simstudy has just been updated (version 0.1.13 on CRAN), and includes one interesting addition (and a couple of bug fixes). I have worked on a post (or two) about intra-cluster correlations (ICCs) and stepped-wedge study designs, and I was getting tired of going through the convoluted process of generating data from a time-dependent treatment assignment process. So, I wrote a new function, trtStepWedge, that should simplify things.

I will take the opportunity of this brief announcement to provide a quick example.

**Data definition**

Stepped-wedge designs are a special class of cluster randomized trial where each cluster is observed in both treatment arms (as opposed to the classic parallel design where only some of the clusters receive the treatment). This is a special case of a cross-over design, where the cross-over is only in one direction: control (or pre-intervention) to intervention.

In this example, the data generating process looks like this:

\[Y\_{ict} = \beta\_0 + b\_c + \beta\_1 \* t + \beta\_2\*X\_{ct} + e\_{ict}\]

where \(Y\_{ict}\) is the outcome for individual \(i\) in cluster \(c\) in time period \(t\), \(b\_c\) is a cluster-specific effect, \(X\_{ct}\) is the intervention indicator that has a value 1 during periods where the cluster is under the intervention, and \(e\_{ict}\) is the individual-level effect. Both \(b\_c\) and \(e\_{ict}\) are normally distributed with mean 0 and variances \(\sigma^2\_{b}\) and \(\sigma^2\_{e}\), respectively. \(\beta\_1\) is the time trend, and \(\beta\_2\) is the intervention effect.

We need to define the cluster-level variables (i.e. the cluster effect and the cluster size) as well as the individual specific outcome. In this case each cluster will have 15 individuals per period, and \(\sigma^2\_b = 0.20\). In addition, \(\sigma^2\_e = 1.75\).

library(simstudy)

library(ggplot2)

defc <- defData(varname = "ceffect", formula = 0, variance = 0.20,

dist = "normal", id = "cluster")

defc <- defData(defc, "m", formula = 15, dist = "nonrandom")

defa <- defDataAdd(varname = "Y",

formula = "0 + ceffect + 0.1\*period + trt\*1.5",

variance = 1.75, dist = "normal")

In this case, there will be 30 clusters and 24 time periods. With 15 individuals per cluster per period, there will be 360 observations for each cluster, and 10,800 in total. (There is no reason the cluster sizes need to be deterministic, but I just did that to simplify things a bit.)

Cluster-level intervention assignment is done after generating the cluster-level and time-period data. The call to trtStepWedge includes 3 key arguments that specify the number of waves, the length of each wave, and the period during which the first clusters begin the intervention.

nWaves indicates how many clusters share the same starting period for the intervention. In this case, we have 5 waves, with 6 clusters each. startPer is the first period of the first wave. The earliest starting period is 0, the first period. Here, the first wave starts the intervention during period 4. lenWaves indicates the length between starting points for each wave. Here, a length of 4 means that the starting points will be 4, 8, 12, 16, and 20.

Once the treatment assignments are made, the individual records are created and the outcome data are generated in the last step.

set.seed(608477)

dc <- genData(30, defc)

dp <- addPeriods(dc, 24, "cluster", timevarName = "t")

dp <- trtStepWedge(dp, "cluster", nWaves = 5, lenWaves = 4,

startPer = 4, grpName = "trt")

dd <- genCluster(dp, cLevelVar = "timeID", "m", "id")

dd <- addColumns(defa, dd)

dd

## cluster period ceffect m timeID startTrt trt id Y

## 1: 1 0 0.628 15 1 4 0 1 1.52

## 2: 1 0 0.628 15 1 4 0 2 0.99

## 3: 1 0 0.628 15 1 4 0 3 -0.12

## 4: 1 0 0.628 15 1 4 0 4 2.09

## 5: 1 0 0.628 15 1 4 0 5 -2.34

## ---

## 10796: 30 23 -0.098 15 720 20 1 10796 1.92

## 10797: 30 23 -0.098 15 720 20 1 10797 5.92

## 10798: 30 23 -0.098 15 720 20 1 10798 4.12

## 10799: 30 23 -0.098 15 720 20 1 10799 4.57

## 10800: 30 23 -0.098 15 720 20 1 10800 3.66

It is easiest to understand the stepped-wedge design by looking at it. Here, we average the outcomes by each cluster for each period and plot the results.

