General Summary

- R Package: Matchit
    - https://www.rdocumentation.org/packages/MatchIt/versions/3.0.2/topics/matchit
    - https://cran.r-project.org/web/packages/MatchIt/MatchIt.pdf
    - https://cran.r-project.org/web/packages/MatchIt/vignettes/MatchIt.html
- Form of subset selection
    - Pruning and weighting of units to arrive at a weighted subset of units from original data
    - Produces new sample where treatment is unassociated with the covariates so comparison of outcomes treatment and control groups are not confounded by measured and balanced covariates
- Weights
    - The formula for computing the weights depends on the argument supplied to estimand. A new stratum "propensity score" (p) is computed as the proportion of units in each stratum that are in the treated group, and all units in that stratum are assigned that propensity score. Weights are then computed using the standard formulas for inverse probability weights: for the ATT, weights are 1 for the treated units and p/(1-p) for the control units; for the ATC, weights are (1-p)/p for the treated units and 1 for the control units; for the ATE, weights are 1/p for the treated units and 1/(1-p) for the control units.

Matching is nonparametric in the sense that the estimated weights and pruning of the sample are not direct functions of estimated model parameters but rather depend on the organization of discrete units in the sample; this is in contrast to propensity score weighting (also known as inverse probability weighting), where the weights come more directly from the estimated propensity score model and therefore are more sensitive to its correct specification

Matching Methods

- Distance Matching (k:1)
    - Considering a focal group (usually the treated group) and selecting members of the non-focal group (i.e., the control group) to pair with each member of the focal group based on the distance between units, which can be computed in one of several ways
    - Nearest
        - Selecting the closest eligible control unit to be paired with each treated unit. It is greedy in the sense that each pairing occurs without reference to how other units will be or have been paired, and therefore does not aim to optimize any criterion
    - Optimal pair
        - Optimal pair matching (often just called optimal matching) is very similar to nearest neighbor matching in that it attempts to pair each treated unit with one or more control units. Unlike nearest neighbor matching, however, it is “optimal” rather than greedy; it is optimal in the sense that it

attempts to choose matches that collectively optimize an overall criterion. Like nearest neighbor matching, optimal pair matching requires the specification of a distance measure between units. Optimal pair matching can be thought of simply as an alternative to selecting the order of the matching for nearest neighbor matching.

- Optimal full
    - It is optimal in the sense that the chosen number of subclasses and the assignment of units to subclasses minimize the mean of the absolute within-subclass distances in the matched sample
- Genetic
    - Genetic matching uses a genetic algorithm, which is an optimization routine used for non-differentiable objective functions, to find scaling factors for each variable in a generalized Mahalanobis distance formula

- Stratum Matching (1:1)
    - Stratum matching involves creating strata based on unique values of the covariates and assigning units with those covariate values into those strata
    - Exact
        - Exact matching is a form of stratum matching that involves creating subclasses based on unique combinations of covariate values and assigning each unit into their corresponding subclass so that only units with identical covariate values are placed into the same subclass.
    - Coarsened exact matching
        - involves first coarsening the covariates by creating bins and then performing exact matching on the new coarsened versions of the covariates
    - Subclassification
        - form of coarsened exact matching with the propensity score as the sole covariate to be coarsened and matched on. The bins are usually based on specified quantiles of the propensity score distribution either in the treated group, control group, or overall, depending on the desired estimand

Distance Measure

- glm
    - The propensity scores are estimated using a generalized linear model
    - Logit
        - The link argument can be specified as a link function supplied to binomial(), e.g., "logit", which is the default. When link is prepended by "linear.", the linear predictor is used instead of the predicted probabilities
- gam
    - The propensity scores are estimated using a generalized additive model
- gbm
    - The propensity scores are estimated using a generalized boosted model
- lasso, ridge, elasticnet

    - The propensity scores are estimated using a lasso, ridge, or elastic net model, respectively
- rpart
    - The propensity scores are estimated using a classification tree
- randomforest
    - The propensity scores are estimated using a random forest
- nnet
    - The propensity scores are estimated using a single-hidden-layer neural network.
- cbps
    - The propensity scores are estimated using the covariate balancing propensity score (CBPS) algorithm, which is a form of logistic regression where balance constraints are incorporated to a generalized method of moments estimation of of the model coefficients
- bart
    - The propensity scores are estimated using Bayesian additive regression trees (BART)
- mahalanobis
    - No propensity scores are estimated. Rather than using the propensity score difference as the distance between units, the Mahalanobis distance is used instead.

