# Validation of RCTdesign

Scott S. Emerson, M.D., Ph.D.

RCTdesign.org

August 13, 2012

**Abstract**

It is of course highly important that software be fully validated before it is used in any application. This is especially true of software that would be used in analyses that would be submitted to regulatory agencies such as the FDA. However, the flexibility and extensibility of R, as well as its generally open source nature, that make it so attractive to many users detracts somewhat from the ability to ensure validated code: the constantly changing code presents a challenge for verification of accuracy and precision. The FDA does not prohibit the use of R in submissions, but it does require that sponsors be able to demonstrate that any code is working correctly. We have instituted a number of policies at RCTdesign.org to increase the reliability of the RCTdesign package across versions of R and RCTdesign. In this document we describe some of the ways that the functionality of RCTdesign has been validated by us. Individual users may draw on and extend these approaches as they seek to verify the accuracy of their results.

## 1 Introduction

The core functionality needed by a package for sequential clinical trial design is the ability to compute the sampling distribution of the sequential test statistic. RCTdesign uses three general approaches (see the *RCTdesign Technical Overview* for detals):

- Numerical integration of the sequential sampling density as described by Armitage, McPherson, and Rowe (1969) for normally distributed incremental statistics.
- Numerical summation of the binomial probability mass function in a manner analogous to that of Armitage, McPherson, and Rowe (1969) in order to handle the one sample binomial probability model.
- Simulation of randomized clinical trials. These routines do not use such R functions as `t.test(), chisq.test(), and survdiff()`. Instead, the analysis of multiple clinical trials is vectorized to save computing time.

Given a particular stopping boundary (no matter how it is derived), the major use of the numerical integration routines is to calculate the tail probabilities of the sampling density in general, and the stopping probabilities in particular. These routines are also used to calculate the expected value of the MLE.

Once the sampling densities are available, however, those functions need to be used in iterative searches in order to find parameter values that correspond to specified criteria:

- The probability of exceeding the boundary critical values most often needs to be equal to some specified type I error under the null and the desired power level under the design alternative. Critical values are

estimated, the stopping probabilities are calculated for that trial stopping rule, and the critical values are modified repeatedly until the valid critical values are found.

- The limits of confidence bounds correspond to parameter values defined such that an observed value of the sequential test statistic is the desired quantile of the sampling distribution. Quantiles for a specified ordering are estimated, the probability of exceeding that estimated quantile are computed, and the trial quantiles are modified repeatedly until the valid quantile is found.
- Bias adjusted estimates are computed as the value of the parameter such that an observed value is the mean of a sampling distribution. The bias adjusted estimate is guessed, the expectation of the MLE is computed when theta is equal to that guess, and the trial bias adjusted estimate is modified repeatedly until the expected value of the MLE is equal to the observed value of the MLE.

(More detail is given below in the description of the approaches to validating inferential methods.)

By the very nature of the problem (i.e., computations too difficult to solve in closed form or with a hand calculator), validation of the routines is not straightforward.

The routines that must be validated include

- The numerical integration routines.
- The search algorithms that are used to find critical values and estimates.
- The specification of parameters to generate particular designs.
- The vectorized statistical analyses in the simulation routines.

The general methods that we have available are

- agreement of results with derived formulas (in very limited settings),
- "n version programming" in which we consider agreement between alternative, independently coded programs,
- internal consistency of the programs in which we ensure that the quantile unbiased estimates and bias adjusted means agree with their sampling distribution definitions, and
- consistency of results across versions of the program.

The following sections describe some of the general routines used to validate RCTdesign. R scripts that implement some of these routines in limited cases are provided in separate files.

We first load the RCTdesign library.

```
> library(RCTdesign)
```

We also define some variables that will be used in the R scripts that invoke the validation routines.

```
> RCTplatform <- substring(version$platform,6,7)
> RCTdirnameV <- paste(ifelse(RCTplatform=="ap","~","z:"),
+         "/Documents/RCTdesign.org/Software/Validation/",sep="")
```

(Because these routines were developed on a MacBook Pro running MacOS as well as Windows XP under Parallels, the first variable is just to deal with the directory naming conventions under the alternative operating systems. However, all the script files only presume that `RCTdirnameV` is defined. A user wanting to reproduce validation will need to define `RCTdirnameV` to some directory where the script files will be found and where the output from the validation routines will be stored.)

