GRAD

STAT 149 Spring 2016 Final Project, Process Workbook #1

Submitted by: Kendrick Lo (Harvard ID: 70984997), Amy Lee (Harvard ID: 60984077)

Introduction

The goal of this project is to predict whether US Chess members will allow their membership to lapsed some time after joining. The first file, `train.csv`, contains a randomly selected 43,436 observations (75%) from the original data set (one observation per respondent) of 57,915 observations. The variable names and descriptions are as follows:

```
lapsed  (response variable: did the membership lapse [N/Y])
age (age in years)
sex (gender [F/M])
region (geographic region)
nregions (number of regions of residences in the previous 6 years)
memtype (N=normal, A=affiliate, F=family)
memmonths (number of months since joining US Chess)
mem_mag1 (subscribed to membership magazine # 1 [N/Y])
mem_mag2 (subscribed to membership magazine # 2 [N/Y])
hasemail (member has an email address on file [N/Y])
r1 (transformed regular chess rating from 2 years prior)
r2 (transformed regular chess rating from 1 year prior)
r3 (transformed current regular chess rating)
r.quick (transformed current quick-chess rating)
extra (does the member play a particular variant of chess [N/Y])
intl (does the member have an international rating [N/Y])
r.intl (member's transformed international rating)
allgames1yr (number of tournament games played in the past year)
allgames5yr (number of tournament games played in the past 5 years)
fastevents (number of quick/blitz tournaments in the past 5 years)
medevents (number of medium-speed tournaments in the past 5 years)
slowevents (number of regular-speed tournaments in the past 5 years)
nfloor (number of times the member's rating dropped to a minimum in the past 5 years)
```

The second file, `test.csv`, contains the remaining 14,479 observations (25% of the original data set) with the same variables as above but with the response variable withheld.

Our objective is to train a model using the given training data to predict the withheld binary response in `test.csv` as accurately as possible.

Evaluation Method

The evaluation metric for this competition is "Log-Loss":

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$$

Baseline scores and notes about evaluation

Several baseline measures were provided beforehand ("Basic Logistic Regression": 0.66486, "Intercept Only Model": 0.67350). Additionally, we tested other baseline models, including models that simply assumed either all members would certainly lapse (p = 1.0), all members would certainly renew (p = 0.0), or all members were equally like to renew or not (p = 0.5).

Interestingly, we attained log losses of 20.69, 13.85, and 0.69 respectively. The very large loss values prompted us to take a closer look at the evaluation metric. Accordingly to the Kaggle site:

> ... this error metric is used where contestants have to predict that something is true or false with a probability (likelihood) ranging from definitely true (1) to equally true (0.5) to definitely false(0).
>
> The use of log on the error provides extreme punishments for being both confident and wrong. In the worst possible case, a single prediction that something is definitely true (1) when it is actually false will add infinite to your error score and make every other entry pointless [...]

Although it seems obvious in retrospect, it then appeared clear that we needed to employ models capable of providing us with predictions in terms of probabilities, rather than just a specific (and thus often overconfident) class prediction.

Loading and exploring the data

```
# setwd("~/Desktop/stat149/StatKaggle")
data = read.csv("~/Desktop/stat149/StatKaggle/train.csv", header=TRUE)
summary(data)
```

```
##  lapsed         age             sex            region         nregions
##  N:17149   Min.   :  3.00   F   : 5639   reg112 : 5209   Min.   :0.00
##  Y:26287   1st Qu.: 10.00   M   :36898   reg127 : 4936   1st Qu.:1.00
##            Median : 13.00   NA's:  899   reg102 : 4058   Median :1.00
##            Mean   : 23.21                reg132 : 2239   Mean   :1.04
##            3rd Qu.: 31.00                reg137 : 2124   3rd Qu.:1.00
##            Max.   :100.00                reg114 : 1972   Max.   :7.00
##            NA's   :127                   (Other):22898
##  memtype     memmonths      mem_mag1  mem_mag2  hasemail        r1
##  A: 1663   Min.   :   0.0   N:31976   N:39275   N:11371   Min.   : 183
##  F: 3333   1st Qu.:  19.0   Y:11460   Y: 4161   Y:32065   1st Qu.:1090
##  N:38440   Median :  56.0                                 Median :2008
##            Mean   : 122.2                                 Mean   :1942
##            3rd Qu.: 177.0                                 3rd Qu.:2757
##            Max.   :1431.0                                 Max.   :4280
##                                                           NA's   :24325
##        r2              r3          r.quick     extra     intl
##  Min.   : 183.0   Min.   : 183   Min.   : 258   N:41713   N:38982
##  1st Qu.: 943.8   1st Qu.: 691   1st Qu.: 908   Y: 1723   Y: 4454
##  Median :1782.0   Median :1382   Median :1747
##  Mean   :1811.6   Mean   :1558   Mean   :2008
##  3rd Qu.:2647.0   3rd Qu.:2373   3rd Qu.:2999
##  Max.   :4280.0   Max.   :4264   Max.   :6162
##  NA's   :19024    NA's   :6672   NA's   :9871
##      r.intl       allgames1yr     allgames5yr        fastevents
```

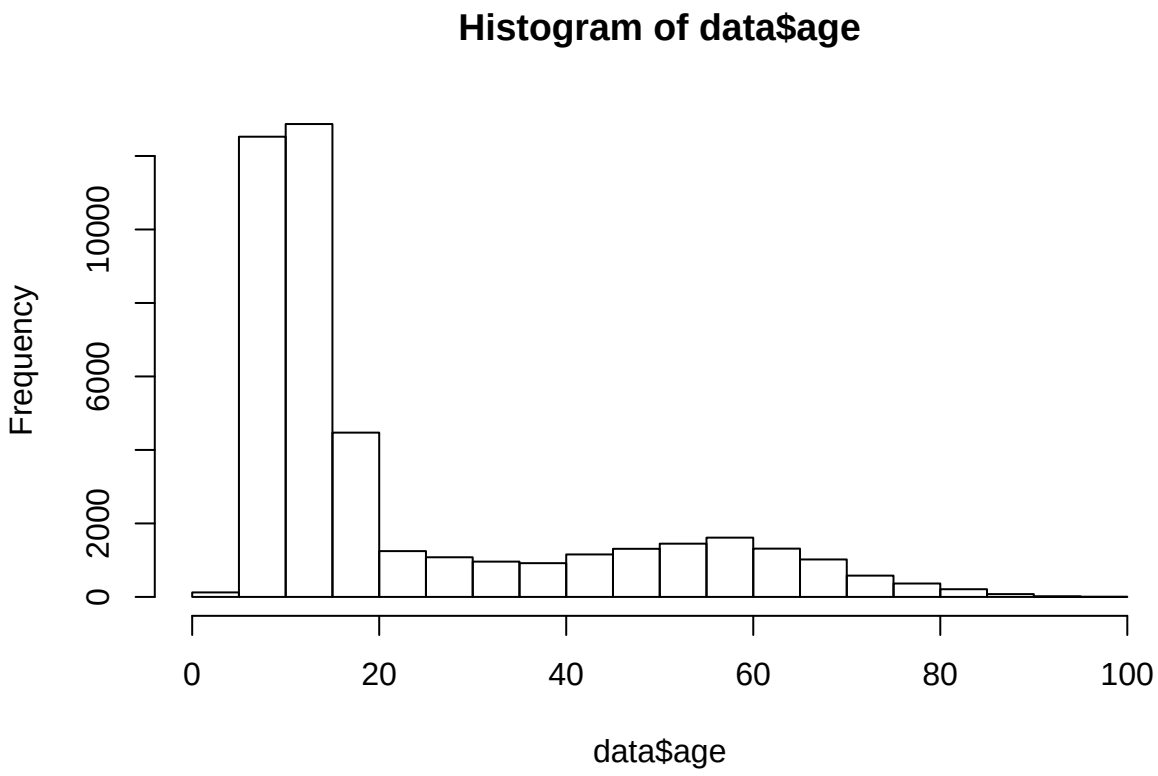
```
##  Min.   :2346    Min.   :  0.0   Min.   :   0.00   Min.   :  0.000
##  1st Qu.:3286    1st Qu.:  0.0   1st Qu.:   4.00   1st Qu.:  0.000
##  Median :3502    Median :  6.0   Median :  14.00   Median :  0.000
##  Mean   :3478    Mean   : 14.3   Mean   :  46.52   Mean   :  1.013
##  3rd Qu.:3714    3rd Qu.: 18.0   3rd Qu.:  49.00   3rd Qu.:  0.000
##  Max.   :4181    Max.   :625.0   Max.   :2687.00   Max.   :434.000
##  NA's   :42259
##    medevents         slowevents          nfloor
##  Min.   :  0.000   Min.   :  0.000   Min.   :  0.0000
##  1st Qu.:  0.000   1st Qu.:  0.000   1st Qu.:  0.0000
##  Median :  2.000   Median :  0.000   Median :  0.0000
##  Mean   :  7.608   Mean   :  4.073   Mean   :  0.1591
##  3rd Qu.:  8.000   3rd Qu.:  2.000   3rd Qu.:  0.0000
##  Max.   :544.000   Max.   :262.000   Max.   :154.0000
##
```

Some initial observations from the summary statistics:

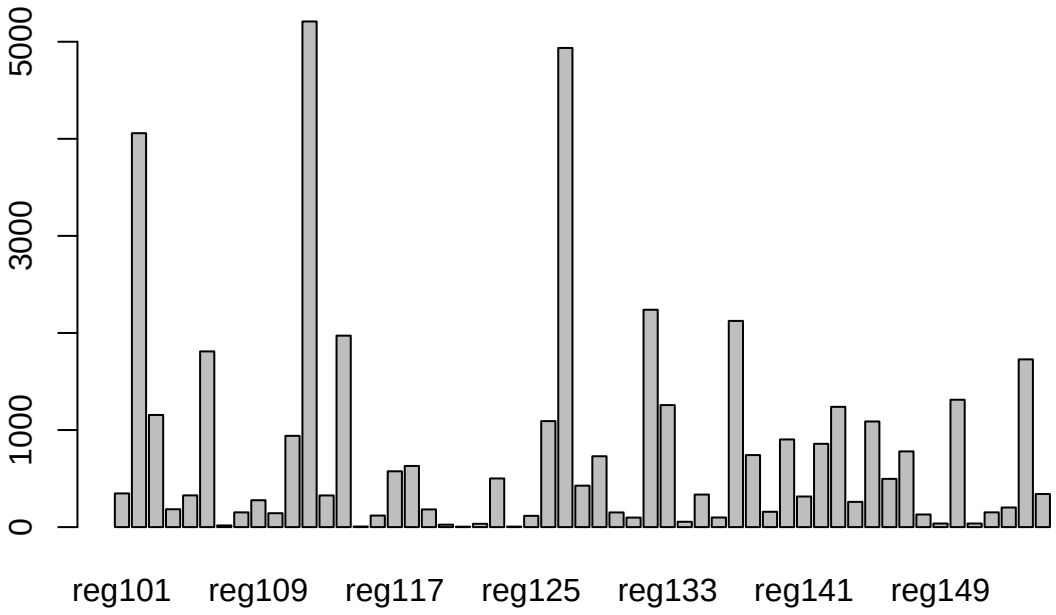
- There are 43,436 observations in the training set. About 60% of the observations are classified as lapsed memberships; therefore, there is some **imbalance** between the two classes
- There is a relatively small set of **missing values** for `age` (127) and `sex` (899) as examples, but a relatively larger set for rating-related data (`r1`: 24325, `r2`: 19024, `r3`: 6672, `r.quick`: 9871, `r.intl`: 42259)
- The maximum value for many of the features is extremely high compared to the mean, suggesting highly **right-skewed distributions**
- Male members outnumber female members by about 7-to-1

We constructed a histogram for various features as part of our exploratory data analysis:

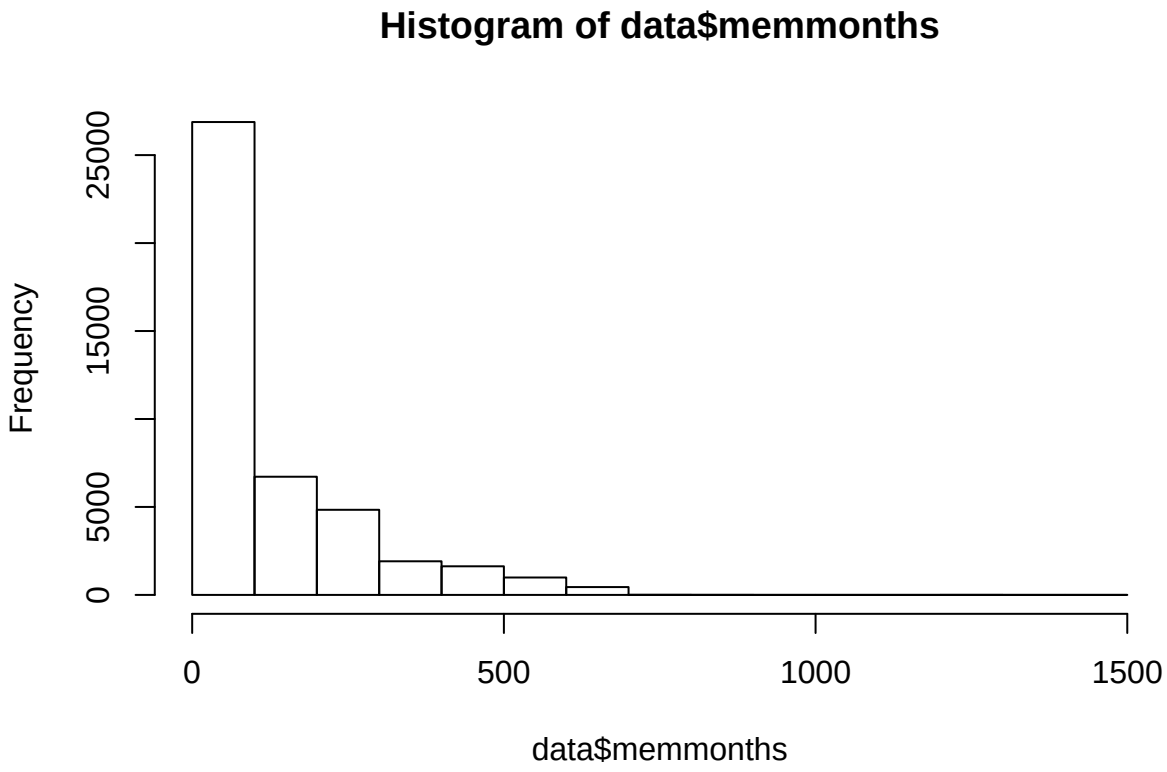
```
hist(data$age)
```



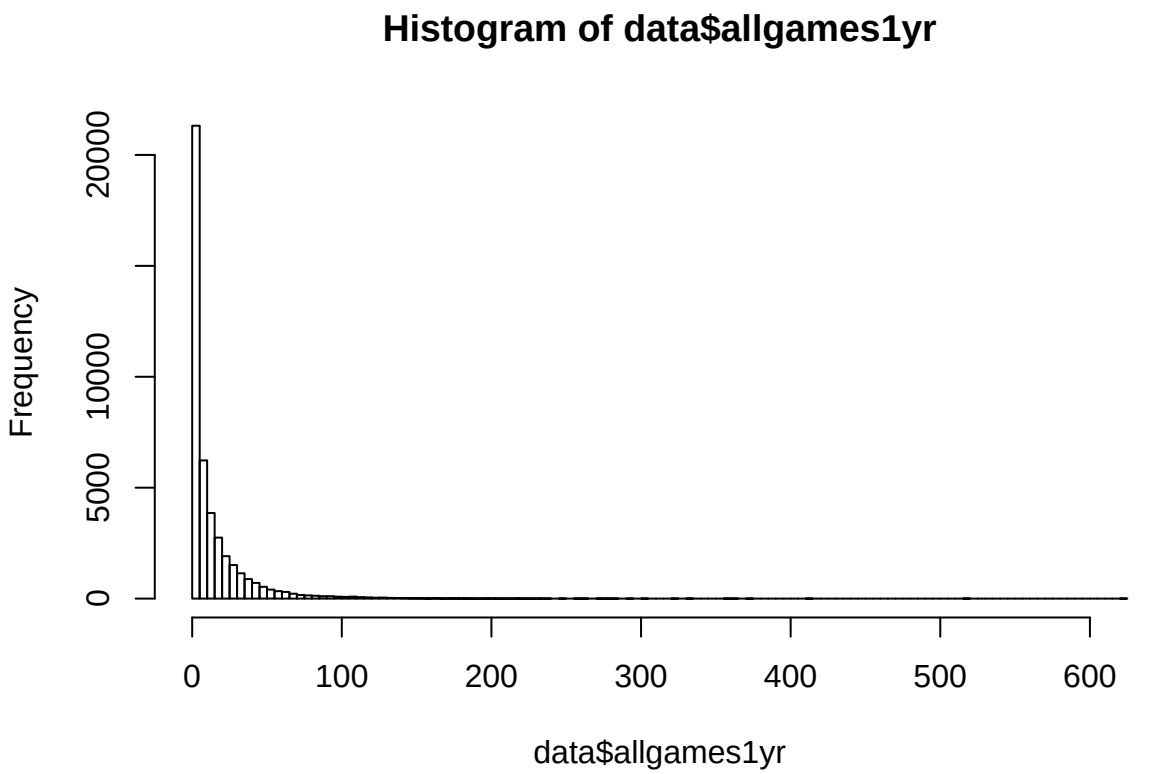

```
plot(data$region)
```



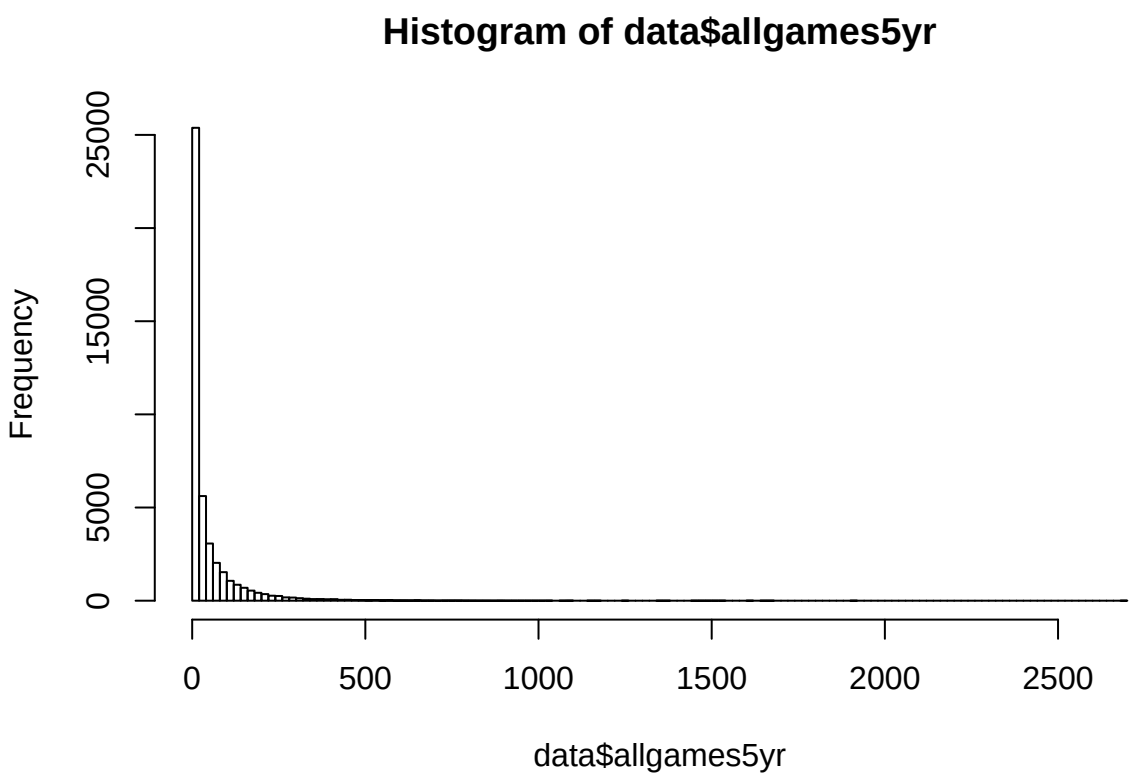

```
hist(data$memmonths)
```




```
hist(data$allgames1yr, breaks=100)
```



```
hist(data$allgames5yr, breaks=100)
```



Some additional observations:

- `age` appears generally unimodal, with the bulk of the observations having a value of less than 20;
- members (`region`) from a handful of regions account for a large proportion of the membership
- the number of months that the member has been a member (`memmonths`) is, expectedly, right-skewed
- ratings data may be roughly uniform for the bulk of the distribution (e.g. `r1`, `r2`), right-skewed (e.g. `r3`, `r.quick`) or even left-skewed (e.g. `r.intl`)
- a significant number of members have an extremely low number of games played causing a significant right-skew (e.g. `allgames1yr`, `allgames5yr`), which may include zeros; similarly some features encode data for relatively few players (e.g. `fastevents`)

Reconstructing the baselines

We now attempted to reconstruct the provided baseline values as a pre-check prior to more extensive modeling. We also reported the Kaggle score of the model's performance on the public test set.

Model 1: Intercept Only

```
model1.glm = glm(lapsed ~ 1, family=binomial, data=data)
summary(model1.glm)
```

```
##
```

```
## Call:
## glm(formula = lapsed ~ 1, family = binomial, data = data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.363  -1.363   1.002   1.002   1.002
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.427135   0.009816   43.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 58278  on 43435  degrees of freedom
## AIC: 58280
##
## Number of Fisher Scoring iterations: 4
```

```
test = read.csv("~/Desktop/stat149/StatKaggle/test.csv", header=TRUE)
summary(test)
Id = test$Id
test$Id = NULL
```

```
model1.newpred = predict(model1.glm, newdata=test, type="response", se.fit=T)
lapsed = model1.newpred$fit
```

```
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model1.csv", row.names=FALSE)
```

Success. Kaggle test score for `model1` is 0.67350.

