UFC Beards

A first Brad-Terry analysis of UFC Beards and wins and losses.

First the libraries that we are using

```
library(BradleyTerry2)
```

```
## Loading required package: lme4
## Loading required package: Matrix
##
## Attaching package: 'BradleyTerry2'
##
## The following object is masked from 'package:plyr':
##
##     baseball
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(knitr)
library(ggplot2)
source("functions_for_prepping_bt_analysis.R")
```

Then load the data which comes from Mike. Note the as.is thing which leaves all the character vectors as strings and does not make them into factors.

```
winner<-read.csv("UFC_winner.csv",as.is=T)
loser<-read.csv("UFC_loser.csv",as.is=T)
predictors<-read.csv("UFC_predictors.csv",as.is=T)
winner<-winner[c(-18,-157),]
loser<-loser[c(-18,-157),]
predictors<-predictors[c(-6,-162),]
predictors$stance[predictors$stance==""]<-"other"
predictors$stance[predictors$stance=="Open Stance "]<-"other"
predictors$stance[predictors$stance=="Switch "]<-"other"
```

Turns out that th weird BradleyTerry thing needs a very specific format to match things up so we label all the unique id columns "ID" and sort the predictor unique ID column to match the order of the levels of the trial-level files

```
names(predictors)[1]<-"ID"
predictors<-arrange(predictors,ID)
row.names(predictors)<-predictors$ID
names(winner)[1]<-"ID"
names(loser)[1]<-"ID"
```

Now stick the winner and loser files together. This is a bit of a hack to get the levels of the factor to play nice

```
all<-rbind(winner,loser)
```

And add a column to keep the winners and losers labeled

```
all$winning<-c(rep("yes",length(winner$ID)),rep("no",length(winner$ID)))
```

Now turn the unique IDs into a factor.

```
all$ID<-as.factor(all$ID)
```

And split the data.frame into winners and losers. Now with the levels correct so that all the matching works

```
winner2<-filter(all,winning=="yes")
loser2<-filter(all,winning=="no")
```

Assemble it into a list of data.frames. This follows the chameleon example from their help thingie. Probably not strictly necessary.

```
beards<-list(winner=winner2,loser=loser2,predictors=predictors)
```

And at last do the model. In this framework the [ID] thing is used for individual level covariates

```
model1<-BTm(player1=winner,player2=loser,
            formula = ~ prev + facehair + ht[ID] + reach[ID] +
            (1|ID), id="ID",data=beards)

summary(model1)
```

```
##
## Call:
## BTm(player1 = winner, player2 = loser, formula = ~prev + facehair +
##     ht[ID] + reach[ID] + (1 | ID), id = "ID", data = beards)
##
## Fixed Effects:
##             Estimate Std. Error z value Pr(>|z|)
## prev        0.003328   0.119863   0.028   0.9779
## facehair   -0.023263   0.097027  -0.240   0.8105
## ht[ID]     -0.044107   0.052209  -0.845   0.3982
## reach[ID]   0.092161   0.036363   2.534   0.0113 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.5048     0.1001   5.045 4.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 2 observations deleted due to missingness
##
## Number of iterations: 15
```

And report some summary stuff from the model

```
model2<-BTm(player1=winner,player2=loser,
            formula = ~ prev + as.factor(facehair)  + ht[ID] + reach[ID] + stance[ID] +
            (1|ID), id="ID",data=beards)

summary(model2)
```

```
##
## Call:
##
## BTm(player1 = winner, player2 = loser, formula = ~prev + as.factor(facehair) +
##     ht[ID] + reach[ID] + stance[ID] + (1 | ID), id = "ID", data = beards)
##
## Fixed Effects:
##                       Estimate Std. Error z value Pr(>|z|)
## prev                  -0.01494    0.12117  -0.123  0.90184
## as.factor(facehair)2 -0.03402    0.15488  -0.220  0.82612
## as.factor(facehair)3 -0.05933    0.20689  -0.287  0.77430
## ht[ID]                -0.05748    0.05319  -1.081  0.27988
## reach[ID]              0.09877    0.03704   2.666  0.00767 **
## stance[ID]other       -1.08387    0.43055  -2.517  0.01182 *
## stance[ID]Southpaw     0.27117    0.16968   1.598  0.11003
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.5199     0.1010   5.145 2.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 2 observations deleted due to missingness
##
## Number of iterations: 15
```

Does this result depend on the type of victory?

