Computer Methods and Programs in Biomedicine 196 (2020) 105514

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

swCRTdesign: An R Package for Stepped Wedge Trial Design and Analysis

Emily C. Voldal[a,*], Navneet R. Hakhu[b], Fan Xia[a], Patrick J. Heagerty[a], James P. Hughes[a]

[a] Department of Biostatistics, University of Washington, Box 357232, Seattle, WA 98195, United States
[b] Department of Statistics, University of California, Irvine, Irvine, CA 92697, United States

ARTICLE INFO

Article history:
Received 29 December 2019
Revised 15 April 2020
Accepted 18 April 2020

Keywords:
Stepped wedge trial
Cluster randomized trial
Shiny
R

ABSTRACT

Background and objective: Stepped wedge trials (SWTs) are a type of cluster-randomized trial that are commonly used to evaluate health care interventions. Most SWT-related software packages have restrictive assumptions about the study design and correlation structure of the data. The objective of this paper is to present a package and corresponding web-based graphical user interface (GUI) that provide researchers with another, more flexible option for SWT design and analysis.

Methods: We developed an R package **swCRTdesign** ('stepped wedge Cluster Randomized Trial design'), which uses a random effects model to account for correlation in the data induced by a SWT design. Possible sources of correlation include clusters, time within clusters, and treatment within clusters.

Results: **swCRTdesign** allows a user to calculate power, simulate SWT data to streamline simulation studies (e.g. to estimate power), and create descriptive summaries and plots. Additionally, a GUI, developed using **shiny**, is available to calculate power and create power curves and design plots.

Conclusions: The **swCRTdesign** package accommodates a wide variety of SWT designs, and makes it easy to account for some sources of correlation which are not found in other packages. The user-friendly web-based GUI makes some **swCRTdesign** features accessible to researchers not familiar with R. These two resources will make appropriately complex SWT calculations more accessible to scientists from a wide variety of backgrounds.

1. Introduction

1.1. Basics of stepped wedge trials

Scientists in medicine, public health, and many other fields rely on randomized controlled trials as the gold standard method to evaluate interventions. Typically, individuals are randomized to intervention or control. However, sometimes clusters of individuals (e.g., families, clinics, or communities) are randomized, in which case individual-level observations are correlated; this is called a cluster randomized trial (CRT). The motivation for choosing a CRT instead of a traditional randomized trial may be due to the nature of the intervention (e.g., policy changes at a hospital) or for practical or logistical reasons. To date, most CRTs have mirrored the design of individually randomized trials and used a parallel or matched parallel two group design. One exception is the stepped wedge trial (SWT) design. The SWT is a more recently developed type of CRT that has been growing in popularity [1]. The most common design for a SWT dictates that all clusters begin in the control state at baseline, and at every subsequent time point (also called periods) some of the clusters cross over until all clusters have received treatment. Cross-over times are pre-specified for each cluster and usually evenly spaced after baseline. A group of clusters that cross over to treatment at the same time is called a sequence or wave. See Hughes, Granston, and Heagerty [2] for a more thorough introduction to stepped wedge designs. SWTs may have logistical or ethical advantages compared to other CRT designs, since the researcher only needs to implement the intervention in a few clusters at a time and every cluster eventually receives the intervention [1]. However, because of the complex design and correlated observations, specialized statistical methods and software are needed for designing SWTs.

The R [3] package **swCRTdesign** [4] is a tool to aid in the design and analysis of SWTs, including calculating power, simulating SWT data, and generating descriptive summaries. Additionally, a web-based graphical user interface (GUI) for calculating power is

* Corresponding Author.
E-mail address: voldal@uw.edu (E.C. Voldal).