Models 2 and 3: Basic Logistic Regression

We ran into a problem when fitting the logistic regression as-is; apparently there is a problem with the factor variable encoding.

```
# http://stackoverflow.com/questions/18171246/
# error-in-contrasts-when-defining-a-linear-model-in-r
l<-sapply(data, function(x)is.factor(x))
m<-data[,names(which(l=="TRUE"))]
ifelse(n<-sapply(m,function(x)length(levels(x)))==1,"DROP","NODROP")
summary(data$region["NA"])
```

We checked for any variables that did not actually have multiple levels, but did not find any. However, we did find a small number of NA values in `sex` (341) and `region1` (1). As a preliminary measure, believing this might be the source of the error, we dropped the rows containing NA values in those variables.

Upon further investigation, it appeared that inclusion of the `r.intl` variable was causing problems. As a preliminary measure, since `intl` and `r.intl` codes the same thing (but in binary form), we ignored the `intl` column for the time being.

```
data$intl = as.integer(data$r.intl)
```

```
model2.glm = glm(lapsed ~ age + sex + region + nregions + memtype + memmonths
                 + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 + r.quick
                 + extra + intl + allgames1yr + allgames5yr + fastevents
                 + medevents + slowevents + nfloor, family=binomial, data=data)
summary(model2.glm)
```

We then encountered a new issue, that the data appeared to have new levels not represented in the training data. We expanded the level definition to include these:

```
test2 = test
test2$intl = as.integer(test2$r.intl)

#http://stackoverflow.com/questions/22315394/factor-has-new-levels-error-for-variable-im-not-using
model2.glm$xlevels[["region"]] <- union(model2.glm$xlevels[["region"]], levels(test2$region))

model2.newpred = predict(model2.glm, newdata=test2, type="response", se.fit=T)
lapsed = model2.newpred$fit
summary(lapsed)
```

Then, there arose a problem that `lapsed` has a series of NA values. We will have to deal with these at some point. We tried substituting a probability of 0.5 for each of the NAs:

```
lapsed[is.na(lapsed)] <- 0.5

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model2.csv", row.names=FALSE)
```

Kaggle score for `model2`: 2.14 (bad)

It was then that we noticed on the website that both `intl` and `region` were simply dropped, and that 0.605189 was substituted for the NA values, to obtain the second baseline measure. So we attempted to replicate that:

```
model3.glm = glm(lapsed ~ age + sex + nregions + memtype + memmonths
                 + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 + r.quick
                 + extra + intl + allgames1yr + allgames5yr + fastevents
                 + medevents + slowevents + nfloor, family=binomial, data=data)
summary(model3.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age + sex + nregions + memtype + memmonths +
##     mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 + r.quick +
##     extra + intl + allgames1yr + allgames5yr + fastevents + medevents +
```

```
##     slowevents + nfloor, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2066  -0.9973   0.5126   0.9189   3.3727
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.073e+00  1.418e-01  14.627  < 2e-16 ***
## age         -2.520e-02  1.444e-03 -17.448  < 2e-16 ***
## sexM         5.305e-02  6.214e-02   0.854 0.393231
## nregions     5.664e-02  5.902e-02   0.960 0.337172
## memtypeF    -2.434e-02  1.214e-01  -0.201 0.841080
## memtypeN     5.061e-01  1.082e-01   4.679 2.89e-06 ***
## memmonths   -4.909e-04  1.558e-04  -3.150 0.001631 **
## mem_mag1Y   -1.790e-01  4.490e-02  -3.986 6.73e-05 ***
## mem_mag2Y   -5.448e-01  7.841e-02  -6.948 3.70e-12 ***
## hasemailY   -6.979e-01  4.752e-02 -14.685  < 2e-16 ***
## r1           2.573e-04  8.834e-05   2.913 0.003583 **
## r2           5.630e-04  1.254e-04   4.490 7.13e-06 ***
## r3          -8.849e-04  1.287e-04  -6.874 6.23e-12 ***
## r.quick     -3.791e-05  7.098e-05  -0.534 0.593258
## extraY      -3.366e-01  7.979e-02  -4.218 2.46e-05 ***
## intlY       -5.115e-01  5.464e-02  -9.360  < 2e-16 ***
## allgames1yr -2.902e-02  1.517e-03 -19.129  < 2e-16 ***
## allgames5yr -4.017e-03  7.595e-04  -5.289 1.23e-07 ***
## fastevents  -9.777e-03  2.581e-03  -3.788 0.000152 ***
## medevents    1.603e-02  2.823e-03   5.678 1.36e-08 ***
## slowevents   6.284e-03  2.814e-03   2.233 0.025549 *
## nfloor      -1.127e-02  9.647e-03  -1.169 0.242506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 23312  on 16829  degrees of freedom
## Residual deviance: 19194  on 16808  degrees of freedom
##   (26606 observations deleted due to missingness)
## AIC: 19238
##
## Number of Fisher Scoring iterations: 5
```

```
model3.newpred = predict(model3.glm, newdata=test2, type="response", se.fit=T)
lapsed = model3.newpred$fit
lapsed[is.na(lapsed)] <- 0.605189
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model3.csv", row.names=FALSE)
```

Success. Kaggle score for `model3` was 0.66486. However, there were many NAs (nearly 15,000) where we simply imputed 0.6 values.

Accordingly, we made some observations regarding this data:

- We needed a strategy to deal with the NA issue. It was unclear whether we should simply delete these training records, or impute missing values.
- From some of the summary statistics above, it looked like certain variables might be particularly helpful (e.g. `allgames1yr`, `age`).

Models 4 and 5: Age and Games Played

With regard to the last observation, we experimented with a simple model that includes only `allgames1yr` and `age`, as well as a model that includes interactions between the two:

```
model4.glm = glm(lapsed ~ age + allgames1yr, family=binomial, data=data)
summary(model4.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age + allgames1yr, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8666  -1.1317   0.7154   0.8656   4.0820
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.6288680  0.0206466   78.89   <2e-16 ***
## age         -0.0264446  0.0005418  -48.81   <2e-16 ***
## allgames1yr -0.0427212  0.0007280  -58.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58140  on 43308  degrees of freedom
## Residual deviance: 51484  on 43306  degrees of freedom
##   (127 observations deleted due to missingness)
## AIC: 51490
##
## Number of Fisher Scoring iterations: 5
```

```
model4.newpred = predict(model4.glm, newdata=test, type="response", se.fit=T)
lapsed = model4.newpred$fit
summary(lapsed)
lapsed[is.na(lapsed)] <- 0.605189
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model4.csv", row.names=FALSE)
```

Kaggle score for `model4`: 0.59537

```
model5.glm = glm(lapsed ~ age * allgames1yr, family=binomial, data=data)
summary(model5.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age * allgames1yr, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8801  -1.1186   0.7107   0.8657   4.0917
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.663e+00  2.215e-02  75.106  < 2e-16 ***
## age             -2.774e-02  6.165e-04 -45.001  < 2e-16 ***
## allgames1yr     -4.638e-02  1.108e-03 -41.847  < 2e-16 ***
## age:allgames1yr  1.908e-04  4.205e-05   4.537  5.7e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58140  on 43308  degrees of freedom
## Residual deviance: 51464  on 43305  degrees of freedom
##   (127 observations deleted due to missingness)
## AIC: 51472
##
## Number of Fisher Scoring iterations: 5
```

```
model5.newpred = predict(model5.glm, newdata=test, type="response", se.fit=T)
lapsed = model5.newpred$fit
summary(lapsed)
lapsed[is.na(lapsed)] <- 0.605189
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model5.csv", row.names=FALSE)
```

Kaggle score for `model5`: 0.59509

We now clear our environment for the proceeding work.

```
rm(list = ls())
```

Further Exploration of Logistic Regression Models

Note re: Cross-Validation

Cross-validation is commonly used when building prediction models as it allows us to use existing training data to obtain unbiased estimates of the error in prediction that we are likely to obtain on the final test set.

In particular, if we have to make a comparison between different models, and to select a subset for use in our "final" predictions, the estimates of the test error become critical. Otherwise, our models will tend to overfit the training data.

Furthermore, if we simply relied upon the Kaggle scores to determine which model will likely perform the best on the withheld data, we are essentially fitting to the public test data. This leaves us without an objective basis for determining how the model will actually perform on the private, withheld test data.

There are a number of packages (e.g. caret) in R that allows us to incorporate cross-validation techniques, and we may also implement it by hand. Some R algorithms also build in means for performing cross-validation (e.g. to tune parameters of the model).

Performing cross-validation on all our models will be very time intensive; due to time constraints we have simply trained on all the given training data for models explored in this notebook, and employed the public test set as our "validation set", although we note that if we were to be more thorough we might build our models on only a portion of the training data. (See Workbook 4).

Missing predictor variable data

We now address the issue of the missing predictor values in greater detail. There are a variety of common approaches; however, for the sake of this exercise we will proceed with the crude method discussed in class: inferring a regression-adjusted effect for a predictor's missing values. In particular, for categorical variables we created a new factor level indicating that the value was missing; for quantitative variables, we impute some value where data is missing (e.g. a mean value), but we accompany that with a corresponding setting of a new binary response variable that indicates the value was initially missing.

The major assumption behind this technique is that the missing data mechanism is "ignorable" for each variable with missing data. This is somewhat simplistic, and there are other approaches (e.g. multiple imputation) that could be used in future investigations.

Note: we labeled new NA factors with a different name other than `NA` to prevent the variable converting back to NA when it is re-read.

```
# reload data
data = read.csv("~/Desktop/stat149/StatKaggle/train.csv", header=TRUE)

# replace function (credit: course website)
na.convert.mean = function (frame)
{
    vars <- names(frame)
    if (!is.null(resp <- attr(attr(frame, "terms"), "response"))) {
        vars <- vars[-resp]
        x <- frame[[resp]]
        pos <- is.na(x)
        if (any(pos)) {
            frame <- frame[!pos, , drop = FALSE]
            warning(paste(sum(pos), "observations omitted due to missing values in the response"))
        }
    }
    for (j in vars) {  #j is variable names
        x <- frame[[j]]
        pos <- is.na(x)
        if (any(pos)) {
            if (length(levels(x))) {   # factors
                xx <- as.character(x)
                xx[pos] <- "Z_dummy_NA"  # changed this
```

```
                x <- factor(xx, exclude = NULL)
            }
            else if (is.matrix(x)) {   # matrices
                ats <- attributes(x)
                x.na <- 1*pos
#               x[pos] <- 0
                w <- !pos
                n <- nrow(x)
                TT <- array(1, c(1, n))
                xbar <- (TT %*% x)/(TT %*% w)
                xbar <- t(TT) %*% xbar
                print(xbar[pos])
                x[pos] <- xbar[pos]
                attributes(x) <- ats
                attributes(x.na) <- ats
                dimnames(x.na)[[2]]=paste(dimnames(x)[[2]],".na",sep='')
                frame[[paste(j,".na",sep='')]] <- x.na
            } else {   # ordinary numerical vector
                ats <- attributes(x)
                x[pos] <- mean(x[!pos])
#               x[pos] <- 0
                x.na <- 1*pos
                frame[[paste(j,".na",sep='')]] <- x.na
                attributes(x) <- ats
            }
            frame[[j]] <- x
        }
    }
    frame
}

data.na = na.convert.mean(data)
summary(data.na)
```

```
##  lapsed         age                sex            region
##  N:17149   Min.   :  3.00   F         : 5639   reg112 : 5209
##  Y:26287   1st Qu.: 10.00   M         :36898   reg127 : 4936
##            Median : 13.00   Z_dummy_NA:  899   reg102 : 4058
##            Mean   : 23.21                      reg132 : 2239
##            3rd Qu.: 31.00                      reg137 : 2124
##            Max.   :100.00                      reg114 : 1972
##                                                (Other):22898
##     nregions    memtype     memmonths      mem_mag1  mem_mag2  hasemail
##  Min.   :0.00   A: 1663   Min.   :   0.0   N:31976   N:39275   N:11371
##  1st Qu.:1.00   F: 3333   1st Qu.:  19.0   Y:11460   Y: 4161   Y:32065
##  Median :1.00   N:38440   Median :  56.0
##  Mean   :1.04             Mean   : 122.2
##  3rd Qu.:1.00             3rd Qu.: 177.0
##  Max.   :7.00             Max.   :1431.0
##
##        r1             r2             r3          r.quick     extra
##  Min.   : 183   Min.   : 183   Min.   : 183   Min.   : 258   N:41713
##  1st Qu.:1942   1st Qu.:1574   1st Qu.: 801   1st Qu.:1132   Y: 1723
```

```
##  Median :1942   Median :1812   Median :1558   Median :2008
##  Mean   :1942   Mean   :1812   Mean   :1558   Mean   :2008
##  3rd Qu.:1942   3rd Qu.:1983   3rd Qu.:2189   3rd Qu.:2574
##  Max.   :4280   Max.   :4280   Max.   :4264   Max.   :6162
##
##  intl          r.intl      allgames1yr     allgames5yr
##  N:38982   Min.   :2346   Min.   :  0.0   Min.   :   0.00
##  Y: 4454   1st Qu.:3478   1st Qu.:  0.0   1st Qu.:   4.00
##            Median :3478   Median :  6.0   Median :  14.00
##            Mean   :3478   Mean   : 14.3   Mean   :  46.52
##            3rd Qu.:3478   3rd Qu.: 18.0   3rd Qu.:  49.00
##            Max.   :4181   Max.   :625.0   Max.   :2687.00
##
##    fastevents        medevents         slowevents          nfloor
##  Min.   :  0.000   Min.   :  0.000   Min.   :  0.000   Min.   :  0.0000
##  1st Qu.:  0.000   1st Qu.:  0.000   1st Qu.:  0.000   1st Qu.:  0.0000
##  Median :  0.000   Median :  2.000   Median :  0.000   Median :  0.0000
##  Mean   :  1.013   Mean   :  7.608   Mean   :  4.073   Mean   :  0.1591
##  3rd Qu.:  0.000   3rd Qu.:  8.000   3rd Qu.:  2.000   3rd Qu.:  0.0000
##  Max.   :434.000   Max.   :544.000   Max.   :262.000   Max.   :154.0000
##
##      age.na             r1.na          r2.na           r3.na
##  Min.   :0.000000   Min.   :0.00   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.000000   Median :1.00   Median :0.000   Median :0.0000
##  Mean   :0.002924   Mean   :0.56   Mean   :0.438   Mean   :0.1536
##  3rd Qu.:0.000000   3rd Qu.:1.00   3rd Qu.:1.000   3rd Qu.:0.0000
##  Max.   :1.000000   Max.   :1.00   Max.   :1.000   Max.   :1.0000
##
##    r.quick.na       r.intl.na
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.2273   Mean   :0.9729
##  3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
##
```

Model 6: NA converted data frame

We try a model with the converted data frame:

```
model6.glm = glm(lapsed ~ ., family=binomial, data=data.na)
summary(model6.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ ., family = binomial, data = data.na)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6225  -1.0461   0.5943   0.8765   4.0049
##
```

```
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.310e+00  8.571e-01   2.695 0.007031 **
## age           -1.769e-02  9.259e-04 -19.105  < 2e-16 ***
## sexM          -6.554e-02  3.483e-02  -1.881 0.059907 .
## sexZ_dummy_NA  1.404e+00  1.560e-01   9.001  < 2e-16 ***
## regionreg102   1.133e-01  1.294e-01   0.875 0.381333
## regionreg103   6.187e-02  1.404e-01   0.441 0.659413
## regionreg104   2.464e-01  2.073e-01   1.189 0.234524
## regionreg105   1.004e-01  1.739e-01   0.577 0.563866
## regionreg106  -8.354e-02  1.351e-01  -0.618 0.536427
## regionreg107   2.745e-01  5.416e-01   0.507 0.612229
## regionreg108   2.881e-01  2.235e-01   1.289 0.197425
## regionreg109   4.371e-01  1.847e-01   2.366 0.017971 *
## regionreg110   1.058e-01  2.330e-01   0.454 0.649854
## regionreg111   1.170e-01  1.439e-01   0.813 0.416058
## regionreg112   2.673e-01  1.285e-01   2.080 0.037523 *
## regionreg113   5.957e-01  1.812e-01   3.287 0.001011 **
## regionreg114   1.394e-01  1.350e-01   1.033 0.301729
## regionreg115   5.851e-01  1.120e+00   0.522 0.601417
## regionreg116   4.489e-01  2.522e-01   1.780 0.075034 .
## regionreg117   4.877e-02  1.548e-01   0.315 0.752696
## regionreg118   1.144e-02  1.524e-01   0.075 0.940172
## regionreg119  -6.290e-02  2.045e-01  -0.308 0.758430
## regionreg120  -6.733e-02  4.420e-01  -0.152 0.878919
## regionreg121   7.505e-01  1.255e+00   0.598 0.549758
## regionreg122   8.070e-01  4.452e-01   1.813 0.069857 .
## regionreg123   4.442e-03  1.583e-01   0.028 0.977618
## regionreg124   4.854e-02  9.315e-01   0.052 0.958438
## regionreg125   2.083e-01  2.401e-01   0.867 0.385681
## regionreg126   2.759e-01  1.424e-01   1.938 0.052683 .
## regionreg127   1.455e-01  1.284e-01   1.133 0.257074
## regionreg128  -7.038e-02  1.633e-01  -0.431 0.666384
## regionreg129   4.005e-01  1.508e-01   2.655 0.007939 **
## regionreg130   1.288e-01  2.247e-01   0.573 0.566729
## regionreg131  -5.930e-02  2.549e-01  -0.233 0.816006
## regionreg132   3.830e-01  1.335e-01   2.868 0.004125 **
## regionreg133   1.559e-01  1.390e-01   1.121 0.262163
## regionreg134   1.095e-01  3.143e-01   0.349 0.727445
## regionreg135  -5.907e-02  1.743e-01  -0.339 0.734669
## regionreg136   1.244e-02  2.525e-01   0.049 0.960701
## regionreg137   9.701e-02  1.330e-01   0.729 0.465838
## regionreg138   6.305e-02  1.490e-01   0.423 0.672161
## regionreg139   1.636e-01  2.141e-01   0.764 0.444828
## regionreg140   4.713e-01  1.466e-01   3.214 0.001310 **
## regionreg141   2.061e-01  1.769e-01   1.166 0.243811
## regionreg142   1.950e-01  1.458e-01   1.337 0.181096
## regionreg143   2.587e-01  1.399e-01   1.849 0.064428 .
## regionreg144  -4.728e-01  1.877e-01  -2.519 0.011774 *
## regionreg145   1.136e-01  1.432e-01   0.793 0.427589
## regionreg146   1.449e-01  1.602e-01   0.905 0.365509
## regionreg147  -1.115e-01  1.473e-01  -0.757 0.449041
## regionreg148  -4.371e-01  2.269e-01  -1.926 0.054061 .
## regionreg149   3.030e-01  3.744e-01   0.809 0.418339
```

```
## regionreg150   1.623e-01  1.382e-01   1.174 0.240430
## regionreg151  -4.340e-01  3.775e-01  -1.150 0.250277
## regionreg152   6.041e-02  2.185e-01   0.276 0.782190
## regionreg153   2.210e-02  1.986e-01   0.111 0.911377
## regionreg154  -2.156e-02  1.357e-01  -0.159 0.873724
## regionreg155  -1.158e-01  1.714e-01  -0.676 0.499075
## nregions       9.561e-02  4.967e-02   1.925 0.054242 .
## memtypeF       6.152e-02  7.125e-02   0.863 0.387881
## memtypeN       6.035e-01  6.035e-02  10.000  < 2e-16 ***
## memmonths     -8.356e-04  1.113e-04  -7.507 6.03e-14 ***
## mem_mag1Y     -1.432e-01  3.316e-02  -4.319 1.57e-05 ***
## mem_mag2Y     -3.490e-01  3.867e-02  -9.026  < 2e-16 ***
## hasemailY     -8.410e-01  2.834e-02 -29.677  < 2e-16 ***
## r1            -1.159e-04  4.331e-05  -2.676 0.007461 **
## r2             7.075e-05  4.859e-05   1.456 0.145429
## r3            -2.381e-04  4.497e-05  -5.295 1.19e-07 ***
## r.quick        1.120e-04  2.820e-05   3.974 7.07e-05 ***
## extraY        -4.728e-01  5.871e-02  -8.053 8.06e-16 ***
## intlY         -6.672e-01  4.971e-02 -13.422  < 2e-16 ***
## r.intl        -3.437e-05  2.389e-04  -0.144 0.885609
## allgames1yr   -3.852e-02  1.075e-03 -35.832  < 2e-16 ***
## allgames5yr   -3.314e-03  7.040e-04  -4.707 2.51e-06 ***
## fastevents    -8.989e-03  2.390e-03  -3.762 0.000169 ***
## medevents      1.815e-02  2.707e-03   6.704 2.02e-11 ***
## slowevents     1.063e-02  2.580e-03   4.119 3.81e-05 ***
## nfloor        -2.465e-02  9.675e-03  -2.548 0.010823 *
## age.na         1.308e+00  3.289e-01   3.978 6.96e-05 ***
## r1.na         -1.413e-01  6.196e-02  -2.281 0.022562 *
## r2.na         -1.801e-01  6.042e-02  -2.981 0.002875 **
## r3.na          3.210e-01  5.531e-02   5.803 6.50e-09 ***
## r.quick.na     1.492e-01  4.552e-02   3.277 0.001047 **
## r.intl.na     -3.620e-01  8.971e-02  -4.035 5.46e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 48911  on 43352  degrees of freedom
## AIC: 49079
##
## Number of Fisher Scoring iterations: 5
```

Interestingly, many of the `.na` are significant, which suggests that their missingness is not truly ignorable in the manner; there is something about their missingness that has explanatory power in the model.

```
# reread the test
test = read.csv("~/Desktop/stat149/StatKaggle/test.csv", header=TRUE)
Id = test$Id
test$Id = NULL

# try performing same transformations on test data
test.na = na.convert.mean(test)
summary(test.na)
```

Initially, we were going to convert the test data to undergo the same transformation using the exact same function as was performed on the training data, but we realized that at least for quantitative variables, it would not make that much sense since we planned to impute missing values using an average. Where does this average come from? Should we use the average from just the test data? From the training data? Or should we combine the two datasets when imputing the missing values?