dSum <- dd[, .(Y = mean(Y)), keyby = .(cluster, period, trt, startTrt)]

ggplot(data = dSum,

aes(x = period, y = Y, group = interaction(cluster, trt))) +

geom\_line(aes(color = factor(trt))) +

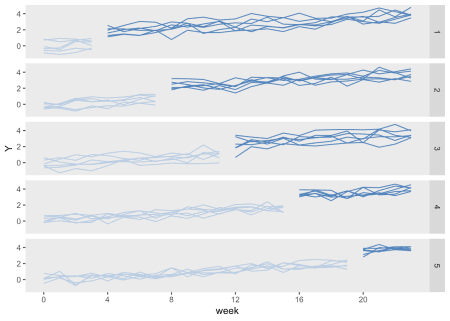
facet\_grid(factor(startTrt, labels = c(1 : 5)) ~ .) +

scale\_x\_continuous(breaks = seq(0, 23, by = 4), name = "week") +

scale\_color\_manual(values = c("#b8cce4", "#4e81ba")) +

theme(panel.grid = element\_blank(),

legend.position = "none")



Key elements of the data generation process are readily appreciated by looking at the graph: (1) the cluster-specific effects, reflected in the variable starting points at period 0, (2) the general upward time trend, and (3), the stepped-wedge intervention scheme.

Since trtStepWedge is a new function, it is still a work in progress. Feel free to get in touch to give me feedback on any enhancements that folks might find useful.

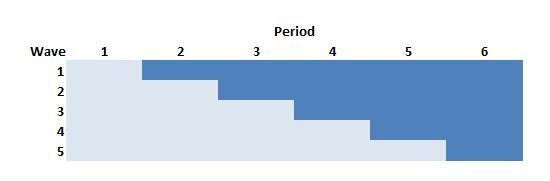
Stepped Wedge Design

The key differences in the various designs are how many sites are exposed to the intervention and what the phase-in schedule looks like. In the examples that follow, I am assuming a study that lasts 24 weeks and with 50 total sites. Each site will include six patients per week. That means if we are collecting data for all sites over the entire study period, we will have 24 \times 6 \times 50 = 720024×6×50=7200 outcome measurements.

The most important assumption I am making, however, is that the investigators can introduce the intervention at a small number of sites during each time period (for example, because the intervention involves extensive training and there is a limited number of trainers.) In this case, I am assuming that at most 10 sites can start the intervention at any point in time, and we must wait at least 4 weeks until the next wave can be started. (We can proceed slower than 4 weeks, of course, which surprisingly may be the best option.)

I am going to walk through the data generation process for each of the variations and then present the results of a series of power analyses to compare and contrast each design.

**Stepped-wedge design**



In the stepped-wedge design, all clusters in a trial will receive the intervention at some point, but the start of the intervention will be staggered. The amount of time in each state (control or intervention) will differ for each site (or group of sites if there are waves of more than one site starting up at the same time).

In this design (and in the others as well) time is divided into discrete data collection/phase-in periods. In the schematic figure, the light blue sections are periods during which the sites are in a control state, and the darker blue are periods during which the sites are in the intervention state. Each period in this case is 4 weeks long.

Following the Thompson et al. [paper](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5718336/), the periods can be characterized as pre-rollout (where no intervention occurs), rollout (where the intervention is introduced over time), and post-rollout (where the all clusters are under intervention). Here, the rollout period includes periods two through five.

First, we define the data, which will largely be the same across the designs: 6 individual patients per week, an intervention effect of 0.33, and a weekly time effect (which unfortunately is parameterized as “period”) of 0.02, and standard deviation within each cluster of 3.

**library**(simstudy)

defS <- defData(varname = "n", formula = 6,

dist = "nonrandom", id = "site")

defS <- defData(defS, varname = "siteInt", formula = 0,

variance = 1, dist = "normal")

defP <- defDataAdd(varname = "rx",

formula = "(start <= period) \* everTrt",

dist = "nonrandom")

defI <- defDataAdd(varname = "Y",

formula = "10 + rx \* 0.33 + period \* 0.02 + siteInt",

variance = 9, dist = "normal")

