Project Report

Introduction

In this project, I explored the relationship between consumers' inferences about twenty business schools' qualities and memory cues. More specifically, I looked at what factors affect consumers' decisions when they are asked to compare two business school, and what factors affect their confidence on the decisions they made from the results of a questionnaire given by my supervisor. Memory cues of a brand are information about this brand based on consumers' own memories.

I looked at 4 memory cues, 3 of which are simple memory cues - the familiarity of the school, the knowledge amount the consumers have of the school, recognition of the school. The other one, which is more complex, is the overlap of the consumer's belief distributions of two brands that are compared. The consumer's belief distribution of a business school, which is a triangle distribution, is constructed by the subjective ranks that consumer made. For each business school, the consumer was asked to guess the most probable rank, the highest rank possible and the lowest rank possible of the school. The overlap of the two distributions is quantified by the overlap area of the two distribution triangles plotted on a two-dimensional space, and is calculated using some geometry packages in R.

I also fitted several different statistical models to the dataset to explore which model is the best to describe the relationship, and which memory cue is the best predictor in the model. In the end, we achieved a model for confidence of almost 50% accuracy and a model for decision of approximately 80% accuracy.

Packages

```r
library(sp)
library(maptools)
library(rgeos)
library(ggplot2)
library(scales)
library(MASS)
library(tidyr)
library(boot)
library(Hmisc)
library(corrplot)
```

The above are the statistical packages that I used during the project. ***sp***, ***maptools*** and ***rgeos*** are used to calculate the area of the overlap.
ggplot2 and ***sclaes*** are used in plotting and visualising the data.
MASS is used to fit an ordinary logistic regression.
tidyr is used to regroup the data.
boot is used to do cross validation.
Hmisc and ***corrplot*** are used to to test the significance of the correlation and draw a correlogram.

Read in the data

When reading in the data, we need to make some changes to the variables. Some variables need to be transformed into factor variables and some need to be standardised. Also, we transformed the confidence into a variable of scale from 0 to 100. We divided dataset into three datasets, traning set, validation set, and the test set. Since there are 125 participants, we included data of the first 75 participants in the training set, data of the next 25 participants in the validation set and rest of them in the test set. That way, the size of our training set is about 60% of the original dataset and the sizes of validation set and test set are 20%.

```r
raw_dat = read.csv("sbj.pair.Inf.Cue_schools.csv")
dat = raw_dat
dat[,"overlap"] = c(1:nrow(dat))
dat[,"confidence"] = c(1:nrow(dat))
dat$obj1.R = factor(dat$obj1.R , levels = 0:1, labels = c("No", "Yes"))
dat$obj2.R = factor(dat$obj2.R , levels = 0:1, labels = c("No", "Yes"))
dat$InfDN = factor(dat$InfDN, levels = 1:2, labels = c("Object 1", "Object 2"))
for (i in 1:nrow(dat)){
  if (dat$InfDN[i] == "Object 1") {
    dat$confidence[i] = dat$InfCD[i] + 50
  }
  else {
    dat$confidence[i] = 51 - dat$InfCD[i]
  }
}
dat$confidence[dat$confidence <= 10] = 1
dat$confidence[dat$confidence > 10 & dat$confidence <= 20] = 2
dat$confidence[dat$confidence > 20 & dat$confidence <= 30] = 3
dat$confidence[dat$confidence > 30 & dat$confidence <= 40] = 4
dat$confidence[dat$confidence > 40 & dat$confidence <= 60] = 5
dat$confidence[dat$confidence > 60 & dat$confidence <= 70] = 6
dat$confidence[dat$confidence > 70 & dat$confidence <= 80] = 7
dat$confidence[dat$confidence > 80 & dat$confidence <= 90] = 8
dat$confidence[dat$confidence > 90 & dat$confidence <= 100] = 9

dat$confidence = factor(dat$confidence, levels = 1:9,
                        labels = c("Very High(2)", "High(2)", "Medium(2)", "Low(2)",
                                   "No confidence",
                                   "Low(1)", "Medium(1)", "High(1)", "Very High(1)") )
dat$obj1.F = scale(dat$obj1.F)
dat$obj1.KnA = scale(dat$obj1.KnA)
dat$obj2.F = scale(dat$obj2.F)
dat$obj2.KnA = scale(dat$obj2.KnA)
```

