-related hypothesis test:
```

```
Type 2 Test:
 H0: all restrictions true(>=)
     vs.
 H1: at least one restriction violated (<)
Test statistic        p-value
          3.36         0.7948

Restricted estimate under H0:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
   0.000     0.000     0.000     0.038     0.256     0.372     0.505     0.914

Unrestricted estimate:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
 -0.1251   -0.0272    0.1489    0.0947    0.3129    0.4290    0.5612    0.9703
```

The next test operates on all coefficients of the analysis of variance model from Section 1.1. This `TP = 11`-type test tests the null hypothesis that all coefficients except for the intercept are zero vs. the alternative that the HSR coefficients follow the restriction outlined above, i.e., coefficients in positions 2 to 9 of the coefficient vector (`index = 2:9`) follow the indicated restrictions, while all other coefficients are free. Here, the null hypothesis is again clearly rejected.

```
R> HSReq.large <- ic.est(coef(limo.grades),
+         ui = ui.treat,
+         Sigma = vcov(limo.grades), index = 2:9, meq = 3)
R> summary(ic.test(HSReq.large, TP = 11,
+         ui0.11 = cbind(rep(0, 16), diag(1, 16))))
```

```
Order-related hypothesis test:

Type 11 Test:
 H0: all original restrictions active plus additional equality restrictions
     vs.
 H1: original restrictions hold
Test statistic        p-value
           488        <0.0001

Restricted estimate under union of H0 and H1 :
(Intercept)     HSR21-30     HSR31-40     HSR41-50     HSR51-60     HSR61-70     HSR71-80
      1.552        0.000        0.000        0.000        0.038        0.256        0.372
   HSR81-90      HSR>=91    ACTC13-15    ACTC16-18    ACTC19-21    ACTC22-24    ACTC25-27
      0.505        0.914        0.300        0.467        0.535        0.624        0.676
  ACTC28-30    ACTC31-33    ACTC34-36
      0.827        0.927        1.044

Restricted estimate under H0:
(Intercept)     HSR21-30     HSR31-40     HSR41-50     HSR51-60     HSR61-70     HSR71-80
       2.63         0.00         0.00         0.00         0.00         0.00         0.00
   HSR81-90      HSR>=91    ACTC13-15    ACTC16-18    ACTC19-21    ACTC22-24    ACTC25-27
       0.00         0.00         0.00         0.00         0.00         0.00         0.00
  ACTC28-30    ACTC31-33    ACTC34-36
       0.00         0.00         0.00
```

The last example demonstrates `TP = 21`: the null hypothesis has three equality restrictions (estimate object `HSReq`), and the first two of these are maintained for the alternative hypothesis (`meq.alt=2`). Note that – as the alternative is unrestricted apart from the first two equality restrictions – a reversal occurs in the estimate under the alternative hypothesis. Nevertheless, like for `TP = 2`, the validity of the restrictions is not rejected.

```
R> summary(ic.test(HSReq, TP = 21, meq.alt = 2))

Order-related hypothesis test:

Type 21 Test:
 H0: all restrictions true(>= or =)
     vs.
 H1: at least one restriction violated (<), some =-restrictions maintained
Test statistic        p-value
          3.03         0.6134

Restricted estimate under H0:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
   0.000     0.000     0.000     0.038     0.256     0.372     0.505     0.914