Now, we actually generate the data, starting with the site level data, then the period data, and then the individual patient level data. Note that the intervention is phased in every 4 weeks so that by the end of the 24 weeks all 5 waves are operating under the intervention:

set.seed(111)

dS <- genData(50, defS)

dS[, start := rep((1:5)\*4, each = 10)]

dS[, everTrt := 1]

dS[site %**in**% c(1, 2, 11, 12, 49, 50)] # review a subset

## site n siteInt start everTrt

## 1: 1 6 0.2352207 4 1

## 2: 2 6 -0.3307359 4 1

## 3: 11 6 -0.1736741 8 1

## 4: 12 6 -0.4065988 8 1

## 5: 49 6 2.4856616 20 1

## 6: 50 6 1.9599817 20 1

# weekly data

dP <- addPeriods(dtName = dS, nPeriods = 24, idvars = "site")

dP <- addColumns(defP, dP)

dP[site %**in**% c(3, 17) & period < 5] # review a subset

## site period n siteInt start everTrt timeID rx

## 1: 3 0 6 -0.31162382 4 1 49 0

## 2: 3 1 6 -0.31162382 4 1 50 0

## 3: 3 2 6 -0.31162382 4 1 51 0

## 4: 3 3 6 -0.31162382 4 1 52 0

## 5: 3 4 6 -0.31162382 4 1 53 1

## 6: 17 0 6 -0.08585101 8 1 385 0

## 7: 17 1 6 -0.08585101 8 1 386 0

## 8: 17 2 6 -0.08585101 8 1 387 0

## 9: 17 3 6 -0.08585101 8 1 388 0

## 10: 17 4 6 -0.08585101 8 1 389 0

# patient data

dI <- genCluster(dtClust = dP, cLevelVar = "timeID", numIndsVar = "n",

level1ID = "id")

dI <- addColumns(defI, dI)

dI

## site period n siteInt start everTrt timeID rx id Y

## 1: 1 0 6 0.2352207 4 1 1 0 1 10.810211

## 2: 1 0 6 0.2352207 4 1 1 0 2 14.892854

## 3: 1 0 6 0.2352207 4 1 1 0 3 12.977948

## 4: 1 0 6 0.2352207 4 1 1 0 4 11.311097

## 5: 1 0 6 0.2352207 4 1 1 0 5 10.760508

## ---

## 7196: 50 23 6 1.9599817 20 1 1200 1 7196 11.317432

## 7197: 50 23 6 1.9599817 20 1 1200 1 7197 7.909369

## 7198: 50 23 6 1.9599817 20 1 1200 1 7198 13.048293

## 7199: 50 23 6 1.9599817 20 1 1200 1 7199 17.625904

## 7200: 50 23 6 1.9599817 20 1 1200 1 7200 7.147883

Here is a plot of the site level averages at each time point:

**library**(ggplot2)

dSum <- dI[, .(Y = mean(Y)), keyby = .(site, period, rx, everTrt, start)]