https://doi.org/10.1016/j.cmpb.2020.105514
0169-2607/

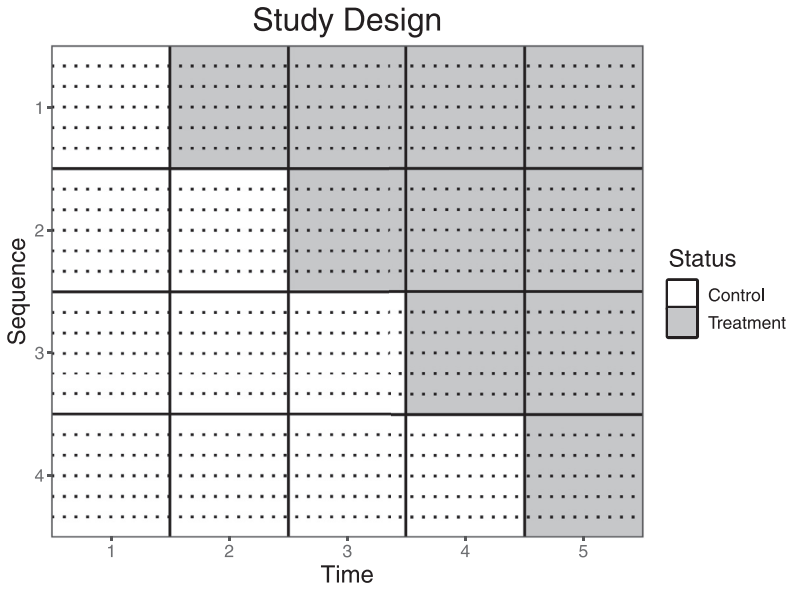

Figure 1. The design of the EPT trial with 4 sequences in which all sequences start in the control group. In each sequence, there are 6 independent clusters, separated by dotted lines.

available, with the hope of making some **swCRTdesign** features more accessible to researchers not familiar with R.

1.2. Motivating example

As a motivating example, we consider the stepped wedge trial on expedited partner therapy (EPT) in Washington state summarized by Golden et al [5]. With EPT, when an individual tests positive for a sexually transmitted infection, their sex partners are then offered treatment without medical evaluation. In this study, the clusters were local health jurisdictions (LHJs), which typically corresponded to counties. Although the authors analyzed several primary and secondary outcomes, here we focus on the effect of EPT on the percent of chlamydia tests that were positive among women between 14 and 25 years old at sentinel clinics within each LHJ. Researchers planned on randomizing 24 LHJs into four sequences of 6 LHJs each, stratified by region and LHJ size. Every sequence was observed in the control state at baseline, and at each subsequent time one sequence crossed over, resulting in a total of five time points. Figure 1 depicts the study design for this motivating example. Note that although the x-axis is labeled "Time 1", "Time 2", etc., the distance between observed times was between six and eight months, so these times could alternatively be labeled "Month 1", "Month 7", etc.

1.3. Summary of methods and software

There are a wide variety of methods for designing and analyzing SWTs [1]. For calculating power, the two most common methods are based on (i) application of normal-theory models via weighted least squares (WLS), with approximations for binary and count outcomes [6] and (ii) simulation [7]. A 'design effect' or 'sample size correction factor' is sometimes also used [8]. Baio et al [7] also provide an overview of these three techniques and their use for SWTs. **swCRTdesign** uses method (i). Very briefly, this method uses a mixed model structure to compute the variance of the WLS estimate of the treatment effect; power is based on a Wald statistic and relies on asymptotic Normality. For details, see [6] and [2] (especially Appendix A).

There are two other R packages that can be used for designing and analyzing SWTs. **SamplingDataCRT** [9] allows the user to simulate data and calculate power, but only for Gaussian outcomes and limited covariance structures. Power calculations are based on Hussey and Hughes [6] for cross-sectional data. **SWSamp** allows the user to simulate data and calculate power based on Hussey and Hughes, a design effect, or simulations, but most functionality is only supported for a limited number of covariance structures [10]. A web-based GUI developed by Hemming et al [11] allows the user to calculate power based on a design effect for Gaussian, binary, or count outcomes and several possible covariance structures. **swCRTdesign** provides support for more complex SWT designs (e.g., a sample size that varies over time and cluster) and alternative ways of specifying correlation structures.

1.4. Overview

The remainder of this manuscript is organized as follows. Section 2 describes the underlying analytic model used to compute power for a SWT design and simulate SWT data. Section 3 contains an overview of the **swCRTdesign** package, followed by examples with R code in section 4. Section 5 describes the web-based GUI tool. We conclude with a discussion in section 6. We hope that making both **swCRTdesign** and the GUI freely available will help support the needs of investigators designing or analyzing a SWT.

2. The model

2.1. Primary parameterization

To account for the correlation within clusters, we use a random effects model [2]. This model is used for both the power calculations, and simulating SWT data. Let Y_{ijk} denote the response recorded for individual k from cluster i at time j. In the **swCRTdesign** package, Y_{ijk} may follow a Gaussian, Bernoulli, or (for simulations only) Poisson distribution. Also, let X_{ij} have a value of one if cluster i is assigned to treatment at time j, and zero otherwise. In this paper and package, time is modeled as an unordered categorical factor. Then the mean response for cluster i at time j is:

$$\mu_{ij} = \mu + \beta_j + \theta * X_{ij} + u_i + w_{ij} + v_i * X_{ij} \quad (1)$$

