A few weeks ago, The annual meeting of the [NIH Collaboratory](https://rethinkingclinicaltrials.org/), which is an innovative collection of collaboratory cores, demonstration projects, and NIH Institutes and Centers that is developing new models for implementing and supporting large-scale health services research. A study I am involved with – *Primary Palliative Care for Emergency Medicine* – is one of the demonstration projects in this collaboratory.

The second day of this meeting included four panels devoted to the design and analysis of embedded pragmatic clinical trials, and focused on the challenges of conducting rigorous research in the real-world context of a health delivery system. The keynote address that started off the day was presented by David Murray of NIH, who talked about the challenges and limitations of cluster randomized trials.

In particular, Dr. Murray talked a great deal about stepped-wedge designs, which have become a quite popular tool in health services research. A big takeaway from the talk was that we must be cognizant of the underlying assumptions of the models used to estimate treatment effects; being unaware can lead to biased estimates of treatment effects, or more likely, biased estimates of uncertainty.

**Intra-cluster correlations**

If outcomes of subjects in a study are correlated in any way (e.g. they received care from the same health care provider), we do not learn as much information from each individual study participant as we would in the case where there is no correlation. In a parallel designed cluster randomized trial (where half of the clusters receive an intervention and the other half do not), we expect that the outcomes will be correlated *within* each cluster, though not *across* clusters. (This is not true if the clusters are themselves clustered, in which case we would have a 2-level clustered study.) This intra-cluster correlation (ICC) increases sample size requirements and reduces precision/power.

A common way to model correlation explicitly in a cluster randomized trial is to conceive of a random effects model like this:

\[(1) \qquad \qquad Y\_{ic} = \mu + \beta\_1X\_{c} + b\_c + e\_{ic},\]

where \(Y\_{ic}\) is a continuous outcome for subject \(i\) in cluster \(c\), and \(X\_c\) is a treatment indicator for cluster \(c\) (either 0 or 1). The underlying structural parameters are \(\mu\), the grand mean, and \(\beta\_1\), the treatment effect. The unobserved random effects are, \(b\_c \sim N(0, \sigma^2\_b)\), the normally distributed group level effect, and \(e\_{ic} \sim N(0, \sigma^2\_e)\), the normally distributed individual-level effect. (This is often referred to as the “error” term, but that doesn’t adequately describe what is really unmeasured individual variation.)

The correlation between any two subjects \(i\) and \(j\) in the *same* cluster \(c\) is:

\[ cor(Y\_{ic}, Y\_{jc}) = \frac{cov(Y\_{ic}, Y\_{jc})} {\sqrt {var(Y\_{ic})var(Y\_{jc})}} \]

\(cov(Y\_{ic}, Y\_{jc})\) can be written in terms of the parameters in the underlying data generating process:

\[  
\begin{aligned}  
cov(Y\_{ic}, Y\_{jc}) &= cov(\mu + \beta\_1X\_c + b\_c + e\_{ic}, \mu + \beta\_1X\_c + b\_c + e\_{jc}) \\  
&=cov(b\_c, b\_c) + cov(e\_{ic},e\_{jc} ) \\  
&=\sigma^2\_b + 0 \\  
&=\sigma^2\_b  
\end{aligned}  
\]

The terms simplify since the cluster level effects are independent of the individual level effects (and all the fixed effects in the model) and the individual level effects are independent of each other. The within-period intra-cluster co-variance depends only on the between cluster variation.

The total variance of the outcomes \(Y\_{ic}\) is:

\[  
\begin{aligned}  
var(Y\_{ic}) &= var(\mu + \beta\_1X\_c + b\_c + e\_{ic}) \\  
&= var(b\_c) + var(e\_{ic}) \\  
&= \sigma^2\_b + \sigma^2\_e  
\end{aligned}  
\]

Substituting all of this into the original equation gives us the intra-cluster correlation for any two subjects in the cluster:

\[  
\begin{aligned}  
cor(Y\_{ic}, Y\_{jc}) &= \frac{cov(Y\_{ic}, Y\_{jc})} {\sqrt {var(Y\_{ic})var(Y\_{jc})}} \\  
\\  
ICC &= \frac{\sigma^2\_b}{\sigma^2\_b + \sigma^2\_e}  
\end{aligned}  
\]

So, the correlation between any two subjects in a cluster increases as the variation *between* clusters increases.

**Cluster randomization when time matters**

Moving beyond the parallel design to the stepped-wedge design, time starts to play a very important role. It is important to ensure that we do not confound treatment and time effects; we have to be careful that we do not attribute the general changes over time to the intervention. This is accomplished by introducing a time trend into the model. (Actually, it seems more common to include a time-specific effect so that each time period has its own effect. However, for simulation purposes, I will will assume a linear trend.)

In the stepped-wedge design, we are essentially estimating within-cluster treatment effects by comparing the cluster with itself pre- and post-intervention. To estimate sample size and precision (or power), it is no longer sufficient to consider a single ICC, because there are now multiple ICC’s – the within-period ICC and the between-period ICC’s. The within-period ICC is what we defined in the parallel design (since we effectively treated all observations as occurring in the same period.) Now we also need to consider the expected correlation of two individuals in the *same* cluster in *different* time periods.

If we do not properly account for within-period ICC and the between-period ICC’s in either the planning or analysis stages, we run the risk of generating biased estimates.

My primary aim is to describe possible data generating processes for the stepped wedge design and what implications they have for both the within-period and between-period ICC’s. I will generate data to confirm that observed ICC’s match up well with the theoretical expectations. This week I will consider the simplest model, one that is frequently used but whose assumptions may not be realistic in many applications. In a follow-up post, I will consider more flexible data generating processes.

