# Implementing a Class of Permutation Tests: The coin Package

**Torsten Hothorn**
Ludwig-Maximilians-Universität München

**Kurt Hornik**
Wirtschaftsuniversität Wien

**Mark A. van de Wiel**
Vrije Universiteit Amsterdam

**Achim Zeileis**
Wirtschaftsuniversität Wien

**Abstract**

The R package **coin** implements a unified approach to permutation tests providing a huge class of independence tests for nominal, ordered, numeric, and censored data as well as multivariate data at mixed scales. Based on a rich and flexible conceptual framework that embeds different permutation test procedures into a common theory, a computational framework is established in **coin** that likewise embeds the corresponding R functionality in a common S4 class structure with associated generic functions. As a consequence, the computational tools in **coin** inherit the flexibility of the underlying theory and conditional inference functions for important special cases can be set up easily. Conditional versions of classical tests—such as tests for location and scale problems in two or more samples, independence in two- or three-way contingency tables, or association problems for censored, ordered categorical or multivariate data—can be easily be implemented as special cases using this computational toolbox by choosing appropriate transformations of the observations. The paper gives a detailed exposition of both the internal structure of the package and the provided user interfaces.

*Keywords*: conditional inference, exact distribution, conditional Monte Carlo, categorical data analysis, R.

## 1. Introduction

Conditioning on all admissible permutations of the data for testing independence hypotheses is a very old, yet very powerful and popular, idea (Fisher 1935; Ernst 2004). Conditional inference procedures, or simply *permutation* or *re-randomization* tests, are implemented in many different statistical computing environments. These implementations, for example `wilcox.test()` for the Wilcoxon-Mann-Whitney test or `mantelhaen.test()` for the Cochran-Mantel-Haenszel $\chi^2$ test in the S language or the tools implemented in **StatXact** (Cytel Inc. 2003), **LogXact** (Cytel Inc. 2006), or Stata (StataCorp. 2003) (see Oster 2002, 2003, for an overview), all follow the classical classification scheme of inference procedures and offer procedures for location problems, scale problems, correlation, or nominal and ordered categorical data. Thus, each test procedure is implemented separately, maybe with the exception of conditional versions of linear rank statistics (Hájek, Šidák, and Sen 1999) in `NPAR1WAY` as available in SAS (SAS Institute Inc. 2003).

Novel theoretical insights by Strasser and Weber (1999) open up the way to a unified treatment of a huge class of permutation tests. The **coin** package for conditional inference is the computational counterpart to this theoretical framework, implemented in the R system for statistical computing (R Development Core Team 2007). Hothorn, Hornik, van de Wiel, and Zeileis (2006) introduce the package and illustrate the transition from theory to practice. Here, we focus on the design principles upon which the **coin** implementation is based as well as on the more technical issues

that need to be addressed in the implementation of such conceptual tools.

Formal S4 classes describe the data model and the conditional test procedures, consisting of multivariate linear statistics, univariate test statistics and a reference distribution. Generic functions for obtaining statistics, conditional expectation and covariance matrices as well as $p$ value, distribution, density and quantile functions for the reference distribution are available. The most important user-visible function is `independence_test()`, the computational counterpart of the theoretical framework of Strasser and Weber (1999), providing the same flexibility in software as in the underlying theory.

# 2. Theory and classes

We first focus on the conceptual framework for conditional inference procedures as proposed by Strasser and Weber (1999) along with the class structure upon which the **coin** package is based. Formal S4 classes defining data objects, objects describing the inference problem and conditional test procedure are introduced and explained.

We deal with variables $\mathbf{Y}$ and $\mathbf{X}$ from sample spaces $\mathcal{Y}$ and $\mathcal{X}$ which may be measured at arbitrary scales and may be multivariate as well. In addition, $b \in \{1, \ldots, k\}$, a factor measured at $k$ levels, indicates a certain block structure of the observations: for example study centers in a multi-center randomized clinical trial where only a re-randomization of observations within blocks is admissible. We are interested in testing the null hypothesis

$$H_0 : D(\mathbf{Y}|\mathbf{X}, b) = D(\mathbf{Y}|b)$$

of conditional independence of $\mathbf{Y}$ and $\mathbf{X}$ within blocks $b$ against arbitrary alternatives, for example shift or scale alternatives, linear trends, association in contingency tables etc.