# 2 External Validation

In this section we describe validation routines that compare RCTdesign results to those based on 1) R base package functions, or 2) sequential design routines written by others.

## 2.1 Agreement with Standard Normal Distribution

RCTdesign performs recursive numerical integration of convolutions of normal densities with truncations of similar convolutions. If a clinical trial design has stopping boundaries that are effectively infinite, then the resulting density should be very close to the density of a normally distributed random variable. We can use a very large value for the P parameter within the unified family to produce boundaries that do not allow early stopping over a wide range of alternatives.

```
> approxStdNorm <- seqDesign(nbr.analyses=3, arms=1, test.type="two.sided", P=2.5)
> approxStdNorm

Call:
seqDesign(arms = 1, nbr.analyses = 3, test.type = "two.sided",
    P = 2.5)

PROBABILITY MODEL and HYPOTHESES:
   Theta is mean response
   Two-sided hypothesis test:
           Null hypothesis : Theta = 0.00    (size  = 0.050)
    Alternative hypothesis : Theta > 3.92    (power = 0.975)
   (Wang & Tsiatis (1987) with Delta= -1.5 (or P= 2.5 ) )

STOPPING BOUNDARIES: Sample Mean scale
                     Low Diff Lo Equiv Hi Equiv High Diff
    Time 1 (N= 0.33) -30.5528       NA       NA   30.5528
    Time 2 (N= 0.67)  -5.4010       NA       NA    5.4010
    Time 3 (N= 1.00)  -1.9600    -1.96     1.96    1.9600

> print(seqOC(approxStdNorm,theta=0), digits= 7)

### Asymptotic Operating Characteristics
Operating characteristics at theta= 0
ASN= 0.9999966
Expected theta= 0
Lower Power= 0.025
Upper Power= 0.025

Stopping Probabilities:
                    Lower Null     Upper     Total
Analysis time 1 0.0000000 0.00 0.0000000 0.0000000
Analysis time 2 0.0000052 0.00 0.0000052 0.0000103
Analysis time 3 0.0249948 0.95 0.0249948 0.9999897

> range( dSeq( approxStdNorm, rep(3,51), (-25:25)/10, theta = 0 )$dens /
+         dnorm( (-25:25)/10) - 1)
```

```
[1] -2.042078e-05 -2.453593e-14

> range( pSeq( approxStdNorm, rep(3,51), (-25:25)/10, theta = 0 )$cdf.meanOrder /
+         pnorm( (-25:25)/10 ) - 1)

[1] -8.484046e-08  1.364503e-05

> range( qSeq( approxStdNorm, (1:100)/100- .005 , theta = 0 )$qObsMean /
+         qnorm( (1:100)/100- .005 ) - 1)

[1] -9.871720e-08  3.032148e-06

> print(seqOC(approxStdNorm,theta=0.25), digits= 7)

### Asymptotic Operating Characteristics
Operating characteristics at theta= 0.25
ASN= 0.999995
Expected theta= 0.2500196
Lower Power= 0.0135538
Upper Power= 0.0436363

Stopping Probabilities:
                    Lower      Null     Upper    Total
Analysis time 1 0.0000000 0.0000000 0.0000000 0.000000
Analysis time 2 0.0000020 0.0000000 0.0000130 0.000015
Analysis time 3 0.0135519 0.9428099 0.0436233 0.999985

> range( dSeq( approxStdNorm, rep(3,51), (-25:25)/10, theta = 0.25)$dens /
+         dnorm( (-25:25)/10, 0.25) - 1)

[1] -2.042078e-05 -2.453593e-14

> range( pSeq( approxStdNorm, rep(3,51), (-25:25)/10, theta = 0.25 )$cdf.meanOrder /
+         pnorm( (-25:25)/10, 0.25 ) - 1)

[1] -1.470880e-07  1.574992e-05

> range( qSeq( approxStdNorm, (1:100)/100- .005 , theta = 0.25 )$qObsMean /
+         qnorm( (1:100)/100- .005, 0.25 ) - 1)

[1] -8.197342e-08  1.315251e-05
```

Note that the choice `P= 2.5` generates boundaries with very low (but not zero) probability of stopping before the 3rd analysis (higher values of `P` could be used, but the computation time can be long owing to the wide continuation region with mass concentrated near the center). As seen from the ranges of proportionate differences between the densities, cumulative probabilities, and quantiles, the numerical integration routines agree to within 0.002% over the values tested even in this setting in which there is truly a 0.00001 probability of stopping prior to the last analysis.