There seems to be more of a basis to use averages from the training data alone to avoid tainting the test data set. That said, there are a large number of data points in the test set that the difference between imputed values may well be negligible (this would not necessarily be the case if the test data set was small, since we could be at the mercy of the other data records having values for a given predictor that were abnormally low or high).

As an experiment, we converted the training and test data separately, and examined the imputed values for the quantitative variables:

- `age`: 23.21233 for training set, 23.3894 for test set
- `r1`: 1942.116 for training set, 1975.189 for test set
- `r2`: 1811.606 for training set, 1831.113 for test set
- `r3`: 1570.301 for training set, 1570.301 for test set
- `r.quick`: 2007.736 for training set, 2018.476 for test set
- `r.intl`: 3477.557 for training set, 3497.182 for test set

The imputed numbers calculated from just test set data do not seem too far off; however, for all variables except `r3` there was a still a difference of about 1%. Since we actually were able to extract the mean values for the quantitative variables based on the training data alone, it seemed more sensible to impute those on the test set as well. The categorical variables can, however, be handled in the similar way (a new `NA` level was created).

Accordingly, we modified the technique to be applied to test data:

```
# values imputed from training set
na_age = 23.21233
na_r1 = 1942.116
na_r2 = 1811.606
na_rquick = 2007.736
na_rintl = 3477.557

# replace with values from training set
test.na$age[is.na(test$age)] <- na_age
test.na$r1[is.na(test$r1)] <- na_r1
test.na$r2[is.na(test$r2)] <- na_r2
test.na$r.quick[is.na(test$r.quick)] <- na_rquick
test.na$r.intl[is.na(test$r.intl)] <- na_rintl
```

```
# make new predictions
model6.newpred = predict(model6.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model6.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model6.csv", row.names=FALSE)
```

Kaggle score for `model6`: 0.56551 (improvement)

As previously mentioned, there may be other ways to impute the missing values, and the fact that the summary of the model suggests that the predictors have missing values is something that needs to be considered (e.g. simply imputing the mean may not be helpful) should lead us to consider using a more sophisticated method. However, in the interests of time and for the purpose of this assignment, we will use these "clean" data sets (data.na, test.na) to explore further models.

Eliminating Predictors

Multi-collinearity and VIFs

It is highly likely that some of the predictors are nearly linearly related to each other (e.g. at first glance some of the ratings measures appear to be transformations of others); collinearity can cause serious problems with the estimation of model coefficients as well as intepretations of these.

That said, when the goal is prediction and not interpretation, it becomes less critical to remedy the collinearity problems. We essentially rely on the incremental amount of information that a potentially related predictor variable adds to the predictive power of the model. However, what this also means is that we may be prone to overfitting to the training data in this regard, and we may want a model that potentially generalizes better.

In any event, in order to demonstrate some of the techniques learned, we will remove variables from our "full" model that may be related to others, by computing variance inflation factors. (Note: we are supposed to use `lm` here but it crashes! Using `glm` instead works, although results may be slightly inaccurate).

```
library(car)
# lm seems to crash, we'll use glm instead
chess.vif1 <- glm(lapsed ~ ., family=binomial, data=data.na)
data.frame(vif = car::vif(chess.vif1))
```

```
##              vif.GVIF vif.Df vif.GVIF..1..2.Df..
## age          2.975618      1            1.724998
## sex          1.118741      2            1.028448
## region       1.498466     54            1.003752
## nregions     1.050225      1            1.024805
## memtype      1.210231      2            1.048859
## memmonths    2.291007      1            1.513607
## mem_mag1     1.874986      1            1.369301
## mem_mag2     1.112067      1            1.054546
## hasemail     1.098047      1            1.047877
## r1           6.598757      1            2.568805
## r2          10.845849      1            3.293304
## r3          13.879276      1            3.725490
## r.quick      8.418773      1            2.901512
## extra        1.142042      1            1.068664
## intl         1.609114      1            1.268508
## r.intl       1.049200      1            1.024305
## allgames1yr  2.297634      1            1.515795
## allgames5yr 17.154791      1            4.141834
## fastevents   1.108781      1            1.052987
## medevents    9.308926      1            3.051053
## slowevents   4.447695      1            2.108956
## nfloor       1.067022      1            1.032968
## age.na       1.005625      1            1.002809
## r1.na        7.908246      1            2.812160
```

```
## r2.na        7.395784      1            2.719519
## r3.na        3.032676      1            1.741458
## r.quick.na   2.990283      1            1.729243
## r.intl.na    1.428945      1            1.195385
```

```
sqrt(10)
```

```
## [1] 3.162278
```

In this form of the table, we check to see if the last column is less than the square root of 10 (or the first column is greater than 10). We will successively remove predictors with the greatest VIFs, one at a time, on this basis:

```
chess.vif2 <- update(chess.vif1, .~. - allgames5yr)
data.frame(vif = car::vif(chess.vif2))
```

```
chess.vif3 <- update(chess.vif2, .~. - r3)
data.frame(vif = car::vif(chess.vif3))
```

```
##             vif.GVIF vif.Df vif.GVIF..1..2.Df..
## age         2.899733      1            1.702860
## sex         1.114190      2            1.027401
## region      1.456533     54            1.003488
## nregions    1.048709      1            1.024065
## memtype     1.209161      2            1.048627
## memmonths   2.281876      1            1.510588
## mem_mag1    1.873155      1            1.368633
## mem_mag2    1.110304      1            1.053710
## hasemail    1.097250      1            1.047497
## r1          6.616478      1            2.572252
## r2          9.569599      1            3.093477
## r.quick     3.837938      1            1.959066
## extra       1.141514      1            1.068417
## intl        1.601941      1            1.265678
## r.intl      1.049641      1            1.024520
## allgames1yr 2.099956      1            1.449122
## fastevents  1.106939      1            1.052112
## medevents   1.992868      1            1.411690
## slowevents  1.643921      1            1.282155
## nfloor      1.063590      1            1.031305
## age.na      1.005331      1            1.002662
## r1.na       7.857775      1            2.803172
## r2.na       6.597179      1            2.568497
## r3.na       3.010572      1            1.735100
## r.quick.na  2.492967      1            1.578913
## r.intl.na   1.410670      1            1.187716
```

According to this heuristic, we would be able to eliminate `allgames5yr` and `r3` if we wanted to reduce the number of predictors in the model.

Analysis of Deviance / Likelihood Ratio Tests

```
summary(chess.vif3)
```

```
##
## Call:
## glm(formula = lapsed ~ age + sex + region + nregions + memtype +
##     memmonths + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r.quick +
##     extra + intl + r.intl + allgames1yr + fastevents + medevents +
##     slowevents + nfloor + age.na + r1.na + r2.na + r3.na + r.quick.na +
##     r.intl.na, family = binomial, data = data.na)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6091  -1.0490   0.5969   0.8776   4.0284
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.134e+00  8.526e-01   2.502 0.012341 *
## age           -1.740e-02  9.138e-04 -19.041  < 2e-16 ***
## sexM          -7.347e-02  3.472e-02  -2.116 0.034349 *
## sexZ_dummy_NA  1.403e+00  1.559e-01   9.003  < 2e-16 ***
## regionreg102   1.279e-01  1.292e-01   0.990 0.322306
## regionreg103   7.669e-02  1.403e-01   0.547 0.584515
## regionreg104   2.560e-01  2.074e-01   1.235 0.217000
## regionreg105   1.005e-01  1.737e-01   0.579 0.562896
## regionreg106  -6.587e-02  1.350e-01  -0.488 0.625534
## regionreg107   2.399e-01  5.435e-01   0.442 0.658849
## regionreg108   3.112e-01  2.240e-01   1.389 0.164890
## regionreg109   4.488e-01  1.847e-01   2.430 0.015090 *
## regionreg110   1.029e-01  2.327e-01   0.442 0.658300
## regionreg111   1.299e-01  1.437e-01   0.904 0.366064
## regionreg112   2.770e-01  1.284e-01   2.158 0.030943 *
## regionreg113   6.184e-01  1.812e-01   3.413 0.000643 ***
## regionreg114   1.499e-01  1.349e-01   1.111 0.266575
## regionreg115   5.638e-01  1.124e+00   0.502 0.615829
## regionreg116   4.596e-01  2.518e-01   1.825 0.067980 .
## regionreg117   4.604e-02  1.545e-01   0.298 0.765769
## regionreg118   1.790e-02  1.522e-01   0.118 0.906372
## regionreg119  -5.172e-02  2.044e-01  -0.253 0.800208
## regionreg120  -5.572e-02  4.418e-01  -0.126 0.899638
## regionreg121   7.133e-01  1.236e+00   0.577 0.563787
## regionreg122   8.385e-01  4.456e-01   1.882 0.059871 .
## regionreg123   2.456e-02  1.582e-01   0.155 0.876589
## regionreg124   5.339e-02  9.319e-01   0.057 0.954315
## regionreg125   2.373e-01  2.397e-01   0.990 0.322244
## regionreg126   2.967e-01  1.423e-01   2.085 0.037045 *
## regionreg127   1.491e-01  1.283e-01   1.162 0.245142
## regionreg128  -7.429e-02  1.631e-01  -0.455 0.648772
## regionreg129   4.160e-01  1.506e-01   2.762 0.005751 **
## regionreg130   1.475e-01  2.248e-01   0.656 0.511791
## regionreg131  -3.465e-02  2.544e-01  -0.136 0.891634
## regionreg132   3.885e-01  1.334e-01   2.913 0.003584 **
```

```
## regionreg133   1.696e-01  1.389e-01   1.221 0.222054
## regionreg134   1.116e-01  3.139e-01   0.356 0.722160
## regionreg135  -2.237e-02  1.741e-01  -0.128 0.897787
## regionreg136   5.948e-02  2.515e-01   0.236 0.813048
## regionreg137   1.024e-01  1.329e-01   0.770 0.441242
## regionreg138   7.780e-02  1.489e-01   0.523 0.601290
## regionreg139   1.782e-01  2.138e-01   0.833 0.404568
## regionreg140   4.985e-01  1.466e-01   3.401 0.000671 ***
## regionreg141   2.144e-01  1.767e-01   1.214 0.224919
## regionreg142   2.118e-01  1.457e-01   1.454 0.145893
## regionreg143   2.723e-01  1.398e-01   1.948 0.051474 .
## regionreg144  -4.565e-01  1.876e-01  -2.433 0.014961 *
## regionreg145   1.229e-01  1.430e-01   0.859 0.390270
## regionreg146   1.488e-01  1.599e-01   0.931 0.351971
## regionreg147  -1.013e-01  1.472e-01  -0.688 0.491308
## regionreg148  -3.969e-01  2.261e-01  -1.755 0.079187 .
## regionreg149   3.226e-01  3.753e-01   0.860 0.389948
## regionreg150   1.799e-01  1.381e-01   1.303 0.192593
## regionreg151  -4.114e-01  3.762e-01  -1.094 0.274113
## regionreg152   7.553e-02  2.184e-01   0.346 0.729496
## regionreg153   5.068e-02  1.984e-01   0.255 0.798373
## regionreg154   7.341e-03  1.355e-01   0.054 0.956788
## regionreg155  -1.124e-01  1.712e-01  -0.657 0.511493
## nregions       1.046e-01  4.966e-02   2.106 0.035242 *
## memtypeF       5.725e-02  7.123e-02   0.804 0.421592
## memtypeN       5.993e-01  6.036e-02   9.929  < 2e-16 ***
## memmonths     -8.006e-04  1.110e-04  -7.211 5.57e-13 ***
## mem_mag1Y     -1.492e-01  3.313e-02  -4.502 6.72e-06 ***
## mem_mag2Y     -3.404e-01  3.861e-02  -8.815  < 2e-16 ***
## hasemailY     -8.453e-01  2.832e-02 -29.849  < 2e-16 ***
## r1            -1.268e-04  4.337e-05  -2.923 0.003466 **
## r2            -2.474e-05  4.565e-05  -0.542 0.587870
## r.quick        2.116e-06  1.904e-05   0.111 0.911510
## extraY        -4.615e-01  5.871e-02  -7.861 3.82e-15 ***
## intlY         -6.954e-01  4.959e-02 -14.023  < 2e-16 ***
## r.intl        -6.531e-06  2.379e-04  -0.027 0.978095
## allgames1yr   -4.070e-02  1.034e-03 -39.364  < 2e-16 ***
## fastevents    -8.668e-03  2.387e-03  -3.632 0.000281 ***
## medevents      7.435e-03  1.270e-03   5.853 4.82e-09 ***
## slowevents     9.323e-04  1.596e-03   0.584 0.559223
## nfloor        -2.158e-02  9.314e-03  -2.317 0.020501 *
## age.na         1.288e+00  3.284e-01   3.922 8.77e-05 ***
## r1.na         -1.048e-01  6.173e-02  -1.698 0.089490 .
## r2.na         -6.270e-02  5.701e-02  -1.100 0.271386
## r3.na          3.044e-01  5.509e-02   5.525 3.30e-08 ***
## r.quick.na     5.505e-02  4.141e-02   1.329 0.183755
## r.intl.na     -3.048e-01  8.895e-02  -3.426 0.000612 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 48960  on 43354  degrees of freedom
```

```
## AIC: 49124
##
## Number of Fisher Scoring iterations: 5
```

From the summary statistics, we notice that that fit is OK, but not great, by comparing the ratio of the residual deviance and the degrees of freedom, to 1.

We proceeded to test whether any further individual factors can be eliminated, one at a time, based on their significance using an analysis of deviance (there are a large number of predictors so this takes some time):

```
drop1(chess.vif3, test="LRT")
```

We refit the model without `r.intl`, which has the highest p-value (least significant). Note this is the actual rating, and fortunately we have a separate `r.intl.na` variable in the event that not having a reported international rating is significant (which may well be the case, if there is a correlation between having such a rating and the seriousness of a player).

```
temp1.glm <- update(chess.vif3, .~. - r.intl)
```

```
drop1(temp1.glm, test="LRT")
```

We refit the model without `r.quick`, which has the highest p-value (least significant):

```
temp2.glm <- update(temp1.glm, .~. - r.quick)
```

```
drop1(temp2.glm, test="LRT")
```

We refit the model without `r2`, which has the highest p-value (least significant):

```
temp3.glm <- update(temp2.glm, .~. - r2)
```

```
drop1(temp3.glm, test="LRT")
```

We refit the model without `slowevents`, which has the highest p-value (least significant):

```
temp4.glm <- update(temp3.glm, .~. - slowevents)
```

```
drop1(temp4.glm, test="LRT")
```

We refit the model without `r.quick.na`, which has the highest p-value (least significant):

```
temp5.glm <- update(temp4.glm, .~. - r.quick.na)
```

```
drop1(temp5.glm, test="LRT")
```

It appears that all the remaining variables are significant at the 0.05 level.

Model 7: Dropping terms, main effects

We proceeded to make predictions based on the reduced model.

```r
model7.glm <- temp5.glm
summary(model7.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age + sex + region + nregions + memtype +
##     memmonths + mem_mag1 + mem_mag2 + hasemail + r1 + extra +
##     intl + allgames1yr + fastevents + medevents + nfloor + age.na +
##     r1.na + r2.na + r3.na + r.intl.na, family = binomial, data = data.na)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6124  -1.0496   0.5976   0.8769   4.0294
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.108e+00  1.813e-01  11.626  < 2e-16 ***
## age           -1.719e-02  8.856e-04 -19.405  < 2e-16 ***
## sexM          -7.491e-02  3.443e-02  -2.176 0.029561 *
## sexZ_dummy_NA  1.400e+00  1.558e-01   8.984  < 2e-16 ***
## regionreg102   1.296e-01  1.293e-01   1.003 0.315886
## regionreg103   7.597e-02  1.403e-01   0.542 0.588030
## regionreg104   2.529e-01  2.073e-01   1.220 0.222287
## regionreg105   1.022e-01  1.738e-01   0.588 0.556499
## regionreg106  -6.084e-02  1.350e-01  -0.451 0.652176
## regionreg107   2.450e-01  5.413e-01   0.453 0.650833
## regionreg108   3.084e-01  2.240e-01   1.377 0.168504
## regionreg109   4.497e-01  1.847e-01   2.435 0.014906 *
## regionreg110   1.023e-01  2.327e-01   0.439 0.660352
## regionreg111   1.292e-01  1.437e-01   0.899 0.368427
## regionreg112   2.753e-01  1.284e-01   2.145 0.031978 *
## regionreg113   6.157e-01  1.812e-01   3.399 0.000677 ***
## regionreg114   1.507e-01  1.349e-01   1.117 0.264115
## regionreg115   5.619e-01  1.125e+00   0.500 0.617403
## regionreg116   4.618e-01  2.519e-01   1.833 0.066784 .
## regionreg117   5.302e-02  1.545e-01   0.343 0.731491
## regionreg118   2.334e-02  1.522e-01   0.153 0.878092
## regionreg119  -5.352e-02  2.044e-01  -0.262 0.793412
## regionreg120  -5.376e-02  4.418e-01  -0.122 0.903153
## regionreg121   7.208e-01  1.238e+00   0.582 0.560309
## regionreg122   8.373e-01  4.456e-01   1.879 0.060201 .
## regionreg123   2.666e-02  1.582e-01   0.169 0.866126
## regionreg124   5.356e-02  9.319e-01   0.057 0.954167
## regionreg125   2.413e-01  2.394e-01   1.008 0.313460
## regionreg126   2.963e-01  1.422e-01   2.083 0.037255 *
## regionreg127   1.498e-01  1.283e-01   1.168 0.242966
## regionreg128  -7.404e-02  1.631e-01  -0.454 0.649948
## regionreg129   4.168e-01  1.507e-01   2.766 0.005669 **
## regionreg130   1.491e-01  2.248e-01   0.663 0.507162
## regionreg131  -2.513e-02  2.543e-01  -0.099 0.921279
## regionreg132   3.893e-01  1.334e-01   2.918 0.003519 **
## regionreg133   1.688e-01  1.389e-01   1.215 0.224358
## regionreg134   1.175e-01  3.135e-01   0.375 0.707733
```

```
## regionreg135  -2.254e-02  1.741e-01  -0.129 0.896984
## regionreg136   7.583e-02  2.507e-01   0.302 0.762341
## regionreg137   1.023e-01  1.329e-01   0.770 0.441357
## regionreg138   7.600e-02  1.489e-01   0.510 0.609769
## regionreg139   1.809e-01  2.136e-01   0.847 0.397039
## regionreg140   4.960e-01  1.465e-01   3.386 0.000710 ***
## regionreg141   2.148e-01  1.767e-01   1.215 0.224216
## regionreg142   2.138e-01  1.457e-01   1.467 0.142259
## regionreg143   2.721e-01  1.398e-01   1.946 0.051616 .
## regionreg144  -4.590e-01  1.876e-01  -2.447 0.014398 *
## regionreg145   1.243e-01  1.430e-01   0.869 0.384703
## regionreg146   1.587e-01  1.596e-01   0.994 0.320000
## regionreg147  -9.986e-02  1.472e-01  -0.678 0.497533
## regionreg148  -3.898e-01  2.258e-01  -1.726 0.084323 .
## regionreg149   3.264e-01  3.756e-01   0.869 0.384800
## regionreg150   1.815e-01  1.381e-01   1.314 0.188955
## regionreg151  -4.048e-01  3.760e-01  -1.077 0.281662
## regionreg152   7.133e-02  2.184e-01   0.327 0.743921
## regionreg153   5.003e-02  1.984e-01   0.252 0.800951
## regionreg154   1.045e-02  1.355e-01   0.077 0.938519
## regionreg155  -1.128e-01  1.713e-01  -0.658 0.510297
## nregions       1.037e-01  4.964e-02   2.089 0.036746 *
## memtypeF       5.768e-02  7.119e-02   0.810 0.417844
## memtypeN       5.996e-01  6.034e-02   9.937  < 2e-16 ***
## memmonths     -8.060e-04  1.107e-04  -7.279 3.36e-13 ***
## mem_mag1Y     -1.473e-01  3.306e-02  -4.455 8.39e-06 ***
## mem_mag2Y     -3.402e-01  3.856e-02  -8.823  < 2e-16 ***
## hasemailY     -8.452e-01  2.823e-02 -29.940  < 2e-16 ***
## r1            -1.430e-04  2.226e-05  -6.421 1.36e-10 ***
## extraY        -4.596e-01  5.854e-02  -7.852 4.10e-15 ***
## intlY         -6.953e-01  4.861e-02 -14.302  < 2e-16 ***
## allgames1yr   -4.065e-02  9.687e-04 -41.968  < 2e-16 ***
## fastevents    -8.721e-03  2.359e-03  -3.696 0.000219 ***
## medevents      7.308e-03  1.264e-03   5.780 7.47e-09 ***
## nfloor        -2.120e-02  9.257e-03  -2.290 0.022031 *
## age.na         1.294e+00  3.284e-01   3.940 8.16e-05 ***
## r1.na         -8.648e-02  4.189e-02  -2.064 0.038992 *
## r2.na         -8.347e-02  3.796e-02  -2.199 0.027872 *
## r3.na          3.481e-01  4.011e-02   8.679  < 2e-16 ***
## r.intl.na     -3.067e-01  8.822e-02  -3.477 0.000508 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 48962  on 43359  degrees of freedom
## AIC: 49116
##
## Number of Fisher Scoring iterations: 5
```

```
model7.newpred = predict(model7.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model7.newpred$fit
```

```
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model7.csv", row.names=FALSE)
```

