randPack overview

VJ Carey, R Gentleman

March 30, 2012

1 Introduction

This package supports tasks of implementing and analyzing clinical trials. Various approaches to randomization are offered.

2 Classes

2.1 High-level containers for trials

Two key organizing classes are defined to manage clinical trials and associated clinical experiments.

```
> library(randPack)
> getClass("ClinicalTrial")

Class "ClinicalTrial" [package "randPack"]

Slots:

Name:         Experiment        PatientData        Randomizers
Class: ClinicalExperiment        environment               list

> getClass("ClinicalExperiment")

Class "ClinicalExperiment" [package "randPack"]

Slots:

Name:            name        factors    treatments randomization        strataFun
Class:      character           list       integer          list         function

Name:      patientIDs
Class:      PatientID
```

```
> getClass("PatientID")

Class "PatientID" [package "randPack"]

Slots:

Name:     strata      start      stop
Class: character    integer    integer
```

Use of these administrative containers is illustrated here.
First the set of strata and patient IDs is defined.

```
> pIDs = new("PatientID",
+           strata = c("Center1", "Center2"),
+           start = c(1000L, 2000L),
+           stop = c(1150L, 2150L)
+ )
> validPID(pIDs)

[1] TRUE
```

Now we create a 'vacuous' randomized experiment because we do not specify randomization methods for strata. These will be introduced later. The labeling and relative frequencies of assignments are created first.

```
> trts = c( A = 3L, B = 4L, C = 1L)
> CEvac = new("ClinicalExperiment",
+    name="My first experiment",
+    treatments = trts,
+    factors = list( F1 = c("A", "B", "C"), F2 = c("t1", "t2")),
+    strataFun = function(pDesc) pDesc@strata,
+    #randomization = list(Center1= list(pbdesc), Center2=list(rd)),
+    #randomization = list(Center1= list(pbdesc), Center2=list(pbdesc)),
+    randomization = list(Center1= list(a=1), Center2=list(a=1)),
+    patientIDs = pIDs
+ )
> CEvac

ClinicalExperiment:  My first experiment
  With 3 treatments
        A B C
  With 2 strata
        Center1 Center2
```

```
> treatmentFactors(CEvac)

$F1
[1] "A" "B" "C"

$F2
[1] "t1" "t2"

> factorNames(CEvac)

[1] "F1" "F2"

> randPack:::numberOfFactorLevels(CEvac)

F1 F2
 3  2
```

We create a trial object on the basis of an experiment specification.

```
> CTvac = createTrial(CEvac, seed=c(301, 401))
```

2.2   Classes for randomization computations

```
> getClass("RandomizerDesc")

Virtual Class "RandomizerDesc" [package "randPack"]

Slots:

Name:   treatments       type
Class:    integer   character

Known Subclasses: "MinimizationDesc", "PermutedBlockDesc", "RandomDesc", "UrnDesc",
"EfronBiasedCoinDesc"

> getClass("Randomizer")

Virtual Class "Randomizer" [package "randPack"]

Slots:

Name:            name treatmentTable stateVariables
Class:      character          integer    environment

Known Subclasses: "Random", "PermutedBlock", "Urn", "Minimization", "EfronBiasedCoin"
```

```
> getClass("PermutedBlockDesc")

Class "PermutedBlockDesc" [package "randPack"]

Slots:

Name:    numBlocks treatments        type
Class:     integer     integer   character

Extends: "RandomizerDesc"

> getClass("PermutedBlock")

Class "PermutedBlock" [package "randPack"]

Slots:

Name:            name treatmentTable stateVariables
Class:      character          integer    environment

Extends: "Randomizer"
```

2.2.1 Permuted blocks

Now we describe a permuted blocks randomization description based on this treatment regimen set.

```
> pbdesc = new("PermutedBlockDesc", treatments = trts, type="PermutedBlock",
+      numBlocks=4L)
```

A hidden function .newPBlock will create a vector of allocations:

```
>    ##should be 24 (no, 32) long and have one C ever 8 allocs
> table(dempb <- randPack:::.newPBlock(pbdesc))

 A  B  C
12 16  4

> bls = rep(1:4, each=8)
> sapply(split(dempb,bls), function(x) sum(x=="C"))

1 2 3 4
1 1 1 1
```

2.2.2 Pure randomization

The .newRandom function will create a collection of unconstrained randomizations.

```
> rd = new("RandomDesc", treatments = trts, type = "Random", numPatients = 32L)
> mmpb = makeRandomizer("Expt1", pbdesc, seed = 101)
> mmr = makeRandomizer("Expr1r", rd, seed=201)
> demrd = randPack:::.newRandom(rd)
> table(demrd)

demrd
 A  B  C
10 17  5
```

3   Creating the components of a trial

First we can create a treatment regimen set, a named vector.

```
> trts = c( A = 3L, B = 4L, C = 1L)
```

Now we use the randomization description objects in the randomization slot to create a real clinical experiment representation.

```
>  CE1 = new("ClinicalExperiment",
+     name="My first experiment",
+     treatments = trts,
+     factors = list( F1 = c("A", "B", "C"), F2 = c("t1", "t2")),
+     strataFun = function(pDesc) pDesc@strata,
+     #randomization = list(Center1= list(pbdesc), Center2=list(rd)),
+     randomization = list(Center1= list(pbdesc), Center2=list(pbdesc)),
+     patientIDs = pIDs
+  )
> CE1

ClinicalExperiment:  My first experiment
  With 3 treatments
        A B C
  With 2 strata
        Center1 Center2
```