Here, μ is the baseline average (i.e., the average response among those assigned to control at the first time point), θ represents the

treatment effect, and β_j represents the mean difference between time j and time 1, with $\beta_1 = 0$. The random effects are represented by u_i, w_{ij}, and v_i; each follows a Gaussian distribution with mean zero. The random intercepts, u_i, represent each cluster's unique baseline and have a standard deviation of τ. The random time effects, w_{ij}, represent cluster- and time- specific deviations and have a standard deviation of γ. The random treatment effects, v_i, represent the cluster-specific variations in the impact of treatment, and have a standard deviation of η. We assume that the correlation between u_i and v_i is ρ, and the correlation between w_{ij} and both u_i and v_i is zero. In addition, we assume the correlation between w_{ij} and $w_{ij'}$ with $j \neq j'$ is zero. So in cluster i with times j=1,...,J, the covariance of the vector of random effects is:

$$Cov\begin{bmatrix} u_i \\ v_i \\ w_{i1} \\ \vdots \\ w_{ij} \end{bmatrix} = \begin{bmatrix} \tau^2 & \rho\tau\eta & 0 & \cdots & 0 \\ \rho\tau\eta & \eta^2 & 0 & \cdots & 0 \\ 0 & 0 & \gamma^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \gamma^2 \end{bmatrix} \quad (2)$$

When the response Y_{ijk} is assumed to be Gaussian, $Y_{ijk} = \mu_{ij} + \varepsilon_{ijk}$ where ε_{ijk} is Gaussian with mean zero and standard deviation σ; that is, the mean of Y_{ijk} is μ_{ij} and the standard deviation is $\text{sd}(Y_{ijk} \mid u_i, w_{ij}, v_i) = \sigma$. The ε_{ijk} are mutually independent, and independent of all random effects. When Y_{ijk} is assumed to be Bernoulli, the mean is μ_{ij} and the standard deviation is defined by the function $sd(Y_{ijk}|u_i, w_{ij}, v_i) = \sqrt{\mu_{ij}(1-\mu_{ij})}$. Similarly, when Y_{ijk} is assumed to be Poisson, the mean is μ_{ij} and the standard deviation is defined by the function $sd(Y_{ijk}|u_i, w_{ij}, v_i) = \sqrt{\mu_{ij}}$

Note that these calculations assume that each subject in the data set is only observed once, at a single time point. If the same subjects are observed at multiple time points, we refer to this as a 'cohort' design; this requires additional random effects [12]. Currently, **swCRTdesign** does not support power calculations or simulations for cohort data. The application developed by Hemming et al [11] allows the user to calculate power for some cohort designs. **SamplingDataCRT** also provides some support for cohort data, but only for Gaussian outcomes and cluster random effects [9].

2.2. Alternative parameterization

Instead of using the variance components above to quantify correlation (which we will refer to as the standard deviation or SD parameterization), some researchers prefer to use measures such as the within-period intra-cluster correlation (ICC) and cluster auto-correlation (CAC) [12]. When there are no random treatment effects ($\eta = 0$), the two parametrizations are equivalent. However, the SD parameterization allows for random treatment effects (through η) and a correlation between the random treatment effects and the random intercepts (ρ), while a single ICC and CAC do not [13]. The ICC/CAC parameterization is less flexible than the SD parameterization since it requires the assumption that there are no random treatment effects (that is, $\eta = 0$ and $\rho = 0$). Despite this limitation, ICC and CAC are widely used in the literature both to summarize SWT data and perform power calculations [1].

The ICC is the correlation between individuals sampled from the same cluster at the same time, or the ratio of between-cluster variance to total variance [12]. In terms of the random effects parameters defined above,

$$ICC = \frac{\tau^2 + \gamma^2}{\tau^2 + \gamma^2 + \sigma^2}. \quad (3)$$

When the outcome is assumed to be Gaussian, σ is well defined. When the outcome is assumed to be Bernoulli, we substitute $\sigma = \sqrt{\bar{\mu}(1-\bar{\mu})}$, where $\bar{\mu}$ is the average of the mean outcomes for the baseline control and baseline treatment groups, $(\mu_0 + \mu_1)/2$. The CAC is the ratio of the between-period ICC to the within-period ICC, or the ratio of the correlation between individuals from the same cluster at different times to the correlation between individuals from the same cluster within the same period [12]. In this scenario,

$$CAC = \frac{\tau^2}{\tau^2 + \gamma^2}. \quad (4)$$

In this package, ICC and CAC input is translated to random effect standard deviations. If the ICC is zero, both τ and γ are zero and the CAC is not well-defined. If the ICC is one, either σ is zero or either τ or γ is extremely large. Since solving for the random effects parameters is not possible in this case, it is best to use the SD parameterization directly when the ICC is close to one.

The formulas below can be used to manually convert from ICC, CAC, and σ to the SD parameterization:

For CAC=1, $\gamma = 0$ and $\tau = \sigma * \sqrt{\text{ICC}/(1-\text{ICC})}$.