Caliper

- Based on the propensity score or other covariates
- Two units whose distance on a calipered covariate is larger than the caliper width for that covariate are not allowed to be matched to each other
- Any units for which there are no available matches within the caliper are dropped from the matched sample

Data Preparation

- Remove cases of missing AMPAC scores
- Limit to patients whose LOSCalc is [3, 14]
- Remove outlier patients
- Create variable for first reported AMPAC score for each patient
- Create variable for days since admission
- Create binary variable for surgical patients
- Create variable for normalized days
- Create variable for number of scores per patient
- Create variable for highest JHFRAT score in first 48 hours
- Isolate data by unique patient ID

Matching

- Call:
  - m.out <- matchit(FallInjuryYN ~ first_amp + AGE + Surgery_binary + num_scores + JHFRAT_48max, data = by.ID, method = "nearest", distance = "logit", caliper = 0.03, ratio = 10)

```
Summary of Balance for Matched Data:
               Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
distance              0.0042        0.0042          0.0005     1.0201    0.0001   0.0192
first_amp            40.8352       40.8272          0.0008     0.8940    0.0253   0.0879
AGE                  61.5000       61.7206         -0.0143     0.9737    0.0173   0.0827
Surgery_binary        0.5577        0.5745         -0.0337          .    0.0168   0.0168
num_scores            9.0385        9.0909         -0.0179     0.9380    0.0161   0.0437
JHFRAT_48max         11.5385       11.3159          0.0509     0.9455    0.0178   0.0681
               Std. Pair Dist.
distance                0.0013
first_amp               1.0895
AGE                     1.0101
Surgery_binary          0.9748
num_scores              0.4022
JHFRAT_48max            1.0385

Sample Sizes:
               Control Treated
All           23401.        52
Matched (ESS)   470.85      52
Matched         504.        52
Unmatched     22897.         0
Discarded         0.         0
```

  - m.out <- matchit(FallInjuryYN ~ first_amp + AGE + Surgery_binary + num_scores + JHFRAT_48max, data = by.ID, method = "full", distance = "logit", caliper = 2, std.caliper = TRUE)

```
Summary of Balance for Matched Data:
               Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
distance              0.0042        0.0039          0.1160     0.9215    0.0737   0.2012
first_amp            40.8352       40.7400          0.0098     0.7595    0.0434   0.0983
AGE                  61.5000       60.4613          0.0673     0.9049    0.0278   0.1229
Surgery_binary        0.5577        0.3626          0.3928          .    0.1951   0.1951
num_scores            9.0385        8.1392          0.3069     0.7449    0.0746   0.1708
JHFRAT_48max         11.5385       11.0405          0.1140     0.8682    0.0287   0.1677
               Std. Pair Dist.
distance                0.1327
first_amp               0.8933
AGE                     0.9383
Surgery_binary          1.1358
num_scores              0.4014
JHFRAT_48max            1.0999

Sample Sizes:
               Control Treated
All           23401.        52
Matched (ESS)    71.96      52
Matched       23392.        52
Unmatched         9.         0
Discarded         0.         0
```

- m.out <- matchit(FallInjuryYN ~ first_amp + AGE + Surgery_binary + num_scores + JHFRAT_48max, data = by.ID, method = "full", distance = "logit", caliper = 1, std.caliper = TRUE)

```
Summary of Balance for Matched Data:
               Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
distance              0.0042        0.0039          0.1168     0.9352    0.0743   0.2014
first_amp            40.8352       40.9080         -0.0075     0.7905    0.0397   0.0930
AGE                  61.5000       60.0134          0.0963     0.8733    0.0307   0.1294
Surgery_binary        0.5577        0.3801          0.3576          .    0.1776   0.1776
num_scores            9.0385        8.1712          0.2960     0.7265    0.0733   0.1578
JHFRAT_48max         11.5385       10.9784          0.1282     0.8949    0.0304   0.1667
               Std. Pair Dist.
distance                0.1325
first_amp               0.8924
AGE                     0.9387
Surgery_binary          1.1360
num_scores              0.4015
JHFRAT_48max            1.1002

Sample Sizes:
               Control Treated
All           23401.        52
Matched (ESS)    74.26      52
Matched       23385.        52
Unmatched        16.         0
Discarded         0.         0
```

