Extending a Lego package.
An extension of the coin package for comparing interventions assigned by dynamic allocation

Johannes Hüsing
Coordination Center for Clinical Trials, Heidelberg Medical School

2008-08-13
Outline

1. Sequential Balancing Designs
2. Re-randomization extension
3. Building blocks of dynamic allocation
4. Result
5. Discussion
History on Randomization

- Randomization initially had to find its place among systematic designs.
- Paul Martini, a forerunner of clinical trials in Germany, used alternation in an early form of cross-over trials (Shelley & Baur, 1999).
- In clinical trials, randomization had found its place at the top.
- Nevertheless, random allocation sometimes leads to unbalanced distribution between treatment arms.
Some Discussion of Randomized Designs

Those who habitually use random assignments may have thought that the issue was finally settled in favour of randomization, but the recent recrudescence of the dispute . . . make[s] it clear that there is still a considerable body of opinion which favour such arrangements.  

Yates (1938)

But in either case, as soon as the experiment is designed, [experimental conditions] are definitely known . . . , and not random any longer.  

Jeffreys (1938)
New Discussion, new Variants

- biased coin (not coin!) Efron (1971)
- minimization Taves (1974)
- (optimal stratification) Sedransk (1974)
- adaptive allocation Pocock & Simon (1975)
- dynamic balanced randomization Signorini (1993)
- generalization to continuous covariates incorporating Kullback-Leibler loss
New Discussion, new Variants

- biased coin (not coin!) Efron (1971)
- minimization Taves (1974)
- (optimal stratification) Sedransk (1974)
- adaptive allocation Pocock & Simon (1975)
- dynamic balanced randomization Signorini (1993)
- generalization to continuous covariates incorporating Kullback-Leibler loss

Idea: achieve balance
Dynamic allocation is strongly discouraged. However, if it is used, then it is imperative that all factors used in the allocation scheme be included as covariates in the analysis. Even with this requirement, it remains controversial whether the analysis adequately reflect the randomisation scheme.

EMEA (2003)
Outline

1. Sequential Balancing Designs
2. Re-randomization extension
3. Building blocks of dynamic allocation
4. Result
5. Discussion
Reflect the Randomization Scheme

Can be accomplished by re-randomization methods (Simon 1979).
Idea: Repeat the same algorithm over original sequence of patients.
Obtain null distribution, define critical region and determine if test statistic is in it.

This idea is very similar to the permutation test.
Freeloading on an existing interface

The package `coin` offers a common interface to permutation test methods. Choice between:

1. exact,
2. asymptotic (*Strasser and Weber* 1999),
3. approximate (Monte Carlo) methods.
For re-randomization methods, Monte Carlo only choice.
Classes in the `coin` package

Want to mess with the approximate null distribution. **ApproxNullDistribution** is a method on **ScalarIndependenceTestStatistic**. Need algorithm for reallocation and possibly a shuffling algorithm.
Classes in the coin package

- IndependenceProblem
  - x
  - y
  - data.frame
  - weights
  - numeric
  - block
  - factor

- IndependenceTestProblem
  - xtrans
  - matrix
  - ytrans
  - matrix
  - xtrafo
  - function
  - ytrafo
  - function

- Variance
  - variance
  - numeric

- CovarianceMatrix
  - covariance
  - matrix

- IndependenceLinearStatistic
  - linearstatistic
  - numeric
  - expectation
  - numeric
  - covariance
  - VarCovar

- IndependenceTestStatistic
  - teststatistic
  - numeric
  - standardizedlinearstatistic
  - numeric
  - estimates
  - list

- IndependenceTest
  - distribution
  - statistic
  - method

- NullDistribution
  - IndependenceTestStatistic
    - character

- MaxTypeIndependenceTestStatistic
  - alternative
  - character

- QuadTypeIndependenceTestStatistic
  - covarianceplus
  - matrix
  - df

- ScalarIndependenceTestStatistic
  - alternative
  - character

- ApproxNullDistribution

- ExactNullDistribution

- AsymptNullDistribution

- NullDistribution
  - q
  - d
  - support
  - parameters

- PValue
  - pvalue
  - function
  - name
  - character

- MaxTypeIndependenceTest

- SymmetryProblem
Messing with the **coin** package

---

**IndependenceProblem**

- **x**: data.frame
- **y**: data.frame
- **weights**: numeric
- **block**: factor

**IndependenceTestProblem**

- **xtrans**: matrix
- **ytrans**: matrix
- **xtrafo**: function
- **ytrafo**: function