For CAC<1, $\gamma = \sigma * \sqrt{\text{ICC} * (1-\text{CAC})/(1-\text{ICC})}$ and $\tau = \gamma * \sqrt{\text{CAC}/(1-\text{CAC})}$.

2.3. Fractional treatment effect

In the case described above, we assume that the clusters receive the full treatment effect θ as soon as they cross over to treatment (i.e., X_{ij} changes from 0 to 1). However, the effect of an intervention is not always immediate, and usually depends on the scientific setting, mechanism of the intervention, and the planned timing of assessments in a trial. For example, perhaps when a cluster first crosses over to treatment the effect size is $\theta/2$, then θ at every subsequent time point. Fractional treatment effects may extend over as many time points as desired by specifying a series of X_{ij} between 0 and 1. Depending on the planned trial duration, some clusters may not be followed long enough for researchers to observe the full treatment effect. For more detail, see Hughes, Granston, and Heagerty [2].

3. swCRTdesign overview

The **swCRTdesign** package contains five main functions: `swDsn` for defining a SWT design, `swPwr` for calculating power, `swSim` for simulating data, and `swPlot` and `swSummary` for plotting and summarizing a SWT data set. The full documentation can be found at https://cran.r-project.org/web/packages/swCRTdesign/swCRTdesign.pdf.

The functions `swPwr` and `swSim` both depend on defining a SWT design through `swDsn`. To define a design, a vector (`clusters`) specifies the number of sequences (the length of the vector) and number of clusters per sequence. Designs can either begin with all sequences in the control setting (default), or begin with the first sequence on treatment. Extra time periods can be added while all sequences are on control, or after all sequences have switched to treatment. Designs may also have extra time points throughout the study during which no sequences cross over, or have transition periods (see Examples and supplemental file).

`swPwr` calculates power for either a Bernoulli or Gaussian outcome. The number of observations per cluster may vary by cluster and over time. When simulating data with `swSim`, the outcome may be Gaussian, log-Gaussian, Poisson (identity or log link), or Bernoulli (identity, log, or logit link). Both `swPwr` and `swSim` allow either ICC/CAC or the SD parameterization in some cases. See supplemental file for an example of how to use `swSim` to estimate power.

`swSummary` summarizes stepped wedge data over time with respect to sequences, or with respect to individual clusters. The

summaries can be by mean, sum, or number of observations. `swPlot` can plot trends by sequence or by cluster, either in a single plot or with one plot for each sequence. Default colors, symbols, and line types differentiate between different clusters or sequences, and indicate when each sequence or cluster was under treatment or control. If desired, these can be customized or eliminated.

More details can be found in the supplemental file on the syntax and flexibility of the **swCRTdesign** package. The examples below are meant to illustrate the capabilities of the package, but not provide a thorough tutorial of the package's use.

4. Examples

4.1. Gaussian example

To illustrate the use of the **swCRTdesign** package, suppose we are interested in a traditional SWT with 5 sequences and 6 clusters per sequence. We start by creating a design, and printing the design matrix (`swDsn.unique.clusters`). Rows and columns correspond to clusters and time points, respectively. Control and treatment status are represented by 0′s and 1′s.

```
> library("swCRTdesign")
> design <- swDsn(clusters =
  c(6, 6, 6, 6, 6))
> design$swDsn.unique.clusters
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0	1	1	1	1	1
[2,]	0	0	1	1	1	1
[3,]	0	0	0	1	1	1
[4,]	0	0	0	0	1	1
[5,]	0	0	0	0	0	1

Next, using our planned design and external information about the assumed variability in the data, we can calculate power. Here, we have Gaussian data with 50 individuals in every cluster at every time point. Argument names mirror the symbols in the model described above.

```
> swPwr(design = design, distn = "gaussian",
  n = 50, mu0 = 0, mu1 = 0.003,
  sigma = .03, tau = .01, eta = 0, rho = 0,
  gamma = 0.001, silent = TRUE)
[1] 0.7399873
```

Alternatively, transforming from the SD parameterization to the ICC/CAC parameterization, we can obtain the same power estimate using ICC and CAC. Note that this is possible only because we assumed there are no random treatment effects ($\eta = 0$) in this scenario.

```
> icc.ex <- (.01 ^ 2 + .001 ^ 2) /
  (.01 ^ 2 + .001 ^ 2 + .03 ^ 2)
> icc.ex
[1] 0.1008991
> cac.ex <- .01 ^ 2 / (.01 ^ 2 + .001 ^ 2)
> cac.ex
[1] 0.990099
> swPwr(design = design, distn = "gaussian",
  n = 50, mu0 = 0, mu1 = 0.003,
  sigma = .03, icc = icc.ex,
  cac = cac.ex,
  silent = TRUE)
[1] 0.7399873
```

Additionally, we can simulate SWT data based on the assumed design and parameters. Note that we added a positive time trend (with `time.effect`); this corresponds to the vector of $\beta_1 \ldots \beta_6$ in the model above. The default output contains the outcome (`response.var`), treatment indicator (`tx.var`), and the time and cluster IDs (`time.var, cluster.var`).