## 2.1. Data

In the following we assume that we are provided with $n$ observations

$$(\mathbf{Y}_i, \mathbf{X}_i, b_i, w_i), \quad i = 1, \ldots, n.$$

In addition to variables $\mathbf{X}$, $\mathbf{Y}$, and $b$, it is convenient (for example to efficiently represent large contingency tables) to allow for some integer-valued case weights $w_i$, indicating that $w_i$ observations with realizations $\mathbf{Y}_i$, $\mathbf{X}_i$ and $b_i$ are available, with default $w_i \equiv 1$. This data structure is represented by class '`IndependenceProblem`':

Class '`IndependenceProblem`'

| Slot | Class |
|---|---|
| `x` | '`data.frame`' |
| `y` | '`data.frame`' |
| `weights` | '`numeric`' |
| `block` | '`factor`' |

## 2.2. Inference problem and linear statistic

Strasser and Weber (1999) suggest to derive scalar test statistics for testing $H_0$ from multivariate linear statistics of the form

$$\mathbf{T} = \sum_{j=1}^{k} \mathbf{T}_j \in \mathbb{R}^{pq} \tag{1}$$

where the linear statistic for each block is given by

$$\mathbf{T}_j = \text{vec}\left(\sum_{i=1}^{n} I(b_i = j) w_i g(\mathbf{X}_i) h(\mathbf{Y}_i)^\top\right) \in \mathbb{R}^{pq}.$$

The function $I(\cdot)$ is the indicator function and vec denotes the vec operator (which stacks the columns of a matrix one underneath the other). Here, $g : \mathcal{X} \to \mathbb{R}^{p \times 1}$ is a transformation of the $\mathbf{X}$ measurements and $h : \mathcal{Y} \to \mathbb{R}^{q \times 1}$ is called *influence function*. The function $h(\mathbf{Y}_i) = h(\mathbf{Y}_i, (\mathbf{Y}_1, \ldots, \mathbf{Y}_n))$ may depend on the full vector of responses $(\mathbf{Y}_1, \ldots, \mathbf{Y}_n)$, however only in a permutation symmetric way, i.e., the value of the function must not depend on the order in which $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ appear. The transformation $g$ and influence function $h$ as well as $g(\mathbf{X}_i)$ and $h(\mathbf{Y}_i), i = 1, \ldots, n$, are attached to the data structure by extending class '`IndependenceProblem`':

Class '`IndependenceTestProblem`'
Contains '`IndependenceProblem`'

| Slot | Class |
|---|---|
| `xtrans` | '`matrix`' |
| `ytrans` | '`matrix`' |
| `xtrafo` | '`function`' |
| `ytrafo` | '`function`' |

The `ytrafo` and `xtrafo` slots correspond to the influence function $h$ and transformation $g$, respectively. The $i$th row of the $n \times q$ matrix `ytrans` corresponds to $h(\mathbf{Y}_i)$. Similar, the rows of `xtrans` ($n \times p$) correspond to $g(\mathbf{X}_i)$.

In the simplest case of both $\mathbf{X}$ and $\mathbf{Y}$ being univariate factors at $p$ and $q$ levels, $g$ and $h$ are the corresponding dummy codings and the linear statistic $\mathbf{T}$ is the (vectorized) $p \times q$ contingency table of $\mathbf{X}$ and $\mathbf{Y}$.

## 2.3. Conditional expectation and covariance

The distribution of $\mathbf{T}$ depends on the joint distribution of $\mathbf{Y}$ and $\mathbf{X}$, which is unknown under almost all practical circumstances. At least under the null hypothesis one can dispose of this dependency by fixing $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and conditioning on all possible permutations $S_j$ of the responses $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ within block $j, j = 1, \ldots, k$.

The conditional expectation $\mu \in \mathbb{R}^{pq}$ and covariance $\Sigma \in \mathbb{R}^{pq \times pq}$ of $\mathbf{T}$ under $H_0$ given all permutations $\sigma \in S$ of the responses are derived by Strasser and Weber (1999) and are re-stated in the following.

Let $w_{\cdot j} = \sum_{i=1}^{n} I(b_i = j) w_i$ denote the sum of the weights in block $j$. The conditional expectation of the influence function $h$ in block $j$

$$\mathsf{E}(h|S_j) = w_{\cdot j}^{-1} \sum_i I(b_i = j) w_i h(\mathbf{Y}_i) \in \mathbb{R}^q$$

with corresponding $q \times q$ covariance matrix

$$\mathsf{COV}(h|S_j) = w_{\cdot j}^{-1} \sum_i I(b_i = j) w_i \left(h(\mathbf{Y}_i) - \mathsf{E}(h|S_j)\right) \left(h(\mathbf{Y}_i) - \mathsf{E}(h|S_j)\right)^\top.$$