Restricted estimate under H1:
HSR21-30  HSR31-40  HSR41-50  HSR51-60  HSR61-70  HSR71-80  HSR81-90   HSR>=91
   0.000     0.000     0.198     0.144     0.362     0.478     0.610     1.020
```

## 1.5. Calculation of weights and $p$ values for the test problems

Function `ic.weights` calculates the mixing weights for a given covariance matrix, using the probabilities for certain faces of the cone as derived in Section **??**. Since it is known that even and odd weights sum to 0.5 each (cf. e.g., Silvapulle and Sen 2004, Proposition 3.6.1, Number 3), the two most demanding weights (in terms of most summands in (**??**)) can always be inferred as the difference of 0.5 to the sum of the other even or odd weights. Even exploiting this possibility, calculation of weights remains computer-intensive for large covariance matrices; for example, it takes about 9 seconds CPU time for a matrix with dimension 10, and already 1265 seconds (about 21 minutes) for a matrix with dimension 15.

Orthant probabilities that are needed for the weights according to (**??**), are calculated using package **mvtnorm** by Monte-Carlo methods, i.e., the weights are subject to slight variation. Because of numerical inaccuracies, it is even possible that calculated $p$ values become slightly negative. Printing and summary functions of package **ic.infer** report all $p$ values below 0.0001 as "<0.0001", since more accuracy should normally not be needed.

For the test problems implemented in function `ic.test`, choice of the covariance matrices for obtaining the weights follows the formulae by Shapiro (1988), based on the `meq`-, the `ui`-, and the `Sigma`-element of the input object: Whenever `meq=0`, the covariance matrix to use is `ui%*%Sigma%*%t(ui)` (assuming that `ui` has $p$ columns if the data are $p$-dimensional, otherwise think of `ui` as suitably enlarged by zero columns (`uiw` in package code)). If `meq>0`, the conditional covariance of the last $m-$`meq` rows given the first `meq` rows of `ui%*%y` must be used instead for calculation of mixing weights (formula (5.9) in Shapiro 1988).

Degrees of freedom corresponding to the weights depend on the test problem at hand and are determined in function `ic.test`, if not provided by the user. Functions `pchibar` and `pbetabar` calculate $p$ values from given vectors of weights and degrees of freedom. Function `pchibar` has been taken from package **ibdreg** by Sinnwell and Schaid (2007), and function `pbetabar` has been analogously defined.

## 1.6. Estimation in the linear model

Function `orlm` uses the other functions in package **ic.infer** for providing a convenient overall analysis of order-restricted linear models. Starting from an unconstrained linear model object (class `lm`) or a covariance matrix of response (first position) and all regressors, the function determines the constrained estimate, $R^2$ for the constrained model and – if requested – bootstraps the estimates of coefficients (the latter is valid only in the implemented case of uncorrelated errors and of course only possible if the input is a linear model with embedded data).

### *Postprocessing the output object*

The output object of class `orlm` can be processed with several S3 methods provided in package **ic.infer**: A `plot` method provides a residual plot, a `print` method gives a brief printout, and a `summary` method gives a more extensive overview on the object, involving bootstrap confidence intervals and overall model and restriction tests, if not suppressed; tests can be suppressed because their calculation may take up substantial time in case of many restrictions because of calculation of weights, cf. also the previous subsection. Furthermore, a `coef` method extracts the coefficients from the object. In addition to these specially-defined methods, some general methods for model objects do also work: functions `fitted` and `residuals` provide fitted values and residuals. Other methods for class `lm` (`predict`, `effects`, `vcov`) do not work on `orlm` objects. Note that model diagnostics cannot be simply transferred to restricted models, as the restricted estimation modifies the distributional properties of the residuals in not easily foreseeable ways. The `plot` method only provides a simple plot of raw residuals vs. fitted values, as it is not even possible to standardize the residuals. Further research might improve the availability of diagnostics on the restricted model. As long as this has not been conducted, model diagnostics, e.g., for normality, can be done on the unrestricted model, which is of course still valid even though it does not exploit the prior knowledge about a restriction.

*Linear model analysis for the two example data sets*

For the grades data, with two ordinal factors, restricting only HSR (because ACTC is automatically in the correct order; indicated by `index=2:9` for the position of HSR-coefficients in the overall coefficient vector) function `orlm` works as follows (contrast matrix `ui.treat` defined in 1.2.2):

```
R> orlimo.grades <- orlm(limo.grades,
+         ui = ui.treat, index = 2:9)
R> summary(orlimo.grades, brief = TRUE)

Order-restricted linear model with restrictions of coefficients of
HSR21-30 HSR31-40 HSR41-50 HSR51-60 HSR61-70 HSR71-80 HSR81-90 HSR>=91


Coefficients from order-restricted model:
  (Intercept)    R HSR21-30    R HSR31-40    R HSR41-50    R HSR51-60    R HSR61-70    R HSR71-80
      1.42444       0.00000       0.04917       0.19181       0.19181       0.38920       0.50551
   R HSR81-90     R HSR>=91     ACTC13-15     ACTC16-18     ACTC19-21     ACTC22-24     ACTC25-27
      0.63772       1.04688       0.29496       0.45714       0.53311       0.61918       0.66945
    ACTC28-30     ACTC31-33     ACTC34-36
      0.82222       0.92091       1.03868

 Note: Coefficients marked with R are involved in restrictions.