Calculate the area of the overlap

```r
for (i in 1:nrow(dat)){
  # Creating 2 data frames which contain the coordinates of two triangles
  object1 = data.frame(x = c(dat$obj1.EstRL[i],dat$obj1.EstRB[i],dat$obj1.EstRH[i]),
                       y = c(0,2/(dat$obj1.EstRL[i]-dat$obj1.EstRH[i]),0))
  object2 = data.frame(x = c(dat$obj2.EstRL[i],dat$obj2.EstRB[i],dat$obj2.EstRH[i]),
                       y = c(0,2/(dat$obj2.EstRL[i]-dat$obj2.EstRH[i]),0))
  # if one of the distribution is a straight line, then we say the overlap is 0
  if (
    (object1[1,"x"] == object1[2,"x"] & object1[2,"x"] == object1[3,"x"])
    |
    (object2[1,"x"] == object2[2,"x"] & object2[2,"x"] == object2[3,"x"])
  )

  {
    dat$overlap[i] = 0
  }
  else{
    #Transform the data frame of the coordinates to geometry object
    #and work out the intersection of two geometries
    object1_dist = Polygons(list(Polygon(object1)), "object1")
    object2_dist = Polygons(list(Polygon(object2)), "object2")
    shape = SpatialPolygons(list(object1_dist, object2_dist))
    overlap = gIntersection(shape["object1"], shape["object2"])
    if (is.null(overlap) == TRUE){
      dat$overlap[i] = 0
    }
    else{
      #Work out the area of the overlap
      dat$overlap[i] = gArea(overlap)
    }
  }
}
dat.training = dat[dat$sbj.ID %in% c(1:75), ]
dat.val = dat[dat$sbj.ID %in% c(76:100), ]
dat.test = dat[dat$sbj.ID %in% c(101:125), ]
```

Model Fitting

In both models for confidence and decision, we decided to include 4 memory cues and the three different types of estimated ranks of the two compared objects as our predictors. The variables of different types of estimated ranks were grouped into three delta variables so the interpretation would be easier. However, there were still some issues need to be discussed. ## Seperate IVs VS delta IVs The first problem was whether or not grouping the same memory cue variables of different objects into a single delta variable. To tackle that, I used Anova function to test the model with all variables against the model with delta variabls. ### Model for confidence

```r
mod.confidence.seperate = polr(confidence ~ overlap + obj1.R*obj2.R +
                                 obj1.F + obj2.F +
                                 obj1.KnA + obj2.KnA +
                                 I(obj1.EstRB - obj2.EstRB) +
                                 I(obj1.EstRH - obj2.EstRH) +
                                 I(obj1.EstRL - obj2.EstRL),
                               data = dat.training)

mod.confidence.delta = polr(confidence ~ overlap + obj1.R*obj2.R +
                              I(obj1.F - obj2.F) +
                              I(obj1.KnA - obj2.KnA) +
                              I(obj1.EstRB - obj2.EstRB) +
                              I(obj1.EstRH - obj2.EstRH) +
                              I(obj1.EstRL - obj2.EstRL),
                            data = dat.training)
```

```
prediction = predict(mod.confidence.seperate, newdata = dat.val)
df1 = data.frame(actual = dat.val$confidence, predicted = prediction)
accuracy.confidence = c(1,2)
accuracy.confidence[1] = sum(df1$actual == df1$predicted)/ nrow(df1)

prediction = predict(mod.confidence.delta, newdata = dat.val)
df1 = data.frame(actual = dat.val$confidence, predicted = prediction)
accuracy.confidence[2] = sum(df1$actual == df1$predicted)/ nrow(df1)
percent(accuracy.confidence)
```

```
## [1] "55.250%" "55.400%"
```