The conditional expectation and covariance of the linear statistic $\mathbf{T}_j$ in block $j$ are

$$\mu_j \quad = \quad \mathsf{E}(\mathbf{T}_j|S_j) = \text{vec}\left(\left(\sum_{i=1}^{n} I(b_i = j) w_i g(\mathbf{X}_i)\right) \mathsf{E}(h|S_j)^\top\right)$$

and

$$\begin{aligned}
\Sigma_j &= \mathsf{COV}(\mathbf{T}_j|S_j) \\
&= \frac{w_{\cdot j}}{w_{\cdot j} - 1}\mathsf{COV}(h|S_j) \otimes \left(\sum_i I(b_i = j) w_i \left(g(\mathbf{X}_i) \otimes g(\mathbf{X}_i)^\top\right)\right) \\
&- \frac{1}{w_{\cdot j} - 1}\mathsf{COV}(h|S_j) \otimes \left(\sum_i I(b_i = j) w_i g(\mathbf{X}_i)\right) \otimes \left(\sum_i I(b_i = j) w_i g(\mathbf{X}_i)\right)^\top
\end{aligned}$$

respectively, where $\otimes$ is the Kronecker product. The conditional expectation and covariance of $\mathbf{T}$, aggregated over all $k$ blocks, are then given by

$$\begin{aligned}
\mathsf{E}(\mathbf{T}|S_j) &= \mu = \sum_{j=1}^{k} \mu_j = \sum_{j=1}^{k} \mathsf{E}(\mathbf{T}_j|S_j), \\
\mathsf{COV}(\mathbf{T}|S_j) &= \Sigma = \sum_{j=1}^{k} \Sigma_j = \sum_{j=1}^{k} \mathsf{COV}(\mathbf{T}_j|S_j).
\end{aligned}$$

The linear statistic $\mathbf{T}$, its conditional expectation $\mu$ and covariance $\Sigma$ are stored in objects of class '`IndependenceLinearStatistic`':

Class '`IndependenceLinearStatistic`'
Contains '`IndependenceTestProblem`'

| Slot | Class |
|---|---|
| `linearstatistic` | '`numeric`' |
| `expectation` | '`numeric`' |
| `covariance` | '`VarCovar`' |

Class '`VarCovar`' represents either a complete covariance matrix or its diagonal elements only.

## 2.4. Test statistics

Univariate test statistics $c$ mapping an observed linear statistic $\mathbf{t} \in \mathbb{R}^{pq}$ into the real line can be of arbitrary form. Having the conditional expectation and covariance at hand we are able to standardize an observed linear statistic $\mathbf{t} \in \mathbb{R}^{pq}$ of the form given in Equation 1 by

$$\mathbf{z} = \frac{\mathbf{t} - \mu}{\mathrm{diag}(\Sigma)^{1/2}}. \tag{2}$$

Test statistics are represented by class '`IndependenceTestStatistic`'

Class '`IndependenceTestStatistic`'
Contains '`IndependenceLinearStatistic`'

| Slot | Class |
|---|---|
| `estimates` | '`list`' |
| `teststatistic` | '`numeric`' |
| `standardizedlinearstatistic` | '`numeric`' |

The slot `standardizedlinearstatistic` contains $\mathbf{z}$, the (possibly multivariate) linear statistic standardized by its conditional expectation and variance (Equation 2). A univariate test statistic $c$ is stored in slot `teststatistic`. The `estimates` slot may contain parameter estimates where available, for example an estimate and corresponding confidence interval for a shift parameter derived from a conditional Wilcoxon-Mann-Whitney test.

In case of univariate linear statistics $\mathbf{t}$ (with $pq = 1$), the test statistic $c$ is simply the standardized linear statistic

$$c_{\text{scalar}}(\mathbf{t}, \mu, \Sigma) = \frac{\mathbf{t} - \mu}{\sqrt{\Sigma}}.$$

In the multivariate case ($pq > 1$), a maximum-type statistic of the form

$$c_{\max}(\mathbf{t}, \mu, \Sigma) = \max |\mathbf{z}|$$

is appropriate. This version for the two-sided situation is to be replaced by

$$\min(\mathbf{z}) \quad \text{(less) and} \quad \max(\mathbf{z}) \quad \text{(greater)}$$

in the one-sided case. The definition of one- and two-sided $p$ values used for the computations in the **coin** package is

$$\begin{array}{ll} \text{less:} & \mathsf{P}(c(\mathbf{T}, \mu, \Sigma) \leq c(\mathbf{t}, \mu, \Sigma)) \\ \text{greater:} & \mathsf{P}(c(\mathbf{T}, \mu, \Sigma) \geq c(\mathbf{t}, \mu, \Sigma)) \\ \text{two-sided:} & \mathsf{P}(|c(\mathbf{T}, \mu, \Sigma)| \leq |c(\mathbf{t}, \mu, \Sigma)|). \end{array}$$

For univariate statistics $c_{\text{scalar}}(\mathbf{t}, \mu, \Sigma)$ a special class

Class '`ScalarIndependenceTestStatistic`'
Contains '`IndependenceTestStatistic`'

| Slot | Class |
|---|---|
| `alternative` | '`character`' |

is available. For the more general case, maximum-type statistics are represented by objects of class '`MaxTypeIndependenceTestStatistic`':

Class '`MaxTypeIndependenceTestStatistic`'
Contains '`IndependenceTestStatistic`'

| Slot | Class |
|---|---|
| `alternative` | '`character`' |

both defining a character vector specifying the alternative to test against (`"two.sided"`, `"greater"` and `"less"`).