Hypothesis tests ( 50 error degrees of freedom ):
Overall model test under the order restrictions:
       Test statistic: 0.9075,   p-value: <0.0001

Type 1 test: H0: all restrictions active(=)
         vs. H1: at least one restriction strictly true (>)
       Test statistic: 0.8132,   p-value: <0.0001
Type 2 test: H0: all restrictions true
         vs. H1: at least one restriction false
       Test statistic: 0.01074,   p-value: 0.9887
Type 3 test: H0: at least one restriction false or active (=)
         vs. H1: all restrictions strictly true (>)
       Test statistic: -0.5481,   p-value: 0.7070

Type 3 test based on t-distribution (one-sided),
all other tests based on mixture of beta distributions
```

Option `brief` suppresses information on restrictions (that has been shown in Section 1.3). For this example, $R^2$ is only slightly reduced by introducing the restriction, and the estimates (not surprisingly) coincide with those from Section 1.3. The overall model test and the test of **??** clearly reject their respective null hypothesis, while the data are compatible with validity of the restriction according to the test for **??** but do not prove strict validity of the inequality restriction (**??**).

The grades example has been about analysis of variance and has worked with aggregated data, which makes bootstrapping useless. The rest of this section uses the body fat data for illustrating functionality for order-restricted regression, including bootstrap confidence intervals:

```
R> orlimo.bodyfat <- orlm(limo.bodyfat,
+         ui = diag(1,3), boot = TRUE)
R> summary(orlimo.bodyfat)
```

```
Order-restricted linear model with restrictions of coefficients of
Triceps Thigh Midarm


 Inequality restrictions:
     Triceps Thigh Midarm
1:   1       0     0      %*%colnames  >=  0
2:   0       1     0      %*%colnames  >=  0
3: A 0       0     1      %*%colnames  >=  0

 Note: Restrictions marked with A are active.


Restricted model: R2 reduced from 0.8014 to 0.7781


Coefficients from order-restricted model
with 95 pct bootstrap confidence intervals( perc ):
              Coeff.   Lower    Upper
  (Intercept) -19.1742 -33.0032  -3.4302
R Triceps       0.2224   0.0000   1.0215
R Thigh         0.6594   0.0000   0.9577
R Midarm        0.0000   0.0000   0.3269

 Note: Coefficients marked with R are involved in restrictions.


Hypothesis tests ( 16 error degrees of freedom ):
Overall model test under the order restrictions:
       Test statistic: 0.7966,   p-value: <0.0001

Type 1 test: H0: all restrictions active(=)
         vs. H1: at least one restriction strictly true (>)
       Test statistic: 0.7966,   p-value: <0.0001
Type 2 test: H0: all restrictions true
         vs. H1: at least one restriction false
       Test statistic: 0.105,   p-value: 0.4100
Type 3 test: H0: at least one restriction false or active (=)
         vs. H1: all restrictions strictly true (>)
       Test statistic: -1.37,   p-value: 0.9052