## 2.2 Agreement with Formula for Expected MLE

In an unpublished dissertation, Emerson (1988) derived a formula for the expected value of the MLE or a normal mean for a group sequential design having a maximum of $J = 2$ analyses. Let $Y_i \sim \mathcal{N}(\theta, \sigma^2)$ in a one arm study, and suppose that analyses are performed when total sample sizes are $N_1$ and $N_2$. We define the MLE at each analysis as $\hat{\theta}_j = \sum_{i=1}^{N_j} Y_i$. Then for stopping boundaries $a_1 < d_1$, we define test statistic $(M, \hat{\theta})$ as

- $M = 1$ if $N_1\hat{\theta}_1 \notin (a_1, d_1)$, and $M = 2$ otherwise, and
- $\hat{\theta} = \hat{\theta}_M$.

We can then find that

$$E[\hat{\theta}] = \theta + \frac{\sigma(N_2 - N_1)}{N_2\sqrt{2\pi N_1}}\left(\exp\left(-\frac{(d_1 - N_1\theta)^2}{2N_1\sigma^2}\right) - \exp\left(-\frac{(a_1 - N_1\theta)^2}{2N_1\sigma^2}\right)\right).$$

The RCTdesign function `seqCheckExpMeanJ2()` will compare the expected MLE as returned by `seqOC()` to the value computed according to the above formula. The file "RCTvalidateExpMeanJ2.R" contains code comparing the numerical integration to the closed form solution implemented in `seqCheckExpMeanJ2()` for arbitrarily selected group sequential designs. It also compares the "seqDesign" and "seqOC" objects with any results obtained and stored from prior executions of this code. In that way, this routine is also used to validate consistency across versions of RCTdesign. (The first time the script is run on a particular machine, the stored objects will be created in appropriately named subdirectories of the directory indicated by `RCTdirnameV`.)

```
> source(paste(RCTdirnameV,"RCTvalidateExpMeanJ2.R",sep=""))

##### RCTvalidateExpMeanJ2 - 20120804
      Expected Mean J=2  : All 21 tests  PASS
```

## 2.3 Agreement with Critical Values Published in the Statistical Literature

The unified family of designs implemented in RCTdesign includes as special cases the following families for which published critical values exist.

- the Wang and Tsiatis (1987) family of designs, which in turn includes Pocock (1977) and O'Brien and Fleming (1979) families.
- the Emerson and Fleming (1989) family of symmetric designs.
- the Pampallona and Tsiatis (1994) family.

The script file "RCTvalidateCriticalValues.R" compares the critical values calculated by RCTdesign to those published values, and verifies the agreement within the precision of the published papers (though see notes below about Pampallona and Tsiatis).

```
> source(paste(RCTdirnameV,"RCTvalidateCriticalValues.R",sep=""))

##### RCTvalidateCriticalValues - 20111124
      Wang & Tsia (1987) : All 4 tests  PASS
      Emer & Flem (1989) : All 2 tests  PASS
      Pamp & Tsia (1994) : All 12 tests  PASS
```

A couple notes are in order regarding these "external" validations:

- Much of the C code in RCTdesign got its start in the Pascal code used for the original definition of the Emerson and Fleming symmetric designs. Hence, those validations are perhaps better viewed as consistency across versions, than independent programming.
- There were two cases in which the `seqDesign()` results disagreed with Pampallona and Tsiatis. Given the agreement with all other values and the plausibility of typographical errors giving rise to the actual disagreements, the disagreement has been suppressed in the output in order that these routines might be used to ensure consistency across RCTdesign versions. (the "RCTvalidateCriticalValues.R" script file provides specific details regarding the discrepancy.)
- Much of the code that was used by Pampallona and Tsiatis was at least partially incorporated into early versions of East.