**Constant ICC’s over time**

Here is probably the simplest model that can be conceived for a process underlying the stepped-wedge design:

\[  
(2) \qquad \qquad Y\_{ict} = \mu + \beta\_0t + \beta\_1X\_{ct} + b\_c + e\_{ict}  
\]

As before, the unobserved random effects are \(b\_c \sim N(0, \sigma^2\_b)\) and \(e\_{ict} \sim N(0, \sigma^2\_e)\). The key differences between this model compared to the parallel design is the time trend and time-dependent treatment indicator. The time trend accounts for the fact that the outcome may change over time regardless of the intervention. And since the cluster will be in both the control and intervention states we need to have an time-dependent intervention indicator. (This model is a slight variation on the *Hussey and Hughes* model, which includes a time-specific effect \(\beta\_t\) rather than a linear time trend.

The *within-period* ICC from this is model is:

\[  
\begin{aligned}  
cor(Y\_{ict}, Y\_{jct}) &= cor(\mu + \beta\_0t + \beta\_1X\_{ct} + b\_c + e\_{ict}, \ \mu + \beta\_0t + \beta\_1X\_{ct} + b\_c + e\_{jct}) \\  
\\  
ICC\_{tt}&= \frac{\sigma^2\_b}{\sigma^2\_b + \sigma^2\_e}  
\end{aligned}  
\]

I have omitted the intermediary steps, but the logic is the same as in the parallel design case. The within-period ICC under this model is also the same as the ICC in the parallel design.

More importantly, in this case the *between-period* ICC turns out to be the same as the *within-period* ICC. For the *between-period* ICC, we are estimating the expected correlation between any two subjects \(i\) and \(j\) in cluster \(c\), one in time period \(t\) and the other in time period \(t^\prime\) \((t \ne t^\prime)\):

\[  
\begin{aligned}  
cor(Y\_{ict}, Y\_{jct^\prime}) &= cor(\mu + \beta\_0t + \beta\_1X\_{ct} + b\_c + e\_{ict}, \ \mu + \beta\_0t^\prime + \beta\_1X\_{ct^\prime} + b\_c + e\_{jct^\prime}) \\  
\\  
ICC\_{tt^\prime}&= \frac{\sigma^2\_b}{\sigma^2\_b + \sigma^2\_e}  
\end{aligned}  
\]

Under this seemingly reasonable (and popular) model, we are making a big assumption that the within-period ICC and between-period ICC’s are equal and constant throughout the study. This may or may not be reasonable – but it is important to acknowledge the assumption and to make sure we justify that choice.

**Generating data to simulate a stepped-wedge design**

I’ve generated data from a stepped-wedge design before, but will repeat the details here. For the data definitions, we define the variance of the cluster-specific effects, the cluster sizes, and the outcome model.

defc <- defData(varname = "ceffect", formula = 0, variance = 0.15,

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

defc <- defData(defc, "m", formula = 10, dist = "nonrandom")

defa <- defDataAdd(varname = "Y",

formula = "0 + 0.10 \* period + 1 \* rx + ceffect",

variance = 2, dist = "normal")

The data generation follows this sequence: cluster data, temporal data, stepped-wedge treatment assignment, and individual (within cluster) data:

dc <- genData(100, defc)

dp <- addPeriods(dc, 7, "cluster")

dp <- trtStepWedge(dp, "cluster", nWaves = 4, lenWaves = 1, startPer = 2)

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

dd <- addColumns(defa, dd)

dd

## cluster period ceffect m timeID startTrt rx id Y

## 1: 1 0 -0.073 10 1 2 0 1 -2.12

## 2: 1 0 -0.073 10 1 2 0 2 -1.79

## 3: 1 0 -0.073 10 1 2 0 3 1.53

## 4: 1 0 -0.073 10 1 2 0 4 -1.44

## 5: 1 0 -0.073 10 1 2 0 5 2.25

## ---

## 6996: 100 6 0.414 10 700 5 1 6996 1.28

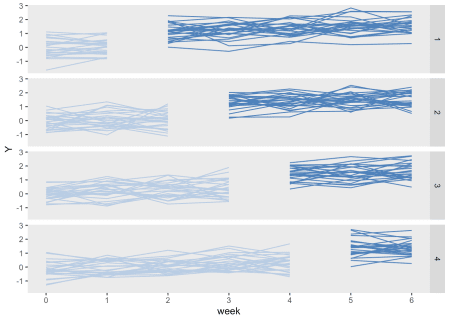
## 6997: 100 6 0.414 10 700 5 1 6997 0.30

## 6998: 100 6 0.414 10 700 5 1 6998 0.94

## 6999: 100 6 0.414 10 700 5 1 6999 1.43

## 7000: 100 6 0.414 10 700 5 1 7000 0.58

It is always useful (and important) to visualize the data (regardless of whether they are simulated or real). This is the summarized cluster-level data. The clusters are grouped together in waves defined by starting point. In this case, there are 25 clusters per wave. The light blue represents pre-intervention periods, and the dark blue represents intervention periods.



**Estimating the between-period within-cluster correlation**

I want to estimate the observed between-period within cluster correlation without imposing any pre-conceived structure. In particular, I want to see if the data generated by the process defined in equation (2) above does indeed lead to constant within- and between-period ICC’s. In a future post, I will estimate the ICC using a model, but for now, I’d prefer to estimate the ICC’s directly from the data.

