Matching for Adjustment and Causal Inference
Class 3: Information, Estimation, Testing

Jake Bowers

August 02, 2023

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

So Far I

1. **Regression is not research design**: Linear additive adjustment must be justified, extrapolation and influential points investigated, hard to do without looking at outcomes and risking multiple testing problems.
2. Stratification is research design (can be done without looking at outcomes, can be **evaluated** without looking at outcomes.)
3. We can create stratified designs by minimizing distances/differences: we can focus on one or more covariates and can represent substantive ideas with calipers, exact matching, and combinations of distance matrices for example: `distMat <- psdist + caliper(agedist,10) + caliper(incomedist,100)`
4. We can evaluate stratified designs (a) substantively (ex. worst differences within set on key covariate) and (b) by comparison to an equivalent randomized design.
5. New challenges:
    - (From Rosenbaum Chapter "Chapter 9 Basic Tools of Multivariate Matching"): Ordinary Mahalanobis distances can trick us into thinking that observations are farther apart than it should be so he suggests a rank-based version.

So Far II

- Propensity scores produced using logit models with many covariates and few observations can make it hard to find matches (because of the "separation problem" in logistic regression).
- Problems of memory for large matching problems. (Use `exactMatch` or `caliper` or the `bigmatch` package etc..)
- How to decide on a single design?

Decision Points in Creating Matched Designs

- Which covariates and their scaling and coding. (For example, exclude covariates with no variation!)
- Which distance matrices (scalar distances for one or two important variables, Mahalanobis distances (rank transformed or not), Propensity distances (using linear predictors)).
- (Possibly) which calipers (and how many, if any, observations to drop. Note about ATT as a random quantity and ATE/ACE as fixed.)
- (Possibly) which exact matching or strata
- (Possibly) which structure of sets (how many treated per control, how many controls per treated)
- Which remaining differences are tolerable from a substantive perspective?
- How well does the resulting research design compare to an equivalent block-randomized study (`xBalance`)?
- (Possibly) How much statistical power does this design provide for the quantity of interest?
- Other questions to ask about a research design aiming to help clarify comparisons.

Example: I

```
thecovs <- unique(c(names(meddat)[c(5:7, 9:24)], "HomRate03"))
balfmla <- reformulate(thecovs[-c(1, 14)], response = "nhTrt")
thebglm <- arm::bayesglm(balfmla, data = meddat, family = binomial(link = "logit"))
mhdist <- match_on(balfmla, data = meddat)
psdist <- match_on(thebglm, data = meddat)
```

Showing here 3 different ways to get a scalar distance:

```
## This is just standardized and centered
hrdist1 <- match_on(nhTrt ~ HomRate03, data = meddat)

## Distance in terms of homicide rate itself
tmp <- meddat$HomRate03
names(tmp) <- rownames(meddat)
hrdist2 <- match_on(tmp, z = meddat$nhTrt, data = meddat)

## By Hand absolute distance after centering and standardizing
tmp <- scale(meddat$HomRate03)[, 1]
names(tmp) <- rownames(meddat)
hrdist3 <- match_on(tmp, z = meddat$nhTrt, data = meddat)

hrdist1[1:3, 1:6]
```

Example: II

```
         control
treatment    401    402    403    404     405    407
      101 0.2496 0.2320 0.5844 0.2188 0.04638 0.6362
      102 0.1911 0.2905 0.5259 0.1603 0.10488 0.5777
      103 0.2168 0.6985 0.1180 0.2476 0.51279 0.1698
```

```
hrdist2[1:3, 1:6]
```

```
         control
treatment    401    402    403    404     405    407
      101 0.4147 0.3854 0.9707 0.3634 0.07703 1.0567
      102 0.3175 0.4826 0.8735 0.2663 0.17422 0.9596
      103 0.3601 1.1602 0.1960 0.4113 0.85180 0.2820
```

```
hrdist3[1:3, 1:6]
```

```
         control
treatment    401    402    403    404     405    407
      101 0.2437 0.2265 0.5705 0.2136 0.04527 0.6210
      102 0.1866 0.2836 0.5134 0.1565 0.10238 0.5639
      103 0.2116 0.6818 0.1152 0.2417 0.50058 0.1657
```

```
## They are all just linear transforms. hrdist2 allows us to talk about homicide rates when we mak
cor(hrdist1[1, ], hrdist2[1, ])
```

```
[1] 1
```

Example: III

```
cor(hrdist1[1, ], hrdist3[1, ])
```

```
[1] 1
```

```
psCal <- quantile(as.vector(psdist), .9)
mhCal <- quantile(as.vector(mhdist), .9)
hrCal <- quantile(as.vector(hrdist2), .9)

## Create a distance matrix reflecting how the covariates relate to treatment,
## to each other (the mahalanobis distance), and also baseline outcome.
matchdist <- psdist + caliper(psdist, psCal) + caliper(mhdist, mhCal) + caliper(hrdist2, 2)
as.matrix(matchdist)[1:3, 1:6]
```

```
       control
treated    401    402    403    404    405 407
    101 2.0277 2.4291 1.8230 1.0042 0.1340 Inf
    102 0.3239 0.7253 0.1191 0.6996 1.5698 Inf
    103 2.2014 2.6028 1.9966 1.1779 0.3077 Inf
```

```
fm0 <- fullmatch(matchdist, data = meddat)
summary(fm0, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

Example: IV

```
Structure of matched sets:
 1:0 11:1  3:1  1:1  1:2 1:14  0:1
   2    1    1    4    1    1    1
Effective Sample Size:  10.5
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       12.1 18   0.844
```

```
fm1 <- fullmatch(matchdist, data = meddat, min.controls = 1)
summary(fm1, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

```
Structure of matched sets:
1:0 1:1 1:3 0:1
  2  19   1   1
Effective Sample Size:  20.5
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       20.9 18   0.283
```

```
fm3 <- fullmatch(matchdist, data = meddat, mean.controls = .9)
summary(fm3, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

Example: V

```
Structure of matched sets:
 1:0 11:1  3:1  1:1  1:2 1:12  0:1
   2    1    1    4    1    1    3
Effective Sample Size:  10.5
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       11.4 18   0.876
```

```
fm0dists <- unlist(matched.distances(fm0, matchdist))
fm1dists <- unlist(matched.distances(fm1, matchdist))

## Next is an example of using a penalty rather than a caliper
maxdist <- max(matchdist[!is.infinite(matchdist)])

psdist01 <- psdist / max(as.matrix(psdist))
mhdist01 <- (mhdist - min(as.matrix(mhdist))) / (max(as.matrix(mhdist)) - min(as.matrix(mhdist)))
hrdist201 <- (hrdist2 - min(as.matrix(hrdist2))) / (max(as.matrix(hrdist2)) - min(as.matrix(hrdis

summary(as.vector(psdist01))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0001  0.1071  0.2133  0.2415  0.3313  1.0000
```

Example: VI

```
summary(as.vector(mhdist01))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.387   0.510   0.509   0.632   1.000
```

```
summary(as.vector(hrdist201))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0266  0.0602  0.1226  0.1211  1.0000
```

Example using Penalties

You can also use penalties rather than calipers:

```
## The larger the differences in psdist, mhdist, and hrdist, the worse the
## matches (by maxdist).
matchdistPen <- psdist + psdist01 * maxdist + mhdist01 * maxdist + hrdist201 * maxdist