ggplot(data = dSum, aes(x = period, y = Y, group = interaction(site, rx))) +

geom\_line(aes(color = factor(rx))) +

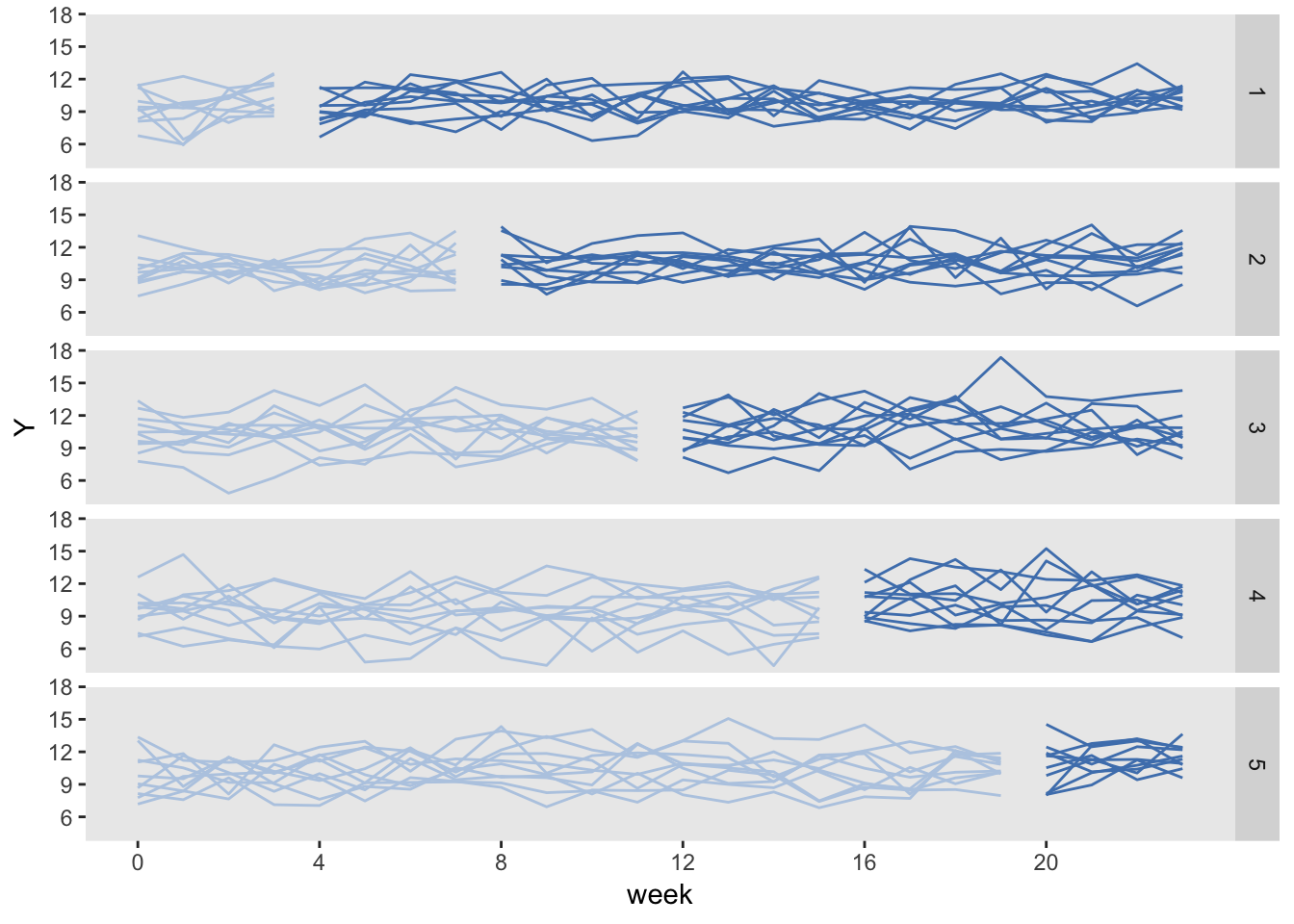
facet\_grid(factor(start, labels = c(1 : 5)) ~ .) +

scale\_x\_continuous(breaks = seq(0, 23, by = 4), name = "week") +

scale\_color\_manual(values = c("#b8cce4", "#4e81ba")) +

theme(panel.grid = element\_blank(),

legend.position = "none")



Finally, we can fit a linear mixed effects model to estimate the treatment effect:

**library**(lme4)

**library**(broom)

tidy(lmer(Y ~ rx + period + (1|site), data = dI))

## term estimate std.error statistic group

## 1 (Intercept) 9.78836231 0.184842722 52.955086 fixed

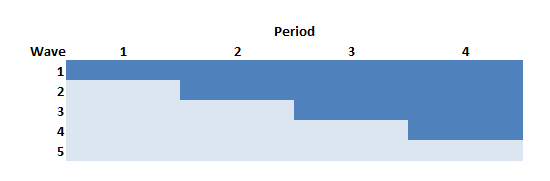
## 2 rx 0.35246094 0.122453829 2.878317 fixed

## 3 period 0.02110481 0.007845705 2.689983 fixed

## 4 sd\_(Intercept).site 1.21303055 NA NA site

## 5 sd\_Observation.Residual 2.99488532 NA NA Residual

**Stepped-wedge using “rollout” stage only**



The Thompson et al. paper argued that if we limit the study to the rollout period only (periods 2 through 5 in the example above) but increase the length of the periods (here, from 4 to 6 weeks), we can actually increase power. In this case, there will be one wave of 10 sites that never receives the intervention.