He gives this set of equations to find the correlation coefficient \(\rho\_{tt^\prime}\) for two time periods \(t\) and \(t^\prime\). In the equations, \(m\_{ct}\) represents the cluster size for cluster \(c\) in time period \(t\), and \(K\) represents the number of clusters:

\[  
\rho\_{tt^\prime} = \frac{\sum\_{c=1}^K \sum\_{i=1}^{m\_{ct}} \sum\_{j=1}^{m\_{ct^\prime}} (Y\_{ict}-\mu\_t)(Y\_{jct^\prime}-\mu\_{t^\prime})} {\left[ \left ( \sum\_{c=1}^K m\_{ct^\prime} \sum\_{i=1}^{m\_{ct}} (Y\_{ict}-\mu\_t)^2 \right ) \left ( \sum\_{c=1}^K m\_{ct} \sum\_{j=1}^{m\_{ct^\prime}} (Y\_{jct^\prime}-\mu\_{t^\prime})^2 \right )\right] ^ \frac {1}{2}}  
\]

\[  
\mu\_t = \frac{\sum\_{c=1}^K m\_{ct} m\_{ct^\prime} \mu\_{ct}}{\sum\_{c=1}^K m\_{ct} m\_{ct^\prime}} \ \ , \ \ \mu\_{t^\prime} = \frac{\sum\_{c=1}^K m\_{ct} m\_{ct^\prime} \mu\_{ct^\prime}}{\sum\_{c=1}^K m\_{ct} m\_{ct^\prime}}  
\]

\[  
\mu\_{ct} = \frac{\sum\_{i=1}^{m\_{ct}} Y\_{ict}}{m\_{ct}} \ \ , \ \ \mu\_{ct^\prime} = \frac{\sum\_{j=1}^{m\_{ct^\prime}} Y\_{jct^\prime}}{m\_{ct^\prime}}  
\]

I’ve implemented the algorithm in R, and the code is included in the addendum. One issue that came up is that as the intervention is phased in over time, the treatment effect is present for each at different times. The algorithm breaks down as a result. However, the between-period ICC can be calculated for each wave, and then we can average across the four waves.

The *within-period* ICC is estimated using a linear mixed effects model applied to each period separately, so that we estimate period-specific within-period ICC’s. The expected (constant) ICC is \(0.07 = \left(\frac{0.15}{0.15 + 2}\right)\).

The function iccs (shown below in the addendum) returns both the estimated *within-* and *between-cluster* ICC’s for a single data set. Here is the within-period ICC for the first period (actually period 0) and the between-period ICC’s using period 0:

set.seed(47463)

iccs(dd, byWave = T)[,c(22, 0:6)]

## wp0 bp01 bp02 bp03 bp04 bp05 bp06

## 1: 0.041 0.068 0.073 0.08 0.067 0.054 0.053

ICC estimates are quite variable and we can’t tell anything about the distribution from any single data set. Generating multiple replications lets us see if the estimates are close, on average, to our assumption of constant ICC’s. Here is a function to generate a single data set:

genDD <- function(defc, defa, nclust, nperiods, waves, len, start) {

dc <- genData(nclust, defc)

dp <- addPeriods(dc, nperiods, "cluster")

dp <- trtStepWedge(dp, "cluster", nWaves = waves,

lenWaves = len, startPer = start)

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

dd <- addColumns(defa, dd)

return(dd[])

}

And here is a function to estimate 200 sets of ICC’s for 200 data sets:

icc <- mclapply(1:200,

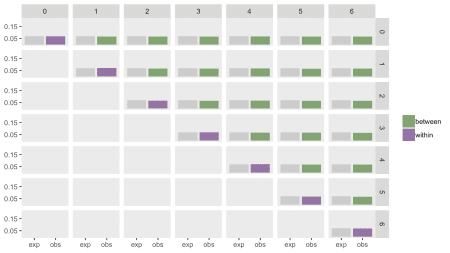
function(x) iccs(genDD(defc, defa, 100, 7, 4, 1, 2), byWave = T),

mc.cores = 4

)

observed <- sapply(rbindlist(icc), function(x) mean(x))

Averages of all the *within-* and *between-period* ICC’s were in fact quite close to the “true” value of 0.07 based on a relatively small number of replications. The plot shows the observed averages along side the expected value (shown in gray) for each of the periods generated in the data. There is little variation across both the *within-* and *between-period* ICC’s.



I’ll give you a little time to absorb this. Next time, I will consider alternative data generating processes where the the ICC’s are not necessarily constant.

References:

Kasza, J., K. Hemming, R. Hooper, J. N. S. Matthews, and A. B. Forbes. “Impact of non-uniform correlation structure on sample size and power in multiple-period cluster randomised trials.” *Statistical methods in medical research* (2017): 0962280217734981.

Rosner, Bernard. “On the estimation and testing of inter-class correlations: the general case of multiple replicates for each variable.” *American journal of epidemiology* 116, no. 4 (1982): 722-730.

**Addendum: R code for simulations**

library(lme4)

library(parallel)

Covar <- function(dx, clust, period1, period2, x\_0, x\_1) {

v0 <- dx[ctemp == clust & period == period1, Y - x\_0]

v1 <- dx[ctemp == clust & period == period2, Y - x\_1]

sum(v0 %\*% t(v1))

}

calcBP <- function(dx, period1, period2) {

# dx <- copy(d2)

# create cluster numbers starting from 1

tt <- dx[, .N, keyby = cluster]

nclust <- nrow(tt)

dx[, ctemp := rep(1:nclust, times = tt$N)]

dx <- dx[period %in% c(period1, period2)]

## Grand means

dg <- dx[, .(m=.N, mu = mean(Y)), keyby = .(ctemp, period)]

dg <- dcast(dg, formula = ctemp ~ period, value.var = c("m","mu"))

setnames(dg, c("ctemp", "m\_0", "m\_1", "mu\_0", "mu\_1"))

x\_0 <- dg[, sum(m\_0 \* m\_1 \* mu\_0)/sum(m\_0 \* m\_1)]

x\_1 <- dg[, sum(m\_0 \* m\_1 \* mu\_1)/sum(m\_0 \* m\_1)]

## Variance (denominator)

dss\_0 <- dx[period == period1, .(ss\_0 = sum((Y - x\_0)^2)),

keyby = ctemp]

dss\_0[, m\_1 := dg[, m\_1]]

v\_0 <- dss\_0[, sum(m\_1 \* ss\_0)]

dss\_1 <- dx[period == period2, .(ss\_1 = sum((Y - x\_1)^2)),

keyby = ctemp]

dss\_1[, m\_0 := dg[, m\_0]]

v\_1 <- dss\_1[, sum(m\_0 \* ss\_1)]