## We could also say, "distances larger than some value are really bad":
matchdistPen2 <- psdist + psdist01 * maxdist + mhdist01 * maxdist + (hrdist2 > 2) * maxdist * 100

as.matrix(matchdist)[5:10, 1:8]
```

```
       control
treated   401   402    403    404    405 407   408   409
    105 1.780 2.181 1.5750 0.7562 0.1140 Inf 2.106   Inf
    106 0.874 1.275 0.6693 0.1495 1.0197 Inf 1.200 3.669
    107 2.131 2.532 1.9262 1.1074 0.2372 Inf 2.457   Inf
    108 2.225   Inf 2.0201 1.2014 0.3312 Inf   Inf   Inf
    109 1.310   Inf 1.1054 0.2867    Inf Inf   Inf   Inf
    110 2.289 2.690    Inf 1.2652 0.3949 Inf 2.615   Inf
```

```
matchdistPen[5:10, 1:8]
```

```
         control
treatment   401   402   403   404    405   407   408    409
      105 5.027 5.268 4.845 3.245 2.0226 14.26 5.101  9.168
      106 2.864 3.364 3.314 1.842 2.6641 12.16 4.106  8.118
      107 4.669 5.005 5.664 3.679 0.7794 14.36 5.710  9.884
      108 5.888 6.108 4.808 3.966 2.9023 14.35 6.440 10.085
      109 4.520 5.663 4.809 3.283 4.0173 13.74 4.862  9.440
      110 6.023 6.573 6.841 4.103 3.5320 14.99 6.909 11.313
```



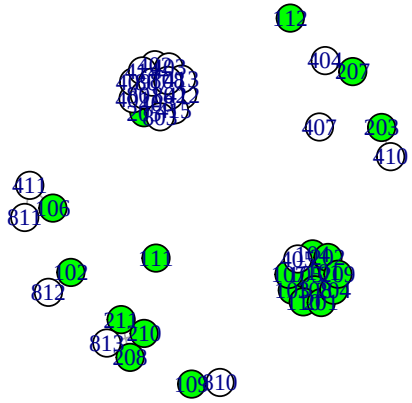
```
matchdistPen2[5:10, 1:8]
```

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

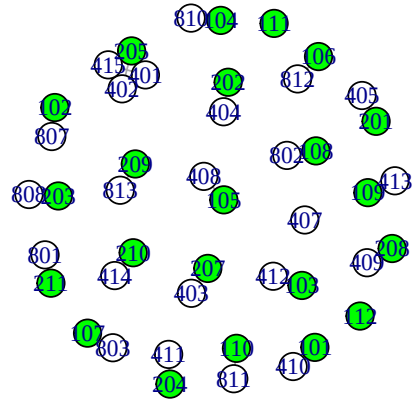
Showing matches

Fullmatching offers us more choices:

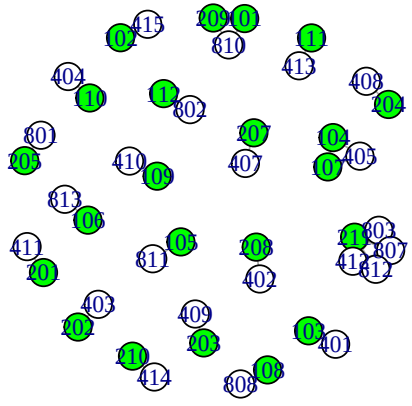
Min Ctrls=0, Max Ctrls=Inf



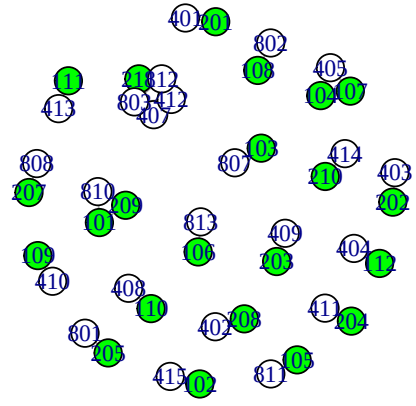
Min Ctrls=1, Max Ctrls=Inf



Penalties,Min Ctrls=.5, Mean Ctrls=23/22



Pen V 2,MinCtrls=.5, MeanCtrls=23/22



Tracking effective sample size

In 2-sample comparisons, total sample size can be a misleading as a measure of information content. Example:

- say Y has same variance, σ^2,in the Tx and the Ctl population.
- Ben H. samples 10 Tx and 40 Ctls, and
- Jake B. samples 25 Tx and 25 Ctls

— so that total sample sizes are the same. However,

$$\begin{aligned} V_{BH}(\bar{y}_t - \bar{y}_c) &= \frac{\sigma^2}{10} + \frac{\sigma^2}{40} = .125\sigma^2; \\ V_{JB}(\bar{y}_t - \bar{y}_c) &= \frac{\sigma^2}{25} + \frac{\sigma^2}{25} = .08\sigma^2. \end{aligned}$$

Similarly, a matched triple is roughly $[(\sigma^2/1 + \sigma^2/2)/(\sigma^2/1 + \sigma^2/1)]^{-1} = 1.33$ times as informative as a matched pair.

Set sizes and precision.

Since we will be estimating effects by weighting the within-set effects, the variance of set sizes (and ratios of treated-to-control units) influences the overall precision. Here, showing this by demonstration rather than diving into the details.

```
fm0a <- fullmatch(psdist, data = meddat)
summary(fm0a, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

```
Structure of matched sets:
14:1  3:1  1:1  1:2 1:15
   1    1    2    2    1
Effective Sample Size:  9.9
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       7.78 18   0.982
```

```
fm0b <- fullmatch(psdist, data = meddat, min.controls = .5, max.controls = 4)
summary(fm0b, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

```
Structure of matched sets:
2:1 1:1 1:2 1:4
  6   7   1   2
Effective Sample Size:  19.5
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       15.6 18   0.623
```

Set sizes and precision.

```
meddat$fm0a <- fm0a
meddat$fm0b <- fm0b

get_se <- function(dat, thefm) {
  ## thefm is a character name of the matching factor
  dat$thefm <- dat[[thefm]]
  ## shuffle treatment so the true effect is always known (and 0)
  ## dplyr is nice but it removes row.names.
  ## optmatch needs rownames to keep track of matches.
  ## So we have to add them back at the end of the data manipulation.
  dat <- dat %>%
    group_by(thefm) %>%
    mutate(newZ = sample(nhTrt)) %>%
    column_to_rownames("nh")
  thelm <- lm_robust(HomRate08 ~ newZ, fixed_effects = ~thefm, data = dat, subset = !is.na(thefm)
  return(thelm$std.error)
}
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.118   0.144   0.166   0.190   0.200   0.395
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.130   0.152   0.157   0.158   0.166   0.178
```

Details

Use pooled 2-sample t statistic SE formula to compare 1-1 vs 1-2 matched sets' contribution to variance:

$$\begin{array}{c|c} \text{1:1} & \text{1:2} \\ M^{-2}\sum_{m=1}^{M}(\sigma^2/1+\sigma^2/1) & M^{-2}\sum_{m=1}^{M}(\sigma^2/1+\sigma^2/2) \\ \frac{2\sigma^2}{M} & \frac{1.5\sigma^2}{M} \end{array}$$

So 20 matched pairs is comparable to 15 matched triples.
(Correspondingly, h-mean of n_t, n_c for a pair is 1, while for a triple it's $[(1/1+1/2)/2]^{-1} = 4/3$.)
The variance of the Z-coeff in `y~Z + match` is

$$\frac{2\sigma^2}{\sum_s h_s}, \qquad h_s = \left(\frac{n_{ts}^{-1}+n_{cs}^{-1}}{2}\right)^{-1},$$

assuming the OLS model and homoskedastic errors. (This is b/c the anova formulation is equivalent to harmonic-mean weighting, under which $V(\sum_s w_s(\bar{v}_{ts}-\bar{v}_{cs})) = \sum_s w_s^2(n_{ts}^{-1}+n_{cs}^{-1})\sigma^2 = \sigma^2\sum_s w_s^2 2h_s^{-1} = 2\sigma^2\sum_s w_s/\sum_s h_s = 2\sigma^2/\sum_s h_s$.)
For matched pairs, of course, $h_s = 1$. Harmonic mean of 1, 2 is $4/3$. Etc.

Matching so as to maximize effective sample size