Kaggle score for `model7`: 0.56564 (not an improvement)

We note that we might have arrived at a different model comprising only main effects if we started from an intercept-only model. We explored this possibility.

```
temp6.glm <- glm(lapsed ~ 1, family=binomial, data=data.na)
# note chess.vif3 is the full model without the terms having high VIF
add1(temp6.glm, scope = chess.vif3, test="LRT")
```

We refit the model adding `allgames1yr`, which had the highest test statistic (and confirmed that the deviance has dropped significantly):

```
temp7.glm <- glm(lapsed ~ allgames1yr, family=binomial, data=data.na)
add1(temp7.glm, scope = chess.vif3, test="LRT")
```

We refit the model adding `age`, which had the highest test statistic (and confirmed that the deviance has dropped significantly):

```
temp8.glm <- glm(lapsed ~ allgames1yr + age, family=binomial, data=data.na)
add1(temp8.glm, scope = chess.vif3, test="LRT")
```

We refit the model adding `hasemail`, which had the highest test statistic (and confirmed that the deviance has dropped significantly):

```
temp9.glm <- glm(lapsed ~ allgames1yr + age + hasemail, family=binomial, data=data.na)
add1(temp9.glm, scope = chess.vif3, test="LRT")
```

We refit the model adding `intl`, which had the highest test statistic (and confirmed that the deviance has dropped significantly):

```
temp10.glm <- glm(lapsed ~ allgames1yr + age + hasemail
                  + intl, family=binomial, data=data.na)
add1(temp10.glm, scope = chess.vif3, test="LRT")
```

We refit the model adding `memtype`, which had the second-highest test statistic (and confirmed that the deviance has dropped significantly). `region` has the highest test statistics, but has many more degrees of freedom, resulting in an expectedly higher p-value.

```
temp11.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl
                  + memtype, family=binomial, data=data.na)
add1(temp11.glm, scope = chess.vif3, test="LRT")
```

Similarly, we refit the model adding `sex`, `r1`, `region`, `extra`, `mem_mag2`, `r3.na`, `medevents`, `memmonths`, `age.na`, `mem_mag1`, `r2.na`, `fastevents`, `r.intl.na`, `nfloor`, `nregions`, and `r1.na` in like fashion until the remaining variables were not considered to have a large enough deviance reduction to be included in the model.

```
temp12.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex, family=binomial, data=data.na)
add1(temp12.glm, scope = chess.vif3, test="LRT")

temp13.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1, family=binomial, data=data.na)
add1(temp13.glm, scope = chess.vif3, test="LRT")

temp14.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region, family=binomial, data=data.na)
add1(temp14.glm, scope = chess.vif3, test="LRT")

temp15.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra, family=binomial, data=data.na)
add1(temp15.glm, scope = chess.vif3, test="LRT")

temp16.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2, family=binomial, data=data.na)
add1(temp16.glm, scope = chess.vif3, test="LRT")

temp17.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2
                  + r3.na, family=binomial, data=data.na)
add1(temp17.glm, scope = chess.vif3, test="LRT")

temp18.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na
                  + medevents, family=binomial, data=data.na)
add1(temp18.glm, scope = chess.vif3, test="LRT")

temp19.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na
                  + medevents + memmonths, family=binomial, data=data.na)
add1(temp19.glm, scope = chess.vif3, test="LRT")

temp20.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na
                  + medevents + memmonths + age.na,
                  family=binomial, data=data.na)
add1(temp20.glm, scope = chess.vif3, test="LRT")

temp21.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1,
                  family=binomial, data=data.na)
add1(temp21.glm, scope = chess.vif3, test="LRT")

temp22.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na,
                  family=binomial, data=data.na)
add1(temp22.glm, scope = chess.vif3, test="LRT")
```

```
temp23.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na
                  + fastevents, family=binomial, data=data.na)
add1(temp23.glm, scope = chess.vif3, test="LRT")

temp24.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na + fastevents
                  + r.intl.na, family=binomial, data=data.na)
add1(temp24.glm, scope = chess.vif3, test="LRT")

temp25.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na + fastevents
                  + r.intl.na + nfloor, family=binomial, data=data.na)
add1(temp25.glm, scope = chess.vif3, test="LRT")

temp26.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na + fastevents
                  + r.intl.na + nfloor + nregions,
                  family=binomial, data=data.na)
add1(temp26.glm, scope = chess.vif3, test="LRT")

temp27.glm <- glm(lapsed ~ allgames1yr + age + hasemail + intl + memtype
                  + sex + r1 + region + extra + mem_mag2 + r3.na + medevents
                  + memmonths + age.na + mem_mag1 + r2.na + fastevents
                  + r.intl.na + nfloor + nregions + r1.na,
                  family=binomial, data=data.na)
add1(temp27.glm, scope = chess.vif3, test="LRT")
```

It appears that we converged to the same model with main effects as above (this will not always be the case).

Adding Interactions

We then looked to add some interaction terms to the model. Given there were so many predictors even in our reduced model (21 in total), it would be quite cumbersome to try to test all pairwise comparisons and combinations of interactions, and so we tried to pick some interactions that seemed to make sense for our model. We knew to keep in mind that there is no single correct model.

For the purpose of this exercise, we decided to arbitrarily take the first 7 terms that were added to previous model (arguably in order of "importance") and add pairwise interactions. We then conducted an analysis of deviance to determine which interactions to keep.

```
temp28.glm <- update(temp27.glm, .~. + allgames1yr:age)
temp29.glm <- update(temp27.glm, .~. + allgames1yr:hasemail)
temp30.glm <- update(temp27.glm, .~. + allgames1yr:intl)
temp31.glm <- update(temp27.glm, .~. + allgames1yr:memtype)
temp32.glm <- update(temp27.glm, .~. + allgames1yr:sex)
temp33.glm <- update(temp27.glm, .~. + allgames1yr:r1)
temp34.glm <- update(temp27.glm, .~. + age:hasemail)
temp35.glm <- update(temp27.glm, .~. + age:intl)
```

```
temp36.glm <- update(temp27.glm, .~. + age:memtype)
temp37.glm <- update(temp27.glm, .~. + age:sex)
temp38.glm <- update(temp27.glm, .~. + age:r1)
temp39.glm <- update(temp27.glm, .~. + hasemail:intl)
temp40.glm <- update(temp27.glm, .~. + hasemail:memtype)
temp41.glm <- update(temp27.glm, .~. + hasemail:sex)
temp42.glm <- update(temp27.glm, .~. + hasemail:r1)
temp43.glm <- update(temp27.glm, .~. + intl:memtype)
temp44.glm <- update(temp27.glm, .~. + intl:sex)
temp45.glm <- update(temp27.glm, .~. + intl:r1)
temp46.glm <- update(temp27.glm, .~. + memtype:sex)
temp47.glm <- update(temp27.glm, .~. + memtype:r1)
temp48.glm <- update(temp27.glm, .~. + sex:r1)

anova(temp28.glm, temp29.glm, temp30.glm, temp31.glm, temp32.glm, temp33.glm,
      temp34.glm, temp35.glm, temp36.glm, temp37.glm, temp38.glm, temp39.glm,
      temp40.glm, temp41.glm, temp42.glm, temp43.glm, temp44.glm, temp45.glm,
      temp46.glm, temp47.glm, temp48.glm,  test="Chi")

# keep age:memtype
temp49.glm <- update(temp36.glm, .~. + allgames1yr:age)
temp50.glm <- update(temp36.glm, .~. + allgames1yr:hasemail)
temp51.glm <- update(temp36.glm, .~. + allgames1yr:intl)
temp52.glm <- update(temp36.glm, .~. + allgames1yr:memtype)
temp53.glm <- update(temp36.glm, .~. + allgames1yr:sex)
temp54.glm <- update(temp36.glm, .~. + allgames1yr:r1)
temp55.glm <- update(temp36.glm, .~. + age:hasemail)
temp56.glm <- update(temp36.glm, .~. + age:intl)
temp57.glm <- update(temp36.glm, .~. + age:sex)
temp58.glm <- update(temp36.glm, .~. + age:r1)
temp59.glm <- update(temp36.glm, .~. + hasemail:intl)
temp60.glm <- update(temp36.glm, .~. + hasemail:memtype)
temp61.glm <- update(temp36.glm, .~. + hasemail:sex)
temp62.glm <- update(temp36.glm, .~. + hasemail:r1)
temp63.glm <- update(temp36.glm, .~. + intl:memtype)
temp64.glm <- update(temp36.glm, .~. + intl:sex)
temp65.glm <- update(temp36.glm, .~. + intl:r1)
temp66.glm <- update(temp36.glm, .~. + memtype:sex)
temp67.glm <- update(temp36.glm, .~. + memtype:r1)
temp68.glm <- update(temp36.glm, .~. + sex:r1)

anova(temp49.glm, temp50.glm, temp51.glm, temp52.glm, temp53.glm, temp54.glm,
      temp55.glm, temp56.glm, temp57.glm, temp58.glm, temp59.glm, temp60.glm,
      temp61.glm, temp62.glm, temp63.glm, temp64.glm, temp65.glm, temp66.glm,
      temp67.glm, temp68.glm,  test="Chi")

# keep memtype:r1
temp69.glm <- update(temp67.glm, .~. + allgames1yr:age)
temp70.glm <- update(temp67.glm, .~. + allgames1yr:hasemail)
temp71.glm <- update(temp67.glm, .~. + allgames1yr:intl)
temp72.glm <- update(temp67.glm, .~. + allgames1yr:memtype)
temp73.glm <- update(temp67.glm, .~. + allgames1yr:sex)
temp74.glm <- update(temp67.glm, .~. + allgames1yr:r1)
```

```
temp75.glm <- update(temp67.glm, .~. + age:hasemail)
temp76.glm <- update(temp67.glm, .~. + age:intl)
temp77.glm <- update(temp67.glm, .~. + age:sex)
temp78.glm <- update(temp67.glm, .~. + age:r1)
temp79.glm <- update(temp67.glm, .~. + hasemail:intl)
temp80.glm <- update(temp67.glm, .~. + hasemail:memtype)
temp81.glm <- update(temp67.glm, .~. + hasemail:sex)
temp82.glm <- update(temp67.glm, .~. + hasemail:r1)
temp83.glm <- update(temp67.glm, .~. + intl:memtype)
temp84.glm <- update(temp67.glm, .~. + intl:sex)
temp85.glm <- update(temp67.glm, .~. + intl:r1)
temp86.glm <- update(temp67.glm, .~. + memtype:sex)
temp87.glm <- update(temp67.glm, .~. + sex:r1)

anova(temp69.glm, temp70.glm, temp71.glm, temp72.glm, temp73.glm, temp74.glm,
      temp75.glm, temp76.glm, temp77.glm, temp78.glm, temp79.glm, temp80.glm,
      temp81.glm, temp82.glm, temp83.glm, temp84.glm, temp85.glm, temp86.glm,
      temp87.glm, test="Chi")

# keep sex:r1
temp88.glm <- update(temp87.glm, .~. + allgames1yr:age)
temp89.glm <- update(temp87.glm, .~. + allgames1yr:hasemail)
temp90.glm <- update(temp87.glm, .~. + allgames1yr:intl)
temp91.glm <- update(temp87.glm, .~. + allgames1yr:memtype)
temp92.glm <- update(temp87.glm, .~. + allgames1yr:sex)
temp93.glm <- update(temp87.glm, .~. + allgames1yr:r1)
temp94.glm <- update(temp87.glm, .~. + age:hasemail)
temp95.glm <- update(temp87.glm, .~. + age:intl)
temp96.glm <- update(temp87.glm, .~. + age:sex)
temp97.glm <- update(temp87.glm, .~. + age:r1)
temp98.glm <- update(temp87.glm, .~. + hasemail:intl)
temp99.glm <- update(temp87.glm, .~. + hasemail:memtype)
temp100.glm <- update(temp87.glm, .~. + hasemail:sex)
temp101.glm <- update(temp87.glm, .~. + hasemail:r1)
temp102.glm <- update(temp87.glm, .~. + intl:memtype)
temp103.glm <- update(temp87.glm, .~. + intl:sex)
temp104.glm <- update(temp87.glm, .~. + intl:r1)
temp105.glm <- update(temp87.glm, .~. + memtype:sex)

anova(temp88.glm, temp89.glm, temp90.glm, temp91.glm, temp92.glm, temp93.glm,
      temp94.glm, temp95.glm, temp96.glm, temp97.glm, temp98.glm, temp99.glm,
      temp100.glm, temp101.glm, temp102.glm, temp103.glm, temp104.glm,
      temp105.glm, test="Chi")

# keep hasemail:memtype
temp106.glm <- update(temp99.glm, .~. + allgames1yr:age)
temp107.glm <- update(temp99.glm, .~. + allgames1yr:hasemail)
temp108.glm <- update(temp99.glm, .~. + allgames1yr:intl)
temp109.glm <- update(temp99.glm, .~. + allgames1yr:memtype)
temp110.glm <- update(temp99.glm, .~. + allgames1yr:sex)
temp111.glm <- update(temp99.glm, .~. + allgames1yr:r1)
temp112.glm <- update(temp99.glm, .~. + age:hasemail)
temp113.glm <- update(temp99.glm, .~. + age:intl)
```

```
temp114.glm <- update(temp99.glm, .~. + age:sex)
temp115.glm <- update(temp99.glm, .~. + age:r1)
temp116.glm <- update(temp99.glm, .~. + hasemail:intl)
temp117.glm <- update(temp99.glm, .~. + hasemail:sex)
temp118.glm <- update(temp99.glm, .~. + hasemail:r1)
temp119.glm <- update(temp99.glm, .~. + intl:memtype)
temp120.glm <- update(temp99.glm, .~. + intl:sex)
temp121.glm <- update(temp99.glm, .~. + intl:r1)
temp122.glm <- update(temp99.glm, .~. + memtype:sex)

anova(temp106.glm, temp107.glm, temp108.glm, temp109.glm, temp110.glm,
      temp111.glm, temp112.glm, temp113.glm, temp114.glm, temp115.glm,
      temp116.glm, temp117.glm, temp118.glm, temp119.glm, temp120.glm,
      temp121.glm, temp122.glm, test="Chi")

# keep age:sex
temp123.glm <- update(temp114.glm, .~. + allgames1yr:age)
temp124.glm <- update(temp114.glm, .~. + allgames1yr:hasemail)
temp125.glm <- update(temp114.glm, .~. + allgames1yr:intl)
temp126.glm <- update(temp114.glm, .~. + allgames1yr:memtype)
temp127.glm <- update(temp114.glm, .~. + allgames1yr:sex)
temp128.glm <- update(temp114.glm, .~. + allgames1yr:r1)
temp129.glm <- update(temp114.glm, .~. + age:hasemail)
temp130.glm <- update(temp114.glm, .~. + age:intl)
temp131.glm <- update(temp114.glm, .~. + age:r1)
temp132.glm <- update(temp114.glm, .~. + hasemail:intl)
temp133.glm <- update(temp114.glm, .~. + hasemail:sex)
temp134.glm <- update(temp114.glm, .~. + hasemail:r1)
temp135.glm <- update(temp114.glm, .~. + intl:memtype)
temp136.glm <- update(temp114.glm, .~. + intl:sex)
temp137.glm <- update(temp114.glm, .~. + intl:r1)
temp138.glm <- update(temp114.glm, .~. + memtype:sex)

anova(temp123.glm, temp124.glm, temp125.glm, temp126.glm, temp127.glm,
      temp128.glm, temp129.glm, temp130.glm, temp131.glm, temp132.glm,
      temp133.glm, temp134.glm, temp135.glm, temp136.glm, temp137.glm,
      temp138.glm, test="Chi")
```

There were no more pairwise interactions (among the 7 selected features) to be added, at a significance level of 0.05. We used this new model as a basis for our predictions.

Model 8: Reduced Model + (selected) 1st order interactions

```
model8.glm <- temp114.glm
summary(model8.glm)

model8.newpred = predict(model8.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model8.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
```

```r
# summary(outputdf)
write.csv(outputdf, file="model8.csv", row.names=FALSE)
```

Kaggle score for `model8`: 0.56524 (slight improvement)

We now decided to repeat a fit on the entire data set (`model6` - imputed values), but adding the five single-order interaction terms we added to the reduced set.

Model 9: Full Model + (selected) 1st order interactions

```r
model9.glm <- update(model6.glm, .~. + age:memtype + memtype:r1 +
    sex:r1 + hasemail:memtype + age:sex)
summary(model9.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age + sex + region + nregions + memtype +
##     memmonths + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 +
##     r.quick + extra + intl + r.intl + allgames1yr + allgames5yr +
##     fastevents + medevents + slowevents + nfloor + age.na + r1.na +
##     r2.na + r3.na + r.quick.na + r.intl.na + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex, family = binomial, data = data.na)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6369  -1.0420   0.5900   0.8764   4.0052
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         4.096e-01  9.204e-01   0.445 0.656301
## age                 1.138e-01  1.851e-02   6.147 7.87e-10 ***
## sexM               -6.164e-02  1.086e-01  -0.567 0.570442
## sexZ_dummy_NA       2.020e+01  1.832e+02   0.110 0.912228
## regionreg102        1.114e-01  1.299e-01   0.857 0.391361
## regionreg103        5.376e-02  1.410e-01   0.381 0.702993
## regionreg104        2.472e-01  2.074e-01   1.192 0.233338
## regionreg105        1.075e-01  1.742e-01   0.617 0.536969
## regionreg106       -8.249e-02  1.357e-01  -0.608 0.543155
## regionreg107        2.976e-01  5.414e-01   0.550 0.582552
## regionreg108        3.012e-01  2.240e-01   1.345 0.178776
## regionreg109        4.473e-01  1.851e-01   2.416 0.015672 *
## regionreg110        1.426e-01  2.334e-01   0.611 0.541160
## regionreg111        1.270e-01  1.443e-01   0.880 0.378788
## regionreg112        2.749e-01  1.291e-01   2.129 0.033247 *
## regionreg113        5.188e-01  1.829e-01   2.836 0.004569 **
## regionreg114        1.441e-01  1.355e-01   1.064 0.287551
## regionreg115        5.993e-01  1.120e+00   0.535 0.592450
## regionreg116        4.590e-01  2.525e-01   1.818 0.069132 .
## regionreg117        5.737e-02  1.552e-01   0.370 0.711678
## regionreg118        2.106e-02  1.529e-01   0.138 0.890454
## regionreg119       -6.542e-02  2.051e-01  -0.319 0.749723
```

```
## regionreg120       -5.026e-02  4.428e-01  -0.113 0.909638
## regionreg121        7.570e-01  1.255e+00   0.603 0.546367
## regionreg122        8.420e-01  4.454e-01   1.891 0.058658 .
## regionreg123        1.099e-02  1.588e-01   0.069 0.944837
## regionreg124        5.785e-02  9.314e-01   0.062 0.950473
## regionreg125        2.096e-01  2.401e-01   0.873 0.382696
## regionreg126        2.785e-01  1.429e-01   1.949 0.051250 .
## regionreg127        1.527e-01  1.290e-01   1.184 0.236295
## regionreg128       -5.671e-02  1.637e-01  -0.346 0.729020
## regionreg129        4.051e-01  1.513e-01   2.677 0.007431 **
## regionreg130        1.365e-01  2.254e-01   0.605 0.544940
## regionreg131       -3.051e-02  2.549e-01  -0.120 0.904728
## regionreg132        3.842e-01  1.341e-01   2.865 0.004171 **
## regionreg133        1.610e-01  1.395e-01   1.154 0.248558
## regionreg134        1.124e-01  3.138e-01   0.358 0.720178
## regionreg135       -4.280e-02  1.749e-01  -0.245 0.806647
## regionreg136        1.974e-02  2.523e-01   0.078 0.937622
## regionreg137        1.043e-01  1.336e-01   0.781 0.434862
## regionreg138        7.177e-02  1.495e-01   0.480 0.631048
## regionreg139        1.573e-01  2.145e-01   0.733 0.463494
## regionreg140        4.924e-01  1.472e-01   3.344 0.000827 ***
## regionreg141        2.232e-01  1.774e-01   1.259 0.208178
## regionreg142        2.049e-01  1.463e-01   1.401 0.161357
## regionreg143        2.610e-01  1.404e-01   1.859 0.063062 .
## regionreg144       -4.689e-01  1.885e-01  -2.488 0.012863 *
## regionreg145        1.143e-01  1.437e-01   0.796 0.426289
## regionreg146        1.560e-01  1.606e-01   0.971 0.331368
## regionreg147       -1.157e-01  1.478e-01  -0.783 0.433812
## regionreg148       -4.369e-01  2.270e-01  -1.924 0.054302 .
## regionreg149        3.264e-01  3.748e-01   0.871 0.383819
## regionreg150        1.879e-01  1.388e-01   1.353 0.175935
## regionreg151       -4.606e-01  3.761e-01  -1.225 0.220712
## regionreg152        7.315e-02  2.191e-01   0.334 0.738441
## regionreg153        3.618e-02  1.990e-01   0.182 0.855741
## regionreg154       -1.590e-02  1.362e-01  -0.117 0.907055
## regionreg155       -1.062e-01  1.718e-01  -0.618 0.536643
## nregions            9.867e-02  4.969e-02   1.986 0.047051 *
## memtypeF            1.622e+00  3.439e-01   4.717 2.40e-06 ***
## memtypeN            2.527e+00  3.184e-01   7.936 2.09e-15 ***
## memmonths          -7.961e-04  1.115e-04  -7.139 9.43e-13 ***
## mem_mag1Y          -1.495e-01  3.335e-02  -4.484 7.34e-06 ***
## mem_mag2Y          -3.521e-01  3.880e-02  -9.075  < 2e-16 ***
## hasemailY          -8.797e-01  1.152e-01  -7.635 2.27e-14 ***
## r1                  1.086e-04  1.301e-04   0.835 0.403857
## r2                  7.116e-05  4.867e-05   1.462 0.143701
## r3                 -2.432e-04  4.508e-05  -5.395 6.86e-08 ***
## r.quick             1.069e-04  2.825e-05   3.783 0.000155 ***
## extraY             -4.594e-01  5.882e-02  -7.810 5.71e-15 ***
## intlY              -6.654e-01  4.976e-02 -13.372  < 2e-16 ***
## r.intl             -3.055e-05  2.393e-04  -0.128 0.898410
## allgames1yr        -3.834e-02  1.078e-03 -35.575  < 2e-16 ***
## allgames5yr        -3.364e-03  7.044e-04  -4.776 1.79e-06 ***
## fastevents         -8.834e-03  2.391e-03  -3.695 0.000220 ***
## medevents           1.796e-02  2.711e-03   6.624 3.49e-11 ***
```

```
## slowevents          1.077e-02  2.577e-03   4.178 2.94e-05 ***
## nfloor             -2.324e-02  9.613e-03  -2.418 0.015614 *
## age.na              1.311e+00  3.291e-01   3.984 6.78e-05 ***
## r1.na              -1.541e-01  6.220e-02  -2.478 0.013225 *
## r2.na              -1.843e-01  6.052e-02  -3.045 0.002326 **
## r3.na               3.367e-01  5.545e-02   6.072 1.26e-09 ***
## r.quick.na          1.566e-01  4.561e-02   3.433 0.000597 ***
## r.intl.na          -3.684e-01  8.983e-02  -4.101 4.11e-05 ***
## age:memtypeF       -1.245e-01  1.857e-02  -6.703 2.04e-11 ***
## age:memtypeN       -1.364e-01  1.839e-02  -7.414 1.23e-13 ***
## memtypeF:r1        -2.537e-04  1.320e-04  -1.922 0.054630 .
## memtypeN:r1        -1.786e-04  1.159e-04  -1.541 0.123365
## sexM:r1            -3.780e-05  5.842e-05  -0.647 0.517533
## sexZ_dummy_NA:r1   -1.003e-02  9.434e-02  -0.106 0.915366
## memtypeF:hasemailY  3.429e-01  1.468e-01   2.336 0.019501 *
## memtypeN:hasemailY  1.394e-02  1.188e-01   0.117 0.906620
## age:sexM            4.111e-03  2.592e-03   1.586 0.112708
## age:sexZ_dummy_NA   5.789e-02  4.143e-02   1.397 0.162307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 48811  on 43342  degrees of freedom
## AIC: 48999
##
## Number of Fisher Scoring iterations: 12
```

```
model9.newpred = predict(model9.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model9.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model9.csv", row.names=FALSE)
```