```
> set.seed(4)
> example.data <- swSim(design = design,
  family = "gaussian", n = 50,
  mu0 = 0, mu1 = 0.003, time.effect = (0:5)
  * .002, sigma = .03,
  tau = .01, eta = 0, rho = 0,
  gamma = 0.001)
> head(example.data)
```

	response.var	tx.var	time.var	cluster.var
1	-0.004442856	0	1	1
2	-0.004531357	0	1	1
3	-0.005274664	0	1	1
4	0.064369213	0	1	1
5	0.049816709	0	1	1
6	-0.028956710	0	1	1

Using `swSummary`, we can examine the mean response for each sequence over time, and plot that using `swPlot` (Figure 2). The additional arguments in `swPlot` customize the location and width of the legends and provide a title; see package documentation for details.

```
> example.summary <- swSummary(
  response.var = response.var,
  tx.var = tx.var, time.var = time.var,
  cluster.var = cluster.var,
  data = example.data, type = "mean")
> #To view the matrix of means by sequence
  and time, we could run:
> #example.summary$response.wave
> swPlot(response.var = response.var,
  tx.var = tx.var, time.var = time.var,
  cluster.var = cluster.var,
  data = example.data, choose.ncol = 5,
  choose.tx.pos = "bottom",
  choose.legend.pos = "bottomright",
  choose.main = "SWT Example Plot")
```

4.2. Bernoulli example (EPT trial)

Recall the motivating example from the introduction, in which Golden et al [5] examined the effect of EPT on the percent of chlamydia tests that were positive. Each cluster is a LHJ, and each individual observation is a chlamydia test result (positive or negative), which follows a Bernoulli distribution. When planning the study, researchers identified 24 eligible LHJs, and planned on randomizing them into four sequences. Below, we construct this design (note that this design is identical to the one in Figure 1).

```
> design.bernoulli <- swDsn(
  clusters = c(6, 6, 6, 6))
```

To estimate power for this SWT design, researchers assumed that there would be 162 tests per LHJ at each observed time, and that 5% of the tests would be positive in the control setting. They also estimated that τ =0.0165, and assumed that there would be no other random effects. The trial was powered to detect a multiplicative change of 0.7; that is, 3.5% positive in the treatment group. With a significance level of 0.05, the calculation below aligns with the researchers' desired power of at least 80%.

```
> swPwr(design = design.bernoulli,
  distn = "binomial", n = 162,
  mu0 = 0.05, mu1 = 0.035,
  tau = 0.0165, eta = 0, rho = 0,
  gamma = 0, silent = TRUE
[1] 0.8468701
```

The researchers stated that they wished to use a log link in their model for the relationship between EPT and chlamydia positivity. We can simulate data that accounts for this. Because we are using a log link, `mu0` and `mu1` are on the log scale, so we enter

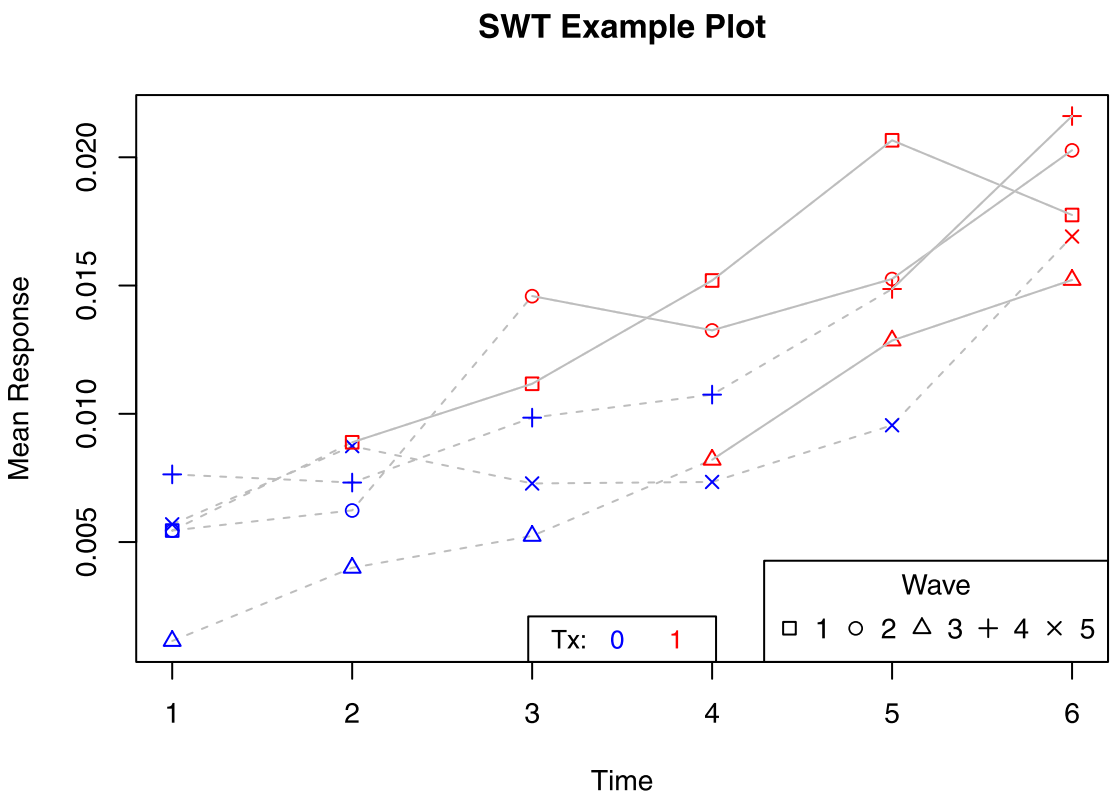

Figure 2. Plot created using `swPlot` for the Gaussian example.

the log of the prevalences. Similarly, `tau`, `eta`, `rho`, and `gamma` correspond to random effects that are linear on the log scale, so we use the predicted coefficient of variation between LHJs for `tau`. Note that this data does not exactly correspond with the mixed effects model used by Golden et al [5], since they also included random effects for individual clinics within the LHJs and a fixed time trend. The WLS technique used by `swPwr` for estimating power cannot easily account for a nonlinear link, but the data can be simulated; this is one scenario where simulation-based power calculations may be useful (see supplemental file).