```
stratumStructure(fm1)
```

```
1:0 1:1 1:3 0:1
  2  19   1   1
```

```
stratumStructure(fm0)
```

```
 1:0 11:1  3:1  1:1  1:2 1:14  0:1
   2    1    1    4    1    1    1
```

```
effectiveSampleSize(fm1)
```

```
[1] 20.5
```

```
effectiveSampleSize(fm0)
```

```
[1] 10.53
```

```
meddat$fm1 <- fm1
meddat$fm0 <- fm0
wtsfm1 <- meddat %>%
  filter(!is.na(fm1)) %>%
  group_by(fm1) %>%
  summarise(
    nb = n(),
    nTb = sum(nhTrt),
    nCb = nb - nTb,
    hwt = (2 * (nCb * nTb) / (nTb + nCb))
  )
wtsfm1
```

```
# A tibble: 20 x 5
   fm1       nb   nTb   nCb   hwt
```

Matching so as to maximize effective sample size

```
wtsfm0 <- meddat %>%
  filter(!is.na(fm0)) %>%
  group_by(fm0) %>%
  summarise(
    nb = n(),
    nTb = sum(nhTrt), nCb = nb - nTb,
    hwt = (2 * (nCb * nTb) / (nTb + nCb))
  )
wtsfm0
```

```
# A tibble: 8 x 5
  fm0      nb   nTb   nCb   hwt
  <fct> <int> <int> <int> <dbl>
1 1.1      12    11     1  1.83
2 1.13      2     1     1  1
3 1.15     15     1    14  1.87
4 1.16      2     1     1  1
5 1.17      4     3     1  1.5
6 1.2       2     1     1  1
7 1.6       3     1     2  1.33
8 1.9       2     1     1  1
```

```
sum(wtsfm0$hwt)
```

```
[1] 10.53
```

```
stratumStructure(fm0)
```

```
 1:0 11:1  3:1  1:1  1:2 1:14  0:1
   2    1    1    4    1    1    1
```

```
mean(unlist(matched.distances(fm0, matchdist)))
```

Why does it matter?

Or see here: Higher effective sample size → lower standard error.

```
stratumStructure(fm2)
```

```
2:1 1:1 1:4
  2  17   1
```

```
stratumStructure(fm4)
```

```
9:1 1:1 1:2 1:9
  1  11   1   1
```

```
effectiveSampleSize(fm2)
```

```
[1] 21.27
```

```
effectiveSampleSize(fm4)
```

```
[1] 15.93
```

```
lm_fm2 <- lm_robust(HomRate08 ~ nhTrt, fixed_effects = ~fm2, data = meddat, subset = !is.na(fm2))
lm_fm4 <- lm_robust(HomRate08 ~ nhTrt, fixed_effects = ~fm4, data = meddat, subset = !is.na(fm4))

lm_fm2$std.error
```

```
 nhTrt
0.1243
```

```
lm_fm4$std.error
```

```
 nhTrt
0.1317
```

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

Missing data and matching

We did not talk about missing data on covariates (and the `fill.NAs` command in optmatch). We also did not dive into the statistical power and set configuration relationship.

Missing data and matching

What if nhPopD had some missing data?

```
set.seed(12345)
meddat$nhPopD[sample(1:45, 10)] <- NA
summary(meddat$nhPopD)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    166     305     365     375     462     574      10
```

We would want to compare units who are equally likely to have nhPopD missing. So, we create a new variable:

```
newdat <- fill.NAs(meddat[, c("nhAboveHS", "nhPopD")])
head(newdat)
```

```
    nhAboveHS nhPopD nhPopD.NA
101   0.00000  448.8     FALSE
102   0.00000  375.0      TRUE
103   0.05556  468.4     FALSE
104   0.00000  462.4     FALSE
105   0.04167  480.6     FALSE
106   0.27273  498.2     FALSE
```

```
stopifnot(all.equal(row.names(newdat), row.names(meddat)))
newdat <- cbind(newdat, meddat[, c("nhTrt", "HomRate08", "HomRate03")])
head(newdat)
```

```
    nhAboveHS nhPopD nhPopD.NA nhTrt HomRate08 HomRate03
101   0.00000  448.8     FALSE     1   0.37072    0.9055
102   0.00000  375.0      TRUE     1   0.18884    1.0027
103   0.05556  468.4     FALSE     1   0.30766    1.6802
104   0.00000  462.4     FALSE     1   0.33785    0.5382
```

Missing data and matching

And we include that variable in our balance testing and matching:

```
theglm <- arm::bayesglm(nhTrt ~ nhAboveHS + nhPopD + nhPopD.NA + HomRate03, data = newdat)
psdist <- match_on(theglm, data = meddat)
maxCaliper(theglm$linear.predictor, z = newdat$nhTrt, widths = c(.1, .5, 1))
```

```
[1] 1
```

```
balfmla <- formula(theglm)
fm0 <- fullmatch(psdist + caliper(psdist, 2), data = newdat)
summary(fm0, min.controls = 0, max.controls = Inf, propensity.model = theglm)
```

```
Structure of matched sets:
1:0 6:1 5:1 4:1 1:1 1:2 1:3 1:4 1:9
  1   1   1   1   2   1   1   1   1
Effective Sample Size:  13.2
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       4.48  4   0.344
```

```
summary(unlist(matched.distances(fm0, psdist)))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0064  0.0531  0.1636  0.3510  0.5767  1.8217
```

```
newdat$fm0 <- fm0
xb0 <- xBalance(balfmla, strata = list(fm0 = ~fm0), data = newdat, report = "all")
xb0$overall
```

```
    chisquare df p.value
fm0     4.485  4  0.3444
```

Missing Data and Matching

So:

1. Missing data on covariates is not a big problems — such data reveals information to us about the units pre-treatment, so we just stratify on it. We treat missing data as just another covariate.
2. Missing data on treatment assignment or the outcome is a bigger problem: we will tend to use bounds to report on the range of possible answers in such cases.

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

How to find a good design?: Design Search for both precision and balance I

Here I demonstrate searching for two calipers and `min.controls` using a grid of possible caliper values.

How to find a good design?: Design Search for both precision and balance II