Kaggle score for `model9`: 0.56513 (an improvement)

We now tried some second order interactions, just involving the five predictors represented in the single order interactions:

```
# second order interactions
temp139.glm <- update(model9.glm, .~. + age:memtype:r1)
temp140.glm <- update(model9.glm, .~. + age:memtype:hasemail)
temp141.glm <- update(model9.glm, .~. + age:memtype:sex)
temp142.glm <- update(model9.glm, .~. + r1:memtype:hasemail)
temp143.glm <- update(model9.glm, .~. + r1:sex:hasemail)
temp144.glm <- update(model9.glm, .~. + r1:age:hasemail)
temp145.glm <- update(model9.glm, .~. + sex:age:r1)
temp146.glm <- update(model9.glm, .~. + sex:memtype:r1)
temp147.glm <- update(model9.glm, .~. + hasemail:memtype:sex)
temp148.glm <- update(model9.glm, .~. + age:sex:hasemail)
```

```
anova(temp139.glm, temp140.glm, temp141.glm, temp142.glm, temp143.glm,
      temp144.glm, temp145.glm, temp146.glm, temp147.glm, temp148.glm,
      test="Chi")

# keep memtype:hasemail:r1
temp149.glm <- update(temp142.glm, .~. + age:memtype:r1)
temp150.glm <- update(temp142.glm, .~. + age:memtype:hasemail)
temp151.glm <- update(temp142.glm, .~. + age:memtype:sex)
temp152.glm <- update(temp142.glm, .~. + r1:sex:hasemail)
temp153.glm <- update(temp142.glm, .~. + r1:age:hasemail)
temp154.glm <- update(temp142.glm, .~. + sex:age:r1)
temp155.glm <- update(temp142.glm, .~. + sex:memtype:r1)
temp156.glm <- update(temp142.glm, .~. + hasemail:memtype:sex)
temp157.glm <- update(temp142.glm, .~. + age:sex:hasemail)

anova(temp149.glm, temp150.glm, temp151.glm, temp152.glm, temp153.glm,
      temp154.glm, temp155.glm, temp156.glm, temp157.glm, test="Chi")

# keep r1:sex:hasemail
temp158.glm <- update(temp152.glm, .~. + age:memtype:r1)
temp159.glm <- update(temp152.glm, .~. + age:memtype:hasemail)
temp160.glm <- update(temp152.glm, .~. + age:memtype:sex)
temp161.glm <- update(temp152.glm, .~. + r1:age:hasemail)
temp162.glm <- update(temp152.glm, .~. + sex:age:r1)
temp163.glm <- update(temp152.glm, .~. + sex:memtype:r1)
temp164.glm <- update(temp152.glm, .~. + hasemail:memtype:sex)
temp165.glm <- update(temp152.glm, .~. + age:sex:hasemail)

anova(temp158.glm, temp159.glm, temp160.glm, temp161.glm, temp162.glm,
      temp163.glm, temp164.glm, temp165.glm, test="Chi")

# keep age:sex:memtype
temp166.glm <- update(temp160.glm, .~. + age:memtype:r1)
temp167.glm <- update(temp160.glm, .~. + age:memtype:hasemail)
temp168.glm <- update(temp160.glm, .~. + r1:age:hasemail)
temp169.glm <- update(temp160.glm, .~. + sex:age:r1)
temp170.glm <- update(temp160.glm, .~. + sex:memtype:r1)
temp171.glm <- update(temp160.glm, .~. + hasemail:memtype:sex)
temp172.glm <- update(temp160.glm, .~. + age:sex:hasemail)

anova(temp166.glm, temp167.glm, temp168.glm, temp169.glm, temp170.glm,
      temp171.glm, temp172.glm, test="Chi")
```

No more second order interaction terms were considered. We double-check our model with second order interactions with the previous model:

```
anova(model8.glm, temp160.glm, test="Chi")
```

We can prefer the model with the selected second order interaction terms.

Model 10: Reduced model + selected first and second order interactions

```
model10.glm <- temp160.glm
summary(model10.glm)

model10.newpred = predict(model10.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model10.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model10.csv", row.names=FALSE)
```

Kaggle score for `model10`: 0.56510

Model 11: Full Model + (selected) 1st and 2nd order interactions

```
model11.glm <- update(model9.glm, .~. + memtype:hasemail:r1 +
    sex:hasemail:r1 + age:sex:memtype)
summary(model11.glm)

anova(model9.glm, model11.glm, test="Chi")

model11.newpred = predict(model11.glm, newdata=test.na, type="response", se.fit=T)
lapsed = model11.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model11.csv", row.names=FALSE)
```

Kaggle score for `model11`: 0.56510 (very slight improvement)

However, at this point, the incremental improvements in score over the previous models are small. However, the residual deviance/degrees of freedom ratio is still not much closer to 1.

We could further explore the fit of these models and create additional ones, but the fact that the level of improvement seems to be plateuing may, among other things, indicate we are missing some variables needed to better explain the data.

For the following sections, we will use the simpler `model9` as our exemplary model (i.e. complete model with first order interactions).

Goodness of Fit Tests

Let's start with the Hosmer-Lemeshow test. We are hoping that the null hypothesis is NOT rejected:

```
# Courtesy of lecture notes
# Hosmer-Lemeshow function
hosmerlem = function (y, yhat, g = 10) {
    cutyhat = cut(yhat, breaks = quantile(yhat, probs = seq(0,
```

```
        1, 1/g)), include.lowest = T)
    obs = xtabs(cbind(1 - y, y) ~ cutyhat)
    expect = xtabs(cbind(1 - yhat, yhat) ~ cutyhat)
    chisq = sum((obs - expect)^2/expect)
    P = 1 - pchisq(chisq, g - 2)
    c("X^2" = chisq, Df = g - 2, "P(>Chi)" = P)
    }

# Hosmer-Lemeshow test
# Partition fitted values into 20 groups
print(hosmerlem(model9.glm$y, fitted(model9.glm), g=20))
```

```
##     X^2      Df P(>Chi)
## 158.006  18.000   0.000
```

```
# Partition fitted values into 10 groups
print(hosmerlem(model9.glm$y, fitted(model9.glm), g=10))
```

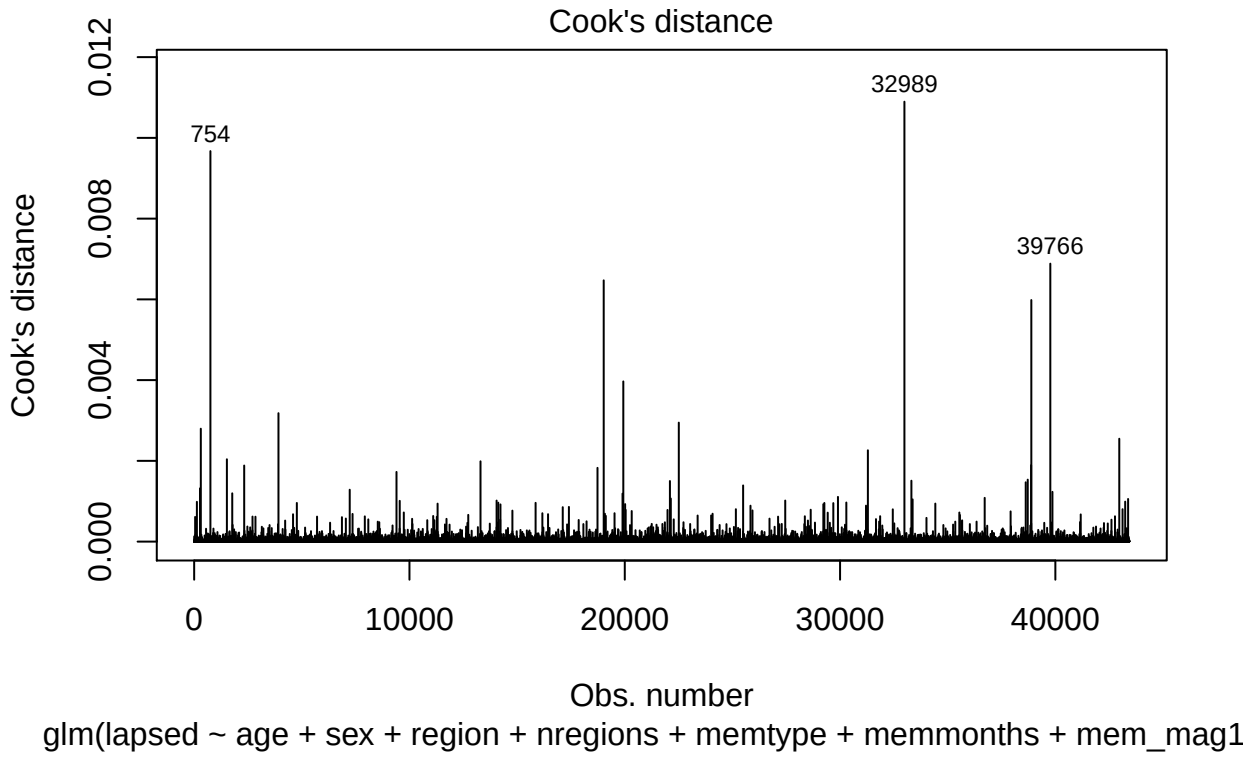
```
##          X^2           Df      P(>Chi)
## 8.310004e+01 8.000000e+00 1.154632e-14
```

```
# Partition fitted values into 6 groups
print(hosmerlem(model9.glm$y, fitted(model9.glm), g=6))
```

```
##          X^2           Df      P(>Chi)
## 3.892029e+01 4.000000e+00 7.235600e-08
```

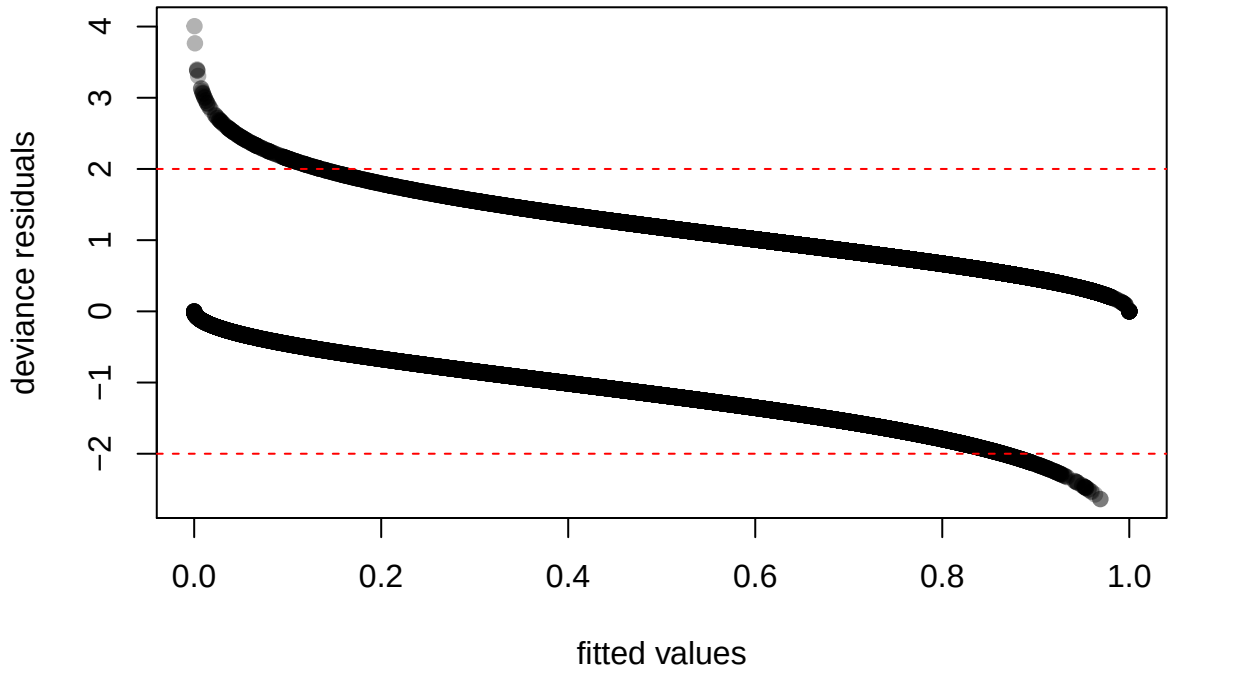
Yikes! This coarse test has low power, but it is saying we have a fit problem... which is definitely problematic!

```
plot(model9.glm, which = 4)
```



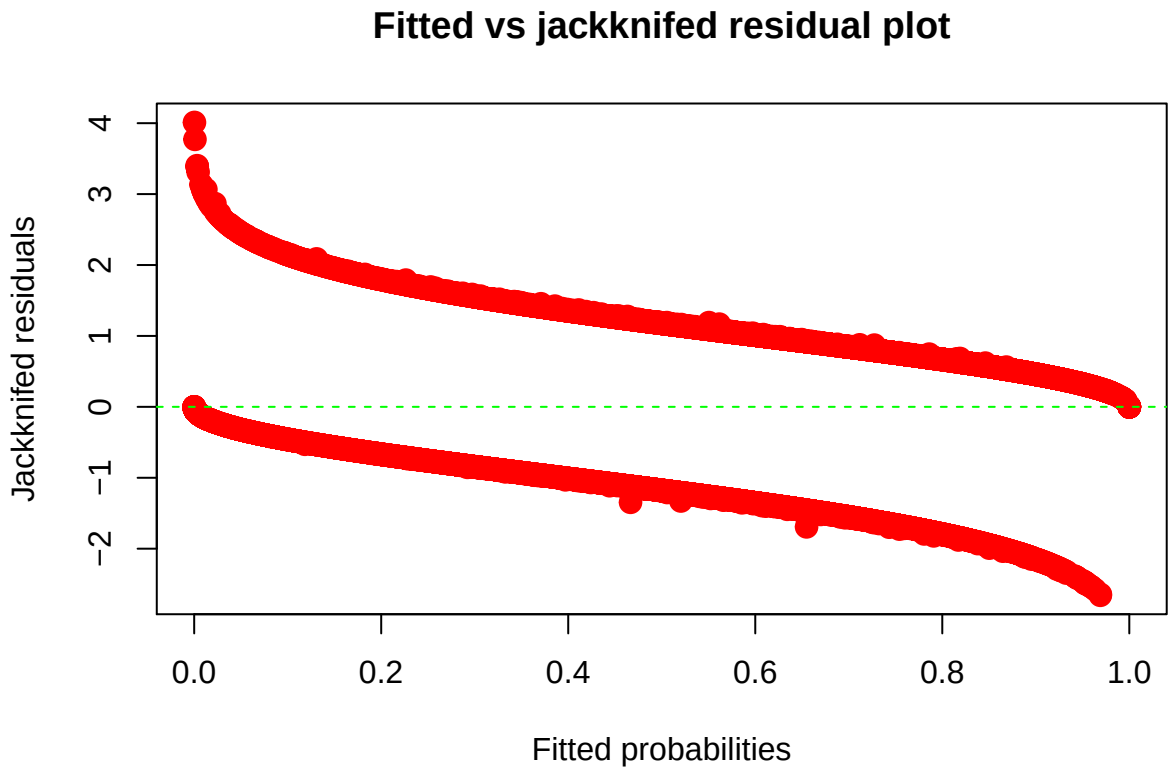
The plot of Cook's distance seems OK; none of the values have a value greater than 1.

```
# deviance residuals
devres <- resid(model9.glm, type = "deviance")
plot(x = model9.glm$fitted.values, y = devres, xlab = "fitted values",
     ylab = "deviance residuals", pch = 19, col = rgb(0,0,0, 0.3))
abline(h = c(2, -2), lty = 2, col = "red")
```



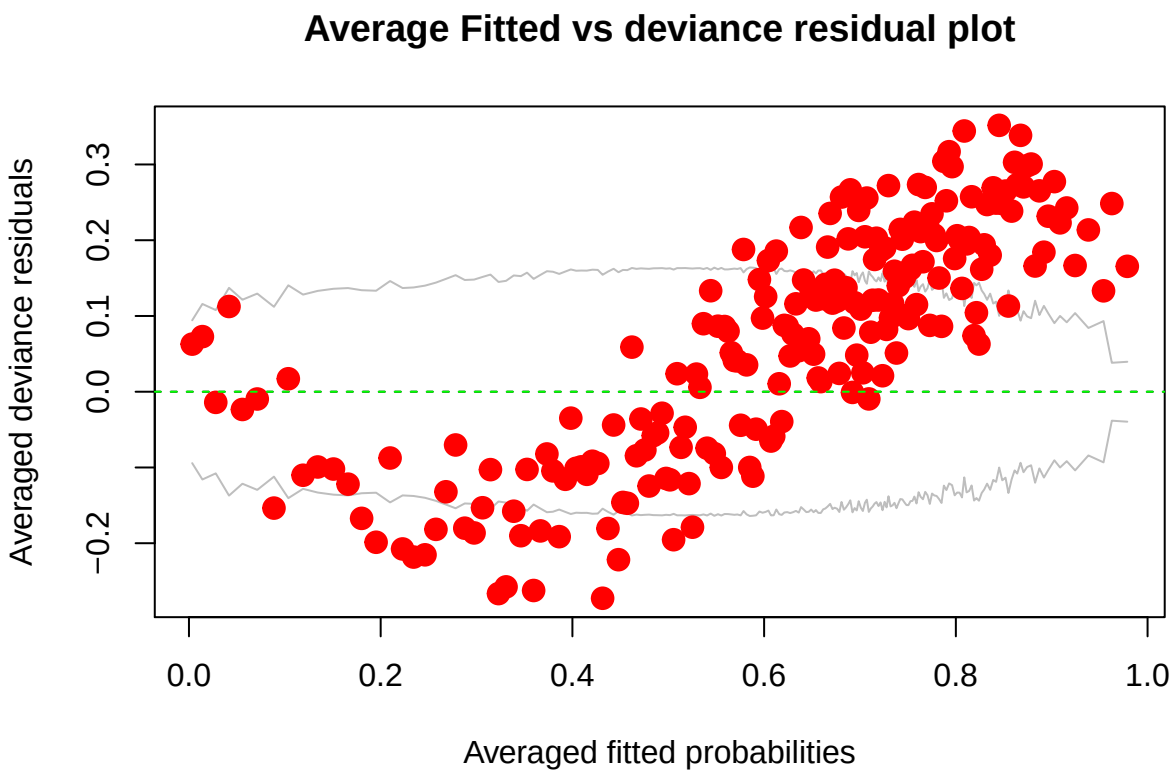
The lines of observations on the plot arise because the responses are discrete. Plots tend to be less striped when counts (fitted values) are larger. Note there are quite a few observations outside of the (+/- 2) bands, although this might be somewhat expected given that we have a lot of observations.

```
plot(fitted(model9.glm), rstudent(model9.glm),
  xlab="Fitted probabilities",
  ylab="Jackknifed residuals",
  pch=19, col="red", cex=1.5,
  main="Fitted vs jackknifed residual plot")
abline(h=0,lty=2,col="green")
```