The data generation process is exactly the same as above, except the statement defining the length of periods (6 weeks instead of 4 weeks) and starting point (week 0 vs. week 4) is slightly changed:

dS[, start := rep((0:4)\*6, each = 10)]

So the site level data set with starting points at 0, 6, 12, and 18 weeks for each of the four waves that ever receive treatment looks like this:

## site n siteInt start everTrt

## 1: 1 6 0.2352207 0 1

## 2: 2 6 -0.3307359 0 1

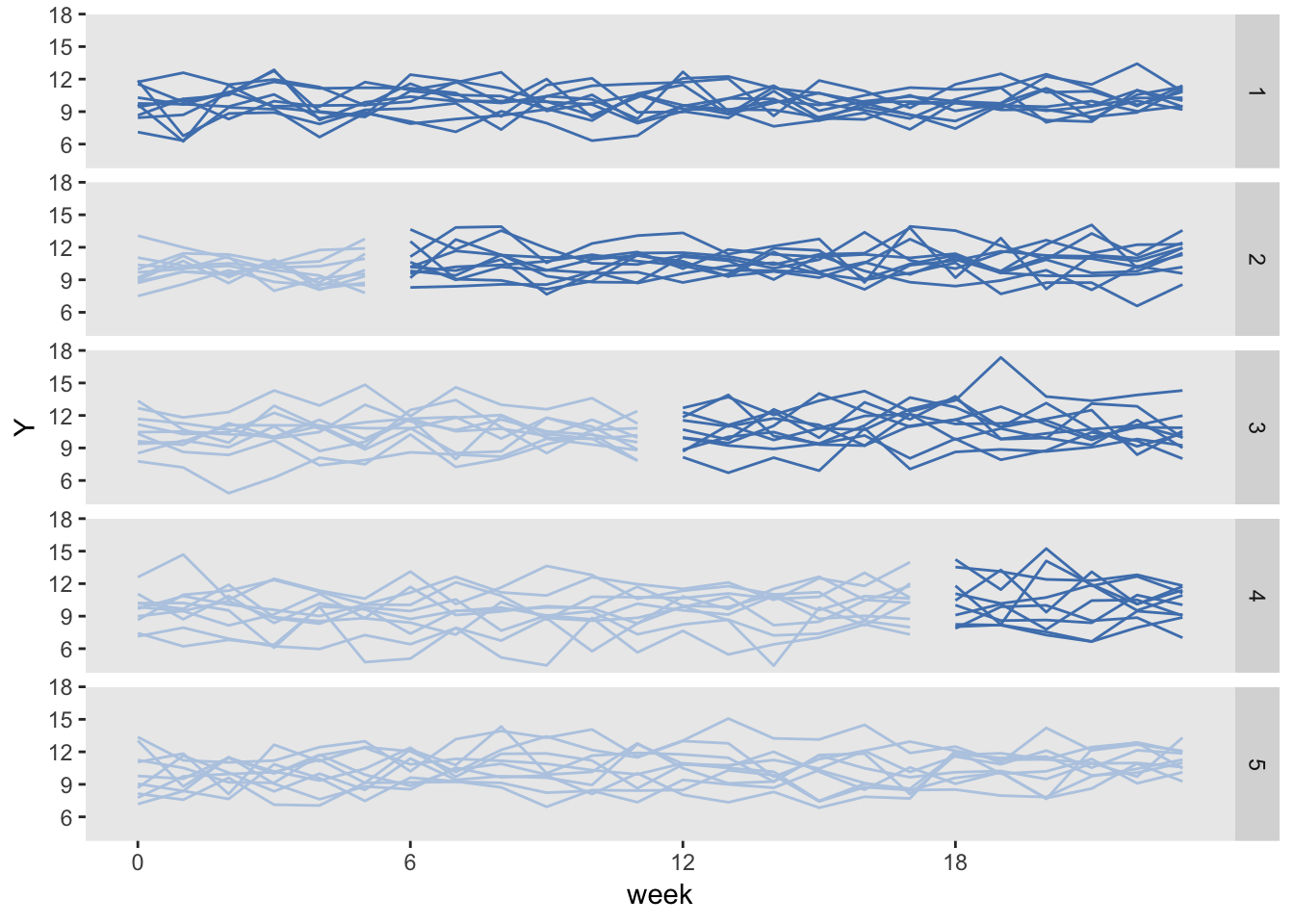
## 3: 11 6 -0.1736741 6 1

## 4: 12 6 -0.4065988 6 1

## 5: 49 6 2.4856616 24 1

## 6: 50 6 1.9599817 24 1

And the data generated under this scenario looks like:



Here is the model estimation:

tidy(lmer(Y ~ rx + period + (1|site), data = dI))