We can see that the accuracy of these two models for confidence is quite the same, all almost 55 percent. However, if we run the anova test, then the model of delta variables is rejected.

```
anova(mod.confidence.delta, mod.confidence.seperate)
```

```
## Likelihood ratio tests of ordinal regression models
##
## Response: confidence
##
## 1 overlap + obj1.R * obj2.R + I(obj1.F - obj2.F) + I(obj1.KnA - obj2.KnA) + I(obj1.EstRB - obj2.EstR
## 2       overlap + obj1.R * obj2.R + obj1.F + obj2.F + obj1.KnA + obj2.KnA + I(obj1.EstRB - obj2.EstR
##   Resid. df Resid. Dev   Test    Df LR stat.      Pr(Chi)
## 1      6483   19749.32
## 2      6481   19731.99 1 vs 2     2 17.33199 0.0001723483
```

In the anova test above, the model with fewer parameters is tested against the model with more parameters. The p-value is very small, which means that the null hypothesis, that the model with more parameters is the same as the one with fewer parametes, is rejected. Therefore, we will choose the model with seperate variables.

Model for decision

In the models for decision, same story is told. The model with seperate variables is still the winner.

```
mod.decision.seperate = glm(InfDN ~ overlap + obj1.R*obj2.R +
                              obj1.F + obj2.F +
                              obj1.KnA + obj2.KnA +
                              I(obj1.EstRB - obj2.EstRB) +
                              I(obj1.EstRH - obj2.EstRH) +
                              I(obj1.EstRL - obj2.EstRL),
                            data = dat.training, family = binomial)

mod.decision.delta = glm(InfDN ~ overlap + obj1.R*obj2.R +
                           I(obj1.F - obj2.F) +
                           I(obj1.KnA - obj2.KnA) +
                           I(obj1.EstRB - obj2.EstRB) +
                           I(obj1.EstRH - obj2.EstRH) +
                           I(obj1.EstRL - obj2.EstRL),
                         data = dat.training, family = binomial)
```

```
prediction = predict(mod.decision.seperate, newdata = dat.val, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df1 = data.frame(actual = dat.val$InfDN, predicted = prediction)
accuracy.decision = c(1,2)
accuracy.decision[1] = sum(df1$actual == df1$predicted)/ nrow(df1)

prediction = predict(mod.decision.delta, newdata = dat.val, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df1 = data.frame(actual = dat.val$InfDN, predicted = prediction)
accuracy.decision[2] = sum(df1$actual == df1$predicted)/ nrow(df1)
percent(accuracy.decision)
```

```
## [1] "76.650%" "76.900%"
```

```
anova(mod.decision.delta, mod.decision.seperate)
```

```
## Analysis of Deviance Table
##
## Model 1: InfDN ~ overlap + obj1.R * obj2.R + I(obj1.F - obj2.F) + I(obj1.KnA -
##     obj2.KnA) + I(obj1.EstRB - obj2.EstRB) + I(obj1.EstRH - obj2.EstRH) +
##     I(obj1.EstRL - obj2.EstRL)
## Model 2: InfDN ~ overlap + obj1.R * obj2.R + obj1.F + obj2.F + obj1.KnA +
##     obj2.KnA + I(obj1.EstRB - obj2.EstRB) + I(obj1.EstRH - obj2.EstRH) +
##     I(obj1.EstRL - obj2.EstRL)
##   Resid. Df Resid. Dev Df Deviance
## 1      6490     5659.7
## 2      6488     5652.4  2   7.2903
```