## 2.4 Agreement with Critical Values Computed by `gsDesign()`

The R package gsDesign was developed by Keaven Anderson and others at Merck. The most commonly used RCTdesign group sequential design families are also included in gsDesign. The script file "RCTvalidateGSDesign.R" compares the stopping rule critical values returned by `seqDesign()` and `gsDesign()` for selected designs within the Wang and Tsiatis (1987) family, a family of single boundary designs, the Hwang, Shih, and DeCani (1990) error spending family, and the Kim and DeMets (1987) power error spending family. The critical values are compared on the Z statistic scale and the error spending scale.

```
> source(paste(RCTdirnameV,"RCTvalidateGSDesign.R",sep=""))

##### RCTvalidateGSDesign - 20120817
      Wang & Tsiatis     : All 162 tests  PASS
      Single Boundary    : All 8 tests  PASS
      Hwang Shih DeCani  : All 8 tests  PASS
      Power Error Spend  : All 8 tests  PASS
```

## 2.5 Validation of Vectorized Computation of Test Statistics for Simulations

In order to gain speed, the RCTdesign capability for simulated clinical trials does not use the R base functions `t.test()`, `chisq.test()`, or `survdiff()`. Instead custom routines written with vectorized code are used. Such code is difficult for a human to read. In order to verify the correct behavior of those routines, the script file "RCTvalidateRSeq.R" compares the statistics returned by `rSeq()` to the results that would have been obtained with the corresponding R base functions `t.test()` and `chisq.test()`, and in that sense is considered by us to be a form of "external" validation.

There are 41 "tests" in the R script file, many of which have multiple parts. Some tests perform extensive simulations to compare numerically integrated and simulated operating characteristics, and these can be time consuming. The R script file will allow the specification of selected tests. The selection below includes all tests that are explicitly comparing the vectorized computation of test statistics to the R base functions.

```
> runRCTvalidateTests <- (1:41)[c(1:2,9:11,18:19,26:28,35:37,39:41)]
> source(paste(RCTdirnameV,"RCTvalidateRSeq.R",sep=""))

##### RCTvalidateRSeq - 20120811
   ## Tests  1 2 9 10 11 18 19 26 27 28 35 36 37 39 40 41
```

```
      One Sample Mean    : All 6 tests  PASS
      One Sample GeomMn  : All 9 tests  PASS
      Two Sample Mean    : All 10 tests  PASS
      Two Sample GeomMn  : All 14 tests  PASS
      One Sample Binom   : All 9 tests  PASS
      Two Sample Binom   : All 9 tests  PASS
```

# 3 Internal Validation

Below we describe some of the validation scripts that assess the internal consistency of RCTdesign functions across computation tasks and across versions.

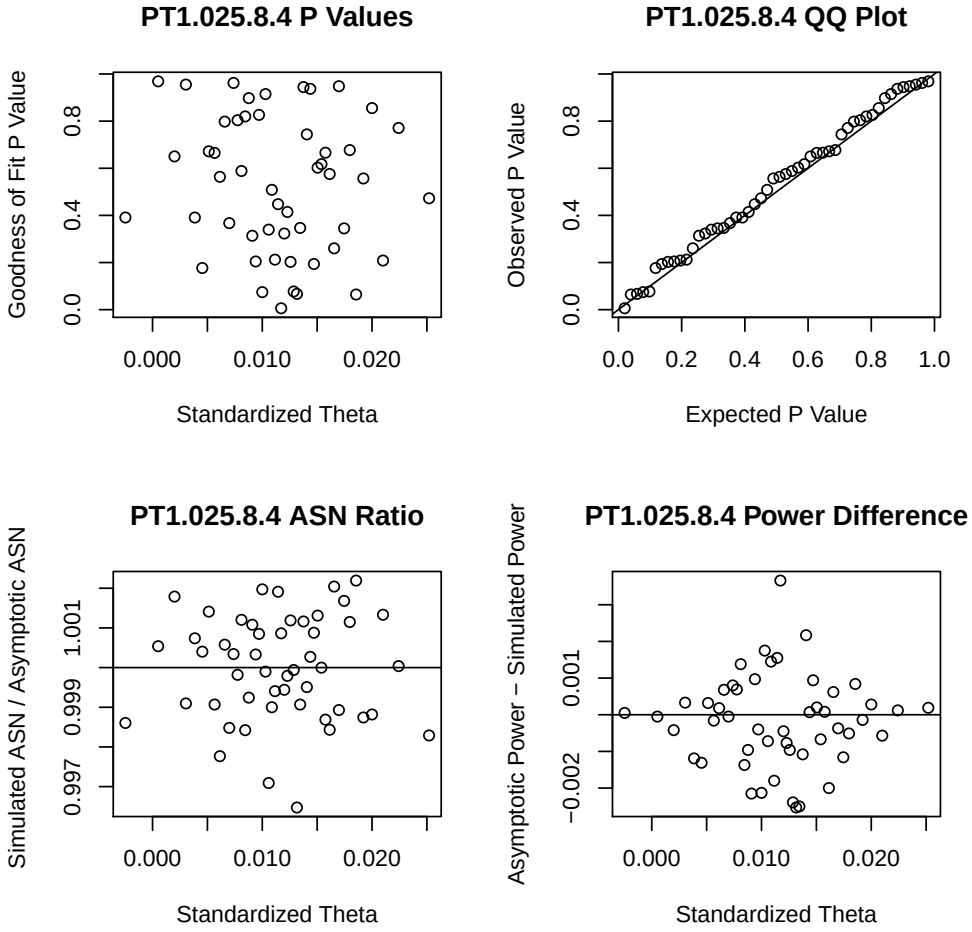
## 3.1 Comparison of Numerically Integrated and Simulated Operating Characteristics