2 or less is a TKO or KO

Greater than 3 is submission or decision or some other outcome. Here we split the dataset into those two categories and show that facial hair still does not have an effect.

```
#other wins including decisions and submissions
w1<-filter(winner,method>=3)
l1<-filter(loser,method>=3)
p1<-subset(predictors,predictors$ID%in%w1$ID)
b.out<-set_up_btm(p1,w1,l1)

model.other.outcomes<-BTm(player1=winner,player2=loser,
            formula = ~  facehair  + reach[ID] +
            (1|ID), id="ID",data=b.out)
```

```
## Warning in rowSums(is.na(vars)) | lev %in% separate.ability: longer object
## length is not a multiple of shorter object length
```

```
summary(model.other.outcomes)
```

```
##
## Call:
## BTm(player1 = winner, player2 = loser, formula = ~facehair +
##     reach[ID] + (1 | ID), id = "ID", data = b.out)
##
## Fixed Effects:
##           Estimate Std. Error z value Pr(>|z|)
## facehair  -0.09745    0.14206  -0.686    0.493
## reach[ID]  0.08432    0.03954   2.133    0.033 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.2428     0.2661   0.912    0.362
##
## 164 observations deleted due to missingness
##
## Number of iterations: 5
```

```
#TKO's and KO's

w1<-filter(winner,method<=2)
l1<-filter(loser,method<=2)
p1<-subset(predictors,predictors$ID%in%w1$ID)
b.out<-set_up_btm(p1,w1,l1)

model3<-BTm(player1=winner,player2=loser,
            formula = ~  facehair  + reach[ID] +
            (1|ID), id="ID",data=b.out)
```

```
## Warning in rowSums(is.na(vars)) | lev %in% separate.ability: longer object
## length is not a multiple of shorter object length
```

```
summary(model3)
```

```
##
## Call:
## BTm(player1 = winner, player2 = loser, formula = ~facehair +
##     reach[ID] + (1 | ID), id = "ID", data = b.out)
##
## Fixed Effects:
##           Estimate Std. Error z value Pr(>|z|)
## facehair   0.47709    0.28322   1.685   0.0921 .
## reach[ID]  0.08311    0.07002   1.187   0.2353
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.5746     0.2835   2.027   0.0426 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 114 observations deleted due to missingness
##
## Number of iterations: 8
```

some stuff

```
model.graphics<-BTm(player1=winner,player2=loser,
            formula = ~ prev + facehair  + ht[ID] + reach[ID]  +
            (1|ID), id="ID",data=beards)

summary(model.graphics)
```

```
##
## Call:
## BTm(player1 = winner, player2 = loser, formula = ~prev + facehair +
##     ht[ID] + reach[ID] + (1 | ID), id = "ID", data = beards)
##
## Fixed Effects:
##            Estimate Std. Error z value Pr(>|z|)
## prev       0.003328   0.119863   0.028   0.9779
## facehair  -0.023263   0.097027  -0.240   0.8105
## ht[ID]    -0.044107   0.052209  -0.845   0.3982
## reach[ID]  0.092161   0.036363   2.534   0.0113 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
## Random Effects:
##           Estimate Std. Error z value Pr(>|z|)
## Std. Dev.   0.5048     0.1001   5.045 4.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 2 observations deleted due to missingness
##
## Number of iterations: 15
```

```
out<-get_bt_abilities(model.graphics,predictors)
predictors$ability<-out$abilities
all<-rbind(winner,loser)
beardy<-summarize(group_by(all,name),mean(facehair))
predictors$beardy<-beardy$`mean(facehair)`[match(predictors$name,beardy$name)]

predictors$beardy[predictors$beardy==1]<-"clean_shaven"
predictors$beardy[predictors$beardy==2]<-"bearded"
predictors$beardy[!predictors$beardy%in%c("clean_shaven","bearded")]<-"variable_or_other"
table(predictors$beardy)
```

```
##
##           bearded      clean_shaven variable_or_other
##               115               109               169
```

```
pdf("plot_hist.pdf")
ggplot(predictors,aes(x=ability, fill = beardy)) +
  geom_histogram(binwidth = 0.05)
dev.off()
```

```
## pdf
##   2
```