**IndependenceLinearStatistic**

- **linearstatistic**: numeric
- **expectation**: numeric
- **covariance**: Scalar

**IndependenceTestStatistic**

- **estimates**: list
- **teststatistic**: numeric
- **standardizedlinearstatistic**: numeric

**ScalarIndependenceTest**

- **nullvalue**: numeric

**ScalarIndependenceTestConfint**

- **confint**: function
- **conf.level**: numeric

**MaxTypeIndependenceTest**

- **alternative**: character

**QuadTypeIndependenceTest**

- **covarianceplus**: matrix
- **df**: numeric

**PValue**

- **p**: function
- **name**: character

**VarCovar**

- **Variance**: numeric
- **covariance**: numeric

**CovarianceMatrix**

- **matrix**: matrix

**SymmetryProblem**

**IndependenceAllocProblem**

- **algorithm**: function
- **shuffle**: function

**IndependenceAllocTestProblem**

- **IndependenceAllocLinearStatistic**: function
- **IndependenceAllocTestStatistic**: function

**ScalarIndependenceAllocTestStatistic**

**MaxIndependenceAllocTestStatistic**

**QuadIndependenceAllocTestStatistic**
Messing with the coin package

IndependenceProblem

x
data.frame
y
data.frame
weights
numeric
block
factor

IndependenceTestProblem

xtrans
matrix
ytrans
matrix
xtrafo
function
ytrafo
function

IndependenceLinearStatistic

linearstatistic
numeric
expectation
numeric
covariance
VarCovar

IndependenceAllocTestProblem

algorithm
shuffle
function
function

ScalarIndependenceTestStatistic

alternative
character
standardizedlinearstatistic
numeric

MaxTypeIndependenceTestStatistic

covarianceplus
matrix
df
numeric

QuadTypeIndependenceTestStatistic

PValue

pvalue
numeric
p
numeric
name
character

NullDistribution

q
function
d
function
support
function
parameters
list

AsymptNullDistribution

ApproxNullDistribution

ExactNullDistribution

IndependenceTest

distribution
NullDistribution

statistic
IndependenceTestStatistic

method
character

ScalarIndependenceAllocTestStatistic

nullvalue
numeric

MaxIndependenceAllocTestStatistic

QuadIndependenceAllocTestStatistic

SymmetryProblem

IndependenceAllocProblem

algorithm
shuffle
function

IndependenceAllocLinearStatistic

IndependenceAllocTestProblem

xtrafo
function
ytrafo
function

ScalarIndependenceAllocTestStatistic

MaxIndependenceAllocTestStatistic

QuadIndependenceAllocTestStatistic
Messing with the coin package

**IndependenceProblem**
- `x`: data.frame
- `y`: data.frame
- `weights`: numeric
- `block`: factor

**IndependenceTestProblem**
- `xtrans`: matrix
- `ytrans`: matrix
- `xtrafo`: function
- `ytrafo`: function

**IndependenceTestStatistic**
- `estimates`: list
- `teststatistic`: numeric
- `standardizedlinearstatistic`: numeric

**IndependenceLinearStatistic**
- `linearstatistic`: numeric
- `expectation`: numeric
- `covariance`: VarCovar

**IndependenceTestStatistic**
- `estimates`: list
- `teststatistic`: numeric
- `standardizedlinearstatistic`: numeric

**ScalarIndependenceTestStatistic**
- `alternative`: character

**MaxTypeIndependenceTestStatistic**
- `alternative`: character

**QuadTypeIndependenceTestStatistic**
- `covarianceplus`: matrix
- `df`: numeric

**PValue**
- `pvalue`: function
- `p`: function
- `name`: character

**NullDistribution**
- `q`: function
- `d`: function
- `support`: function
- `parameters`: list

**ApproxNullDistribution**

**ExactNullDistribution**

**IndependenceAllocTestProblem**
- `xtrans`: matrix
- `ytrans`: matrix
- `algorithm`: function

**IndependenceAllocLinearStatistic**
- `nullvalue`: numeric

**IndependenceAllocTestStatistic**

**ScalarIndependenceAllocTestStatistic**

**MaxIndependenceAllocTestStatistic**

**QuadIndependenceAllocTestStatistic**
Messing with the *coin* package

[Diagram of the *coin* package structure]
Messing with the `coin` package
Outline

1. Sequential Balancing Designs
2. Re-randomization extension
3. Building blocks of dynamic allocation
4. Result
5. Discussion
More Lego bricks: goodarms