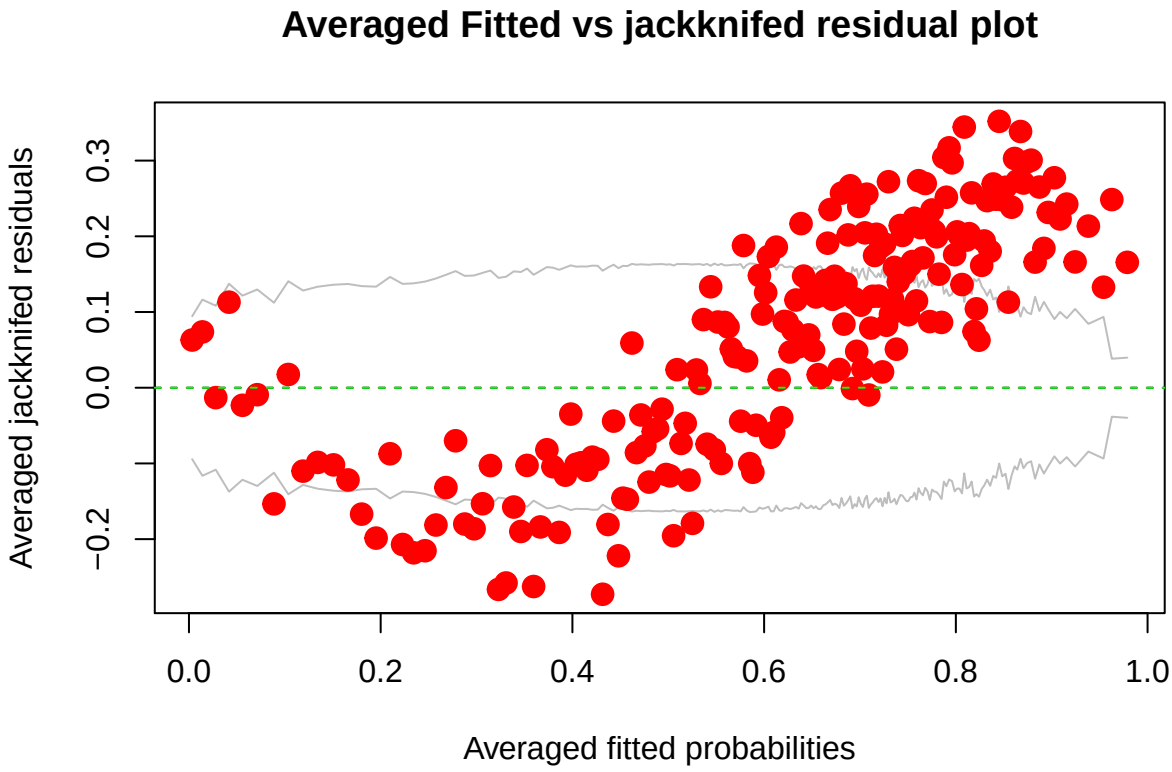


We also tried some binned plots.

```
library(arm)
binnedplot(fitted(model9.glm), residuals(model9.glm, type="deviance"),
  xlab="Averaged fitted probabilities",
  ylab="Averaged deviance residuals",
  pch=19, col.pts="red", cex.pts=1.5,
  main="Average Fitted vs deviance residual plot")
abline(h=0,lty=2,col="green")
```




```
binnedplot(fitted(model9.glm), rstudent(model9.glm),
  xlab="Averaged fitted probabilities",
  ylab="Averaged jackknifed residuals",
  pch=19, col.pts="red", cex.pts=1.5,
  main="Averaged Fitted vs jackknifed residual plot")
abline(h=0,lty=2,col="green")
```



These binned plots were quite interesting! It appears that the residuals have some pattern to them. Perhaps we are missing some higher-order term or we are missing important predictors for the model. In particular, it appears we could look more closely to see if there is some way to better predict the higher probability (lapsed) members. We will explore this further in the "Smoothing" and "Feature Engineering" sections below.

Interpretation of Coefficients / Confidence Intervals

```
summary(model9.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age + sex + region + nregions + memtype +
##     memmonths + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 +
##     r.quick + extra + intl + r.intl + allgames1yr + allgames5yr +
##     fastevents + medevents + slowevents + nfloor + age.na + r1.na +
##     r2.na + r3.na + r.quick.na + r.intl.na + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex, family = binomial, data = data.na)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6369  -1.0420   0.5900   0.8764   4.0052
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          4.096e-01  9.204e-01   0.445 0.656301
## age                  1.138e-01  1.851e-02   6.147 7.87e-10 ***
## sexM                -6.164e-02  1.086e-01  -0.567 0.570442
```

```
## sexZ_dummy_NA       2.020e+01  1.832e+02   0.110 0.912228
## regionreg102        1.114e-01  1.299e-01   0.857 0.391361
## regionreg103        5.376e-02  1.410e-01   0.381 0.702993
## regionreg104        2.472e-01  2.074e-01   1.192 0.233338
## regionreg105        1.075e-01  1.742e-01   0.617 0.536969
## regionreg106       -8.249e-02  1.357e-01  -0.608 0.543155
## regionreg107        2.976e-01  5.414e-01   0.550 0.582552
## regionreg108        3.012e-01  2.240e-01   1.345 0.178776
## regionreg109        4.473e-01  1.851e-01   2.416 0.015672 *
## regionreg110        1.426e-01  2.334e-01   0.611 0.541160
## regionreg111        1.270e-01  1.443e-01   0.880 0.378788
## regionreg112        2.749e-01  1.291e-01   2.129 0.033247 *
## regionreg113        5.188e-01  1.829e-01   2.836 0.004569 **
## regionreg114        1.441e-01  1.355e-01   1.064 0.287551
## regionreg115        5.993e-01  1.120e+00   0.535 0.592450
## regionreg116        4.590e-01  2.525e-01   1.818 0.069132 .
## regionreg117        5.737e-02  1.552e-01   0.370 0.711678
## regionreg118        2.106e-02  1.529e-01   0.138 0.890454
## regionreg119       -6.542e-02  2.051e-01  -0.319 0.749723
## regionreg120       -5.026e-02  4.428e-01  -0.113 0.909638
## regionreg121        7.570e-01  1.255e+00   0.603 0.546367
## regionreg122        8.420e-01  4.454e-01   1.891 0.058658 .
## regionreg123        1.099e-02  1.588e-01   0.069 0.944837
## regionreg124        5.785e-02  9.314e-01   0.062 0.950473
## regionreg125        2.096e-01  2.401e-01   0.873 0.382696
## regionreg126        2.785e-01  1.429e-01   1.949 0.051250 .
## regionreg127        1.527e-01  1.290e-01   1.184 0.236295
## regionreg128       -5.671e-02  1.637e-01  -0.346 0.729020
## regionreg129        4.051e-01  1.513e-01   2.677 0.007431 **
## regionreg130        1.365e-01  2.254e-01   0.605 0.544940
## regionreg131       -3.051e-02  2.549e-01  -0.120 0.904728
## regionreg132        3.842e-01  1.341e-01   2.865 0.004171 **
## regionreg133        1.610e-01  1.395e-01   1.154 0.248558
## regionreg134        1.124e-01  3.138e-01   0.358 0.720178
## regionreg135       -4.280e-02  1.749e-01  -0.245 0.806647
## regionreg136        1.974e-02  2.523e-01   0.078 0.937622
## regionreg137        1.043e-01  1.336e-01   0.781 0.434862
## regionreg138        7.177e-02  1.495e-01   0.480 0.631048
## regionreg139        1.573e-01  2.145e-01   0.733 0.463494
## regionreg140        4.924e-01  1.472e-01   3.344 0.000827 ***
## regionreg141        2.232e-01  1.774e-01   1.259 0.208178
## regionreg142        2.049e-01  1.463e-01   1.401 0.161357
## regionreg143        2.610e-01  1.404e-01   1.859 0.063062 .
## regionreg144       -4.689e-01  1.885e-01  -2.488 0.012863 *
## regionreg145        1.143e-01  1.437e-01   0.796 0.426289
## regionreg146        1.560e-01  1.606e-01   0.971 0.331368
## regionreg147       -1.157e-01  1.478e-01  -0.783 0.433812
## regionreg148       -4.369e-01  2.270e-01  -1.924 0.054302 .
## regionreg149        3.264e-01  3.748e-01   0.871 0.383819
## regionreg150        1.879e-01  1.388e-01   1.353 0.175935
## regionreg151       -4.606e-01  3.761e-01  -1.225 0.220712
## regionreg152        7.315e-02  2.191e-01   0.334 0.738441
## regionreg153        3.618e-02  1.990e-01   0.182 0.855741
## regionreg154       -1.590e-02  1.362e-01  -0.117 0.907055
```

```
## regionreg155       -1.062e-01  1.718e-01  -0.618 0.536643
## nregions            9.867e-02  4.969e-02   1.986 0.047051 *
## memtypeF            1.622e+00  3.439e-01   4.717 2.40e-06 ***
## memtypeN            2.527e+00  3.184e-01   7.936 2.09e-15 ***
## memmonths          -7.961e-04  1.115e-04  -7.139 9.43e-13 ***
## mem_mag1Y          -1.495e-01  3.335e-02  -4.484 7.34e-06 ***
## mem_mag2Y          -3.521e-01  3.880e-02  -9.075  < 2e-16 ***
## hasemailY          -8.797e-01  1.152e-01  -7.635 2.27e-14 ***
## r1                  1.086e-04  1.301e-04   0.835 0.403857
## r2                  7.116e-05  4.867e-05   1.462 0.143701
## r3                 -2.432e-04  4.508e-05  -5.395 6.86e-08 ***
## r.quick             1.069e-04  2.825e-05   3.783 0.000155 ***
## extraY             -4.594e-01  5.882e-02  -7.810 5.71e-15 ***
## intlY              -6.654e-01  4.976e-02 -13.372  < 2e-16 ***
## r.intl             -3.055e-05  2.393e-04  -0.128 0.898410
## allgames1yr        -3.834e-02  1.078e-03 -35.575  < 2e-16 ***
## allgames5yr        -3.364e-03  7.044e-04  -4.776 1.79e-06 ***
## fastevents         -8.834e-03  2.391e-03  -3.695 0.000220 ***
## medevents           1.796e-02  2.711e-03   6.624 3.49e-11 ***
## slowevents          1.077e-02  2.577e-03   4.178 2.94e-05 ***
## nfloor             -2.324e-02  9.613e-03  -2.418 0.015614 *
## age.na              1.311e+00  3.291e-01   3.984 6.78e-05 ***
## r1.na              -1.541e-01  6.220e-02  -2.478 0.013225 *
## r2.na              -1.843e-01  6.052e-02  -3.045 0.002326 **
## r3.na               3.367e-01  5.545e-02   6.072 1.26e-09 ***
## r.quick.na          1.566e-01  4.561e-02   3.433 0.000597 ***
## r.intl.na          -3.684e-01  8.983e-02  -4.101 4.11e-05 ***
## age:memtypeF       -1.245e-01  1.857e-02  -6.703 2.04e-11 ***
## age:memtypeN       -1.364e-01  1.839e-02  -7.414 1.23e-13 ***
## memtypeF:r1        -2.537e-04  1.320e-04  -1.922 0.054630 .
## memtypeN:r1        -1.786e-04  1.159e-04  -1.541 0.123365
## sexM:r1            -3.780e-05  5.842e-05  -0.647 0.517533
## sexZ_dummy_NA:r1   -1.003e-02  9.434e-02  -0.106 0.915366
## memtypeF:hasemailY  3.429e-01  1.468e-01   2.336 0.019501 *
## memtypeN:hasemailY  1.394e-02  1.188e-01   0.117 0.906620
## age:sexM            4.111e-03  2.592e-03   1.586 0.112708
## age:sexZ_dummy_NA   5.789e-02  4.143e-02   1.397 0.162307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 48811  on 43342  degrees of freedom
## AIC: 48999
##
## Number of Fisher Scoring iterations: 12
```

As an exercise, we interpret select coefficients of this model:

- quantitative predictor: for `age`, an increase of 1 year in age corresponds to a 0.114 increase in the probability of lapsing, on the logit scale, holding all other predictors fixed;

- categorical predictor: for `hasemail`, the odds of a person's membership lapsing for members having an e-mail address on file is $exp(0.88) = 2.41$ times smaller compared to members not having an e-mail address on file, holding all other predictors fixed;
- interaction: for `memtypeN:r1`, 0.179 is the additional effect of a 1 unit increase in the transformed regular chess rating from 2 years prior on the probability of lapsing (on the logit scale) for being a normal member compared to being an affiliate member.

The standard errors in the summary can be used to calculate Wald confidence intervals (e.g. for `age`, this may be given by 1.138e-01 +/- 1.96 * 1.851e-02 or [0.08, 0.15]), or we could have used the `confint` command in R for greater accuracy (which uses profile likelihoods).

```
# confint(model9.glm)  # takes a very long time
```

```
# save dataframes with imputed values
write.csv(data.na, file="train_no_na.csv", row.names=FALSE)
write.csv(test.na, file="test_no_na.csv", row.names=FALSE)
```

Generalized Additive Models

A generalized additive model is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables. The "flexibility to allow non-parametric fits with relaxed assumptions on the actual relationship between response and predictor, provides the potential for better fits to data than purely parametric models, but arguably with some loss of interpretability" (Wikipedia). In general, this model will allow for some flexibility in modeling the relationship between the response variable and the predictors in the model.

For comparison purposes, we left the first order interactions in, and built up the GAM:

```
library(gam)

model9a.gam = gam(lapsed ~ s(age) + sex + region + s(nregions) + memtype
                  + s(memmonths) + mem_mag1 + mem_mag2 + hasemail + s(r1)
                  + s(r2) + s(r3) + s(r.quick) + extra + intl + s(r.intl)
                  + s(allgames1yr) + s(allgames5yr) + s(fastevents)
                  + s(medevents) + s(slowevents) + s(nfloor) + age.na + r1.na
                  + r2.na + r3.na + r.quick.na + r.intl.na + age:memtype
                  + memtype:r1 + sex:r1 + memtype:hasemail + age:sex,
                  family = binomial, data = data.na)

summary(model9a.gam)
```

```
##
## Call: gam(formula = lapsed ~ s(age) + sex + region + s(nregions) +
##     memtype + s(memmonths) + mem_mag1 + mem_mag2 + hasemail +
##     s(r1) + s(r2) + s(r3) + s(r.quick) + extra + intl + s(r.intl) +
##     s(allgames1yr) + s(allgames5yr) + s(fastevents) + s(medevents) +
##     s(slowevents) + s(nfloor) + age.na + r1.na + r2.na + r3.na +
##     r.quick.na + r.intl.na + age:memtype + memtype:r1 + sex:r1 +
##     memtype:hasemail + age:sex, family = binomial, data = data.na)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6877 -0.9705  0.5432  0.8444  2.8421
```

```
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 58278.21 on 43435 degrees of freedom
## Residual Deviance: 47148.31 on 43300 degrees of freedom
## AIC: 47420.31
##
## Number of Local Scoring Iterations: 10
##
## Anova for Parametric Effects
##                     Df Sum Sq Mean Sq   F value    Pr(>F)
## s(age)               1    989  989.31 1013.8769 < 2.2e-16 ***
## sex                  2    256  128.06  131.2419 < 2.2e-16 ***
## region              54    175    3.24    3.3220 2.506e-15 ***
## s(nregions)          1     40   40.01   41.0056 1.533e-10 ***
## memtype              2    114   56.91   58.3270 < 2.2e-16 ***
## s(memmonths)         1    577  576.54  590.8536 < 2.2e-16 ***
## mem_mag1             1     76   76.17   78.0568 < 2.2e-16 ***
## mem_mag2             1     16   15.52   15.9090 6.657e-05 ***
## hasemail             1    930  930.17  953.2638 < 2.2e-16 ***
## s(r1)                1    360  359.99  368.9325 < 2.2e-16 ***
## s(r2)                1    157  156.62  160.5055 < 2.2e-16 ***
## s(r3)                1    640  640.08  655.9758 < 2.2e-16 ***
## s(r.quick)           1     28   27.82   28.5087 9.375e-08 ***
## extra                1     17   17.38   17.8072 2.450e-05 ***
## intl                 1    465  465.29  476.8430 < 2.2e-16 ***
## s(r.intl)            1     10   10.43   10.6904 0.0010778 **
## s(allgames1yr)       1   2263 2263.09 2319.2857 < 2.2e-16 ***
## s(allgames5yr)       1     32   31.69   32.4778 1.213e-08 ***
## s(fastevents)        1     21   21.08   21.6057 3.358e-06 ***
## s(medevents)         1     15   15.07   15.4428 8.517e-05 ***
## s(slowevents)        1      1    0.64    0.6591 0.4168863
## s(nfloor)            1      0    0.00    0.0020 0.9645876
## age.na               1      3    2.62    2.6846 0.1013275
## r1.na                1     22   21.88   22.4214 2.196e-06 ***
## r2.na                1      0    0.08    0.0821 0.7744967
## r3.na                1    109  109.25  111.9631 < 2.2e-16 ***
## r.quick.na           1      2    1.69    1.7341 0.1878990
## r.intl.na            1      4    3.83    3.9218 0.0476693 *
## memtype:age          2      8    4.22    4.3244 0.0132473 *
## memtype:r1           2      1    0.65    0.6710 0.5112006
## sex:r1               2      0    0.04    0.0392 0.9615131
## memtype:hasemail     2     15    7.70    7.8932 0.0003738 ***
## sex:age              2     17    8.54    8.7561 0.0001578 ***
## Residuals        43300  42251    0.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                   Npar Df Npar Chisq    P(Chi)
## (Intercept)
## s(age)                  3    1171.47 < 2.2e-16 ***
## sex
## region
```

```
## s(nregions)            3       1.86 0.6016175
## memtype
## s(memmonths)           3      58.52 1.217e-12 ***
## mem_mag1
## mem_mag2
## hasemail
## s(r1)                  3       1.81 0.6127179
## s(r2)                  3       1.52 0.6783426
## s(r3)                  3      11.03 0.0115876 *
## s(r.quick)             3      18.50 0.0003474 ***
## extra
## intl
## s(r.intl)              3       6.89 0.0755708 .
## s(allgames1yr)         3     351.87 < 2.2e-16 ***
## s(allgames5yr)         3       7.75 0.0514471 .
## s(fastevents)          3       9.06 0.0284450 *
## s(medevents)           3       8.12 0.0435855 *
## s(slowevents)          3       3.82 0.2812100
## s(nfloor)              3      11.05 0.0114845 *
## age.na
## r1.na
## r2.na
## r3.na
## r.quick.na
## r.intl.na
## memtype:age
## memtype:r1
## sex:r1
## memtype:hasemail
## sex:age
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model9.glm, model9a.gam, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: lapsed ~ age + sex + region + nregions + memtype + memmonths +
##     mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 + r.quick +
##     extra + intl + r.intl + allgames1yr + allgames5yr + fastevents +
##     medevents + slowevents + nfloor + age.na + r1.na + r2.na +
##     r3.na + r.quick.na + r.intl.na + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex
## Model 2: lapsed ~ s(age) + sex + region + s(nregions) + memtype + s(memmonths) +
##     mem_mag1 + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3) +
##     s(r.quick) + extra + intl + s(r.intl) + s(allgames1yr) +
##     s(allgames5yr) + s(fastevents) + s(medevents) + s(slowevents) +
##     s(nfloor) + age.na + r1.na + r2.na + r3.na + r.quick.na +
##     r.intl.na + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail +
##     age:sex
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     43342      48811
## 2     43300      47148 42   1662.2 < 2.2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

At first glance, it does appear that the more flexible GAM may be preferable over the GLM we've been using as the likelihood ratio test suggests it is a better fit. We noted the GAM and GLM are in, fact, nested models; this allows us to make the comparison using the likelihood ratio test.

```
model9a.newpred = predict(model9a.gam, newdata=test.na, type="response", se.fit=T)
lapsed = model9a.newpred

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model9a.csv", row.names=FALSE)
```

Kaggle score for `model9a`: 0.54542 (nice improvement)

Feature Engineering

Even a moderate amount of **feature engineering** might allow us to improve our predictions. Including interactions and higher-order terms is part of this; we can transform the data in such a way that makes it easier for important patterns to be uncovered. Often, domain knowledge can be helpful in this regard.

We decided to focus on two features we believe will be particularly relevant to whether the membership of a person will lapse: the number of months since joining US chess (`memmonths`) and the number of tournament games played in the past year (`allgames1yr`). It may be that there is some structure in this data that we can tease out and help R with its modeling.

```
summary(data.na$memmonths)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    19.0    56.0   122.2   177.0  1431.0
```

First, with respect to the number of months, the data is extremely right-skewed. Long time members are likely going to remain members, but it is probably the members who have just joined or who are relatively new, who may be more likely to quit. It may be worthwhile looking for some patterns here. For example, the difference in 12 months for someone who has already been a member for 5 years on the probability of lapsing is likely going to be completely different for someone who recently joined.