Alternatively, a quadratic form $c_{\text{quad}}(\mathbf{t}, \mu, \Sigma) = (\mathbf{t} - \mu)\Sigma^{+}(\mathbf{t} - \mu)^{\top}$ can be used as test statistic. It is computationally more expensive because the Moore-Penrose inverse $\Sigma^{+}$ of $\Sigma$ is involved. Such statistics are represented by objects of class '`QuadTypeIndependenceTestStatistic`' defining slots for $\Sigma^{+}$ and its rank (degrees of freedom):

Class '`QuadTypeIndependenceTestStatistic`'
Contains '`IndependenceTestStatistic`'

| Slot | Class |
|---|---|
| `covarianceplus` | '`matrix`' |
| `df` | '`numeric`' |

## 2.5. Conditional null distribution

The conditional distribution (or an approximation thereof) and thus the $p$ value corresponding to the statistic $c(\mathbf{t}, \mu, \Sigma)$ can be computed in several different ways. For some special forms of

the linear statistic, the exact distribution of the test statistic is tractable. For 2-sample problems, the shift-algorithm by Streitberg and Röhmel (1986, 1987) and the split-up algorithm by van de Wiel (2001) are implemented as part of the package. Conditional Monte-Carlo procedures can always be used to approximate the exact distribution. Strasser and Weber (1999, Theorem 2.3) proved that the conditional distribution of linear statistics $\mathbf{T}$ with conditional expectation $\mu$ and covariance $\Sigma$ tends to a multivariate normal distribution with parameters $\mu$ and $\Sigma$ as $w_{\cdot j} \to \infty$ for all $j = 1, \ldots, k$. Thus, the asymptotic conditional distribution of test statistics of the form $c_{\max}$ is normal and can be computed directly in the univariate case ($pq = 1$) or approximated by numerical algorithms (quasi-randomized Monte-Carlo procedures Genz 1992) in the multivariate setting. For quadratic forms $c_{\text{quad}}$ which follow a $\chi^2$ distribution with degrees of freedom given by the rank of $\Sigma$ (see Johnson and Kotz 1970, Chapter 29), precise asymptotic probabilities can be computed efficiently.

A null distribution is represented by either a distribution (and $p$ value) function only

Class '`PValue`'

| Slot | Class |
|---|---|
| `pvalue` | '`function`' |
| `p` | '`function`' |
| `name` | '`character`' |

or, where possible, is enriched by its density and quantile function:

Class '`NullDistribution`'
Contains '`PValue`'

| Slot | Class |
|---|---|
| `q` | '`function`' |
| `d` | '`function`' |
| `support` | '`function`' |
| `parameters` | '`list`' |

Currently, there are three classes extending '`NullDistribution`' (without defining additional slots at the moment): '`ExactNullDistribution`' (exact conditional null distribution, computed for example via the shift-algorithm), '`ApproxNullDistribution`' (approximations of the exact conditional distribution using conditional Monte-Carlo procedures) and '`AsymptNullDistribution`' (asymptotic approximations via multivariate normal or $\chi^2$ distribution). A new method for computing or approximating the conditional distribution can be implemented by defining a dedicated class (and corresponding methods) extending '`NullDistribution`'.

For maximum-type statistics $c_{\max}$, single-step and step-down multiplicity adjusted $p$ values based on the limiting distribution and conditional Monte-Carlo methods (see Westfall and Young 1993) are available as well.

## 2.6. Conditional tests

A conditional test is represented by a test statistic and its conditional null distribution (or an approximation thereof). In addition, a character string giving the name of the test procedure is defined in class '`IndependenceTest`':

Class '`IndependenceTest`'

| Slot | Class |
|---|---|
| `distribution` | '`NullDistribution`' |
| `statistic` | '`IndependenceTestStatistic`' |
| `method` | '`character`' |

# 3. Generic functions

Methods for the following generic functions are defined for class 'IndependenceTestStatistic':

`statistic()`: extracts the linear statistic $\mathbf{t}$, the standardized statistic $\mathbf{z}$ or test statistic $c(\mathbf{t}, \mu, \Sigma)$.

`expectation()`: extracts the conditional expectation $\mu$.

`covariance()`: extracts the complete conditional covariance matrix $\Sigma$ (if available).

`variance()`: extracts the diagonal elements of the conditional covariance matrix $\text{diag}(\Sigma)$.

For conditional null distributions (class 'NullDistribution'), the following methods are available:

`pvalue()`: computes the $p$ value (plus a confidence interval if Monte-Carlo procedures have been used) based on an observed test statistic $c$ and its conditional null distribution.

`pperm()`: evaluates the cumulative distribution function.

`dperm()`: evaluates the probability density function.

`qperm()`: evaluates the quantile function.

`support()`: returns the support of the null distribution.