The associated trial is:

```
>  CT1 = createTrial(CE1, seed=c(301, 401))
```

Here is how we can create 4 patient data objects for randomization.

```
> pD1 = new("PatientData", name="Sally H", date=Sys.Date(),
+ covariates=list(sex="F", age=33), strata="Center1")
> pD2 = new("PatientData", name="Sally Z", date=Sys.Date(),
+ covariates=list(sex="F", age=34), strata="Center1")
> pD3 = new("PatientData", name="Tom Z", date=Sys.Date(),
+ covariates=list(sex="M", age=44), strata="Center2")
> pD4 = new("PatientData", name="Jack Z", date=Sys.Date(),
+ covariates=list(sex="M", age=54), strata="Center2")
```

Treatment codes are obtained as follows:

```
> trt1 = getTreatment(CT1, pD1)
> trt2 = getTreatment(CT1, pD2)
> trt3 = getTreatment(CT1, pD3)
> trt4 = getTreatment(CT1, pD4)
```

We can get covariate plus allocation information using:

```
> getEnrolleeInfo(CT1)

$Center1
      name sex age alloc
1 Sally H   F  33     A
2 Sally Z   F  34     B

$Center2
     name sex age alloc
1  Tom Z   M  44     B
2 Jack Z   M  54     A
```

4   Illustrating the use of other randomizers

4.1   Efron's biased coin

This procedure requires a probability $p \in (0.5, 1.0)$ that specifies the probability that a biased coin falls in such a way that treatment A is selected.

```
> trts = c( A = 1L, B= 1L)
> bcdesc = new("EfronBiasedCoinDesc", treatments = trts, type="EfronBiasedCoin",
+      numPatients=1000L, p=2/3)
> CE1@randomization = list(Center1= list(bcdesc), Center2=list(bcdesc))
> CT2 = createTrial(CE1, seed=c(301, 401))
> btrt1 = getTreatment(CT2, pD1)
```

```
> btrt2 = getTreatment(CT2, pD2)
> btrt3 = getTreatment(CT2, pD3)
> btrt4 = getTreatment(CT2, pD4)
> c(btrt1, btrt2, btrt3, btrt4)

[1] "B" "A" "B" "B"
```

4.2   Wei's urn design

This procedure requires parameters α and β nonnegative integers indicating how balls labeled A and B are added to an urn as allocations are made by drawing from the urn. Initially α balls of both types are present, and when a patient is randomized, they are assigned the treatment corresponding to the label of the ball drawn, which is then replaced. If the ball drawn was of type A, then β B balls are added. If the ball drawn was of type B, then α A balls are added.

```
> urndesc = new("UrnDesc", treatments = trts, type="Urn",
+      numPatients=1000L, alpha=1, beta=3)
> CE1@randomization = list(Center1= list(urndesc), Center2=list(urndesc))
> CT3 = createTrial(CE1, seed=c(301, 401))
> utrt1 = getTreatment(CT3, pD1)
> utrt2 = getTreatment(CT3, pD2)
> utrt3 = getTreatment(CT3, pD3)
> utrt4 = getTreatment(CT3, pD4)
> c(utrt1, utrt2, utrt3, utrt4)

[1] "B" "A" "A" "A"
```

5   Work related to minimization

We create minimization-based allocator descriptions as follows:

```
> md = new("MinimizationDesc", treatments=c(A=1L, B=1L), method=minimizePocSim,
+     type="Minimization", featuresInUse="sex")
```

Now bind to an experiment:

```
> randomization(CE1) = list(Center1=list(md), Center2 = list(md))
```

Create the trial and obtain treatments:

```
> CT4 = createTrial(CE1, seed=c(301,401))
> getTreatment(CT4, pD1)
```

```
[1] "B"

> getTreatment(CT4, pD2)

[1] "A"

> getTreatment(CT4, pD3)

[1] "A"

> getTreatment(CT4, pD4)

[1] "A"
```

6 Software Design Overview

Here we briefly describe the ideas that underlie our design, which may seem a bit overly complex initially, however, we have found that they are needed to provide sufficient flexibility to support practical use of the software.

6.1 Randomizers

There are two components here, first is the description of the randomizer. This description is a complete description of all the components needed to create a randomizer for use. The *randomizerDesc* class is used to hold the information. Then, for any instance of this class one can create a realization, which is a randomizer that can then be used to randomize patients.

Perhaps the most important reason for this separation is that if one wants to use re-randomization as a basis for inference, then this approach makes that particularly simple. One need only obtain an instance of the *randomizerDesc* to create as many randomizers as needed for inference.

6.2 Patient Identifiers

Within any study patient identifiers are used once patients have been entered into the study. These identifiers are then used to identify specific patient records in the database that is used to store them.

Our implementation provides patient IDs for each strata and allows for a starting number and a stoping number (this is one way to indicate that the strata, and perhaps the trial, has reached its accrual goals). This could easily be extended to support other mechanisms of assigning patient IDs.