```
findbalance <- function(x, mhdist = mhdist, psdist = psdist, absdist = hrdist2, thedat = meddat,
  ## message(paste(x,collapse=" "))
  thefm <- try(fullmatch(psdist + caliper(mhdist, x[2]) +
    caliper(psdist, x[1]) + caliper(absdist, x[4]), min.controls = x[3], data = thedat, tol = .00

  if (inherits(thefm, "try-error")) {
    return(c(x = x, d2p = NA, maxHR03diff = NA, n = NA, effn = NA))
  }

  thedat$thefm <- thefm

  thexb <- try(balanceTest(update(thebalfmla, . ~ . + strata(thefm)), data = thedat), silent = TR

  if (inherits(thexb, "try-error")) {
    return(c(x = x, d2p = NA, maxHR03diff = NA, n = NA, effn = NA))
  }

  maxHomRate03diff <- max(unlist(matched.distances(thefm, distance = absdist)))

  return(c(
    x = x, d2p = thexb$overall["thefm", "p.value"],
    maxHR03diff = maxHomRate03diff,
    n = sum(!is.na(thefm)),
    effn = summary(thefm)$effective.sample.size
  ))
}
```

Design Search for both precision and balance

```
## Test the function
thecovs <- unique(c(names(meddat)[c(5:7, 9:24)], "HomRate03"))
balfmla <- reformulate(thecovs, response = "nhTrt")
findbalance(c(psCal, mhCal, 0, 2), thedat = meddat, psdist = psdist, mhdist = mhdist)
```

```
     x.90%      x.90%         x3         x4        d2p maxHR03diff          n       effn
    4.0648     7.4291     0.0000     2.0000     0.5473     1.7812    43.0000    14.8111
```

```
## Don't worry about errors for certain combinations of parameters
maxmhdist <- max(as.vector(mhdist))
minmhdist <- min(as.vector(mhdist))
maxpsdist <- max(as.vector(psdist))
minpsdist <- min(as.vector(psdist))
```

```
set.seed(123455)
system.time({
  resultsTemp <- replicate(10, findbalance(x = c(
    runif(1, minpsdist, maxpsdist),
    runif(1, minmhdist, maxmhdist),
    sample(seq(0, 1, length = 100), size = 1),
    runif(1, min(hrdist2), max(hrdist2))
  ), thedat = meddat, psdist = psdist, mhdist = mhdist))
})
## Notice that balanceTest has trouble when number of covariates is too large (it returns a p=.453
```

```
   user  system elapsed
136.959   2.041  19.383
```

Which matched design might we prefer? I

Now, how might we interpret the results of this search for matched designs? Here are a few ideas.

```
if (any(class(results) == "list")) {
  resAnyNA <- sapply(results, function(x) {
    any(is.na(x))
  })
  resNoNA <- simplify2array(results[!resAnyNA])
} else {
  resAnyNA <- apply(results, 2, function(x) {
    any(is.na(x))
  })
  resNoNA <- simplify2array(results[, !resAnyNA])
}
apply(resNoNA, 1, summary)
```

```
              x1    x2     x3       x4    d2p maxHR03diff     n   effn
Min.     0.02048 3.071 0.0000  0.06684 0.1835     0.06122  2.00  1.000
1st Qu. 2.08324 4.473 0.2323  2.74567 0.3679     1.02853 19.00  8.983
Median  3.79437 5.888 0.4949  5.10018 0.4373     2.34082 38.00 13.923
Mean    3.78876 5.948 0.4922  5.27021 0.4384     2.59880 31.13 12.128
3rd Qu. 5.48746 7.360 0.7500  7.94537 0.4797     4.00156 43.00 16.314
Max.    7.25948 8.956 1.0000 10.75499 0.8712    10.06636 45.00 21.500
```

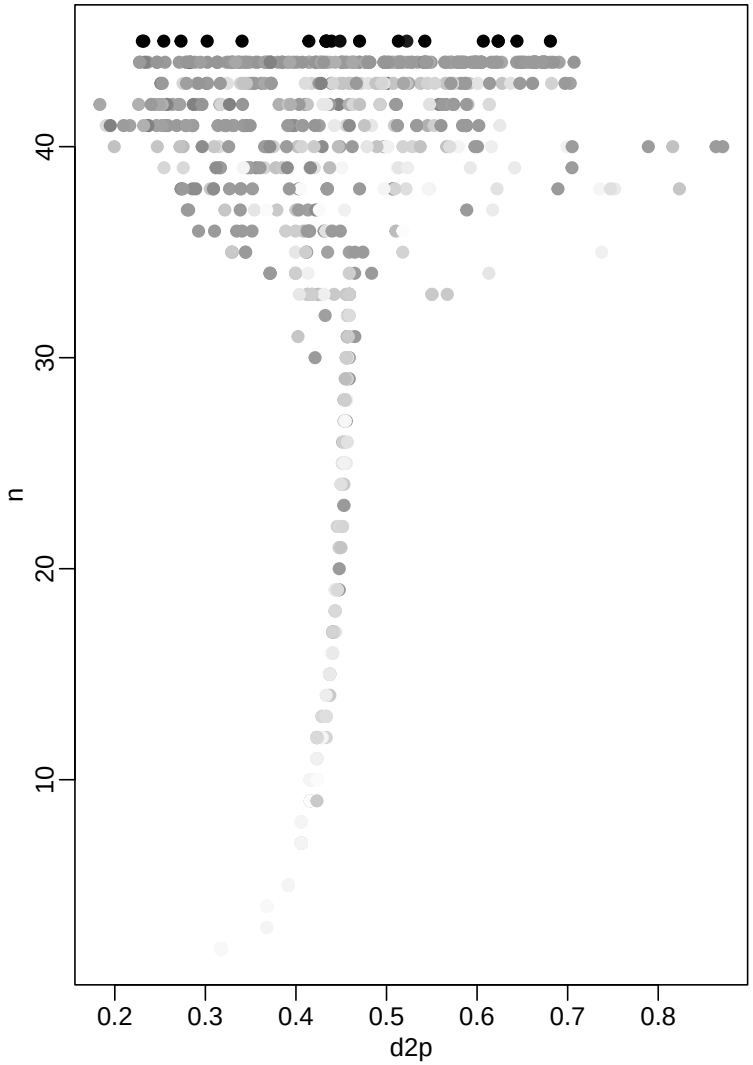
Which matched design might we prefer? II

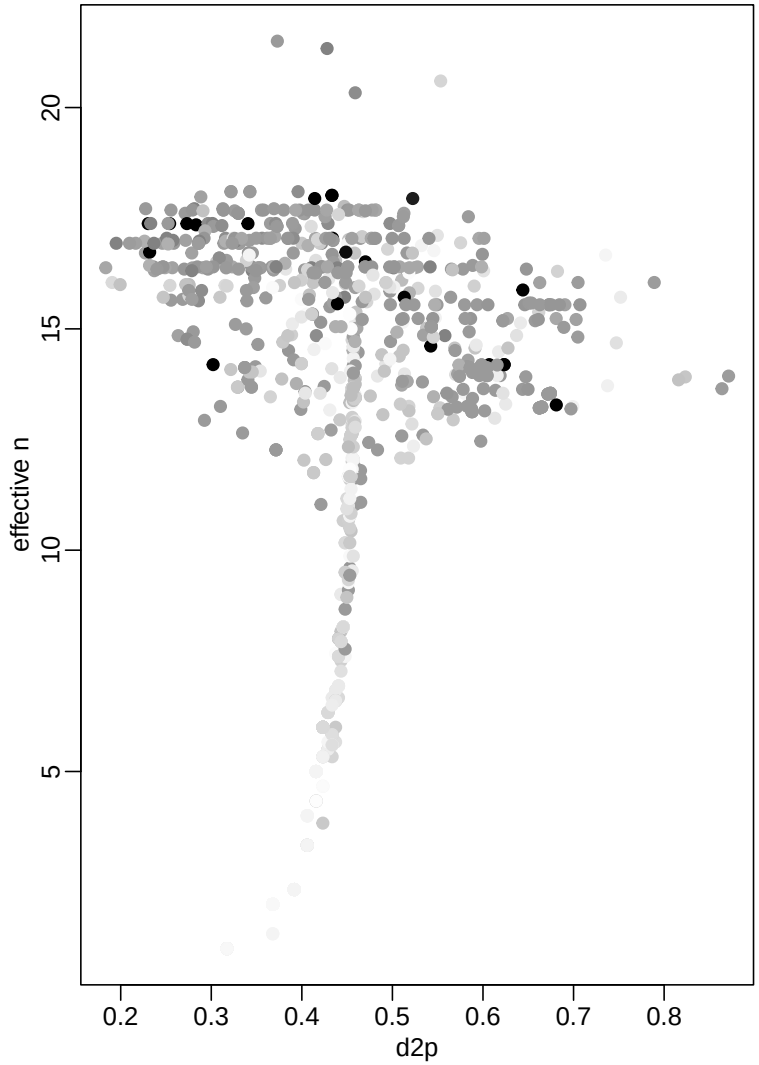
```
highbalres <- resNoNA[, resNoNA["d2p", ] > .5]
apply(highbalres, 1, summary)
```