Type 3 test based on t-distribution (one-sided),
all other tests based on mixture of beta distributions
```

Again, $R^2$ is not dramatically reduced, the overall test – in this case identical to the test for **??** – is clearly significant, while the other two tests do not reject their null hypothesis. While the unrestricted model had two negative estimated coefficients, the restricted model has one active restriction. The other previously negative coefficient has now been estimated to be positive. Note that still none of the individual coefficients is significantly different from 0, since all bootstrap confidence intervals include this boundary value.

### *Bootstrapping regression models*

Confidence intervals in **ic.infer** are obtained via the bootstrap. The implemented bootstrap is valid for uncorrelated observations only, since observations are independently sampled. When bootstrapping regression models, there are two principally different reasonable approaches (cf. e.g. Davison and Hinkley 1997; Fox 2002): The regressors can be considered fixed in some situations, e.g., for experimental data. In this case, only the error terms are random. Contrary, in observational studies, like e.g., customer satisfaction surveys, it makes far more sense to consider also the regressors as random, since the observations are a random sample from a larger population. These two scenarii prompt two different approaches for bootstrapping: For fixed regressors, bootstrapping is based on repeated sampling from the residuals of the regression model, while for random regressors, the complete observation rows – consisting of regressors and response – are resampled. **ic.infer** offers both possibilities, defaulting to random regressors (`fixed = FALSE`). Bootstrapping in **ic.infer** is implemented in function `orlm` based on the function `boot` from R package **boot**. Bootstrap confidence intervals are then calculated by the summary method for the output object from function `orlm`, relying on function `boot.ci` of package **boot**. Percentile intervals, BCa intervals, normal intervals and basic intervals are supported (default: percentile intervals). For further information on bootstrapping in general, cf. e.g., Davison and Hinkley (1997).

### *Overall tests*

As mentioned above and shown in the example output, the summary method for objects of class `orlm` calculates an overall model test, similar to the overall $F$ test in the unconstrained linear model, and several tests for or against the restrictions. These can be suppressed, because their calculation can be very time-consuming in case of large sets of restrictions.

If they are not suppressed, function `summary.orlm` calculates an overall test that all parameters except the intercept are 0 ($H_0$) vs. the restriction set (this is a test of type `TP = 11` or `TP = 1`, depending on whether or not the original restrictions refer to all parameters in the model). In addition, all tests for the three test problems **??** to **??** are calculated. (Test problem **??** is only applicable if there are no equality restrictions (i.e., `meq=0`).) Note that the time-consuming aspect is calculation of weights for the null distributions of test statistics. These are calculated only once and are then handed to function `ic.test` for the further tests. Nevertheless, calculation of weights for large problems takes a long time or is even impossible because of storage space restrictions.

It would be desirable to have a function for sequential testing of sources, analogous to `anova`, for order-restricted linear models. However, this would require the possibility to test a cone-shaped null hypothesis vs. a larger cone-shaped alternative hypothesis, which is far from trivial. So far, it has not been figured out how to implement such a test.

## 1.7. $R^2$ decomposition

It has been mentioned earlier that function `or.relimp` decomposes $R^2$ into contributions of individual regressors. The method is implemented by handing the $2^p$-vector of $R^2$ values for all sub-models to function `Shapley.value` from R package **kappalab** (Grabisch, Kojadinovic, and Meyer 2009). The result is illustrated for the body fat example:

```
R> or.relimp(limo.bodyfat, ui = diag(1, 3))

 Triceps    Thigh   Midarm
0.354115 0.416395 0.007542
```

Note that – in this example – although the coefficients are quite different from those of the unrestricted model, the $R^2$ decomposition is very similar (**relaimpo** must be loaded for the following calculation):