```
m = data.na$memmonths
n = m[m<60]
hist(n)
old = m[m>60]
hist(old)

a = data.na$lapsed[m<=60]
plot(a, m[m<=60])

N = 120
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
```

```
}
plot(lapsePercentage)

N = 500
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  print(i)
  print(p)
}

N = 120
lapsePercentage = rep(NA, N)
for (i in 60:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage[60:N])

N = 1000  # tried a few numbers
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage)

N = 300
lapsePercentage = rep(NA, N)
for (i in 200:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage[200:300])
```

There seemed to be a steady pattern up until `i` reaches 31 months, at which point there seems to be very sharp dip in lapse percentages. This increases back to a level a higher average probability level around 0.6 around the 36 month point. It is unknown whether this number has some special signifiance, Additionally, around the 84 month mark, there also seems to be another "breakpoint".

We can create some new indicator variables that split up this region, for use with our linear model. We decided to try: [<=30, 31, 32, 33, 34, 35, 36, 37-60, 61-84, 85-120, 121-263, >264]. We do recognize that the problem with being too specific with these "bands" is that we may well be overfitting. Using a smoother, as discussed above with respect to GAMs, may be better a more efficient way to handle this.

```
data2 <- data.na
data2$mon_less30 = as.factor(data2$memmonths <= 30)
data2$mon_31 = as.factor(data2$memmonths == 31)
data2$mon_32 = as.factor(data2$memmonths == 32)
data2$mon_33 = as.factor(data2$memmonths == 33)
data2$mon_34 = as.factor(data2$memmonths == 34)
data2$mon_35 = as.factor(data2$memmonths == 35)
data2$mon_36 = as.factor(data2$memmonths == 36)
data2$mon_37_60 = as.factor(data2$memmonths >= 37 & data2$memmonths <= 60)
```

```
data2$mon_61_84 = as.factor(data2$memmonths >= 61 & data2$memmonths <= 84)
data2$mon_85_120 = as.factor(data2$memmonths >= 85 & data2$memmonths <= 120)
data2$mon_121_263 = as.factor(data2$memmonths >= 121 & data2$memmonths <= 263)
data2$mon_264_plus = as.factor(data2$memmonths >= 264)

test2 <- test.na
test2$mon_less30 = as.factor(test2$memmonths <= 30)
test2$mon_31 = as.factor(test2$memmonths == 31)
test2$mon_32 = as.factor(test2$memmonths == 32)
test2$mon_33 = as.factor(test2$memmonths == 33)
test2$mon_34 = as.factor(test2$memmonths == 34)
test2$mon_35 = as.factor(test2$memmonths == 35)
test2$mon_36 = as.factor(test2$memmonths == 36)
test2$mon_37_60 = as.factor(test2$memmonths >= 37 & test2$memmonths <= 60)
test2$mon_61_84 = as.factor(test2$memmonths >= 61 & test2$memmonths <= 84)
test2$mon_85_120 = as.factor(test2$memmonths >= 85 & test2$memmonths <= 120)
test2$mon_121_263 = as.factor(test2$memmonths >= 121 & test2$memmonths <= 263)
test2$mon_264_plus = as.factor(test2$memmonths >= 264)
```

```
summary(data.na$allgames1yr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     6.0    14.3    18.0   625.0
```

The distribution of values for this predictor is also highly right skewed. It's quite unbelievable that someone has played 625 tournament games in one year!! Perhaps we should investigate whether this is in fact an outlier.

```
m = data.na$allgames1yr
n = m[m<20]
hist(n)

N = 100  # we played around with this number
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage)

N = 50
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(data.na$lapsed[m<i & m>=(i-1)] == "Y")
  print(i)
  print(p)
}

sum(data.na$lapsed[m<1 & m>=0] == "Y")

sum(data.na$lapsed[m<2 & m>=1] == "Y")
```

A couple of interesting things we noted: there seems to be a lot of values that are just below 5. There are a lot of people who have played 0 tournament games... AND have lapsed memberships! It seems necessary to treat these members separately. It may be that these people had no intention of being active (perhaps they joined through some promotion or the like).

The other interesting observation is that until 20 games are played, all of the fractions of lapsed players are greater than half! And as the number of games played increases, the percentage of lapsed players steadily decreases.

Since the number of months is a quantitative variable, we can probably model this decreasing pattern readily, but perhaps we should draw explicit attention to the small numbers of games played.

We can create some new indicator variables that split up this critical region. We decided to try: [0, 1-5, 6-10, 11-20, 21-34, 35-49, 50+]. We do recognize again, however, that the problem with being too specific with these "bands" is that we may well be overfitting.

```
data2$games_0 = as.factor(data2$allgames1yr < 1)
data2$games_1_5 = as.factor(data2$allgames1yr >= 1 & data2$allgames1yr <= 5)
data2$games_6_10 = as.factor(data2$allgames1yr >= 6 & data2$allgames1yr <= 10)
data2$games_11_20 = as.factor(data2$allgames1yr >= 11 & data2$allgames1yr <= 20)
data2$games_21_34 = as.factor(data2$allgames1yr >= 21 & data2$allgames1yr <= 34)
data2$games_35_49 = as.factor(data2$allgames1yr >= 35 & data2$allgames1yr <= 49)
data2$games_50_plus = as.factor(data2$memmonths >= 50)

test2$games_0 = as.factor(test2$allgames1yr < 1)
test2$games_1_5 = as.factor(test2$allgames1yr >= 1 & test2$allgames1yr <= 5)
test2$games_6_10 = as.factor(test2$allgames1yr >= 6 & test2$allgames1yr <= 10)
test2$games_11_20 = as.factor(test2$allgames1yr >= 11 & test2$allgames1yr <= 20)
test2$games_21_34 = as.factor(test2$allgames1yr >= 21 & test2$allgames1yr <= 34)
test2$games_35_49 = as.factor(test2$allgames1yr >= 35 & test2$allgames1yr <= 49)
test2$games_50_plus = as.factor(test2$memmonths >= 50)
```

Model 12: Full Model + two added variables

```
model12.glm <- glm(lapsed ~ ., family=binomial, data=data2)
summary(model12.glm)

anova(model6.glm, model12.glm, test="Chi")  # compared to model without

model12.newpred = predict(model12.glm, newdata=test2, type="response", se.fit=T)
lapsed = model12.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model12.csv", row.names=FALSE)
```

Kaggle score for `model12`: not run

Model 13: Full Model + added variables + 1st and 2nd order interactions

We added some of the interactions we added before:

```
model13.glm <- update(model12.glm, .~. + age:memtype + memtype:r1
                + sex:r1 + hasemail:memtype + age:sex + memtype:hasemail:r1
                + sex:hasemail:r1 + age:sex:memtype)
summary(model13.glm)

anova(model12.glm, model13.glm, test="Chi")

model13.newpred = predict(model13.glm, newdata=test2, type="response", se.fit=T)
lapsed = model13.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model13.csv", row.names=FALSE)
```

Kaggle score for `model13`: not run

Model 14: Add interactions with between added binary predictors and main effect

```
model14.glm <- glm(lapsed ~ age + sex + region + nregions + memtype
                   + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3
                   + r.quick + extra + intl + r.intl + allgames5yr
                   + fastevents + medevents + slowevents + nfloor + age.na
                   + r1.na + r2.na + r3.na + r.quick.na + r.intl.na
                   + memmonths * (mon_less30 + mon_31 + mon_32 + mon_33
                                  + mon_34 + mon_35 + mon_36 + mon_37_60
                                  + mon_61_84 + mon_85_120 + mon_121_263
                                  + mon_264_plus)
                   + allgames1yr * (games_0 + games_1_5 + games_6_10
                                    + games_11_20 + games_21_34 + games_35_49
                                    + games_50_plus) + age:memtype
                   + memtype:r1 + sex:r1 + memtype:hasemail + age:sex
                   + memtype:hasemail:r1 + sex:hasemail:r1
                   + age:sex:memtype, family = "binomial", data = data2)

summary(model14.glm)

anova(model13.glm, model14.glm, test="Chi")

model14.newpred = predict(model14.glm, newdata=test2, type="response", se.fit=T)
lapsed = model14.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model14.csv", row.names=FALSE)
```

Kaggle score for `model14`: 0.55723 (an improvement over GLM)

```
write.csv(data2, file="full_train.csv", row.names=FALSE)
write.csv(test2, file="full_test.csv", row.names=FALSE)
write.csv(Id, file="id.csv", row.names=FALSE)

train = read.csv("~/Desktop/stat149/StatKaggle/full_train.csv", header=TRUE)
test = read.csv("~/Desktop/stat149/StatKaggle/full_test.csv", header=TRUE)
Id = read.csv("~/Desktop/stat149/StatKaggle/id.csv", header=TRUE)
```

```
# run this if not reading from file above
train = data2
test = test2
```

Given the information from the importance plots obtained when constructing a Random Forest model (see other workbook), we see that `allgames5` as well as age are important factors. Let's see whether fleshing out `allgames5` and `age`, to create some additional features, makes a difference.

```
m = train$allgames5yr
n = m[m<20]
hist(n)

N = 20 # we played around with this number
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(train$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage)

N = 20
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(train$lapsed[m<i & m>=(i-1)] == "Y")
  print(i)
  print(p)
}
```

We generally observed that the patterns for `allgames5yr` look the same as with `allgames1yr`, so we chose to just reuse the previously created indicator variable for that variable as a proxy, to avoid duplication. With respect to age:

```
m = train$age
n = m[m<20]
hist(n)

N = 50 # we played around with this number
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(train$lapsed[m<i & m>=(i-1)] == "Y")
  lapsePercentage[i] = p
}
plot(lapsePercentage)

N = 50
```

```
lapsePercentage = rep(NA, N)
for (i in 1:N){
  p = mean(train$lapsed[m<i & m>=(i-1)] == "Y")
  print(i)
  print(p)
}
```

Memberships probably differ by age, and at the point of renewal it is highly likely someone will not renew (usually due to a fee increase). This largely follows the renewal schedule: https://secure2.uschess.org/webstore/member.php

We can create some new indicator variables that split up this region. We decided to try: [<=12, 13-15, 16-18, 19-24, 25-64, 65+]. We do recognize that the problem with being too specific with these "bands" is that we may well be overfitting.

```
train$age_12_under = as.factor(train$age <= 12)
train$age_13_15 = as.factor(train$age >= 13 & train$age <= 15)
train$age_16_18 = as.factor(train$age >= 16 & train$age <= 18)
train$age_19_24 = as.factor(train$age >= 19 & train$age <= 24)
train$age_25_64 = as.factor(train$age >= 25 & train$age <= 64)
train$age_65_plus = as.factor(train$age >= 65)

test$age_12_under = as.factor(test$age <= 12)
test$age_13_15 = as.factor(test$age >= 13 & test$age <= 15)
test$age_16_18 = as.factor(test$age >= 16 & test$age <= 18)
test$age_19_24 = as.factor(test$age >= 19 & test$age <= 24)
test$age_25_64 = as.factor(test$age >= 25 & test$age <= 64)
test$age_65_plus = as.factor(test$age >= 65)
```

(for intermediate model numbers, see other workbook)

Model 26: Add age categories plus interactions

```
model26.glm <- glm(lapsed ~ age * (age_12_under + age_13_15 + age_16_18
                                   + age_19_24 + age_25_64 + age_65_plus)
                   + sex + region + nregions + memtype + mem_mag1 + mem_mag2
                   + hasemail + r1 + r2 + r3 + r.quick + extra + intl
                   + r.intl + allgames5yr * (games_0 + games_1_5 + games_6_10
                                             + games_11_20 + games_21_34
                                             + games_35_49 + games_50_plus)
                   + fastevents + medevents + slowevents + nfloor + age.na
                   + r1.na + r2.na + r3.na + r.quick.na + r.intl.na
                   + memmonths * (mon_less30 + mon_31 + mon_32 + mon_33
                                  + mon_34 + mon_35 + mon_36 + mon_37_60
                                  + mon_61_84 + mon_85_120 + mon_121_263
                                  + mon_264_plus)
                   + allgames1yr * (games_0 + games_1_5 + games_6_10
                                    + games_11_20 + games_21_34 + games_35_49
                                    + games_50_plus) + age:memtype
                   + memtype:r1 + sex:r1 + memtype:hasemail + age:sex
                   + memtype:hasemail:r1 + sex:hasemail:r1
```

```
                  + age:sex:memtype, family = "binomial", data = train)

summary(model26.glm)
```

```
##
## Call:
## glm(formula = lapsed ~ age * (age_12_under + age_13_15 + age_16_18 +
##     age_19_24 + age_25_64 + age_65_plus) + sex + region + nregions +
##     memtype + mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 +
##     r.quick + extra + intl + r.intl + allgames5yr * (games_0 +
##     games_1_5 + games_6_10 + games_11_20 + games_21_34 + games_35_49 +
##     games_50_plus) + fastevents + medevents + slowevents + nfloor +
##     age.na + r1.na + r2.na + r3.na + r.quick.na + r.intl.na +
##     memmonths * (mon_less30 + mon_31 + mon_32 + mon_33 + mon_34 +
##         mon_35 + mon_36 + mon_37_60 + mon_61_84 + mon_85_120 +
##         mon_121_263 + mon_264_plus) + allgames1yr * (games_0 +
##     games_1_5 + games_6_10 + games_11_20 + games_21_34 + games_35_49 +
##     games_50_plus) + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail +
##     age:sex + memtype:hasemail:r1 + sex:hasemail:r1 + age:sex:memtype,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8754  -0.9437   0.4966   0.8416   2.8725
##
## Coefficients: (11 not defined because of singularities)
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -2.282e+00  1.103e+00  -2.069 0.038511 *
## age                           9.275e-03  2.455e-02   0.378 0.705579
## age_12_underTRUE             -9.369e-01  5.146e-01  -1.821 0.068677 .
## age_13_15TRUE                 8.923e-01  7.471e-01   1.194 0.232374
## age_16_18TRUE                -9.897e+00  1.078e+00  -9.184  < 2e-16 ***
## age_19_24TRUE                 3.637e+00  8.322e-01   4.370 1.24e-05 ***
## age_25_64TRUE                 3.034e+00  5.134e-01   5.910 3.43e-09 ***
## age_65_plusTRUE                      NA         NA      NA       NA
## sexM                         -2.804e-01  1.119e-01  -2.507 0.012176 *
## sexZ_dummy_NA                 2.004e+01  1.796e+02   0.112 0.911123
## regionreg102                  2.273e-01  1.350e-01   1.684 0.092154 .
## regionreg103                 -4.470e-03  1.465e-01  -0.031 0.975659
## regionreg104                  7.273e-02  2.170e-01   0.335 0.737470
## regionreg105                  2.115e-02  1.817e-01   0.116 0.907322
## regionreg106                 -1.514e-01  1.411e-01  -1.073 0.283229
## regionreg107                  4.808e-01  5.544e-01   0.867 0.385821
## regionreg108                  1.701e-01  2.320e-01   0.733 0.463585
## regionreg109                  3.465e-01  1.928e-01   1.797 0.072319 .
## regionreg110                  5.886e-02  2.421e-01   0.243 0.807864
## regionreg111                  1.038e-01  1.498e-01   0.693 0.488304
## regionreg112                  3.289e-01  1.342e-01   2.451 0.014259 *
## regionreg113                  4.212e-01  1.900e-01   2.217 0.026612 *
## regionreg114                  1.371e-01  1.408e-01   0.974 0.330058
## regionreg115                  6.621e-01  1.146e+00   0.578 0.563599
## regionreg116                  4.442e-01  2.619e-01   1.696 0.089899 .
## regionreg117                  6.780e-02  1.613e-01   0.420 0.674279
```

```
## regionreg118               -9.601e-02  1.594e-01  -0.602 0.547097
## regionreg119               -1.977e-01  2.126e-01  -0.930 0.352264
## regionreg120               -2.540e-01  4.672e-01  -0.544 0.586669
## regionreg121                4.444e-01  1.423e+00   0.312 0.754808
## regionreg122                7.347e-01  4.654e-01   1.579 0.114450
## regionreg123               -8.528e-02  1.657e-01  -0.515 0.606706
## regionreg124               -9.766e-02  1.005e+00  -0.097 0.922615
## regionreg125                2.428e-02  2.494e-01   0.097 0.922444
## regionreg126                3.654e-01  1.481e-01   2.466 0.013652 *
## regionreg127                2.278e-01  1.341e-01   1.699 0.089352 .
## regionreg128               -1.364e-01  1.700e-01  -0.802 0.422453
## regionreg129                5.475e-01  1.568e-01   3.491 0.000481 ***
## regionreg130                1.137e-01  2.337e-01   0.486 0.626698
## regionreg131                7.675e-02  2.614e-01   0.294 0.769092
## regionreg132                4.131e-01  1.393e-01   2.965 0.003025 **
## regionreg133                1.462e-01  1.450e-01   1.009 0.313113
## regionreg134               -1.229e-02  3.244e-01  -0.038 0.969786
## regionreg135               -3.174e-02  1.812e-01  -0.175 0.860999
## regionreg136               -3.492e-02  2.637e-01  -0.132 0.894635
## regionreg137                9.639e-02  1.389e-01   0.694 0.487643
## regionreg138               -2.714e-02  1.559e-01  -0.174 0.861780
## regionreg139                1.064e-01  2.241e-01   0.475 0.634784
## regionreg140                4.936e-01  1.528e-01   3.229 0.001241 **
## regionreg141                2.070e-01  1.844e-01   1.122 0.261747
## regionreg142                2.190e-01  1.520e-01   1.441 0.149471
## regionreg143                2.230e-01  1.460e-01   1.527 0.126759
## regionreg144               -7.335e-01  1.993e-01  -3.680 0.000233 ***
## regionreg145                8.631e-02  1.496e-01   0.577 0.563946
## regionreg146                4.813e-02  1.670e-01   0.288 0.773220
## regionreg147               -2.371e-01  1.541e-01  -1.539 0.123833
## regionreg148               -5.453e-01  2.388e-01  -2.284 0.022386 *
## regionreg149                3.508e-01  3.978e-01   0.882 0.377916
## regionreg150                2.210e-01  1.442e-01   1.532 0.125490
## regionreg151               -5.922e-01  3.859e-01  -1.535 0.124845
## regionreg152               -8.331e-02  2.274e-01  -0.366 0.714110
## regionreg153               -3.289e-02  2.078e-01  -0.158 0.874202
## regionreg154               -5.644e-02  1.416e-01  -0.399 0.690092
## regionreg155               -6.182e-02  1.781e-01  -0.347 0.728438
## nregions                    6.794e-02  5.132e-02   1.324 0.185555
## memtypeF                    2.862e-01  4.803e-01   0.596 0.551297
## memtypeN                    1.170e+00  4.065e-01   2.879 0.003993 **
## mem_mag1Y                  -2.750e-01  3.580e-02  -7.681 1.58e-14 ***
## mem_mag2Y                  -1.729e-01  4.080e-02  -4.237 2.27e-05 ***
## hasemailY                  -3.444e-01  4.105e-01  -0.839 0.401464
## r1                          3.626e-04  1.876e-04   1.933 0.053225 .
## r2                          7.038e-05  5.073e-05   1.387 0.165336
## r3                         -4.884e-04  4.834e-05 -10.104  < 2e-16 ***
## r.quick                     2.288e-05  3.001e-05   0.762 0.445812
## extraY                     -3.066e-01  6.116e-02  -5.014 5.34e-07 ***
## intlY                      -7.327e-01  5.166e-02 -14.183  < 2e-16 ***
## r.intl                      1.453e-04  2.405e-04   0.604 0.545674
## allgames5yr                -9.294e-04  3.980e-03  -0.234 0.815366
## games_0TRUE                 1.425e+00  1.609e-01   8.858  < 2e-16 ***
## games_1_5TRUE               1.742e+00  1.971e-01   8.841  < 2e-16 ***
```

```
## games_6_10TRUE                 1.578e+00  2.331e-01   6.769 1.30e-11 ***
## games_11_20TRUE                1.451e+00  2.142e-01   6.773 1.26e-11 ***
## games_21_34TRUE                8.554e-01  2.736e-01   3.127 0.001769 **
## games_35_49TRUE                1.049e+00  5.074e-01   2.068 0.038631 *
## games_50_plusTRUE             -2.827e-01  1.336e-01  -2.116 0.034335 *
## fastevents                    -9.389e-03  2.435e-03  -3.856 0.000115 ***
## medevents                      1.108e-02  2.665e-03   4.157 3.22e-05 ***
## slowevents                     5.461e-03  2.598e-03   2.102 0.035572 *
## nfloor                        -3.852e-03  8.587e-03  -0.449 0.653699
## age.na                         6.637e-01  3.516e-01   1.888 0.059034 .
## r1.na                         -4.246e-01  8.012e-02  -5.299 1.16e-07 ***
## r2.na                         -2.426e-01  8.590e-02  -2.824 0.004748 **
## r3.na                          5.958e-01  6.957e-02   8.564  < 2e-16 ***
## r.quick.na                     5.371e-02  4.946e-02   1.086 0.277550
## r.intl.na                     -3.342e-01  9.228e-02  -3.622 0.000293 ***
## memmonths                      4.375e-04  2.275e-04   1.923 0.054426 .
## mon_less30TRUE                 2.781e-01  1.754e-01   1.585 0.112864
## mon_31TRUE                    -4.092e-01  4.414e-01  -0.927 0.353858
## mon_32TRUE                     1.048e-01  3.899e-01   0.269 0.788010
## mon_33TRUE                     3.449e-01  3.520e-01   0.980 0.327113
## mon_34TRUE                     8.222e-01  2.841e-01   2.894 0.003805 **
## mon_35TRUE                     5.287e-01  3.035e-01   1.742 0.081530 .
## mon_36TRUE                     1.071e+00  2.437e-01   4.395 1.11e-05 ***
## mon_37_60TRUE                  4.556e-01  5.419e-01   0.841 0.400540
## mon_61_84TRUE                  1.619e+00  4.922e-01   3.288 0.001008 **
## mon_85_120TRUE                -7.696e-03  4.795e-01  -0.016 0.987194
## mon_121_263TRUE                1.017e+00  1.648e-01   6.171 6.80e-10 ***
## mon_264_plusTRUE                      NA         NA      NA       NA
## allgames1yr                   -1.334e-02  5.164e-03  -2.583 0.009784 **
## age:age_12_underTRUE           2.214e-01  1.265e-02  17.508  < 2e-16 ***
## age:age_13_15TRUE              6.137e-02  4.020e-02   1.527 0.126807
## age:age_16_18TRUE              7.186e-01  5.730e-02  12.541  < 2e-16 ***
## age:age_19_24TRUE             -6.839e-02  3.211e-02  -2.130 0.033161 *
## age:age_25_64TRUE             -4.391e-02  7.267e-03  -6.042 1.52e-09 ***
## age:age_65_plusTRUE                   NA         NA      NA       NA
## allgames5yr:games_0TRUE        3.395e-03  9.244e-04   3.672 0.000241 ***
## allgames5yr:games_1_5TRUE     -5.933e-04  9.838e-04  -0.603 0.546410
## allgames5yr:games_6_10TRUE     9.325e-04  9.098e-04   1.025 0.305370
## allgames5yr:games_11_20TRUE    2.188e-03  7.098e-04   3.082 0.002055 **
## allgames5yr:games_21_34TRUE    1.456e-03  6.850e-04   2.126 0.033531 *
## allgames5yr:games_35_49TRUE    5.124e-04  7.287e-04   0.703 0.481990
## allgames5yr:games_50_plusTRUE -2.597e-03  3.939e-03  -0.659 0.509712
## memmonths:mon_less30TRUE       3.270e-02  2.925e-03  11.178  < 2e-16 ***
## memmonths:mon_31TRUE                  NA         NA      NA       NA
## memmonths:mon_32TRUE                  NA         NA      NA       NA
## memmonths:mon_33TRUE                  NA         NA      NA       NA
## memmonths:mon_34TRUE                  NA         NA      NA       NA
## memmonths:mon_35TRUE                  NA         NA      NA       NA
## memmonths:mon_36TRUE                  NA         NA      NA       NA
## memmonths:mon_37_60TRUE        6.631e-03  9.842e-03   0.674 0.500487
## memmonths:mon_61_84TRUE       -1.221e-02  6.459e-03  -1.891 0.058629 .
## memmonths:mon_85_120TRUE       5.528e-03  4.521e-03   1.223 0.221454
## memmonths:mon_121_263TRUE     -3.301e-03  7.278e-04  -4.535 5.75e-06 ***
## memmonths:mon_264_plusTRUE            NA         NA      NA       NA
```