Decision as a variable in the model for confidence

Sincethe new confidence variable was transformed from the original scale based on the decision, we might need to include decision as a variable in the mode to avoid bias in the model. So I tested which one is more suitable.

```
mod.confidence.decision = polr(confidence ~ overlap*InfDN + obj1.R*obj2.R*InfDN +
                                 obj1.F*InfDN + obj2.F*InfDN +
                                 obj1.KnA*InfDN + obj2.KnA*InfDN +
                                 I(obj1.EstRB - obj2.EstRB)*InfDN +
                                 I(obj1.EstRH - obj2.EstRH)*InfDN +
                                 I(obj1.EstRL - obj2.EstRL)*InfDN,
                               data = dat.training)
ctable = coef(summary(mod.confidence.decision))
p = pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
p = round(p, digit = 3)
ctable = cbind(ctable, "p value" = p)
ctable
```

```
##                                                  Value Std. Error
## overlap                                   -1.211505858 0.13334854
## InfDNObject 2                          -13.458708661 3.68730712
## obj1.RYes                                0.152516773 0.11055883
## obj2.RYes                                0.343026548 0.19144000
## obj1.F                                   0.591485703 0.06442083
## obj2.F                                  -0.128258978 0.07870976
## obj1.KnA                                 0.804205430 0.05474510
## obj2.KnA                                 0.389903473 0.06950841
## I(obj1.EstRB - obj2.EstRB)               0.018298212 0.01890280
## I(obj1.EstRH - obj2.EstRH)              -0.008911014 0.01744588
## I(obj1.EstRL - obj2.EstRL)              -0.078489719 0.01198875
## overlap:InfDNObject 2                    2.145996773 0.18123395
## obj1.RYes:obj2.RYes                     -0.554796158 0.19802984
## InfDNObject 2:obj1.RYes                 -0.318251136 0.20991862
## InfDNObject 2:obj2.RYes                 -0.664571692 0.22862373
## InfDNObject 2:obj1.F                    -0.718859887 0.10356493
## InfDNObject 2:obj2.F                    -0.285194454 0.10398568
## InfDNObject 2:obj1.KnA                  -0.920179348 0.09240501
## InfDNObject 2:obj2.KnA                  -1.220129416 0.09052569
## InfDNObject 2:I(obj1.EstRB - obj2.EstRB)  -0.034840508 0.02810734
## InfDNObject 2:I(obj1.EstRH - obj2.EstRH)   0.022816448 0.02565940
## InfDNObject 2:I(obj1.EstRL - obj2.EstRL)   0.011886433 0.01790225
## InfDNObject 2:obj1.RYes:obj2.RYes        0.858379584 0.27524429
## Very High(2)|High(2)                   -15.502089004 3.68674894
## High(2)|Medium(2)                      -14.202146257 3.68638690
## Medium(2)|Low(2)                       -13.204132731 3.68615301
## Low(2)|No confidence                   -12.564070449 3.68599065
## No confidence|Low(1)                    -1.073548068 0.09383026
## Low(1)|Medium(1)                        -0.473666713 0.09275623
## Medium(1)|High(1)                        0.381922111 0.09284622
## High(1)|Very High(1)                     1.600799563 0.09691479
##                                            t value p value
## overlap                                  -9.0852576   0.000
## InfDNObject 2                            -3.6500102   0.000
## obj1.RYes                                 1.3795078   0.168
## obj2.RYes                                 1.7918227   0.073
## obj1.F                                    9.1815915   0.000
## obj2.F                                   -1.6295182   0.103
## obj1.KnA                                 14.6899996   0.000
## obj2.KnA                                  5.6094433   0.000
## I(obj1.EstRB - obj2.EstRB)                0.9680157   0.333
## I(obj1.EstRH - obj2.EstRH)               -0.5107805   0.610
## I(obj1.EstRL - obj2.EstRL)               -6.5469493   0.000
## overlap:InfDNObject 2                    11.8410307   0.000
## obj1.RYes:obj2.RYes                      -2.8015786   0.005
## InfDNObject 2:obj1.RYes                  -1.5160691   0.130
## InfDNObject 2:obj2.RYes                  -2.9068360   0.004
## InfDNObject 2:obj1.F                     -6.9411515   0.000
## InfDNObject 2:obj2.F                     -2.7426321   0.006
## InfDNObject 2:obj1.KnA                   -9.9581115   0.000
## InfDNObject 2:obj2.KnA                  -13.4782675   0.000
## InfDNObject 2:I(obj1.EstRB - obj2.EstRB)  -1.2395518   0.215
## InfDNObject 2:I(obj1.EstRH - obj2.EstRH)   0.8892044   0.374
## InfDNObject 2:I(obj1.EstRL - obj2.EstRL)   0.6639631   0.507
## InfDNObject 2:obj1.RYes:obj2.RYes         3.1186100   0.002
## Very High(2)|High(2)                        -4.2048128   0.000
## High(2)|Medium(2)                           -3.8525924   0.000
## Medium(2)|Low(2)                            -3.5820902   0.000
## Low(2)|No confidence                        -3.4086007   0.001
## No confidence|Low(1)                       -11.4413840   0.000
## Low(1)|Medium(1)                            -5.1065755   0.000
## Medium(1)|High(1)                            4.1134913   0.000
## High(1)|Very High(1)                        16.5175976   0.000
```