Pre-Fall

- Based off match, cut fallers to day before fall and non-fallers to a pseudo normalized day before fall
  - Ex. faller fell 0.6 normalized days, matched non-fallers cut there too

Data Preparation

Create DF with all patients, subset important variables for later use, convert data into working format by omitting NA data and converting date data type. This is a cleaner version of the large data set that I will be using to subset smaller DF's from for intended purposes.

```
MobLineDat <- subset(PatientFalls, select=c('PAT_ENC_CSN_ID','AGE',
'RECORDED_TIME.day','ADMIT_DATE', 'FALL_DAY1', 'HOSP_DISCH_TIME',
'LOScalc' ,'IMP.AMPAC_MOB_RAW.ALL', 'IMP.AMPAC_MOB_TSCORE.ALL',
'JHHLM.max', 'FallYN', 'HOSPSERVICE', 'ADMFORSURGERY'))

MobLineDat.clean <- na.omit(MobLineDat)

#convert dates into date format
MobLineDat.clean$RECORDED_TIME.day <-
mdy(MobLineDat.clean$RECORDED_TIME.day)
MobLineDat.clean$ADMIT_DATE <- mdy_hm(MobLineDat.clean$ADMIT_DATE,
tz="US/Eastern")
MobLineDat.clean$ADMIT_DATE <- as.Date(MobLineDat.clean$ADMIT_DATE)

MobLineDat.clean$HOSP_DISCH_TIME <-
mdy_hm(MobLineDat.clean$HOSP_DISCH_TIME, tz="US/Eastern")
MobLineDat.clean$HOSP_DISCH_TIME <-
as.Date(MobLineDat.clean$HOSP_DISCH_TIME)
MobLineDat.clean$FALL_DAY1 <- mdy(MobLineDat.clean$FALL_DAY1)
```

Construct the DF that I will be working with by limiting the patients to have LOS between 3 to 14. Create the first mobility scores variables "first_amp" and "first_hlm".

```
Mob_3_14 <- subset((MobLineDat.clean), MobLineDat.clean$LOScalc <= 14
& MobLineDat.clean$LOScalc >= 3)

Mob_3_14 <- Mob_3_14 %>% group_by(PAT_ENC_CSN_ID) %>%
mutate(first_amp = first(IMP.AMPAC_MOB_TSCORE.ALL))
Mob_3_14 <- Mob_3_14 %>% group_by(PAT_ENC_CSN_ID) %>%
mutate(first_hlm = first(JHHLM.max))
```

Create the "Days_since_admission" variable to relate each daily mobility measurement to the admission date. Then normalize this across each patient so the day of admission is 0 and the last day of score is 1.

```
Mob_3_14$Days_since_admission <-
as.numeric(Mob_3_14$RECORDED_TIME.day - Mob_3_14$ADMIT_DATE)
```

```
Mob_3_14 <- Mob_3_14 %>% group_by(PAT_ENC_CSN_ID) %>% mutate(
  normalized_LOS =
(Days_since_admission-min(Days_since_admission))/(max(Days_since_admi
ssion)-min(Days_since_admission)))
```

Create the “num_scores” variable that counts the number of scores each patient has.

```
Mob_3_14 <- Mob_3_14 %>% group_by(PAT_ENC_CSN_ID) %>%
mutate(num_scores = n())
```