## Covariance

v0v1 <- sapply(1:nclust,

function(x) Covar(dx, x, period1, period2, x\_0, x\_1))

bp.icc <- sum(v0v1)/sqrt(v\_0 \* v\_1)

bp.icc

}

btwnPerICC <- function(dd, period1, period2, byWave = FALSE) {

if (byWave) {

waves <- dd[, unique(startTrt)]

bpICCs <- sapply(waves, function(x)

calcBP(dd[startTrt==x], period1, period2))

return(mean(bpICCs))

} else {

calcBP(dd, period1, period2)

}

}

withinPerICC <- function(dx) {

lmerfit <- lmer(Y~rx + (1|cluster), data = dx)

vars <- as.data.table(VarCorr(lmerfit))[, vcov]

vars[1]/sum(vars)

}

genPairs <- function(n) {

x <- combn(x = c(1:n-1), 2)

lapply(seq\_len(ncol(x)), function(i) x[,i])

}

iccs <- function(dd, byWave = FALSE) {

nperiods <- dd[, length(unique(period))]

bperiods <- genPairs(nperiods)

names <-

unlist(lapply(bperiods, function(x) paste0("bp", x[1], x[2])))

bp.icc <- sapply(bperiods,

function(x) btwnPerICC(dd, x[1], x[2], byWave))

system(paste("echo ."))

bdd.per <- lapply(1:nperiods - 1, function(x) dd[period == x])

wp.icc <- lapply(bdd.per,

function(x) withinPerICC(x))

wp.icc <- unlist(wp.icc)

nameswp <- sapply(1:nperiods - 1, function(x) paste0("wp", x))

do <- data.table(t(c(bp.icc, wp.icc)))

setnames(do, c(names, nameswp))

return(do[])

}

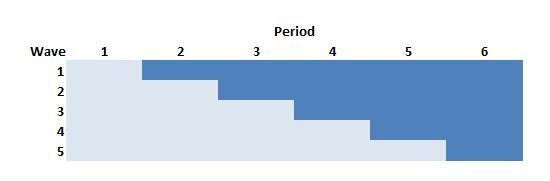
Stepped Design

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))) +

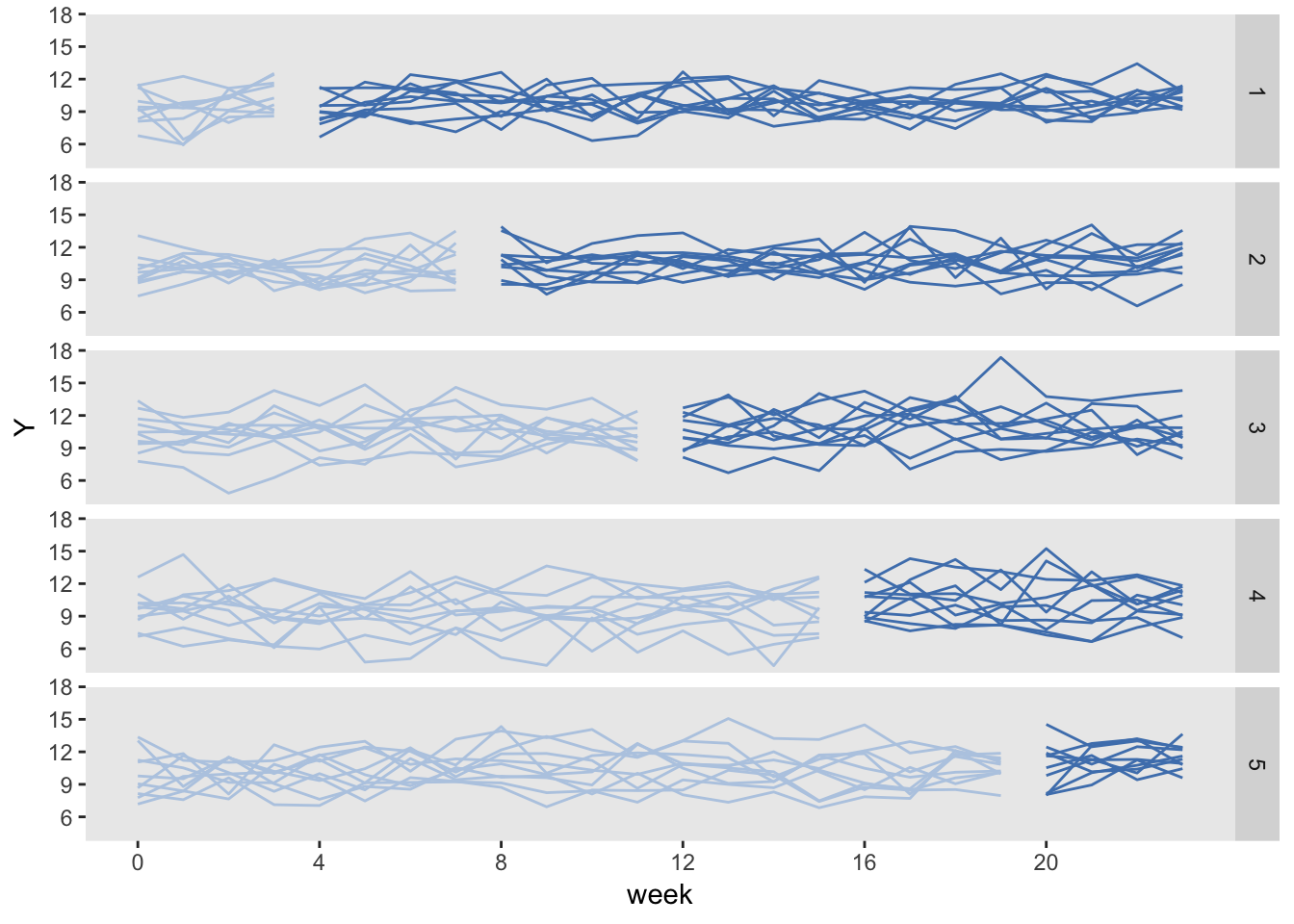
facet\_grid(factor(start, labels = c(1 : 5)) ~ .) +