```
            x1    x2     x3      x4    d2p maxHR03diff     n  effn
Min.    0.6633 4.743 0.0000  0.1307 0.5013      0.1234 33.00 12.08
1st Qu. 2.3417 6.721 0.1136  3.0494 0.5425      2.3408 41.00 13.55
Median  3.9341 7.635 0.2525  4.8151 0.5868      4.0016 43.00 14.51
Mean    3.9446 7.470 0.3102  5.2533 0.5952      3.4655 42.25 14.71
3rd Qu. 5.3161 8.305 0.4343  7.8528 0.6440      4.0016 44.00 15.54
Max.    7.2595 8.943 1.0000 10.6465 0.8712     10.0664 45.00 20.60
```

Which matched design might we prefer?

The darker points have smaller maximum within set differences on the baseline outcome.





Which matched design might we prefer?

```
interestingDesigns <- (resNoNA["d2p", ] > .3 & resNoNA["n", ] >= 40 &
  resNoNA["maxHR03diff", ] < 10 & resNoNA["effn", ] > 17)
candDesigns <- resNoNA[, interestingDesigns, drop = FALSE]
str(candDesigns)
```

```
 num [1:8, 1:80] 5.004 5.873 0.98 7.999 0.584 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:8] "x1" "x2" "x3" "x4" ...
  ..$ : NULL
```

```
apply(candDesigns, 1, summary)
```

```
           x1    x2     x3     x4    d2p maxHR03diff     n  effn
Min.    1.865 5.624 0.4141  1.153 0.3013       1.118 40.00 17.02
1st Qu. 3.678 6.524 0.6313  4.757 0.3653       3.456 43.00 17.04
Median  5.099 7.025 0.7525  5.858 0.4146       4.002 44.00 17.53
Mean    4.885 7.026 0.7518  6.074 0.4181       3.829 43.11 17.60
3rd Qu. 5.917 7.708 0.8889  8.009 0.4667       4.125 44.00 17.68
Max.    7.259 8.799 1.0000 10.440 0.6011       9.991 45.00 21.50
```

```
candDesigns <- candDesigns[, order(candDesigns["d2p", ], decreasing = TRUE)]
candDesigns <- candDesigns[, 1]
```

How would we use this information in `fullmatch`?

```
stopifnot(nrow(candDesigns) == 1)
fm4 <- fullmatch(psdist + caliper(psdist, candDesigns["x1"]) + caliper(mhdist, candDesigns["x2"]),
summary(fm4, min.controls = 0, max.controls = Inf, propensity.model = thebglm)
```

```
Structure of matched sets:
1:0 2:1 1:1 1:4 1:8
  1   8   3   1   1
Effective Sample Size:  17
(equivalent number of matched pairs).

Balance test overall result:
  chisquare df p.value
       18.5 18   0.422
```

```
meddat$fm4 <- NULL ## this line exists to prevent confusion with new fm4 objects
meddat[names(fm4), "fm4"] <- fm4
xb3 <- balanceTest(update(balfmla, . ~ . + strata(fm0) + strata(fm1) + strata(fm2) + strata(fm4))
  data = meddat
)
xb3$overall[, 1:3]
```

```
    chisquare df p.value
fm0     17.79 21  0.6623
fm1     21.87 21  0.4069
fm2     21.98 21  0.4008
fm4     18.75 21  0.6011
--      25.12 21  0.2421
```

```
zapsmall(xb3$results["HomRate03", , ])
```

```
          strata
stat          fm0    fm1    fm2    fm4    --
```

Another approach: Optimization I

Here is another approach that tries to avoid searching the whole space. It focuses on getting close to a target p-value from the omnibus/overall balance test and ignores effective sample size. Here we are just looking for one caliper value that gets us close to a particular target balance using one distance matrix. But, of course we care about **both** effective sample size **and** omnibus balance test results.

Another approach: Optimization II

```
matchAndBalance2 <- function(x, distmat, alpha) {
  # x is a caliper widths
  if (x > max(as.vector(distmat)) | x < min(as.vector(distmat))) {
    return(99999)
  }
  thefm <- fullmatch(distmat + caliper(distmat, x), data = meddat, tol = .00001)
  balfmla_to_use <- update(balfmla, . ~ . + strata(thefm))
  thexb <- balanceTest(balfmla_to_use, data = data.frame(cbind(meddat, thefm)))
  return(thexb$overall["thefm", "p.value"])
}

maxpfn <- function(x, distmat, alpha) {
  ## here x is the targeted caliper width and x2 is the next wider
  ## caliper width
  p1 <- matchAndBalance2(x = x[1], distmat, alpha)
  p2 <- matchAndBalance2(x = x[2], distmat, alpha)
  return(abs(max(p1, p2) - alpha))
}

maxpfn(c(minpsdist, minpsdist + 5), distmat = psdist, alpha = .25)
```

```
[1] 0.431
```

Another approach: Optimization III

```
maxpfn(c(minpsdist + .31, minpsdist + 1), distmat = psdist, alpha = .25)
```

```
[1] 0.3376
```

```
## Try basically no caliper:
maxpfn(c(maxpsdist - .01, maxpsdist), distmat = psdist, alpha = .25)
```

```
[1] 0.431
```

```
# quantile(as.vector(psdist),seq(0,1,.1))
# sort(as.vector(psdist))[1:10]
```

Another approach: more fine tuned optimization

```
### This takes a long time
library(Rsolnp)
results3 <- gosolnp(
  fun = maxpfn,
  ineqfun = function(x, distmat, alpha) {
    x[2] - x[1]
  },
  ineqLB = 0,
  ineqUB = maxpsdist,
  LB = c(minpsdist + .31, minpsdist + .32),
  UB = c(maxpsdist - .01, maxpsdist),
  n.restarts = 2,
  alpha = .5,
  distmat = psdist,
  n.sim = 500,
  rseed = 12345,
  control = list(trace = 1)
)
```

```
Calculating Random Initialization Parameters...ok!

Excluding Inequality Violations...

...Excluded 484/1000 Random Sequences

Evaluating Objective Function with Random Sampled Parameters...ok!