## term estimate std.error statistic group

## 1 (Intercept) 9.79022407 0.185294936 52.835897 fixed

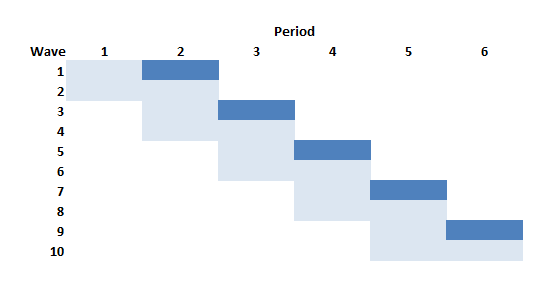
## 2 rx 0.30707559 0.122414620 2.508488 fixed

## 3 period 0.02291619 0.006378367 3.592800 fixed

## 4 sd\_(Intercept).site 1.21153700 NA NA site

## 5 sd\_Observation.Residual 2.99490926 NA NA Residual

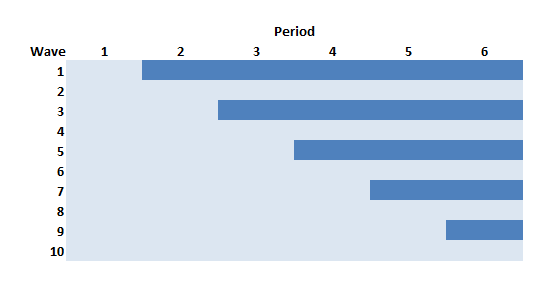
**Staggered cluster randomized trial**



If we wanted to conduct a cluster randomized trial but were able to phase in the intervention over time as we have been assuming, this design is the closest we could get. In this example with 50 sites and five phase-in periods, the intervention waves (in this example 1, 3, 5, 7, and 9) would each include five clusters. The respective control waves (2, 4, 6, 8, and 10) would also have five clusters each. And since we are assuming five waves, each wave will be in the study for eight: the first four weeks comprise “pre” measurement period, and the second four week period is the “post” measurement period.

The problem with this design relative to all the others discussed here is that the amount of data collected for each site is considerably reduced. As a result, this design is going to be much less efficient (hence less powerful) than the others. So much so, that I do not even generate data for this design (though I did actually confirm using simulations not shown here.)