```
> example.data.bernoulli <-
  swSim(design = design.bernoulli,
  family = binomial(link = "log"), n = 162,
  mu0 = log(0.05), mu1 = log(0.035),
  time.effect = 0,
  tau = .33, eta = 0, rho = 0, gamma = 0)
> #To view the simulated data, we could run:
> #head(example.data.bernoulli)
```

Unfortunately, while the researchers planned on 24 LHJs and 162 tests per LHJ at each time point, when the study was conducted they only had 22 LHJs and about 107 tests per LHJ at each time point. Researchers assigned six LHJs to each of the first three waves, and only four LHJs to the last wave, resulting in the design below.

```
> design.bernoulli.post <- swDsn(
  clusters = c(6, 6, 6, 4))
```

For the purpose of this example, suppose we know the exact number of observations in each LHJ at each time point. A plausible matrix of sample sizes is simulated below. Each row represents a LHJ, and each column represents a time point. LHJs are ordered by sequence, from first to cross over to last to cross over, so the sample sizes printed below correspond to the six LHJs in the first sequence to cross over. Within a sequence, the order of clusters is arbitrary.

```
> n.bernoulli.post <- matrix(
  data = rpois(n = 110,
  lambda = 107),
  nrow = 22, ncol = 5)
> head(n.bernoulli.post)
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	99	95	118	108	110
[2,]	105	139	104	93	121
[3,]	100	95	118	105	111
[4,]	95	113	85	124	96
[5,]	112	101	107	102	120
[6,]	116	117	113	97	128

Now we can see how these changes impact the power of the study.

```
> swPwr(design = design.bernoulli.post,
  distn = "binomial",
  n = n.bernoulli.post, mu0 = 0.05,
  mu1 = 0.035,
  tau = 0.0165, eta = 0, rho = 0,
  gamma = 0, silent = TRUE)
[1] 0.6432843
```

5. Web-based power calculator

The web-based GUI power calculator provides investigators with an alternative tool (that does not require knowledge of R) to compute power for SWTs and explore the impact on power of candidate SWT designs. This was developed using the **shiny** package [14]. The app can be accessed online at https://swcrtdesign.shinyapps.io/stepped_wedge_power_calculation/ or downloaded and run in R from https://github.com/swCRTdesign/Stepped-wedge-power-calculation. The language and inputs are very similar to the **swCRTdesign** package, described above, and a manual is provided in the 'Help and References' tab.