Sorting and Choosing Best Candidates for starting Solver...ok!
```

Another approach: more fine tuned optimization

Results of the optimization search:

```
maxpfn(results3$pars, distmat = psdist, alpha = .25)
```

```
[1] 0.3376
```

```
## This is the d2 p-value for the chosen caliper
matchAndBalance2(results3$pars[1], distmat = psdist, alpha = .25)
```

```
[1] 0.4516
```

Cardinality Matching Example I

Another approach to matching combines different constraints — attempting to, for example minimize the sum of distances between units within set while also maximizing the number of units in the design. See the citations in the `designmatch` package for papers explaining and applying these ideas.

Cardinality Matching Example II

```
library(designmatch)
## library(gurobi)

## Let's match on the propensity score
meddat$pscore <- thebglm$linear.predictors
## designmatch needs the observations on the data to be in order of treatment assignment
meddat_new <- meddat[order(meddat$nhTrt, decreasing = TRUE), ]
z <- as.vector(meddat_new$nhTrt)
Xmat <- model.matrix(~pscore, data = meddat_new)
thedistmat <- distmat(z, Xmat, digits = 2)

thecovs <- unique(c(names(meddat)[c(6:7, 9:24)], "HomRate03"))
balfmla <- reformulate(thecovs, response = "nhTrt")

psdist_new <- match_on(nhTrt ~ pscore, data = meddat_new)
distmat <- as.matrix(psdist)
distmat_scaled <- round(distmat / mean(distmat), 2)
dimnames(distmat_scaled) <- dimnames(thedistmat)
## Minimize distances but keep HomRate03 diffs below 2
nearlist <- list(covs = as.matrix(meddat_new$HomRate03), pairs = 2)
```

Cardinality Matching Example I

```
## solverlist <- list(name='gurobi',approximate=0,t_max=2000,trace=1)
solverlist <- list(name = "highs", approximate = 1, t_max = 1000)
res <- bmatch(
  t_ind = z,
  dist_mat = thedistmat,
  near = nearlist,
  solver = solverlist,
  subset_weight = 1
)
```

```
  Building the matching problem...
  HiGHS optimizer is open...
  Finding the optimal matches...
  Optimal matches found
  HiGHS optimizer is open...
  Finding the optimal matches...
  Optimal matches found
```

```
## library(cobalt)
## bal.tab(res,balfmla,data=meddat_new)
```

```
    chisquare df p.value
dm1     14.60 14 0.40638
pm1     16.48 14 0.28499
--      21.81 14 0.08268
```

Cardinality Matching Example II

	strata		
vars	dm1	pm1	--
HomRate03	0.09515	0.52443	0.52574
nhAboveHS	-0.22576	-0.55182	-0.67076
nhHS	0.09196	0.01783	0.05169
nhRent	0.07752	0.17613	0.14432
nhOwn	-0.09322	-0.21526	-0.20101
nhSepDiv	-0.79340	-0.48858	-0.56808
nhMarDom	0.01128	0.06498	0.09524
nhAgeYoung	0.24822	0.35250	0.38216
nhSisben	0.09260	0.68627	0.74027
nhPopD	0.64196	0.77550	0.81720
nhQP03	0.04704	0.34461	0.35557
nhPV03	0.17054	0.36363	0.37321
nhTP03	-0.41555	-0.92015	-0.96781
(nhPopD)	0.47310	0.21505	0.18700

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

Overview: Estimate and Test "as if block-randomized" I

What are we estimating? Most people would say ACE=$\bar{\tau} = \bar{y}_1 - \bar{y}_0$. What estimator estimates this without bias?

```
meddat[names(fm0), "fm0"] <- fm0
datB <- meddat %>%
  filter(!is.na(fm0)) %>%
  group_by(fm0) %>%
  summarise(
    Y = mean(HomRate08[nhTrt == 1]) - mean(HomRate08[nhTrt == 0]),
    nb = n(),
    nbwt = unique(nb / nrow(meddat)),
    nTb = sum(nhTrt),
    nCb = sum(1 - nhTrt),
    pb = mean(nhTrt),
    pbwt = pb * (1 - pb),
    hbwt1 = pbwt * nb,
    hbwt2 = pbwt * nbwt,
    hbwt3 = (2 * (nCb * nTb) / (nTb + nCb))
  )
```

```
datB
```

Overview: Estimate and Test "as if block-randomized" II

```
# A tibble: 9 x 11
  fm0          Y    nb   nbwt   nTb   nCb    pb  pbwt hbwt1  hbwt2 hbwt3
  <fct>    <dbl> <int>  <dbl> <int> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>
1 1.1    0.0630     7 0.156      6     1 0.857 0.122 0.857 0.0190  1.71
2 1.10  -0.612      4 0.0889     1     3 0.25  0.188 0.75  0.0167  1.5
3 1.13  -0.276     10 0.222      1     9 0.1   0.09  0.9   0.02    1.8
4 1.14  -0.524      2 0.0444     1     1 0.5   0.25  0.5   0.0111  1
5 1.15  -0.577      3 0.0667     1     2 0.333 0.222 0.667 0.0148  1.33
6 1.2   -1.87       2 0.0444     1     1 0.5   0.25  0.5   0.0111  1
7 1.3   -0.265      5 0.111      4     1 0.8   0.16  0.8   0.0178  1.6
8 1.6    0.136      6 0.133      5     1 0.833 0.139 0.833 0.0185  1.67
9 1.9   -0.229      5 0.111      1     4 0.2   0.16  0.8   0.0178  1.6
```

```r
## Notice that all of these different ways to express the harmonic mean weight are the same.
datB$hbwt101 <- datB$hbwt1 / sum(datB$hbwt1)
datB$hbwt201 <- datB$hbwt2 / sum(datB$hbwt2)
datB$hbwt301 <- datB$hbwt3 / sum(datB$hbwt3)
stopifnot(all.equal(datB$hbwt101, datB$hbwt201))
stopifnot(all.equal(datB$hbwt101, datB$hbwt301))
```

Using the weights: Set size weights

First, we could estimate the set-size weighted ATE. Our estimator uses the size of the sets to estimate this quantity.

```
## The set-size weighted version
atewnb <- with(datB, sum(Y * nb / sum(nb)))
atewnb
```

```
[1] -0.2941
```

Using the weights: Set size weights

Sometimes it is convenient to use `lm` (or the more design-friendly `lm_robust`) because there are R functions for design-based standard errors and confidence intervals.

```
meddat$id <- row.names(meddat)
meddat$nhTrtF <- factor(meddat$nhTrt)
## See Gerber and Green section 4.5 and also Chapter 3 on block randomized experiments. Also Hanse
## Here just making a new dataset with no missing data for ease of use later.
wdat <- meddat %>%
  filter(!is.na(fm0)) %>%
  group_by(fm0) %>%
  mutate(
    pb = mean(nhTrt),
    nbwt = nhTrt / pb + (1 - nhTrt) / (1 - pb),
    gghbwt = 2 * (n() / nrow(meddat)) * (pb * (1 - pb)), ## GG version,
    gghbwt2 = 2 * (nbwt) * (pb * (1 - pb)), ## GG version,
    nb = n(),
    nTb = sum(nhTrt),
    nCb = nb - nTb,
    hbwt1 = (2 * (nCb * nTb) / (nTb + nCb)),
    hbwt2 = nbwt * (pb * (1 - pb))
  )

row.names(wdat) <- wdat$id ## dplyr strips row.names
wdat$nhTrtF <- factor(wdat$nhTrtF)
lm0b <- lm_robust(HomRate08 ~ nhTrt, data = wdat, weight = nbwt)
lm0b
```