**Staggered cluster randomized trial with continued measurement**

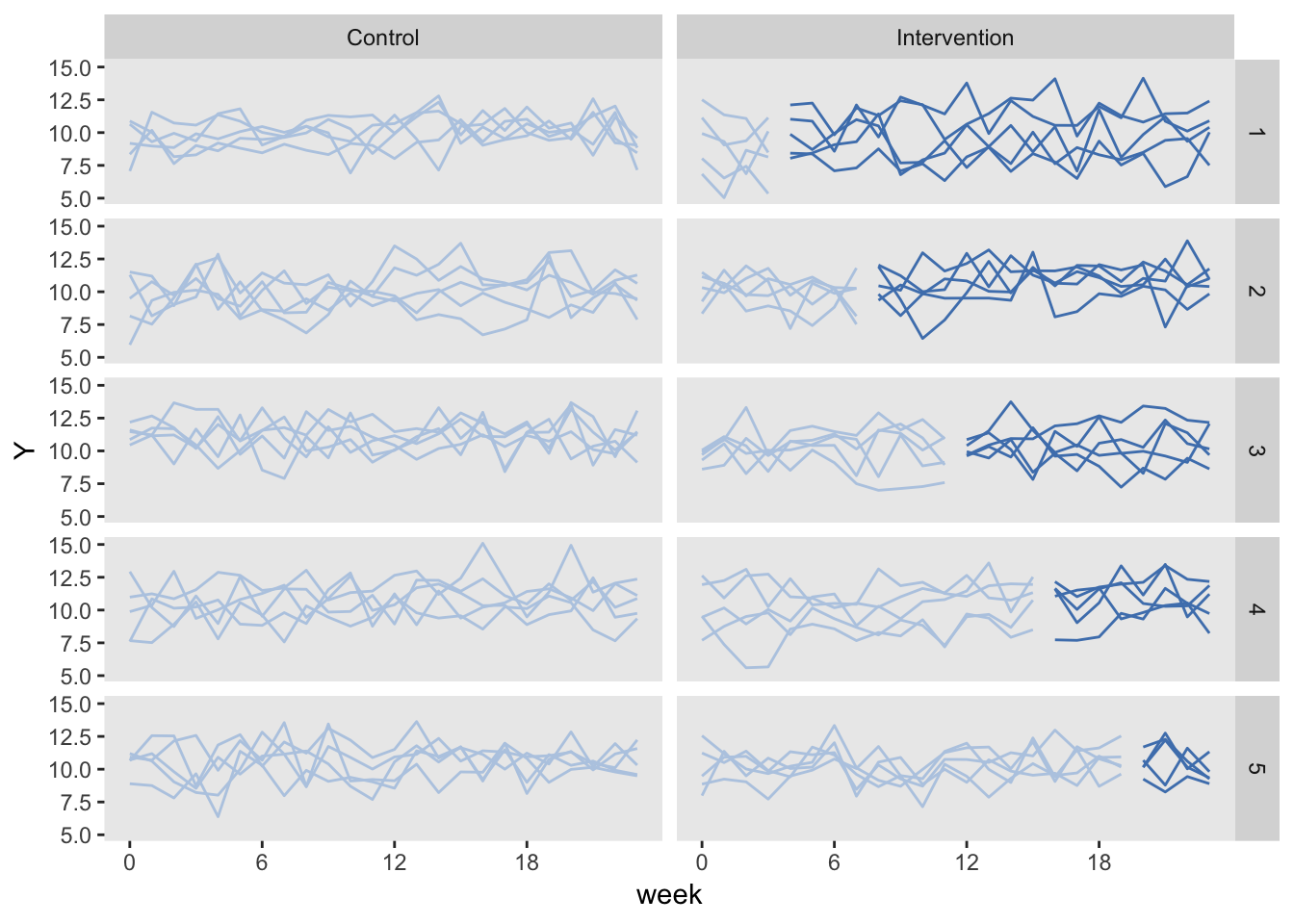


This is the staggered CRT just described, but we collect data for all 24 weeks for all of the sites. In this case, we are not at disadvantage with respect to the number of measurements, so it might be a competitive design. This version of staggered CRT could also be viewed as a traditional stepped-wedge design with controls.

The data generation is identical to the traditional stepped-wedge design we started with, except the only half of the sites are “ever treated”:

dS[, everTrt := rep(0:1)]

Here is the plot, with the control arm on the left, and the intervention arm on the right. The control arm is never introduced to the intervention.



**Conducting a power analysis using simulation**

We are ultimately interested in assessing how much information each study design can provide. Power analyses under different conditions are one way to measure this.

Since one of my missions here is to illustrate as much R code as possible, here is how I do conduct the power analysis of the traditional stepped-wedge design:

powerStepWedge1 <- **function**(x) {

# generate data

dS <- genData(50, defS)

dS[, start := rep((1:5)\*4, each = 10)]

dS[, everTrt := 1]

dP <- addPeriods(dtName = dS, nPeriods = 24, idvars = "site")

dP <- addColumns(defP, dP)

dI <- genCluster(dtClust = dP, cLevelVar = "timeID",

numIndsVar = "n", level1ID = "id")

dI <- addColumns(defI, dI)

# fit model

data.frame(summary(lmer(Y ~ rx + period + (1|site), data = dI))$coef)

}

res <- vector("list", length = 5)

i <- 0

**for** (icc **in** seq(0, 0.04, .01)) {

i <- i + 1

# update data definition based on new ICC

between.var <- iccRE(ICC = icc, dist = "normal", varWithin = 9)

defS <- updateDef(defS, changevar = "siteInt", newvariance = between.var)

# generate 200 data sets and fit models

resSW1<- lapply(1:200, FUN = powerStepWedge1)

# estimate and store power

pSW1 <- mean( unlist(lapply(resSW1, `[`, 2, 3 )) >= 1.96)

res[[i]] <- data.table(icc, pSW1)

}

rbindlist(res)

## icc pSW1

## 1: 0.00 0.940

## 2: 0.01 0.855

## 3: 0.02 0.850

## 4: 0.03 0.830

## 5: 0.04 0.780

**Comparing power of three different designs**

The next figure shows the estimated power for all three designs based on the same effect size and a range of ICC’s. The SW rollout only design consistently equals or outperforms the others. When the ICC is moderate to large (in this case > 0.06), the traditional SW design performs equally well. The design that comes closest to a staggered cluster randomized trial, the SW + controls performs well here on the lower range of ICCs, but is less compelling with more between site variation.