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

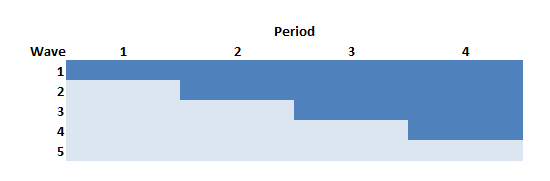
## 2 rx 0.35246094 0.122453829 2.878317 fixed

## 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

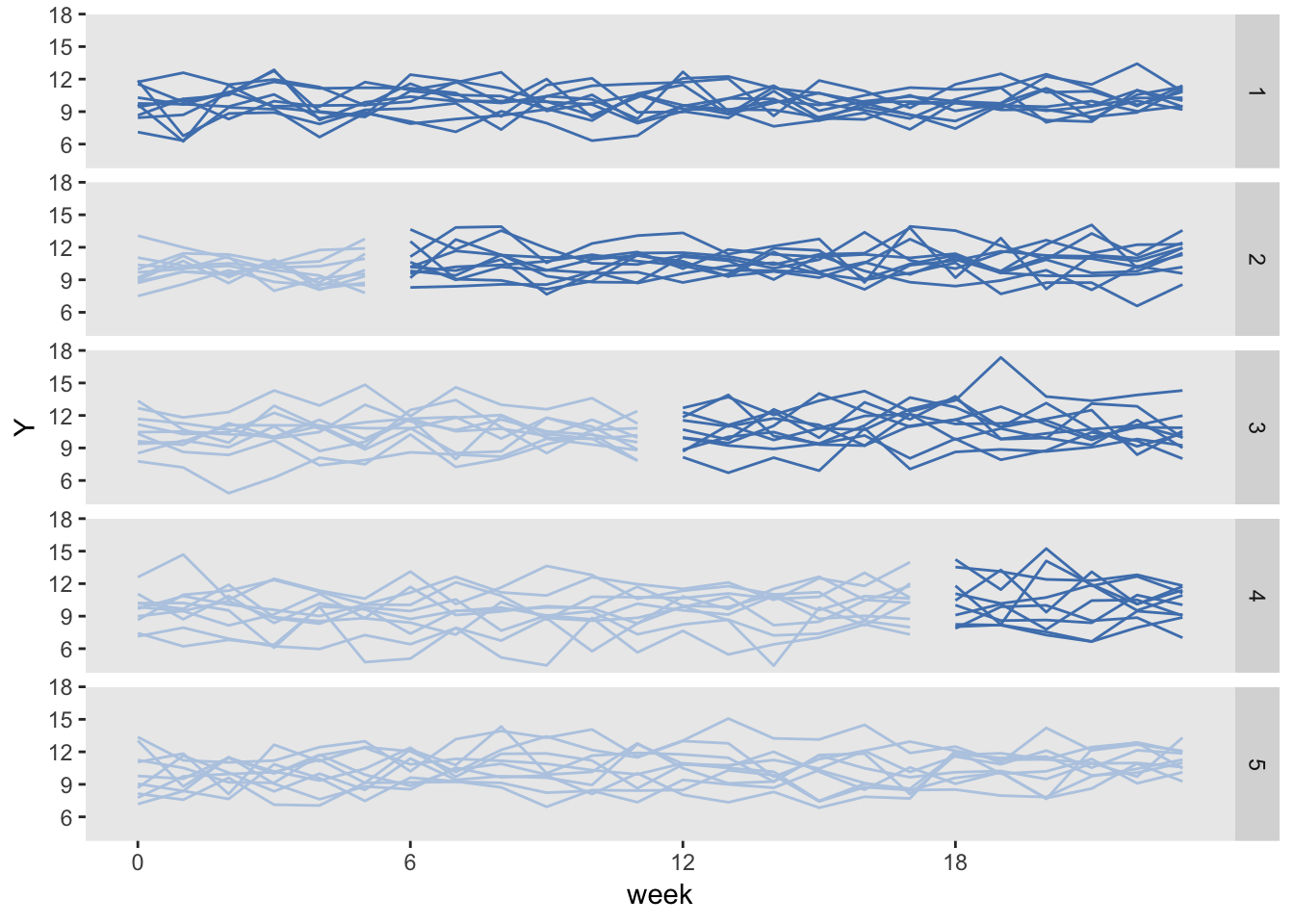
## 3: 11 6 -0.1736741 6 1

## 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

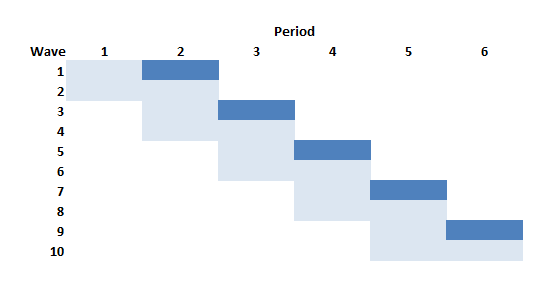
## 2 rx 0.30707559 0.122414620 2.508488 fixed