```
            Estimate Std. Error t value   Pr(>|t|) CI Lower CI Upper DF
(Intercept)   0.6036     0.1216   4.965 0.00001191   0.3583  0.84898 42
nhTrt        -0.2941     0.1353  -2.173 0.03546446  -0.5672 -0.02098 42
```

Using the weights: precision weights

Set-size weighting is easy to explain but may differ in terms of precision:

```
atewhb <- with(datB, sum(Y * hbwt1 / sum(hbwt1)))
atewhb
```

```
[1] -0.3811
```

```
lm1 <- lm_robust(HomRate08 ~ nhTrt + fm0, data = wdat)
summary(lm1)$coef[2, ]
```

```
  Estimate Std. Error    t value   Pr(>|t|)   CI Lower   CI Upper         DF
  -0.38106    0.16247   -2.34543    0.02498   -0.71123   -0.05088   34.00000
```

```
summary(lm0b)$coef[2, ]
```

```
  Estimate Std. Error    t value   Pr(>|t|)   CI Lower   CI Upper         DF
  -0.29410    0.13534   -2.17308    0.03546   -0.56723   -0.02098   42.00000
```

```
## Notice that fixed_effects is same as indicator variables is same as weighting
lm1a <- lm_robust(HomRate08 ~ nhTrt, fixed_effects = ~fm0, data = wdat)
summary(lm1a)$coef[1, ]
```

```
  Estimate Std. Error    t value   Pr(>|t|)   CI Lower   CI Upper         DF
  -0.38106    0.16247   -2.34543    0.02498   -0.71123   -0.05088   34.00000
```

Precision weighting

Block-mean centering is another approach although notice some precision gains for not "estimating fixed effects" — in quotes because there is nothing to estimate here — set or block-means are fixed quantities and need not be estimated in this framework.

```
wdat$HomRate08Cent <- with(wdat, HomRate08 - ave(HomRate08, fm0))
wdat$nhTrtCent <- with(wdat, nhTrt - ave(nhTrt, fm0))

lm2 <- lm_robust(HomRate08Cent ~ nhTrtCent, data = wdat)
summary(lm2)$coef[2, ]
```

```
  Estimate Std. Error    t value   Pr(>|t|)   CI Lower   CI Upper         DF
-0.381059   0.131125  -2.906068   0.005819  -0.645680  -0.116437  42.000000
```

What about random effects?

Notice that one problem we have here is too few sets. Maybe better to use a fully Bayesian version if we wanted to do this. Why would we **model** the variability between sets? When might this be useful? How might we evaluate this approach?

```
## This had troubles with convergence
## library(lme4)
## lmer1 <- lmer(HomRate08 ~ nhTrt + (1 | fm0),
##   data = wdat,
##   verbose = 2, start = 0,
##   control = lmerControl(optimizer = "bobyqa", restart_edge = TRUE, optCtrl = list(maxfun = 1000
## )
## summary(lmer1)$coef
## Here is the more directly bayesian version of adjusting for the strata as random intercepts
library(rstanarm)
lmer2 <- stan_lmer(HomRate08 ~ nhTrt + (1 | fm0),
  data = wdat, seed = 12345
)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000212 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.12 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

Which estimator to choose? I

The block-sized weighted approach is unbiased. But unbiased is not the only indicator quality in an estimator.

```
library(DeclareDesign)

thepop <- declare_population(wdat)
theassign <- declare_assignment(blocks = fm0, block_m_each = table(
  fm0,
  nhTrt
), legacy = TRUE)
po_fun <- function(data) {
  data$Y_Z_1 <- data$HomRate08
  data$Y_Z_0 <- data$HomRate08
  data
}
thepo <- declare_potential_outcomes(handler = po_fun)
thereveal <- declare_reveal(Y, Z) ## how does assignment reveal potential outcomes
thedesign <- thepop + theassign + thepo + thereveal

oneexp <- draw_data(thedesign)
## Test
origtab <- with(wdat, table(trt = nhTrt, b = fm0))
all.equal(origtab, with(oneexp, table(trt = Z, b = fm0)))
```

```
[1] TRUE
```

Which estimator to choose? II

```
estimand1 <- declare_inquiry(ACE = mean(Y_Z_1 - Y_Z_0))

est1 <- declare_estimator(Y ~ Z,
  .method = difference_in_means,
  label = "E1: Ignoring Blocks",
  inquiry = "ACE"
)
est2 <- declare_estimator(Y ~ Z,
  fixed_effects = ~fm0,
  .method = lm_robust,
  label = "E2: precision weights fe1",
  inquiry = "ACE"
)
est3 <- declare_estimator(Y ~ Z + fm0,
  .method = lm_robust,
  label = "E3: precision weights fe2",
  inquiry = "ACE"
)

nbwt_est_fun <- function(data) {
  data$newnbwt <- with(data, (Z / pb) + ((1 - Z) / (1 - pb)))
  obj <- lm_robust(Y ~ Z, data = data, weights = newnbwt)
  res <- tidy(obj) %>% filter(term == "Z")
  return(res)
}

hbwt_est_fun <- function(data) {
  data$newnbwt <- with(data, (Z / pb) + ((1 - Z) / (1 - pb)))
  data$newhbwt <- with(data, newnbwt * (pb * (1 - pb)))
```

Diagnosands and diagnosis

```
set.seed(12345)

thediagnosands <- declare_diagnosands(
  bias = mean(estimate - estimand),
  rmse = sqrt(mean((estimate - estimand)^2)),
  power = mean(p.value < .05),
  coverage = mean(estimand <= conf.high & estimand >= conf.low),
  mean_estimate = mean(estimate),
  sd_estimate = sd(estimate),
  mean_se = mean(std.error),
  mean_estimand = mean(estimand)
)

library(future)
library(future.apply)
plan(multicore) ## only works easily on Mac/Linux machines
diagnosis <- diagnose_design(thedesign_plus_est,
  sims = 1000, bootstrap_sims = 0,
  diagnosands = thediagnosands
)
save(diagnosis, file = "day10diag.rda")
plan(sequential)
```

Results of the Simulation

	Estimator	Term	Bias	RMSE	Power	Coverage	Mean Estimate	SD Estimate	Mea
1	E1: Ignoring Blocks	Z	-0.1453	0.1912	0.1850	0.8150	-0.1453	0.1244	0.
2	E2: precision weights fe1	Z	0.0014	0.2066	0.0520	0.9480	0.0014	0.2067	0.
3	E3: precision weights fe2	Z	0.0014	0.2066	0.0520	0.9480	0.0014	0.2067	0.
4	E4: direct block size weights	Z	0.0023	0.1929	0.0800	0.9200	0.0023	0.1930	0.
5	E5: direct precision weights	Z	0.0014	0.2066	0.0850	0.9150	0.0014	0.2067	0.
6	E6: Direct Demeaning	Z	0.0014	0.2066	0.1410	0.8590	0.0014	0.2067	0.
7	E7: Random Effects	Z	-0.1327	0.2009	0.1860	0.8050	-0.1327	0.1508	0.