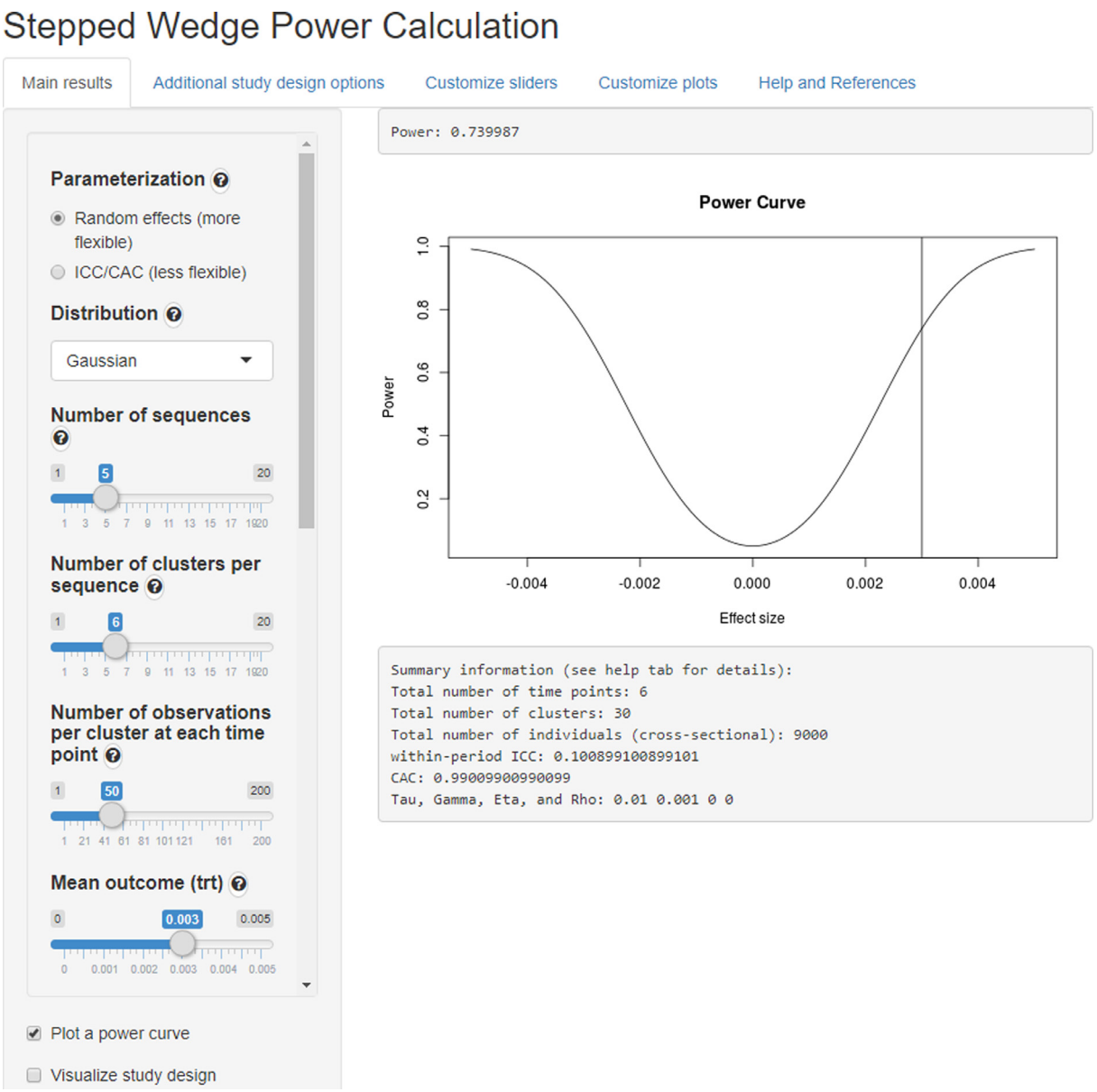

Figure 3. Power calculations for the Gaussian example, done in the web-based GUI. Inputs for all the arguments were entered using the sliders in the left-hand column. The range and appearance of the plot were changed in the "Customize plots" tab.

In addition to calculating the power for a specific design, the app can also plot a power curve, visualize and summarize the study design, and help translate between the ICC/CAC and SD parameterization. Some plots use the package **ggplot2** [15]. Figure 3 shows the power calculations from the Gaussian example above. Not pictured is the plot of the study design which the GUI also generates, in the same style as Figure 1.

6. Discussion

The R package **swCRTdesign** provides useful tools for the design and analysis of SWTs. These functions make it easy to calculate power, simulate data, and plot and summarize SWT data. The online power calculator app makes power calculations accessible without knowledge of R, and facilitates exploration of different designs. We believe these will be useful tools for applied researchers from a variety of backgrounds, as the use of SWTs continues to grow.

A limitation of **swCRTdesign** is that, currently, power cannot be calculated for cohort designs. Calculating power for SWTs with more complex correlation structures is an important extension that may be added in a future version of the package. Other potential additions include allowing more flexible specification of the design matrix and random effects covariance matrix, and adding power calculations for logit and log link models.