```
R> calc.relimp(limo.bodyfat)$lmg

Triceps   Thigh  Midarm
0.37439 0.39914 0.02782
```

So far, such similarity has been observed for all examples for which the restrictions employed were plausible and adequate.

It has been mentioned in Section 1.3 that automatic generation of restrictions for sub models is naturally done by deleting the respective columns from the restriction matrix ($R$ or `ui`, respectively). It is emphasized here once more that this is not adequate for all conceivable situations. *It is in the responsibility of the user to ensure that restrictions for sub models are sensible and meaningful.*

Decomposition of $R^2$ requires calculation of $2^p$ *constrained* estimates. This involves significantly higher computational burden than for the unconstrained case: For example, calculations on a 2.4GHz Dual Core Windows XP machine in `calc.relimp` took 0.5 seconds for 10 regressors, about 17 seconds for 15 regressors and about 580 seconds for 20 regressors. For the same scenarios, calculations in `or.relimp` with all non-intercept coefficients restricted to be non-negative took 2.5 seconds for 10 regressors, about 109 seconds for 15 regressors, and about 14800 seconds for 20 regressors. In case of fewer restrictions than regressors, computing time is somewhat reduced; for example, when restricting only 10 of the 15 coefficients in the 15 regressor situation, `or.relimp` computing time was about 90 seconds. Given that unconstrained models gave very similar $R^2$ decompositions in all reasonable applications that have so far been examined, decompositions from unconstrained models may very well be used at least as first checks.

# 2. Final remarks

Inequality-constrained inference and its implementation in R package **ic.infer** have been explained and illustrated in this article. While **ic.infer** offers the most important possibilities for normal means and linear models, some wishes remain to be fulfilled with future developments. These will be discussed below.

Within the linear model context, it would be desirable to implement some factor-related functionality for function `orlm`, supporting e.g., an overall test of significance for a factor as a whole or hypothesis tests corresponding to sequential analysis of variance (analogously to function `anova`). It has been mentioned before that these topics may prove difficult because they will often require testing a cone-shaped null hypothesis within a larger cone-shaped alternative. Their feasibility will be investigated, and even if not all situations can be covered, some may prove feasible (e.g., no restrictions on the factor, inequality restrictions on the factor but on nothing else, ...).

For non-linear models with asymptotically normal parameter estimates, users can apply inequality-restricted inference on the coefficients through functions `ic.est` or `ic.test`. A more direct approach would be desirable. It is intended to extend coverage of the package to (selected) non-normal situations with linear equality and inequality restrictions, for which it is known that the asymptotic distribution of the likelihood ratio test statistic is also a mixture of $\chi^2$ distributions (cf. section 4 of Silvapulle and Sen 2004). Of course, inference is local and less robust, if we leave the linear model.

Calculation of weights is a computational road block in case of many restrictions. It will be explored if direct calculation of weights using Monte-Carlo methods is more efficient than using Equation (**??**) together with package **mvtnorm**.

Function `or.relimp` is currently restricted to linear models without factors. It would be possible to include factors by grouping their dummies, like in **relaimpo**. Also, it might be possible to enable usage of `or.relimp` for larger problems than currently possible by a different programming approach – however, as long as no reasonable examples have been encountered for which constrained and unconstrained decompositions make a relevant difference, improvements on `or.relimp` have low priority.

## Acknowledgments

This vignette is based on Grömping (2010a). My colleague Frank Hausser helped overcome initial numerical problems. The package uses code by John Fox (functions `RREF` and `GaussianElimination`), Wolfgang Huber (function `nchoosek`), and Sinnwell and Schaid (2007, function `pchibar`). A referee made very useful comments that improved both the paper and the software.

## References

Davison A, Hinkley D (1997). *Bootstrap Methods and Their Application*. Cambridge University Press, Cambridge.

Fox J (2002). "Bootstrapping Regression Models." In *An R and S-PLUS Companion to Applied Regression: A Web Appendix to the Book*. Sage, Thousand Oaks, CA. URL http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-bootstrapping.pdf.

Grabisch M, Kojadinovic I, Meyer P (2009). ***kappalab**: Non-Additive Measure and Integral Manipulation Functions*. R package version 0.4-4, URL http://CRAN.R-project.org/package=kappalab.

Grömping U (2010a). "Inference with Linear Equality and Inequality Constraints Using R: The Package ic.infer." *Journal of Statistical Software*, **33**(10), 1–31.

Grömping U (2010b). ***ic.infer**: Inequality Constrained Inference in Linear Normal Situations.* R package version 1.1-3, URL http://CRAN.R-project.org/package=ic.infer.

Kutner M, Nachtsheim C, Neter J (2004). *Applied Linear Statistical Models*. 4th edition. McGraw-Hill, Boston.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Robertson T, Wright F, Dykstra R (1988). *Order-Restricted Inference*. John Wiley & Sons, New York.

Shapiro A (1988). "Towards a Unified Theory in Inequality-Constrained Testing in Multivariate Analysis." *International Statistical Review*, **56**, 49–62.

Silvapulle M, Sen P (2004). *Constrained Statistical Inference*. John Wiley & Sons, New York.

Sinnwell J, Schaid D (2007). ***ibdreg**: Regression Methods for IBD Linkage With Covariates.* R package version 0.1.1, URL http://CRAN.R-project.org/package=ibdreg.

### Affiliation:

Ulrike Grömping
Department II – Mathematics, Physics, Chemistry
BHT Berlin – University of Applied Sciences
D-13353 Berlin
E-mail: groemping@bht-berlin.de
URL: http://prof.tfh-berlin.de/groemping/