```
## games_0TRUE:allgames1yr               NA         NA      NA       NA
## games_1_5TRUE:allgames1yr     -4.509e-02  2.879e-02  -1.566 0.117343
## games_6_10TRUE:allgames1yr    -5.557e-02  2.103e-02  -2.642 0.008240 **
## games_11_20TRUE:allgames1yr   -5.144e-02  9.799e-03  -5.249 1.53e-07 ***
## games_21_34TRUE:allgames1yr   -2.070e-02  8.726e-03  -2.372 0.017671 *
## games_35_49TRUE:allgames1yr   -2.381e-02  1.207e-02  -1.973 0.048495 *
## games_50_plusTRUE:allgames1yr  9.736e-03  4.930e-03   1.975 0.048309 *
## age:memtypeF                  -2.124e-02  2.355e-02  -0.902 0.367083
## age:memtypeN                  -5.572e-03  2.335e-02  -0.239 0.811408
## memtypeF:r1                   -1.359e-04  2.267e-04  -0.600 0.548829
## memtypeN:r1                   -2.077e-04  1.784e-04  -1.164 0.244413
## sexM:r1                       -1.920e-05  6.976e-05  -0.275 0.783132
## sexZ_dummy_NA:r1              -9.776e-03  9.246e-02  -0.106 0.915795
## memtypeF:hasemailY             1.784e-01  5.106e-01   0.349 0.726861
## memtypeN:hasemailY            -4.841e-01  4.222e-01  -1.147 0.251537
## age:sexM                       1.161e-02  1.398e-02   0.831 0.406011
## age:sexZ_dummy_NA             -1.218e-01  6.101e-02  -1.997 0.045836 *
## memtypeA:hasemailY:r1         -2.709e-04  2.324e-04  -1.166 0.243700
## memtypeF:hasemailY:r1         -2.677e-04  1.680e-04  -1.594 0.110991
## memtypeN:hasemailY:r1         -3.788e-05  6.677e-05  -0.567 0.570485
## sexM:hasemailY:r1              2.836e-05  4.538e-05   0.625 0.532040
## sexZ_dummy_NA:hasemailY:r1     1.295e-04  1.775e-04   0.730 0.465635
## age:sexM:memtypeF              1.787e-02  1.393e-02   1.282 0.199689
## age:sexZ_dummy_NA:memtypeF     1.500e+00  2.970e+01   0.051 0.959721
## age:sexM:memtypeN             -5.397e-03  1.360e-02  -0.397 0.691572
## age:sexZ_dummy_NA:memtypeN     1.403e-01  5.758e-02   2.436 0.014848 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 58278  on 43435  degrees of freedom
## Residual deviance: 46573  on 43287  degrees of freedom
## AIC: 46871
##
## Number of Fisher Scoring iterations: 12
```

```
# anova(model14.glm, model26.glm, test="Chi")

model26.newpred = predict(model26.glm, newdata=test, type="response", se.fit=T)
lapsed = model26.newpred$fit

# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model26.csv", row.names=FALSE)
```

Kaggle score for `model26`: 0.53906 (an improvement + nice jump)

We experimented with some other models (logging response variables, polynomials, other interactions, etc.) (code not shown).

Kaggle score for `model26b`: 0.53882 Kaggle score for `model26c`: 0.53904 Kaggle score for `model26d`: 0.53915 Kaggle score for `model26e`: 0.53916

We also noticed only a slight improvement with the additional log transformation of the `allgames1yr` variable. We are becoming a bit wary of overfitting.

```
write.csv(train, file="train_add3.csv", row.names=FALSE)
write.csv(test, file="test_add3.csv", row.names=FALSE)
```

Finally, we incorporate the additional features into our GAM. We expect there to be some redundancy, as we have already used smoothers to deal with some of the non-linearities. However, we thought it would be interesting to see if the additional indicators with the particular splits we chose would capture something extra to help improve our predictions.

```
model26f.gam <- gam(lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18
                                       + age_19_24 + age_25_64 + age_65_plus)
                    + sex + region + s(nregions) + memtype + mem_mag1
                    + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3)
                    + s(r.quick) + extra + intl + s(r.intl)
                    + s(allgames5yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + s(fastevents) + s(medevents) + s(slowevents)
                    + s(nfloor) + age.na + r1.na + r2.na + r3.na
                    + r.quick.na + r.intl.na
                    + s(memmonths) * (mon_less30 + mon_31 + mon_32 + mon_33
                                      + mon_34 + mon_35 + mon_36 + mon_37_60
                                      + mon_61_84 + mon_85_120 + mon_121_263
                                      + mon_264_plus)
                    + s(allgames1yr) * (games_0 + games_1_5 + games_6_10
                                        + games_11_20 + games_21_34
                                        + games_35_49 + games_50_plus)
                    + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail
                    + age:sex + memtype:hasemail:r1 + sex:hasemail:r1
                    + age:sex:memtype, family = "binomial", data = train)

summary(model26f.gam)
```

```
##
## Call: gam(formula = lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18 +
##     age_19_24 + age_25_64 + age_65_plus) + sex + region + s(nregions) +
##     memtype + mem_mag1 + mem_mag2 + hasemail + s(r1) + s(r2) +
##     s(r3) + s(r.quick) + extra + intl + s(r.intl) + s(allgames5yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + s(fastevents) + s(medevents) +
##     s(slowevents) + s(nfloor) + age.na + r1.na + r2.na + r3.na +
##     r.quick.na + r.intl.na + s(memmonths) * (mon_less30 + mon_31 +
##     mon_32 + mon_33 + mon_34 + mon_35 + mon_36 + mon_37_60 +
##     mon_61_84 + mon_85_120 + mon_121_263 + mon_264_plus) + s(allgames1yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex + memtype:hasemail:r1 +
##     sex:hasemail:r1 + age:sex:memtype, family = "binomial", data = train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8502 -0.9419  0.4942  0.8433  2.8681
```

```
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 58278.21 on 43435 degrees of freedom
## Residual Deviance: 46464.29 on 43245 degrees of freedom
## AIC: 46846.29
##
## Number of Local Scoring Iterations: 10
##
## Anova for Parametric Effects
##                               Df Sum Sq Mean Sq  F value    Pr(>F)
## s(age)                         1    722  722.46 725.3471 < 2.2e-16 ***
## age_12_under                   1    115  114.70 115.1609 < 2.2e-16 ***
## age_13_15                      1      5    5.40   5.4206 0.0199056 *
## age_16_18                      1     22   21.53  21.6154 3.341e-06 ***
## age_19_24                      1     28   28.07  28.1846 1.108e-07 ***
## age_25_64                      1      0    0.12   0.1237 0.7250904
## sex                            2    189   94.39  94.7698 < 2.2e-16 ***
## region                        54    121    2.25   2.2563 3.862e-07 ***
## s(nregions)                    1      9    8.66   8.6935 0.0031951 **
## memtype                        2     89   44.26  44.4379 < 2.2e-16 ***
## mem_mag1                       1     45   44.51  44.6830 2.345e-11 ***
## mem_mag2                       1     18   17.91  17.9777 2.240e-05 ***
## hasemail                       1    784  783.58 786.7110 < 2.2e-16 ***
## s(r1)                          1    455  454.93 456.7472 < 2.2e-16 ***
## s(r2)                          1    169  168.50 169.1750 < 2.2e-16 ***
## s(r3)                          1    652  652.02 654.6219 < 2.2e-16 ***
## s(r.quick)                     1     31   31.04  31.1618 2.388e-08 ***
## extra                          1     27   26.97  27.0782 1.963e-07 ***
## intl                           1    487  486.93 488.8796 < 2.2e-16 ***
## s(r.intl)                      1      9    9.45   9.4887 0.0020687 **
## s(allgames5yr)                 1    656  655.70 658.3179 < 2.2e-16 ***
## games_0                        1    290  290.33 291.4930 < 2.2e-16 ***
## games_1_5                      1    623  622.75 625.2404 < 2.2e-16 ***
## games_6_10                     1    326  325.71 327.0144 < 2.2e-16 ***
## games_11_20                    1    272  271.95 273.0378 < 2.2e-16 ***
## games_21_34                    1    133  132.89 133.4197 < 2.2e-16 ***
## games_35_49                    1     25   25.40  25.4979 4.447e-07 ***
## games_50_plus                  1     23   23.05  23.1407 1.511e-06 ***
## s(fastevents)                  1     17   16.90  16.9632 3.818e-05 ***
## s(medevents)                   1     12   12.14  12.1869 0.0004817 ***
## s(slowevents)                  1      0    0.36   0.3570 0.5501700
## s(nfloor)                      1      0    0.40   0.4056 0.5241893
## age.na                         1      3    3.45   3.4677 0.0625834 .
## r1.na                          1      2    2.28   2.2873 0.1304445
## r2.na                          1      2    1.92   1.9271 0.1650843
## r3.na                          1    108  107.95 108.3814 < 2.2e-16 ***
## r.quick.na                     1      3    2.95   2.9655 0.0850637 .
## r.intl.na                      1      3    2.68   2.6945 0.1007011
## s(memmonths)                   1     41   41.19  41.3573 1.281e-10 ***
## mon_less30                     1      1    1.48   1.4886 0.2224335
## mon_31                         1      6    6.47   6.4958 0.0108161 *
## mon_32                         1      3    2.87   2.8830 0.0895278 .
## mon_33                         1      1    1.16   1.1669 0.2800537
```

```
## mon_34                           1      0    0.02   0.0189 0.8906167
## mon_35                           1      1    0.87   0.8721 0.3503785
## mon_36                           1      2    2.20   2.2073 0.1373700
## mon_37_60                        1      8    8.07   8.1022 0.0044233 **
## mon_61_84                        1      7    6.52   6.5423 0.0105370 *
## mon_85_120                       1      8    8.39   8.4235 0.0037058 **
## mon_121_263                      1     48   48.48  48.6688 3.074e-12 ***
## s(allgames1yr)                   1     26   25.58  25.6828 4.041e-07 ***
## s(age):age_12_under              1    662  661.54 664.1810 < 2.2e-16 ***
## s(age):age_13_15                 1      4    4.40   4.4210 0.0355045 *
## s(age):age_16_18                 1    168  167.55 168.2225 < 2.2e-16 ***
## s(age):age_19_24                 1      2    2.25   2.2580 0.1329311
## s(age):age_25_64                 1     34   34.49  34.6236 4.030e-09 ***
## s(allgames5yr):games_0           1     22   22.43  22.5164 2.090e-06 ***
## s(allgames5yr):games_1_5         1      0    0.24   0.2364 0.6268205
## s(allgames5yr):games_6_10        1      0    0.19   0.1871 0.6653179
## s(allgames5yr):games_11_20       1      4    4.05   4.0657 0.0437686 *
## s(allgames5yr):games_21_34       1      2    1.55   1.5595 0.2117474
## s(allgames5yr):games_35_49       1      0    0.16   0.1644 0.6851206
## s(allgames5yr):games_50_plus     1      2    2.45   2.4625 0.1165969
## s(memmonths):mon_less30          1    108  108.13 108.5659 < 2.2e-16 ***
## s(memmonths):mon_37_60           1      1    1.16   1.1665 0.2801362
## s(memmonths):mon_61_84           1      4    4.05   4.0617 0.0438730 *
## s(memmonths):mon_85_120          1      2    1.68   1.6903 0.1935656
## s(memmonths):mon_121_263         1     26   25.56  25.6590 4.091e-07 ***
## games_1_5:s(allgames1yr)         1      2    2.23   2.2379 0.1346738
## games_6_10:s(allgames1yr)        1      6    6.13   6.1503 0.0131424 *
## games_11_20:s(allgames1yr)       1     26   26.04  26.1425 3.185e-07 ***
## games_21_34:s(allgames1yr)       1      7    6.70   6.7303 0.0094823 **
## games_35_49:s(allgames1yr)       1      5    4.97   4.9931 0.0254541 *
## games_50_plus:s(allgames1yr)     1      3    3.34   3.3542 0.0670390 .
## memtype:age                      2      0    0.14   0.1419 0.8677095
## memtype:r1                       2      1    0.35   0.3561 0.7003836
## sex:r1                           2      0    0.05   0.0526 0.9487808
## memtype:hasemail                 2     11    5.68   5.7012 0.0033445 **
## sex:age                          2     23   11.46  11.5070 1.009e-05 ***
## memtype:hasemail:r1              3      3    0.92   0.9227 0.4288060
## sex:hasemail:r1                  2      1    0.51   0.5133 0.5985211
## sex:memtype:age                  4     40   10.12  10.1635 3.197e-08 ***
## Residuals                    43245  43073    1.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                               Npar Df Npar Chisq   P(Chi)
## (Intercept)
## s(age)                             3     0.9507 0.813202
## age_12_under
## age_13_15
## age_16_18
## age_19_24
## age_25_64
## age_65_plus
## sex
```

```
## region
## s(nregions)                      3     1.0490 0.789384
## memtype
## mem_mag1
## mem_mag2
## hasemail
## s(r1)                            3     1.8262 0.609265
## s(r2)                            3     2.0095 0.570472
## s(r3)                            3    12.6152 0.005549 **
## s(r.quick)                       3    24.3710 2.09e-05 ***
## extra
## intl
## s(r.intl)                        3     8.9196 0.030373 *
## s(allgames5yr)                   3     2.5897 0.459269
## games_0
## games_1_5
## games_6_10
## games_11_20
## games_21_34
## games_35_49
## games_50_plus
## s(fastevents)                    3     7.9842 0.046339 *
## s(medevents)                     3     8.3311 0.039641 *
## s(slowevents)                    3     8.6608 0.034157 *
## s(nfloor)                        3    10.7432 0.013199 *
## age.na
## r1.na
## r2.na
## r3.na
## r.quick.na
## r.intl.na
## s(memmonths)                     3     3.1128 0.374540
## mon_less30
## mon_31
## mon_32
## mon_33
## mon_34
## mon_35
## mon_36
## mon_37_60
## mon_61_84
## mon_85_120
## mon_121_263
## mon_264_plus
## s(allgames1yr)                   3     1.8344 0.607469
## s(age):age_12_under
## s(age):age_13_15
## s(age):age_16_18
## s(age):age_19_24
## s(age):age_25_64
## s(age):age_65_plus
## s(allgames5yr):games_0
## s(allgames5yr):games_1_5
## s(allgames5yr):games_6_10
```

```
## s(allgames5yr):games_11_20
## s(allgames5yr):games_21_34
## s(allgames5yr):games_35_49
## s(allgames5yr):games_50_plus
## s(memmonths):mon_less30
## s(memmonths):mon_31
## s(memmonths):mon_32
## s(memmonths):mon_33
## s(memmonths):mon_34
## s(memmonths):mon_35
## s(memmonths):mon_36
## s(memmonths):mon_37_60
## s(memmonths):mon_61_84
## s(memmonths):mon_85_120
## s(memmonths):mon_121_263
## s(memmonths):mon_264_plus
## games_0:s(allgames1yr)
## games_1_5:s(allgames1yr)
## games_6_10:s(allgames1yr)
## games_11_20:s(allgames1yr)
## games_21_34:s(allgames1yr)
## games_35_49:s(allgames1yr)
## games_50_plus:s(allgames1yr)
## memtype:age
## memtype:r1
## sex:r1
## memtype:hasemail
## sex:age
## memtype:hasemail:r1
## sex:hasemail:r1
## sex:memtype:age
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model26.glm, model26f.gam, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: lapsed ~ age * (age_12_under + age_13_15 + age_16_18 + age_19_24 +
##     age_25_64 + age_65_plus) + sex + region + nregions + memtype +
##     mem_mag1 + mem_mag2 + hasemail + r1 + r2 + r3 + r.quick +
##     extra + intl + r.intl + allgames5yr * (games_0 + games_1_5 +
##     games_6_10 + games_11_20 + games_21_34 + games_35_49 + games_50_plus) +
##     fastevents + medevents + slowevents + nfloor + age.na + r1.na +
##     r2.na + r3.na + r.quick.na + r.intl.na + memmonths * (mon_less30 +
##     mon_31 + mon_32 + mon_33 + mon_34 + mon_35 + mon_36 + mon_37_60 +
##     mon_61_84 + mon_85_120 + mon_121_263 + mon_264_plus) + allgames1yr *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex + memtype:hasemail:r1 +
##     sex:hasemail:r1 + age:sex:memtype
## Model 2: lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18 + age_19_24 +
##     age_25_64 + age_65_plus) + sex + region + s(nregions) + memtype +
##     mem_mag1 + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3) +
```

```
##     s(r.quick) + extra + intl + s(r.intl) + s(allgames5yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + s(fastevents) + s(medevents) +
##     s(slowevents) + s(nfloor) + age.na + r1.na + r2.na + r3.na +
##     r.quick.na + r.intl.na + s(memmonths) * (mon_less30 + mon_31 +
##     mon_32 + mon_33 + mon_34 + mon_35 + mon_36 + mon_37_60 +
##     mon_61_84 + mon_85_120 + mon_121_263 + mon_264_plus) + s(allgames1yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex + memtype:hasemail:r1 +
##     sex:hasemail:r1 + age:sex:memtype
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     43287      46573
## 2     43245      46464 42   108.43 8.801e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model9a.gam, model26f.gam, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: lapsed ~ s(age) + sex + region + s(nregions) + memtype + s(memmonths) +
##     mem_mag1 + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3) +
##     s(r.quick) + extra + intl + s(r.intl) + s(allgames1yr) +
##     s(allgames5yr) + s(fastevents) + s(medevents) + s(slowevents) +
##     s(nfloor) + age.na + r1.na + r2.na + r3.na + r.quick.na +
##     r.intl.na + age:memtype + memtype:r1 + sex:r1 + memtype:hasemail +
##     age:sex
## Model 2: lapsed ~ s(age) * (age_12_under + age_13_15 + age_16_18 + age_19_24 +
##     age_25_64 + age_65_plus) + sex + region + s(nregions) + memtype +
##     mem_mag1 + mem_mag2 + hasemail + s(r1) + s(r2) + s(r3) +
##     s(r.quick) + extra + intl + s(r.intl) + s(allgames5yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + s(fastevents) + s(medevents) +
##     s(slowevents) + s(nfloor) + age.na + r1.na + r2.na + r3.na +
##     r.quick.na + r.intl.na + s(memmonths) * (mon_less30 + mon_31 +
##     mon_32 + mon_33 + mon_34 + mon_35 + mon_36 + mon_37_60 +
##     mon_61_84 + mon_85_120 + mon_121_263 + mon_264_plus) + s(allgames1yr) *
##     (games_0 + games_1_5 + games_6_10 + games_11_20 + games_21_34 +
##         games_35_49 + games_50_plus) + age:memtype + memtype:r1 +
##     sex:r1 + memtype:hasemail + age:sex + memtype:hasemail:r1 +
##     sex:hasemail:r1 + age:sex:memtype
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     43300      47148
## 2     43245      46464 55   684.02 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Kaggle score for `model26f`: 0.53897

```
model26f.newpred = predict(model26f.gam, newdata=test, type="response", se.fit=T)
lapsed = model26f.newpred
```

```
# Write as submission file
outputdf = cbind(Id, lapsed)
# summary(outputdf)
write.csv(outputdf, file="model26f.csv", row.names=FALSE)
```

Finally, we saved the predictions on the test set for later use (e.g. we tried ensemble methods, see Workbook 4):

```
model26f.preds = predict(model26f.gam, type="response")
write.csv(model26f.preds, file="model26f_preds.csv", row.names=FALSE)
```

```
rm(list = ls())
```