Below are the subsets I created from the second DF of Mob_3_14 in order to constrain matching to defined boundaries in surg/non_surg and starting ampac. I will not show matching for each of them; I will only show the matching for the Mob_3_14 DF.

```
Mob_surg <- subset(Mob_3_14, Mob_3_14$ADMFORSURGERY == "Yes")
Mob_nonsurg <- subset(Mob_3_14, Mob_3_14$ADMFORSURGERY == "No")

Mob_surg_30 <- subset(Mob_surg, Mob_surg$first_amp <= 30)
Mob_surg_50 <- subset(Mob_surg, Mob_surg$first_amp <= 50 &
Mob_surg$first_amp > 30)
Mob_surg_80 <- subset(Mob_surg, Mob_surg$first_amp <= 80 &
Mob_surg$first_amp > 50)

Mob_nonsurg_30 <- subset(Mob_nonsurg, Mob_nonsurg$first_amp <= 30)
Mob_nonsurg_50 <- subset(Mob_nonsurg, Mob_nonsurg$first_amp <= 50 &
Mob_nonsurg$first_amp > 30)
Mob_nonsurg_80 <- subset(Mob_nonsurg, Mob_nonsurg$first_amp <= 80 &
Mob_nonsurg$first_amp > 50)

Mob_surg_2 <- subset(Mob_surg, Mob_surg$first_hlm <= 2)
Mob_surg_3 <- subset(Mob_surg, Mob_surg$first_hlm <= 5 &
Mob_surg$first_hlm > 2)
Mob_surg_6 <- subset(Mob_surg, Mob_surg$first_hlm >=  6)

Mob_nonsurg_2 <- subset(Mob_nonsurg, Mob_nonsurg$first_hlm <= 2)
Mob_nonsurg_3 <- subset(Mob_nonsurg, Mob_nonsurg$first_hlm <= 5 &
Mob_nonsurg$first_hlm > 2)
Mob_nonsurg_6 <- subset(Mob_nonsurg, Mob_nonsurg$first_hlm >=  6)
```

Create a DF from Mob_3_14 that has unique patient ID’s by collapsing the former DF. This is so that matching can occur by linking unique patient ID’s.

```
Unique_ID <-
Mob_3_14[match(unique(Mob_3_14$PAT_ENC_CSN_ID),Mob_3_14$PAT_ENC_CSN_I
D),]
```

Matching Process

In the function, “FallYN” is the variable we are using to separate the control (non-fallers) from the “treated” (fallers), the matching variables follow the “~”.

```
m.out <- matchit(FallYN ~ first_amp + AGE + ADMFORSURGERY +
num_scores, data = Unique_ID, method = "nearest",
                  distance = "logit", caliper = 0.03, ratio = 10 )
summary(m.out, un=FALSE)
```

```
Call:
matchit(formula = FallYN ~ first_amp + AGE + ADMFORSURGERY +
    num_scores, data = Unique_ID, method = "nearest", distance = "logit",
    caliper = 0.03, ratio = 10)

Summary of Balance for Matched Data:
                 Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean eCDF Max
distance            0.0057        0.0057          0.0001     1.0098    0.0001   0.0106
first_amp          42.2624       42.5120         -0.0228     0.8365    0.0306   0.0700
AGE                60.5319       60.4912          0.0028     0.7725    0.0212   0.0600
ADMFORSURGERYNo     0.4787        0.4799         -0.0024          .    0.0012   0.0012
ADMFORSURGERYYes    0.5213        0.5201          0.0024          .    0.0012   0.0012
num_scores          8.5213        8.5382         -0.0058     0.9902    0.0166   0.0543
                 Std. Pair Dist.
distance                  0.0010
first_amp                 0.8744
AGE                       0.7862
ADMFORSURGERYNo           0.8294
ADMFORSURGERYYes          0.8294
num_scores                0.3224

Sample Sizes:
               Control Treated
All           34149.        94
Matched (ESS)   908.36      94
Matched         922.        94
Unmatched     33227.         0
Discarded         0.         0
```

As shown above, there are in total 34149 patients with 94 fallers, and there are 922 matched non-fallers with 94 fallers, all fallers are matched.

Extracting Matches

We can obtain matches through the function get_matches.

```
Matches <- get_matches(m.out)
```

Patient_Falls_Analysis_Summer_Matchin...* × | FallNMatch × | m.out ×

Show Attributes

Name	Type	Value
m.out	list [14] (S3: matchit)	List of length 14
match.matrix	character [94 x 10]	'6309' '22461' '8425' '32115' '10771' '549' '12710' '25652' '17142' '24144' '228 ...
subclass	factor	Factor with 94 levels: "1", "2", "3", "4", "5", "6", ...
weights	double [34243]	0 0 0 0 0 0 ...
model	list [30] (S3: glm, lm)	List of length 30
X	list [34243 x 4] (S3: data.frame)	A data.frame with 34243 rows and 4 columns
call	language	matchit(formula = FallYN ~ first_amp + AGE + ADMFORSURGERY + num_scores, da ...
info	list [11]	List of length 11
estimand	character [1]	'ATT'
formula	formula	FallYN ~ first_amp + AGE + ADMFORSURGERY + num_scores
treat	integer [34243]	0 0 0 0 0 0 ...
distance	double [34243]	0.001264 0.000690 0.000677 0.000668 0.000657 0.000601 ...
discarded	logical [34243]	FALSE FALSE FALSE FALSE FALSE FALSE ...
caliper	double [1]	9.294468e-05
nn	double [6 x 2]	34149 34149 908 922 33227 0 94 94 94 94 0 0 ...