Of course, all methods are defined for objects inheriting from class 'IndependenceTest' as well. In addition, `show()` methods are defined for classes 'ScalarIndependenceTest', 'MaxTypeIndependenceTest' and 'QuadTypeIndependenceTest', converting these S4 objects to an informal S3 object of class 'htest' for which a `print()` method is available that most R users are familiar with.

For the conditional versions of 2-sample linear rank statistics for location and scale parameters (Hájek *et al.* 1999), e.g., Wilcoxon-Mann-Whitney, normal scores and Ansari-Bradley tests, parameter estimates and confidence intervals based on the conditional distribution of the test statistics are implemented following the methods proposed by Bauer (1972). A `confint()` method is available for these special cases.

# 4. User interfaces

While the internal structures presented in the previous sections make use of the S4 class system, the user interface is written in S3 style to mimic the familiar user interfaces of the classical tests. The workhorse is the `independence_test()` method for 'IndependenceProblem' objects. In addition, corresponding 'formula' and 'table' methods provide the user with the same type of interfaces they are used to from base R. Existing methods for 'htest' objects are utilized for formatting output.

## 4.1. General independence tests

The S3 generic function `independence_test()` has a method for 'IndependenceProblem' objects (as defined in Section 2.1). Additional convenience interfaces for 'formula' and 'table' objects (in case both $\mathbf{X}$ and $\mathbf{Y}$ are univariate factors) are provided which simply set up an 'IndependenceProblem' and call the corresponding method. The latter is defined as

```
independence_test(object,
    teststat = c("max", "quad", "scalar"),
    distribution = c("asymptotic", "approximate", "exact"),
    alternative = c("two.sided", "less", "greater"),
    xtrafo = trafo, ytrafo = trafo, scores = NULL,
    check = NULL, ...)
```

Here, `object` describes the data and thus the null hypothesis. Argument `xtrafo` refers to the transformation $g$ and `ytrafo` to the influence function $h$ (function `trafo()` implements reasonable defaults, see below) defining the linear statistic and its conditional expectation and covariance. Three types of test statistics are hard-coded. The (approximation) of the null distribution to be used as reference distribution can be chosen either by a character string or by functions `exact()`, `approximate()` and `asymptotic()` which also take care of the correct specification of additional arguments, such as the number of permutations to draw randomly in a conditional Monte-Carlo procedure. This mechanism allows for user-supplied algorithms to compute or approximate the exact conditional distribution: A function taking an object inheriting from '`IndependenceTestStatistic`' and returning an object of class '`NullDistribution`' can be specified as `distribution()` argument to `independence_test()`.

By default, the identify transformation is used for both $g$ and $h$ in case of numeric variables $\mathbf{X}$ and $\mathbf{Y}$, respectively (function `id_trafo()`). Factors are dummy-encoded by a set of indicator variables (function `f_trafo()`) and censored variables are transformed to log rank scores (function `logrank_trafo()`):

```
trafo(data, numeric_trafo = id_trafo,
    factor_trafo = f_trafo, surv_trafo = logrank_trafo,
    var_trafo = NULL, block = NULL)
```

The framework is extensible by user-supplied transformations $g$ or influence functions $h$ specified as arguments to `trafo()`.

### 4.2. Methods for '`formula`' and '`table`' objects

A '`formula`' interface as well as a '`table`' method for `independence_test()` are available in addition. The left hand side variables of a formula are interpreted as $\mathbf{Y}$ variables (univariate or possibly multivariate), the right hand side is taken for $\mathbf{X}$ (univariate or multivariate as well). The blocking factor can specified after a vertical bar. So, for example, the formula

```
y1 + y2 ~ x1 + x2 | block
```

leads to a test of independence between two $\mathbf{Y}$ variables and two $\mathbf{X}$ variables (in case all variables are numeric the linear statistic is four-dimensional with $p = 2$ and $q = 2$) for each level in `block`. As usual, `data`, `weights` and `subset` arguments can be specified as well.

Two- and three-dimensional tables can be supplied to the '`table`' method of `independence_test()`. The third variable is then interpreted as block.

### 4.3. Conditional versions of classical tests

For a variety of classical tests (some of them already implemented in base package **stats**), their conditional counterpart is made easily accessible. Some of the most important procedures, such as the Wilcoxon-Mann-Whitney or Cochran-Mantel-Haenszel tests, can be obtained as listed in Table 1, just by setting the `xtrafo`, `ytrafo` and `teststat` arguments appropriately. Almost all special-purpose functionality implemented in packages **exactRankTests** (Hothorn 2001; Hothorn and Hornik 2002, 2006) and **maxstat** (Hothorn and Lausen 2002; Hothorn 2005) can conveniently be provided within the **coin** framework, so that both these packages will become deprecated in the future.