## 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

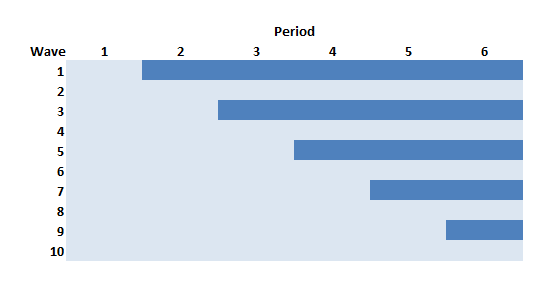
**Staggered cluster randomized trial**



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**

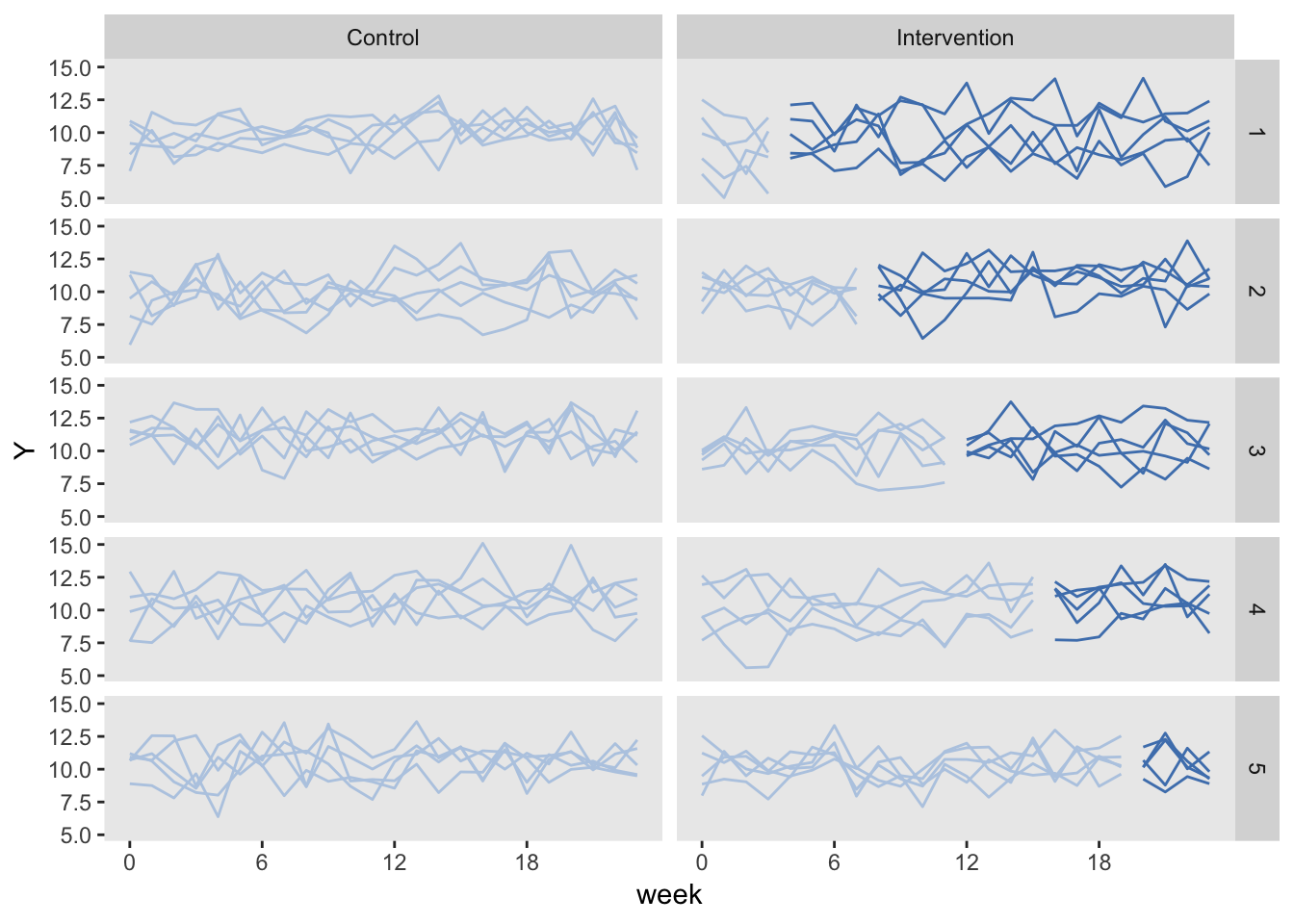


This is the staggered CRT just described, but we collect data for all 24 weeks for all of the sites. In this case, we are not at disadvantage with respect to the number of measurements, so it might be a competitive design. This version of staggered CRT could also be viewed as a traditional stepped-wedge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