Create DF “Matchmob” that extracts the matched patients from “Mob_3_14” by linking patient ID in “Matches” with those in “Mob_3_14”.

```
Matchmob <- subset(Mob_3_14, Mob_3_14$PAT_ENC_CSN_ID %in%
Matches$PAT_ENC_CSN_ID)
```

Link matched non-fallers to their faller (control) through matching patient ID between “Matchmob” and “Matches”.

```
Matchmob$control <- Matches$subclass[match(Matchmob$PAT_ENC_CSN_ID,
Matches$PAT_ENC_CSN_ID)]
```

lls_Analysis_Summer_Matchin...* × | FallNMatch × | m.out ×

Filter

ADMFORSURGERY	first_amp	first_hlm	Days_since_admission	normalized_LOS	num_scores	control	group_id
Yes	41.05	8	1	0.00000000	4	15	1
Yes	41.05	8	2	0.33333333	4	15	1
Yes	41.05	8	3	0.66666667	4	15	1
Yes	41.05	8	4	1.00000000	4	15	1
Yes	41.05	2	0	0.00000000	4	15	2
Yes	41.05	2	1	0.33333333	4	15	2
Yes	41.05	2	2	0.66666667	4	15	2
Yes	41.05	2	3	1.00000000	4	15	2
Yes	41.05	1	0	0.00000000	6	88	3
Yes	41.05	1	1	0.20000000	6	88	3
Yes	41.05	1	2	0.40000000	6	88	3
Yes	41.05	1	3	0.60000000	6	88	3
Yes	41.05	1	4	0.80000000	6	88	3
Yes	41.05	1	5	1.00000000	6	88	3

Separate fallers and non-fallers into two different DF's.

```
FallNMatch <- subset(Matchmob, Matchmob$FallYN == 0)
FallYMatch <- subset(Matchmob, Matchmob$FallYN == 1)
```

For fallers, create the "Days Before Fall Counter" that we will use later to extract data from before fall date for one day before fall date … etc.

```
FallYMatch$BeforeFallDay <- FallYMatch$FALL_DAY1 - 1

FallYMatch <- FallYMatch %>% group_by(PAT_ENC_CSN_ID) %>% mutate(
  DaysBeforeFall = as.numeric(BeforeFallDay-ADMIT_DATE))
FallYMatch <- FallYMatch %>% group_by(PAT_ENC_CSN_ID) %>% mutate(
  DaysBeforeFallCounter= as.numeric(DaysBeforeFall -
Days_since_admission))

FallYMatch <- subset(FallYMatch, FallYMatch$DaysBeforeFallCounter >=
0)
```

Give both matched fallers and matched non-fallers new group id.

```
FallNMatch <- FallNMatch %>% group_by(PAT_ENC_CSN_ID) %>%
mutate(group_id = cur_group_id())
FallYMatch <- FallYMatch %>% group_by(PAT_ENC_CSN_ID) %>%
mutate(group_id = cur_group_id())
```

Normalization

Normalize the timeline for both fallers and non-fallers in a comparable way. I am using fall date as % of LOS and I apply this % for non-fallers in their respective LOS to find their “mimic fall day”.

First, find the median % for fall day as a % of LOS for fallers:

```
FallYMatch <- FallYMatch %>% group_by(group_id) %>% mutate(
  normalized_days =
(Days_since_admission-min(Days_since_admission))/(max(Days_since_admi
ssion)-min(Days_since_admission))
  )
YNorm <- subset(FallYMatch, FallYMatch$normalized_days == 1)
YNorm %>% summary(normalized_LOS)
```

From the summary, extract this % and hard-code it for matching fallers:

```
FallNMatch <- subset(FallNMatch, FallNMatch$normalized_LOS <= 0.6125)
FallNMatch <- FallNMatch %>% group_by(PAT_ENC_CSN_ID) %>% mutate(
normalized_days=(normalized_LOS-min(normalized_LOS))/(max(normalized_
LOS)-min(normalized_LOS))
  )
```