## 5. Internal functionality

The core functionality, i.e., a small set of functions computing the linear statistic $\mathbf{T}$ (both for the original and permuted data), the conditional expectation $\mu$ and conditional covariance matrix $\Sigma$, is coded in C. The shift and split-up algorithms (Streitberg and Röhmel 1986, 1987; van de Wiel 2001)

| Test | `xtrafo` $g$ | `ytrafo` $h$ | `teststat` $c$ |
|---|---|---|---|
| | Independent Samples | | |
| Wilcoxon-Mann-Whitney | `f_trafo()` | `rank()` | `"scalar"` |
| Normal quantiles | `f_trafo()` | `normal_trafo()` | `"scalar"` |
| Median | `f_trafo()` | `median_trafo()` | `"scalar"` |
| Ansari-Bradley | `f_trafo()` | `ansari_trafo()` | `"scalar"` |
| Log rank | `f_trafo()` | `logrank_trafo()` | `"quad"` |
| Kruskal-Wallis | `f_trafo()` | `rank()` | `"quad"` |
| Fligner | `f_trafo()` | `fligner_trafo()` | `"quad"` |
| Spearman | `rank()` | `rank()` | `"scalar"` |
| Cochran-Mantel-Haenszel | `f_trafo()` | `f_trafo()` | `"quad"` |
| Pearson's $\chi^2$ | `f_trafo()` | `f_trafo()` | `"quad"` |
| Cochran-Armitage / Linear Association | scores | any | `"scalar"` |
| $K$-sample permutation test | `f_trafo()` | any | any |
| Maximally selected statistics | `maxstat_trafo()` | any | `"max"` |
| | Dependent Samples | | |
| Friedman | `f_trafo()` | `rank()` | `"quad"` |
| Maxwell-Stuart | `f_trafo()` | `f_trafo()` | `"quad"` |
| Wilcoxon signed rank | `f_trafo()` | `rank()` | `"scalar"` |

Table 1: Convenience functions implementing the conditional counterparts of the most important classical tests.

for computing the exact null distribution in 2-sample problems with univariate response as well as conditional Monte-Carlo procedures for approximating the exact conditional null distribution are implemented in C as well. (In addition, some helper functions, e.g., the Kronecker product etc., are coded in C.) The complete C source code and its documentation can be accessed via

```
R> browseURL(system.file("documentation", "html", "index.html",
+                        package = "coin"))
```

The naming scheme of the C routines distinguishes between functions only called at C level (`C_`*foo*) and functions which can be called from R via the `.Call()` interface (`R_`*foo*). Such functions are available for most of the internal C functions to enable unit testing.

The **coin** package imports packages **mvtnorm** (Genz, Bretz, and Hothorn 2006) for the evaluation of the multivariate normal distribution and package **modeltools** (Hothorn, Leisch, and Zeileis 2007) for formula parsing.

## 6. An example

The job satisfaction data (Table 7.8, Agresti 2002), see Figure 1, is a three-dimensional '`table`' with variables `Income` and `Job.Satisfaction` according to `Gender` (labels slightly modified for convenience):

```
R> js <- jobsatisfaction
R> dimnames(js)[[2]] <- c("VeryDiss", "ModDiss", "ModSat", "VerySat")
R> ftable(Job.Satisfaction ~ Gender + Income, data = js)

                  Job.Satisfaction VeryDiss ModDiss ModSat VerySat
Gender Income
Female <5000                             1       3     11       2
       5000-15000                        2       3     17       3
       15000-25000                       0       1      8       5
       >25000                            0       2      4       2
Male   <5000                             1       1      2       1
       5000-15000                        0       3      5       1
       15000-25000                       0       0      7       3
       >25000                            0       1      9       6
```

Here, we focus on conditional tests for independence of income and job satisfaction. The conditional Cochran-Mantel-Haenszel test is based on a $c_{\text{quad}}$ statistic derived from the contingency table and a $\chi^2$ approximation of the null distribution is utilized traditionally:

```
R> it <- independence_test(js, teststat = "quad", distribution = asymptotic())
R> it

        Asymptotic General Independence Test

data:  Job.Satisfaction by
         Income (<5000, 5000-15000, 15000-25000, >25000)
         stratified by Gender
chi-squared = 10.2001, df = 9, p-value = 0.3345
```