	design	sim_ID	inquiry	estimand	estimator	term	estimate	std.error	stat
1	thedesign_plus_est	1	ACE	0	E7: Random Effects	Z	-0.1030665	0.1408	0.378677775
2	thedesign_plus_est	2	ACE	0	E7: Random Effects	Z	0.1808758	0.1502	0.203880366
3	thedesign_plus_est	3	ACE	0	E7: Random Effects	Z	-0.2238955	0.1374	0.492888755
4	thedesign_plus_est	4	ACE	0	E7: Random Effects	Z	0.0679462	0.1476	0.259488941
5	thedesign_plus_est	5	ACE	0	E7: Random Effects	Z	-0.2609635	0.1359	0.526891790
6	thedesign_plus_est	6	ACE	0	E7: Random Effects	Z	0.1020024	0.1488	0.243844272
7	thedesign_plus_est	7	ACE	0	E7: Random Effects	Z	0.1998938	0.1502	0.192200270
8	thedesign_plus_est	8	ACE	0	E7: Random Effects	Z	-0.2257334	0.1373	0.494586483
9	thedesign_plus_est	9	ACE	0	E7: Random Effects	Z	-0.0309529	0.1416	0.308159112
10	thedesign_plus_est	10	ACE	0	E7: Random Effects	Z	-0.1026737	0.1408	0.378298333
11	thedesign_plus_est	11	ACE	0	E7: Random Effects	Z	-0.0526477	0.1415	

1. Overview and Review
2. Information and Balance: Matching structure and effective sample size
3. Missing data and matching
4. Design Search Approaches
5. Estimation
6. Testing Hypotheses by Randomization Inference in a Block-Randomized Trial

Testing Approach: By Hand

Testing by hand

```
wdat <- meddat %>% filter(!is.na(meddat$fm0))

obsmd1 <- with(wdat, mdwt1(y = HomRate08, trt = nhTrt, b = fm0))
obsmd2 <- with(wdat, mdwt2(y = HomRate08, trt = nhTrt, b = fm0))

origtab <- with(wdat, table(trt = nhTrt, b = fm0))
testtab <- with(wdat, table(trt = newexp(trt = nhTrt, b = fm0), b = fm0))
all.equal(origtab, testtab)
```

```
[1] TRUE
```

Testing by hand

```
set.seed(12345)
nulldist1 <- replicate(1000, with(wdat, mdwt1(y = HomRate08, trt = newexp(trt = nhTrt, b = fm0),
set.seed(12345)
nulldist2 <- replicate(1000, with(wdat, mdwt2(y = HomRate08, trt = newexp(trt = nhTrt, b = fm0),

p1 <- mean(nulldist1 <= obsmd1)
p2 <- mean(nulldist2 <= obsmd2)

var(nulldist1)
```

```
[1] 0.0354
```

```
var(nulldist2)
```

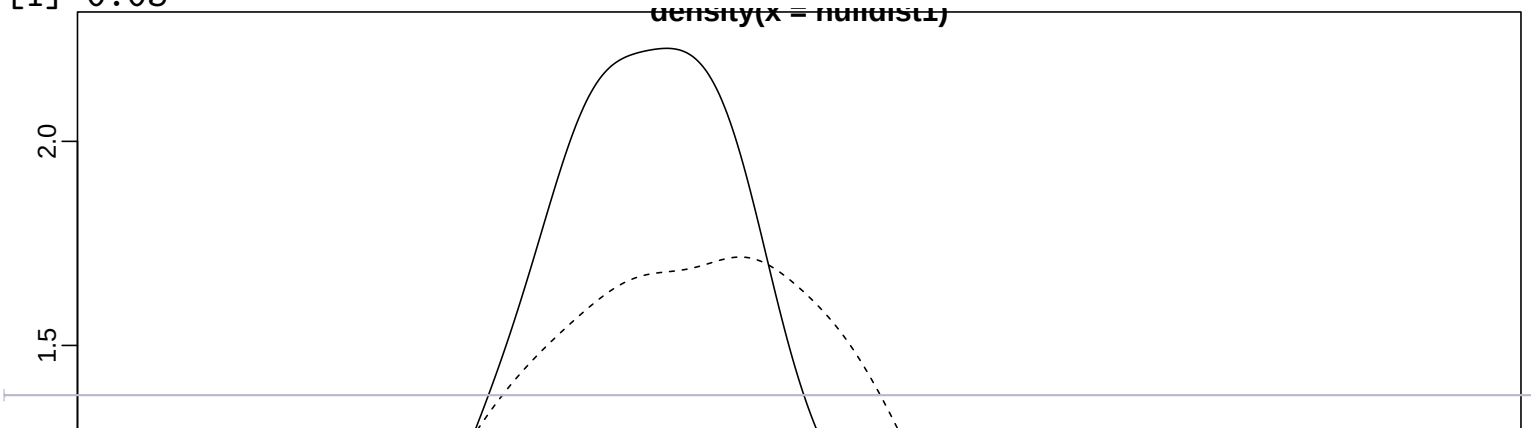
```
[1] 0.04113
```

```
2 * min(mean(nulldist1 <= obsmd1), mean(nulldist1 >= obsmd1))
```

```
[1] 0.076
```

```
2 * min(mean(nulldist2 <= obsmd2), mean(nulldist2 >= obsmd2))
```

```
[1] 0.03
```



Testing Approach: Faster

These are faster because they use the Central Limit Theorem — under the belief that our current data are large enough (informative enough) that our reference distribution would be well approximated by a Normal distribution.

```
## This uses the precision or harmonic mean weighting approach
xbTest1 <- balanceTest(nhTrt ~ HomRate08 + strata(fm0), data = wdat)
xbTest1$results[, , "fm0"]
```

```
  Control Treatment  std.diff  adj.diff pooled.sd         z         p
  0.50922   0.31434  -0.46296  -0.19488   0.42095  -1.86431   0.06228
```

Testing Approach: Faster

The coin package does something similar — it also allows for permutation based distributions using the approximate() function.

```
wdat$nhTrtF <- factor(wdat$nhTrt)
meanTestAsym <- oneway_test(HomRate08 ~ nhTrtF | fm0, data = wdat, distribution = "asymptotic")
set.seed(12345)
meanTestPerm <- oneway_test(HomRate08 ~ nhTrtF | fm0, data = wdat, distribution = approximate(nre

pvalue(meanTestAsym)
```

```
[1] 0.06228
```

```
pvalue(meanTestPerm)
```

```
[1] 0.051
99 percent confidence interval:
 0.03476 0.07166
```

```
rankTestAsym <- wilcox_test(HomRate08 ~ nhTrtF | fm0, data = wdat, distribution = "asymptotic")
set.seed(12345)
rankTestPerm <- wilcox_test(HomRate08 ~ nhTrtF | fm0, data = wdat, distribution = approximate(nre

pvalue(rankTestAsym)
```

```
[1] 0.0486
```

```
pvalue(rankTestPerm)
```

```
[1] 0.056
99 percent confidence interval:
 0.03893 0.07744
```

Notice the two distributions:

Summary

- A good research design allows us to interpret comparisons (or relationships) with some clarity (i.e. "My est. of $Z \to Y$ at least does not reflect age. Nor does it mostly reflect noise"). i.e. at least some way to address alternative explanations and reasonable statistical power (high probability of detecting an effect when an effect exists).
- Different stratifications pose different combinations of information (i.e. statistical power) and clarity.
- Since we create a research design without looking at outcomes, we are free to explore the space of possible designs.
- We are not limited to only distance matrices (and combinations thereof), calipers, and exact matching. We could also specify certain criteria — to think of finding the design as a constrained optimization problem (which is what `designmatch` does).
- Once we have a defensible design, we estimate and test following the design — just as we would in a randomized experiment.
- We can investigate our choices of estimators (and test) using simulation.

Next time:

- Matching when we have more than one group (non-bipartite matching)

Remaining questions?

References