- Takes the data frame of predictors and allocations up to now
- Returns treatment arms with the least loss

\[
\text{goodarms} \rightarrow \text{lossfun()} \rightarrow \text{arms with smallest loss}
\]
More Lego pieces: sumloss

sumloss

- Takes the data frame of predictors and list of loss functions
- Returns sum of loss functions applied to data frame

\[
\sum_{p} \text{contribloss}(dfr)
\]

arms with smallest loss
go
goodarms()
More Lego pieces: contribloss

- contribloss
  - Takes the data frame of predictors, factor and loss function on a contingency table
  - Returns loss function applied to the data frame

sumloss()

goodarms()

contribloss

\[ \sum^p \text{contribloss}(dfr) \]

factor, dfr, lossfun(table)

\text{lossfun(table vs treatment in dfr)}
More Lego pieces: contribloss

contribloss

- Takes the data frame of predictors, factor and loss function on a contingency table
- Returns loss function applied to the data frame
Example: Minimize $X^2$ loss over all marginal tables

$$X2\text{loss} \leftarrow \text{function}(\text{conttab}) \text{ prop. test}(\text{conttab}+1)$$ $\text{statistic}$

$$X2\text{alg} \leftarrow \text{function}(\text{preddf}, x\text{prev}=\text{NULL}, \text{factors}=\text{NULL}) \{$$

- $\text{current} \leftarrow \text{length}(x\text{prev})$
- $\text{dataframe} \leftarrow \text{setupec}(\text{preddf}, \text{levels}(x\text{prev}))$
- if (current == 0) $\text{dataframe}\$treatment[1] \leftarrow \text{sample( levels}(\text{dataframe}$treatment), 1)
- else {
  - $\text{dataframe}\$treatment[1:current] \leftarrow x\text{prev}$
  - $\text{ga} \leftarrow \text{goodarms}(\text{dataframe}, \text{current},$
    - function (dfr, crt) {
      - $\text{sumloss}(\text{mapply}(\text{function}(\text{fac})$
        - $\text{contribloss}(X2\text{loss}, \text{fac}, \text{dataframe})$,\n          - factors ),
            - dfr, crt))$}
  - $\text{dataframe}\$treatment[current + 1] \leftarrow \text{sample}(\text{ga}, 1)$
}
- $\text{dataframe}$
}
Usage of \texttt{X2alg} in \texttt{coin} package

For instance, perform variation of Mann-Whitney U test on trial data.

\begin{verbatim}
  independence_test(outcome ~ treatment, data = trial,
                   teststat = "scalar", ytrafo = rank,
                   algorithm = X2alg, preddf = stratvars)
\end{verbatim}
Outline

1 Sequential Balancing Designs
2 Re-randomization extension
3 Building blocks of dynamic allocation
4 Result
5 Discussion
Re-analysis of clinical trial on Graves’ disease

- Effect of levothyroxine on relapse of Graves’ disease
- Relapse rates to be similar in the levothyroxine group (20% after 1 year) and the randomized controls (18%) ($p = 0.67$)
- If all (seven, including site!) factors are included in the model, no convergence.
- No effect with re-randomisation analysis ($p = 0.57$)
- Standard error of Wald statistic of hazard ratio seems to be considerably lower than 1
Effect of levothyroxine on relapse of Graves’ disease

- Relapse rates to be similar in the levothyroxine group (20% after 1 year) and the randomized controls (18%) ($p = 0.67$)
- If all (seven, including site!) factors are included in the model, no convergence.
- No effect with re-randomisation analysis ($p = 0.57$)
- Standard error of Wald statistic of hazard ratio seems to be considerably lower than 1
Collaboration with Randi2 team

- Project out of Medical Informatic thesis
- Online randomisation tool in Java
- R integration to interface with algorithms is discussed
The package is mostly vapour

This is work in progress. As of now, two different packages:

- dynamic allocation
  (including Wei’s urn model, biased coin, big stick)
- extension of coin
Outline

1. Sequential Balancing Designs
2. Re-randomization extension
3. Building blocks of dynamic allocation
4. Result
5. Discussion
Computing speed issues

- Focus was on clean rather than speedy implementation
- Heavy usage of data frames seems costly
- Usage of dreaded `for` loops seems unavoidable
Versatility of R

The object system and the possibility to pass functions as arguments allow the users to achieve big structures out of small bricks.

Thanks to:

- Herbert Hirche for using and advocating dynamic allocation.
- Rudolf Hörmann for letting me use his data.
- Ulla Roggenbuck for transferring and explaining the data.
- Elmar Hüsing for letting me use his bricks.