The R script file "RCTvalidateOCsimulate.R" uses simulation to validate the numerically integrated stopping probabilities. This function uses `seqCheckOC()`, which in turn calls `seqOC(...,Nsimul=100000)`, and thus is regarded only as internal validation, because of its heavy reliance on other RCTdesign code. (The simulation code has been subjected to a measure of external validation as described above, however.) For each of a set of arbitrarily chosen stopping rules, simulated stopping probabilities are compared to those calculated using numerical integration of the truncated normal densities. Test statistics used in the simulation are meant to ensure exact behavior under the normal probability model. Chi square goodness of fit statistics are calculated for each of 50 (for one-sided tests) or 100 (for two-sided tests) values of theta. The function displays plots of the p values, as well as comparing differences in power and ASN between the simulated and numerically integrated results. The routine also compares the design and numerically integrated operating characteristics against previously stored results, thereby also serving as a check for consistency across versions of RCTdesign. (See the script file for additional comments regarding a single case (test 7) in which original tests failed. This was presumed to be an issue with multiple comparisons, and an alternative simulation substituted.)

There are 38 different designs included in the file, and for each design comparisons are made on the `"seqDesign"` object to any stored value, the `"seqOC"` object to any stored value, and the agreement between simulated and numerically integrated results. Ultimately there are thus 114 tests reported if all 38 cases are run. Because these validations are time consuming, you can specify that only a few cases are run. (Note that the script file will produce plots for each case, but in this Sweave document, only one such plot will be displayed.)

```
> runRCTvalidateTests <- (1:38)[c(17,34,38)]
> source(paste(RCTdirnameV,"RCTvalidateOCsimulate.R",sep=""))

##### RCTvalidateOCsimulate - 20120818
   ## Tests  17 34 38
      One Sample Normal  : All 3 tests  PASS
      Two Sample Normal  : All 3 tests  PASS
      Err Spend Normal   : All 3 tests  PASS
```




## 3.2 Validation of Inference

The R script file "RCTvalidateInference.R" verifies the internal consistency between computed operating characteristics and the corresponding inferential quantities.

Notationally, we can without loss of generality consider that we have data $Y_i \sim (\theta, \sigma^2)$, where the individual measurement $Y_i$ might represent a contrast across individuals (e.g., difference in measurements) or some other transformation of observations (e.g., the $i$th subject's contribution to the score or partial score function). Our sequential sampling scheme involves performing analyses after accrual of $N_1, N_2, \ldots, N_J$ observations, and comparing some statistic $T_j$ to stopping boundaries $a_j \leq b_j \leq c_j \leq d_j$. The group sequential test statistic $(M, T)$ is defined as

1. $M = \min\{1 \leq j \leq J : T_j \notin (a_j, b_j) \cup (c_j, d_j)\}$, and
2. $T = T_M$.

For the purposes of this documentation, we can without loss of generality focus on the setting in which $T_j$

is the maximum likelihood estimator

$$T_j = \hat{\theta}_j = \overline{Y}_{N_j} = \frac{1}{N_j}\sum_{i=1}^{N_j} Y_i.$$

Under sequential sampling, the MLE is not even approximately normally distributed, and it is generally biased and subject to high mean squared error (MSE). We thus define adjusted estimators based on the true sampling distribution of the minimally sufficient group sequential test statistic $(M, T)$ under various values of $\theta$ as calculated using the methods of Armitage, McPherson, and Rowe (1969).