with linear statistic

```
R> statistic(it, "linear")
```

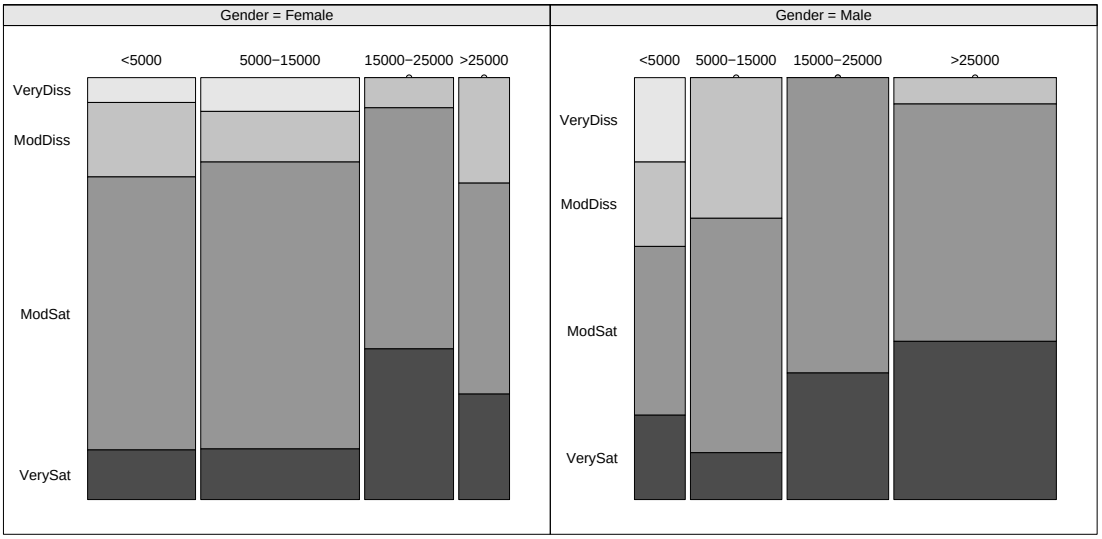


Figure 1: Conditional mosaic plot of job satisfaction and incoming given gender.

```
            VeryDiss ModDiss ModSat VerySat
<5000               2       4     13       3
5000-15000          2       6     22       4
15000-25000         0       1     15       8
>25000              0       3     13       8
```

This is exactly the two-way classification

```
R> margin.table(js, 1:2)
```

```
             Job.Satisfaction
Income        VeryDiss ModDiss ModSat VerySat
  <5000              2       4     13       3
  5000-15000         2       6     22       4
  15000-25000        0       1     15       8
  >25000             0       3     13       8
```

i.e., the three-dimensional table aggregated over the block factor `Gender`. This contingency table in standardized form reads

```
R> statistic(it, "standardized")
```

```
               VeryDiss     ModDiss     ModSat    VerySat
<5000         1.3112789  0.69201053 -0.2478705 -0.9293458
5000-15000    0.6481783  0.83462550  0.5175755 -1.6257547
15000-25000 -1.0958361 -1.50130926  0.2361231  1.4614123
>25000      -1.0377629 -0.08983052 -0.5946119  1.2031648
```

Instead of using a $\chi^2$ statistic collapsing the whole table via a quadratic form, one might want to use the maximum of the absolute values of the standardized cells as test statistic. This maximum-type test is set up easily:

```
R> independence_test(js, teststat = "max")

        Asymptotic General Independence Test

data:  Job.Satisfaction by
         Income (<5000, 5000-15000, 15000-25000, >25000)
         stratified by Gender
maxT = 1.6258, p-value = 0.7214
```

with its conditional asymptotical null distribution being available immediately (due to the joint multivariate normal distribution for the contingency table $\mathbf{T}$). Single-step adjusted $p$ values for each cell of the contingency table corresponding to this maximum test can be computed via

```
R> pvalue(independence_test(js, teststat = "max"),
+        method = "single-step")

              VeryDiss   ModDiss    ModSat   VerySat
<5000        0.9009726 0.9987769 0.9999998 0.9888039
5000-15000   0.9992718 0.9948674 0.9998852 0.7214172
15000-25000 0.9660244 0.8034652 0.9999999 0.8268942
>25000       0.9761806 1.0000000 0.9996381 0.9394831
```

Taking the ordinal scale level of both variables into account, a linear by linear association test (Agresti 2002) is easily performed

```
R> it <- independence_test(js,
+     scores = list(Job.Satisfaction = c(1, 3, 4, 5),
+                   Income = c(3, 10, 20, 35)),
+     distribution = approximate(B = 10000))
R> pvalue(it)

[1] 0.0107
99 percent confidence interval:
 0.008233232 0.013643997
```

For more practical examples, including applications with numeric variables, we refer to Hothorn *et al.* (2006).

## 7. Quality assurance

The test procedures implemented in package **coin** are continuously checked against results obtained by the corresponding implementations in package **stats** (where available). In addition, the test statistics and exact, approximate and asymptotic $p$ values for data examples given in the **StatXact** 6 user manual (Cytel Inc. 2003) are compared with the results reported there. Step-down multiple adjusted $p$ values have been checked against results reported by `mt.maxT()` from package **multtest** (Pollard, Ge, and Dudoit 2005). For details on the test procedures we refer to the R transcript files in directory `coin/tests` of the **coin** package sources.

## 8. Computational details

The class structure, internal functionality, user interface and examples are based on **coin** version 0.6-7, available under the terms of the General Public License from http://CRAN.R-project.org/. R version 2.6.0 was used for the computations, see http://www.R-project.org/.