Above is the model with every variable interacting with decision, and we can see that some variables are not significant, so we might be able to some variables.

```
drop1(mod.confidence.decision)
```

```
## Single term deletions
##
## Model:
## confidence ~ overlap * InfDN + obj1.R * obj2.R * InfDN + obj1.F *
##     InfDN + obj2.F * InfDN + obj1.KnA * InfDN + obj2.KnA * InfDN +
##     I(obj1.EstRB - obj2.EstRB) * InfDN + I(obj1.EstRH - obj2.EstRH) *
##     InfDN + I(obj1.EstRL - obj2.EstRL) * InfDN
##                                  Df   AIC
## <none>                              14910
## overlap:InfDN                     1 15050
## InfDN:obj1.F                      1 14957
## InfDN:obj2.F                      1 14916
## InfDN:obj1.KnA                    1 15008
## InfDN:obj2.KnA                    1 15095
## InfDN:I(obj1.EstRB - obj2.EstRB)  1 14910
## InfDN:I(obj1.EstRH - obj2.EstRH)  1 14909
## InfDN:I(obj1.EstRL - obj2.EstRL)  1 14909
## InfDN:obj1.R:obj2.R               1 14918
```

However, by running the single-term deletion test, we see that the AIC actually will not improve much if we delete any terms. But we can delete the three terms in the table which are decision interacting with the delta of ranks.

```
mod.confidence.decision = polr(confidence ~ overlap*InfDN + obj1.R*obj2.R*InfDN +
                                 obj1.F*InfDN + obj2.F*InfDN +
                                 obj1.KnA*InfDN + obj2.KnA*InfDN +
                                 I(obj1.EstRB - obj2.EstRB) +
                                 I(obj1.EstRH - obj2.EstRH) +
                                 I(obj1.EstRL - obj2.EstRL),
                               data = dat.training)
prediction = predict(mod.confidence.decision, newdata = dat.val)
df1 = data.frame(actual = dat.val$confidence, predicted = prediction)
accuracy = sum(df1$actual == df1$predicted)/ nrow(df1)
percent(accuracy)
```

```
## [1] "58.0%"
```

```r
anova(mod.confidence.seperate, mod.confidence.decision)
```

```
## Likelihood ratio tests of ordinal regression models
##
## Response: confidence
##
## 1                                                 overlap + obj1.R * obj2.R + obj1.F + obj2.F + obj1
## 2 overlap * InfDN + obj1.R * obj2.R * InfDN + obj1.F * InfDN + obj2.F * InfDN + obj1.KnA * InfDN + o
##   Resid. df Resid. Dev   Test    Df LR stat. Pr(Chi)
## 1      6481   19731.99
## 2      6472   14850.16 1 vs 2     9 4881.827       0
```

And, we compare this model with the old model and clearly the model with decision is significantly better. Also, the accuracy is about 3% higher than the model without decision.