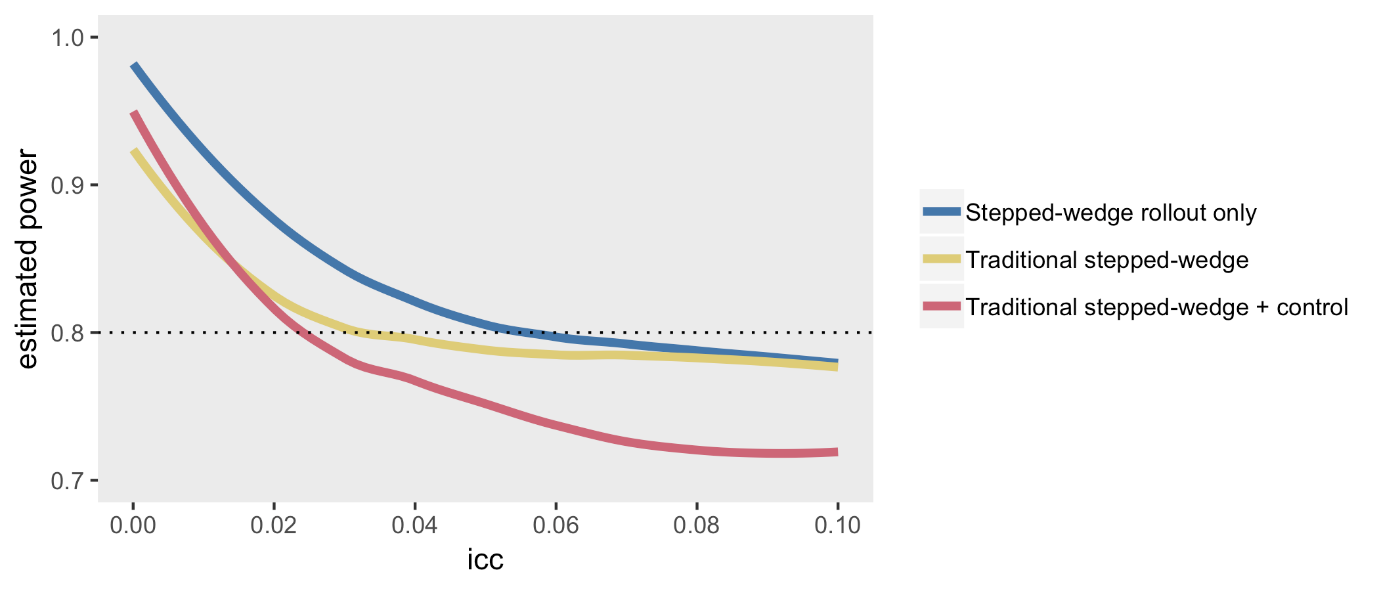
## 3: 0.02 0.850

## 4: 0.03 0.830

## 5: 0.04 0.780

**Comparing power of three different designs**

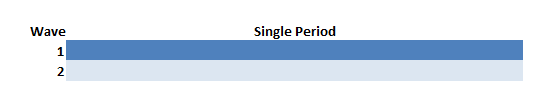
The next figure shows the estimated power for all three designs based on the same effect size and a range of ICC’s. The SW rollout only design consistently equals or outperforms the others. When the ICC is moderate to large (in this case > 0.06), the traditional SW design performs equally well. The design 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.



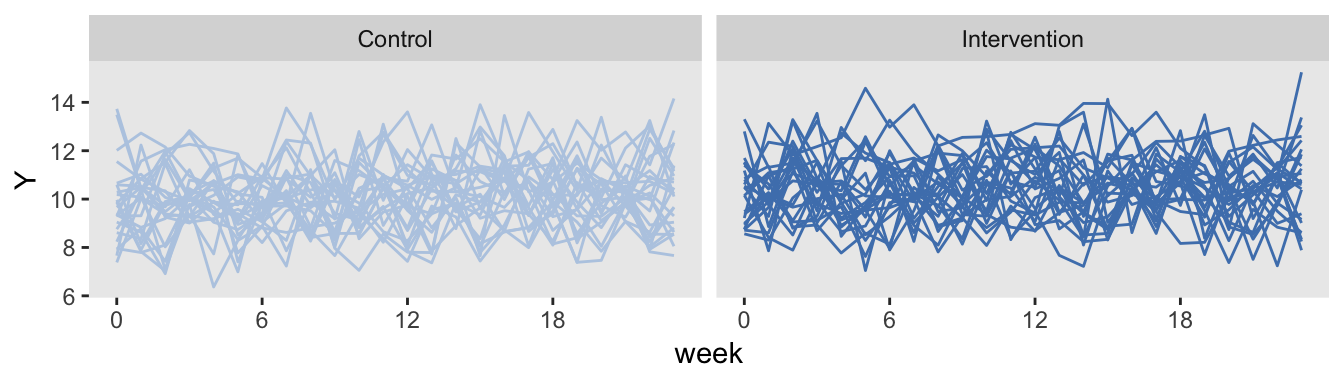
[Thompson et al.](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5718336/) provide more nuance that can improve power under different conditions - mostly involving changing period lengths or adding control-only sites, or both - but these simulations suggest that some sort of stepped-wedge design (either limited to the rollout phase or not) will generally be advantageous, at least under the strict requirements that I established to frame the design.

All of this has been done in the context of a normally distributed outcome. At some point, I will certainly re-do this comparison with a binary outcome.