Upon observing a group sequential test statistic $(M, T) = (m, t)$, the following describes the relationship between the sampling distribution of the test statistic under

1. The bias adjusted mean (BAM) is defined as that estimator $\breve{\theta}$ such that for observed $(M, T) = (m, t)$

$$E[T \,|\, \theta = \breve{\theta}] = t.$$

2. The median unbiased estimator (MUE) is defined as that estimator $\tilde{\theta}$ such that for observed $(M, T) = (m, t)$

$$Pr\left( (M, T) \prec (m, t) \,|\, \theta = \tilde{\theta} \right) = 0.5.$$

   (See the discussion about the definition of an ordering for "$\prec$" below.)

3. A $100(1-\alpha)\%$ confidence interval is defined as the interval $(\ddot{\theta}_L, \ddot{\theta}_U)$ such that for observed $(M, T) = (m, t)$

$$Pr\left( (M, T) \prec (m, t) \,|\, \theta = \ddot{\theta}_L \right) = \frac{1-\alpha}{2}$$
$$Pr\left( (M, T) \prec (m, t) \,|\, \theta = \ddot{\theta}_U \right) = \frac{\alpha}{2}$$

   (Again, see the discussion about the definition of an ordering for "$\prec$" below.)

4. An upper, one-sided p value for a test of $H_0 : \theta \leq \theta_0$ versus $H_1 : \theta > \theta_0$ based on observed $(M, T) = (m, t)$ is

$$P = Pr\left( (M, T) \succeq (m, t) \,|\, \theta = \theta_0 \right) = 1 - Pr\left( (M, T) \prec (m, t) \,|\, \theta = \theta_0 \right).$$

   (Again, see the discussion about the definition of an ordering for "$\prec$" below.)

Key, then, to calculation of estimates is the ability to integrate the sampling distribution of $(M, T)$ under arbitrary values of $\theta$, as well as to find the expectation of arbitrary functions of $(M, T)$ in those same settings. However, as the group sequential test statistic is bivariate, we must first define a suitable ordering of the sample space. RCTdesign currently implements three orderings:

1. The "stagewise ordering" (referred to as the "analysis time ordering" in S+SeqTrial and RCTdesign) explored by Armitage (1957), Siegmund (1978), Fairbanks and Madsen (1982), Jennison and Turnbull (1983), and Tsiatis, Rosner, and Mehta (1984), among others.

2. The "MLE ordering" (referred to as the "mean ordering" in S+SeqTrial and RCTdesign) explored by Armitage (1958), Duffy and Santner (1987), and Emerson and Fleming (1990).

3. The "likelihood ratio ordering" explored by Chang and O'Brien (1986), Rosner and Tsiatis (1988), and Chang (1989).

Results are provided for all three orderings by the RCTdesign functions `pSeq()`, `qSeq()`, and `seqInference()`.

In order to verify the methods for inference this script file defines the following functions, some of which have more widespread application:

- `gstMLEle()`: returns indicators of "less-than" for each ordering. This function is called by the routines that verify the accuracy of the quantiles returned by `qSeq()`.
- `seqCheckQuantilesBAM()`: uses simulations to verify the quantiles returned by `qSeq()` and the expected value for the MLE returned by `seqOC` are correct. (Note that the calculation of confidence intervals depends upon the accuracy of quantiles, and the calculation of the bias adjusted mean (BAM) is defined based on the expected value of the MLE.)
- `seqCheckConsistentInference()`: verifies the relationship between quantiles and CI and between the expected MLE and BAM.
- `seqSimCoverageBias()`: uses simulations to verify coverage probability of CIs (and also to compute bias and RMSE of estimators, though this functionality is not used in these validation tests).

Using these functions, the script file verifies the inferential quantities for arbitrarily selected designs and values of $\theta$. Because these validations are time consuming, you can specify that only a few cases are run.

```
> runRCTvalidateTests <- (1:4)[c(1,4)]
> source(paste(RCTdirnameV,"RCTvalidateInference.R",sep=""))

##### RCTvalidateInference - 20120804
   ## Tests  1 4
      One Sample Normal  : All 8 tests  PASS
```

## 3.3 Validation of Error Spending Functions

The R script file "RCTvalidateErrSpendExmpls.R" validates agreement of `"seqDesign"` objects based on error spending functions with previously stored results. Hence, this routine is primarily of use to validate consistency across versions of RCTdesign.

```
> source(paste(RCTdirnameV,"RCTvalidateErrSpendExmpls.R",sep=""))

##### RCTvalidateErrSpendExmpls - 20120804
      Err Spend Family   : All 12 tests  PASS
```