Computational details

The results in this paper were obtained using R 3.6.1 with the **swCRTdesign** 3.1 package. In addition, the **ggplot2** 3.2.0 package was used to create Figure 1. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

Declaration of Competing Interest

The authors have declared no conflict of interest.

Acknowledgments

Funding: Research reported in this work was partially funded through a Patient-Centered Outcomes Research Institute (PCORI) Award (ME-1507-31750). The statements in this work are solely the responsibility of the authors and do not necessarily represent the views of the Patient-Centered Outcomes Research Institute (PCORI), its Board of Governors, or Methodology Committee. PCORI had no direct involvement in this work.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2020.105514.

References

[1] D Barker, P McElduff, C D'Este, M Campbell, Stepped Wedge Cluster Randomised Trials: A Review of the Statistical Methodology Used and Available, BioMed Central Medical Research Methodology 16 (69) (2016), doi:10.1186/s12874-016-0176-5.

[2] JP Hughes, TS Granston, PJ Heagerty, Current Issues in the Design and Analysis of Stepped Wedge Trials, Contemporary Clinical Trials 45 (2015) 55–60, doi:10.1016/j.cct.2015.07.006.

[3] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2019 https://www.R-project.org/ 29 December 2019.

[4] J Hughes, NR Hakhu, E Voldal, **swCRTdesign**: Stepped Wedge Cluster Randomized Trial (SW CRT) Design, R package version 3.1, 2019 https://CRAN.R-project.org/package=swCRTdesign 29 December 2019.

[5] MR Golden, RP Kerani, M Stenger, JP Hughes, M Aubin, C Malinski, KK Holmes, Uptake and Population-Level Impact of Expedited Partner Therapy (EPT) on *Chlamydia trachomatis* and *Neisseria gonorrhoeae*: The Washington State Community-Level Randomized Trial of EPT, Public Library of Science Medicine 12 (1) (2015), doi:10.1371/journal.pmed.1001777.

[6] MA Hussey, JP Hughes, Design and Analysis of Stepped Wedge Cluster Randomized Trials, Contemporary Clinical Trials 28 (2007) 182–191, doi:10.1016/j.cct.2006.05.007.

[7] G Baio, A Copas, G Ambler, J Hargreaves, E Beard, RZ Omar, Sample Size Calculation for a Stepped Wedge Trial, Trials 16 (354) (2015), doi:10.1186/s13063-015-0840-9.

[8] W Woertman, E de Hoop, M Moerbeek, S Zuidema, D Gerritsen, S Teerenstra, Stepped Wedge Designs Could Reduce the Required Sample Size in Cluster Randomized Trials, Journal of Clinical Epidemiology 66 (7) (2013) 752–758, doi:10.1016/j.jclinepi.2013.01.009.

[9] D Trutschel, H Treutler, **samplingDataCRT**: Sampling Data Within Different Study Designs for Cluster Randomized Trials, R package version 1.0, 2017 https://CRAN.R-project.org/package=samplingDataCRT 29 December 2019.

[10] G Baio, R Leech, **SWSamp**, R package version 0.3.1, 2019 29 December 2019. https://github.com/giabaio/SWSamp/.

[11] K Hemming, J Kasza, R Hooper, A Forbes, M Taljaard, A Tutorial on Sample Size Calculation for Multiple-Period Cluster Randomised Parallel, Cross-Over, and Stepped Wedge Trials Using the **shiny** CRT Calculator, International Journal of Epidemiology (2020), doi:10.1093/ije/dyz237.

[12] R Hooper, S Teerenstra, E de Hoop, S Eldridge, Sample Size Calculation for Stepped Wedge and Other Longitudinal Cluster Randomised Trials, Statistics in Medicine 35 (2016) 4718–4728, doi:10.1002/sim.7028.

[13] K Hemming, M Taljaard, JE McKenzie, R Hooper, A Copas, JA Thompson, M Dixon-Woods, A Aldcroft, A Doussau, M Grayling, C Kristunas, CE Goldstein, MK Campbell, A Girling, S Eldridge, MJ Campbell, RJ Lilford, C Weijer, AB Forbes, JM Grimshaw, Reporting of Stepped Wedge Cluster Randomised Trials: Extension of the CONSORT 2010 Statement with Explanation and Elaboration, British Medical Journal 363 (k1614) (2018), doi:10.1136/bmj.k1614.

[14] W Chang, J Cheng, J Allaire, Y Xie, J McPherson, **shiny**: Web Application Framework for R, R package version 1.3.2, 2019 https://CRAN.R-project.org/package=shiny 29 December 2019.

[15] H Wickham, **ggplot2**: Elegant Graphics for Data Analysis, Springer-Verlag, New York, 2016 ISBN 978-3-319-24277-4https://ggplot2.tidyverse.org 29 December 2019.