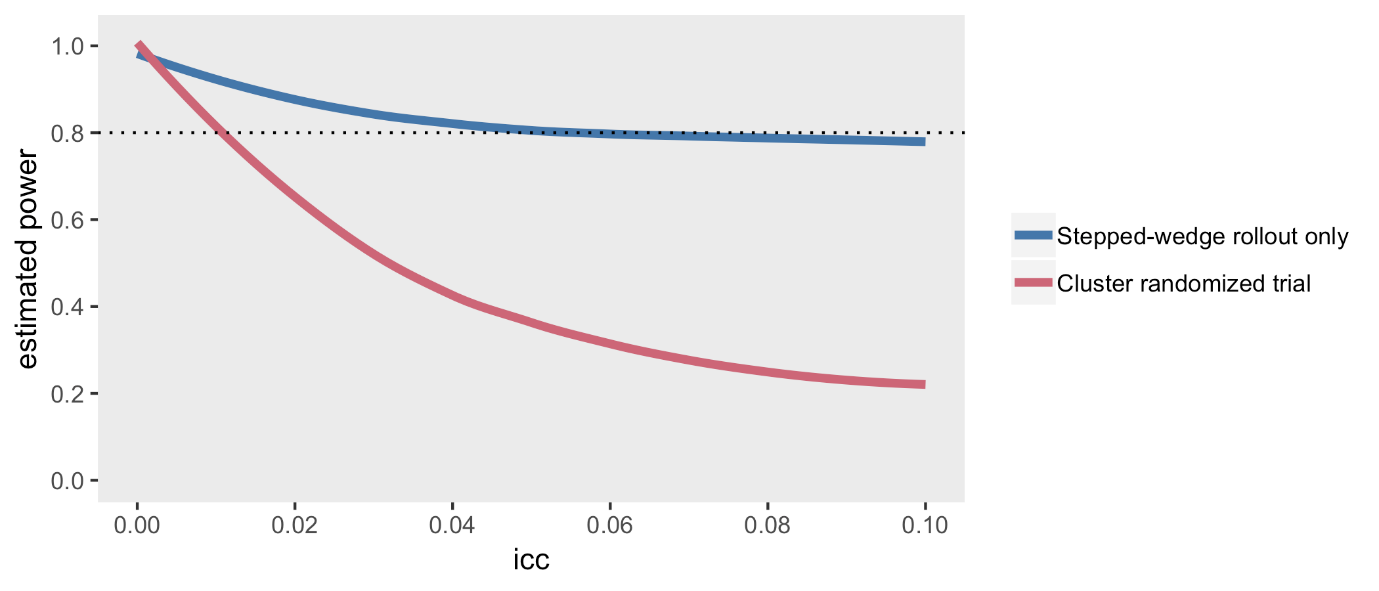
**Addendum: cluster randomized trial**



A traditional cluster randomized trial was not really under consideration because we declared that we could only deliver the intervention to 10 sites at any one time. However, it is illustrative to compare this design to make it clear that CRT is really best used when variability across sites is at its lowest (i.e. when the ICC is at or very close to zero). In this example, 25 sites are randomized to receive the intervention starting in the first week and 25 sites never receive the intervention. Data are collected for all 24 weeks for each of the 50 clusters.



The simulations confirm findings that the CRT is more efficient than stepped-wedge designs when the ICC is close to zero, but pales in comparison even with ICCs as low as 0.01:



Data Definition of Stepped-Wedge Design

### 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}*Yict*​=*β*0​+*bc*​+*β*1​∗*t*+*β*2​∗*Xct*​+*eict*​

where Y\_{ict}*Yict*​ is the outcome for individual i*i* in cluster c*c* in time period t*t*, b\_c*bc*​ is a cluster-specific effect, X\_{ct}*Xct*​ is the intervention indicator that has a value 1 during periods where the cluster is under the intervention, and e\_{ict}*eict*​ is the individual-level effect. Both b\_c*bc*​ and e\_{ict}*eict*​ are normally distributed with mean 0 and variances \sigma^2\_{b}*σb*2​ and \sigma^2\_{e}*σe*2​, respectively. \beta\_1*β*1​ is the time trend, and \beta\_2*β*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*σb*2​=0.20. In addition, \sigma^2\_e = 1.75*σe*2​=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))) +

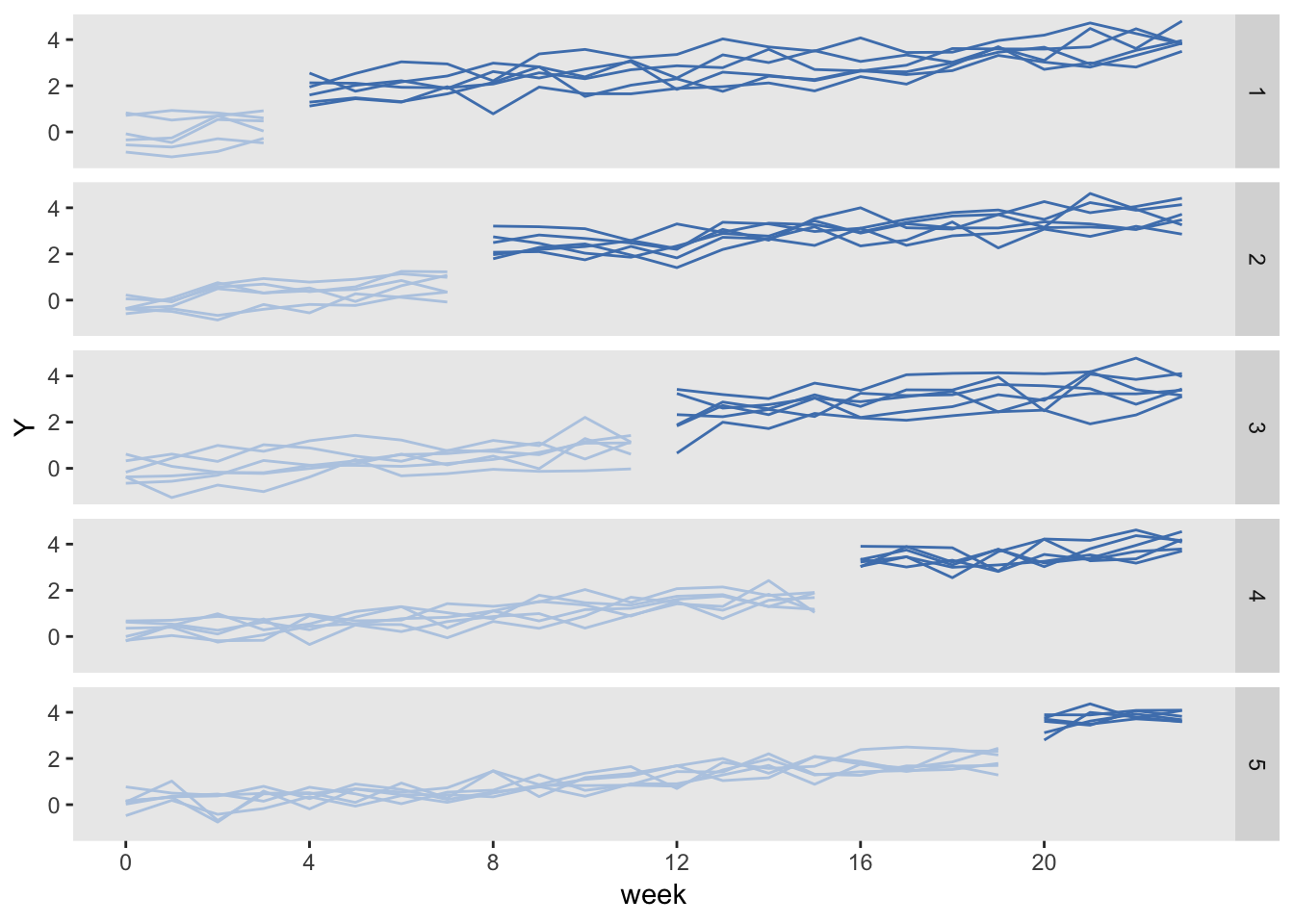
facet\_grid(factor(startTrt, labels = c(1 : 5)) ~ .) +

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.