## Acknowledgments

We would like to thank Helmut Strasser for discussions on the theoretical framework. Henric Nilsson provided clarification and examples for the Maxwell-Stuart test and helped identifying bugs. The work of T. Hothorn was supported by Deutsche Forschungsgemeinschaft (DFG) under grant HO 3242/1-3.

## References

Agresti A (2002). *Categorical Data Analysis*. John Wiley & Sons, Hoboken, New Jersey, 2nd edition.

Bauer DF (1972). "Constructing Confidence Sets Using Rank Statistics." *Journal of the American Statistical Association*, **67**(339), 687–690.

Cytel Inc (2003). **StatXact** *6: Statistical Software for Exact Nonparametric Inference*. Cytel Software Corporation, Cambridge, MA. URL http://www.cytel.com/.

Cytel Inc (2006). **LogXact** *8: Discrete Regression Software Featuring Exact Methods*. Cytel Software Corporation, Cambridge, MA. URL http://www.cytel.com/.

Ernst MD (2004). "Permutation Methods: A Basis for Exact Inference." *Statistical Science*, **19**(4), 676–685.

Fisher RA (1935). *The Design of Experiments*. Oliver and Boyd, Edinburgh, UK.

Genz A (1992). "Numerical Computation of Multivariate Normal Probabilities." *Journal of Computational and Graphical Statistics*, **1**, 141–149.

Genz A, Bretz F, Hothorn T (2006). **mvtnorm***: Multivariate Normal and T Distribution*. R package version 0.7-5, URL http://CRAN.R-project.org/.

Hájek J, Šidák Z, Sen PK (1999). *Theory of Rank Tests*. Academic Press, London, 2nd edition.

Hothorn T (2001). "On Exact Rank Tests in R." *R News*, **1**(1), 11–12. ISSN 1609–3631. URL http://CRAN.R-project.org/doc/Rnews/.

Hothorn T (2005). **maxstat***: Maximally Selected Rank Statistics*. R package version 0.7-9, URL http://CRAN.R-project.org/.

Hothorn T, Hornik K (2002). "Exact Nonparametric Inference in R." In W Härdle, B Rönz (eds.), "Proceedings in Computational Statistics: COMPSTAT 2002," pp. 355–360. Physica-Verlag, Heidelberg.

Hothorn T, Hornik K (2006). **exactRankTests***: Exact Distributions for Rank and Permutation Tests*. R package version 0.8-15, URL http://CRAN.R-project.org/.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2006). "A Lego System for Conditional Inference." *The American Statistician*, **60**(3), 257–263. doi:10.1198/000313006X118430.

Hothorn T, Lausen B (2002). "Maximally Selected Rank Statistics in R." *R News*, **2**(1), 3–5. ISSN 1609–3631. URL http://CRAN.R-project.org/doc/Rnews/.

Hothorn T, Leisch F, Zeileis A (2007). **modeltools***: Tools and Classes for Statistical Models*. R package version 0.2-12, URL http://CRAN.R-project.org/.

Johnson NL, Kotz S (1970). *Distributions in Statistics: Continuous Univariate Distributions 2*. John Wiley & Sons, New York.

Oster RA (2002). "An Examination of Statistical Software Packages for Categorical Data Analysis Using Exact Methods." *The American Statistician*, **56**(3), 235–246.

Oster RA (2003). "An Examination of Statistical Software Packages for Categorical Data Analysis Using Exact Methods–Part II." *The American Statistician*, **57**(3), 201–213.

Pollard KS, Ge Y, Dudoit S (2005). **multtest***: Resampling-based Multiple Hypothesis Testing.* R package version 1.8.0, URL http://CRAN.R-project.org/.

R Development Core Team (2007). *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/.

SAS Institute Inc (2003). *SAS/STAT Software, Version 9.1.* Cary, NC. URL http://www.sas.com/.

StataCorp (2003). *Stata Statistical Software: Release 8.* StataCorp LP, College Station, TX. URL http://www.stata.com/.

Strasser H, Weber C (1999). "On the Asymptotic Theory of Permutation Statistics." *Mathematical Methods of Statistics*, **8**, 220–250. Preprint available from http://epub.wu-wien.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_94c.

Streitberg B, Röhmel J (1986). "Exact Distributions for Permutation and Rank Tests: An Introduction to Some Recently Published Algorithms." *Statistical Software Newsletter*, **12**(1), 10–17. ISSN 1609-3631.

Streitberg B, Röhmel J (1987). "Exakte Verteilungen für Rang- und Randomisierungstests im allgemeinen $c$-Stichprobenfall." *EDV in Medizin und Biologie*, **18**(1), 12–19.

van de Wiel MA (2001). "The Split-up Algorithm: A Fast Symbolic Method for Computing $p$ Values of Rank Statistics." *Computational Statistics*, **16**, 519–538.

Westfall PH, Young SS (1993). *Resampling-based Multiple Testing.* John Wiley & Sons, New York.