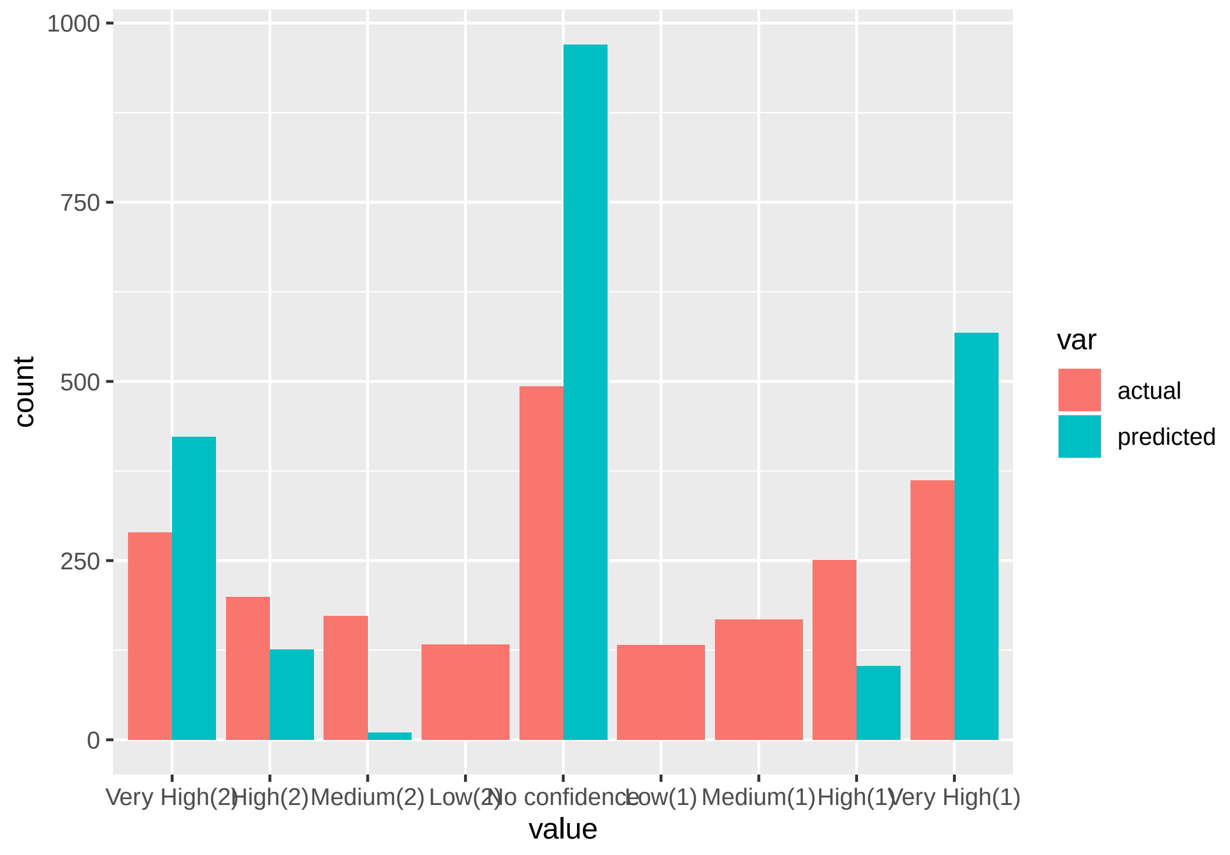
Prediction performance

Now, we evaluate our chosen models' performance on prediction using the test dataset. ## Model for confidence

```r
prediction = predict(mod.confidence.decision, newdata = dat.test)
df1 = data.frame(actual = dat.test$confidence, predicted = prediction)
df1_long = gather(df1, key = var, value = value, actual, predicted)
table(df1$actual,df1$predicted)
```

```
##
##                   Very High(2) High(2) Medium(2) Low(2) No confidence Low(1)
##   Very High(2)             253      27         3      0             6      0
##   High(2)                  112      42         2      0            43      0
##   Medium(2)                 34      35         3      0           101      0
##   Low(2)                    17      16         1      0            99      0
##   No confidence              7       6         1      0           446      0
##   Low(1)                     0       0         0      0            92      0
##   Medium(1)                  0       0         0      0           100      0
##   High(1)                    0       0         0      0            58      0
##   Very High(1)               0       0         0      0            25      0
##
##                   Medium(1) High(1) Very High(1)
##   Very High(2)            0       0            0
##   High(2)                 0       0            0
##   Medium(2)               0       0            0
##   Low(2)                  0       0            0
##   No confidence           0      12           21
##   Low(1)                  0      19           21
##   Medium(1)               0      25           43
##   High(1)                 0      29          164
##   Very High(1)            0      18          319
```

```r
ggplot(data = df1_long, aes(x = value, fill = var)) + geom_bar(position = position_dodge()) +
  scale_x_discrete(limits=c("Very High(2)", "High(2)", "Medium(2)", "Low(2)",
                            "No confidence",
                            "Low(1)", "Medium(1)", "High(1)", "Very High(1)"))
```



```
accuracy = sum(df1$actual == df1$predicted)/ nrow(df1)
percent(accuracy)
```

```
## [1] "49.6%"
```

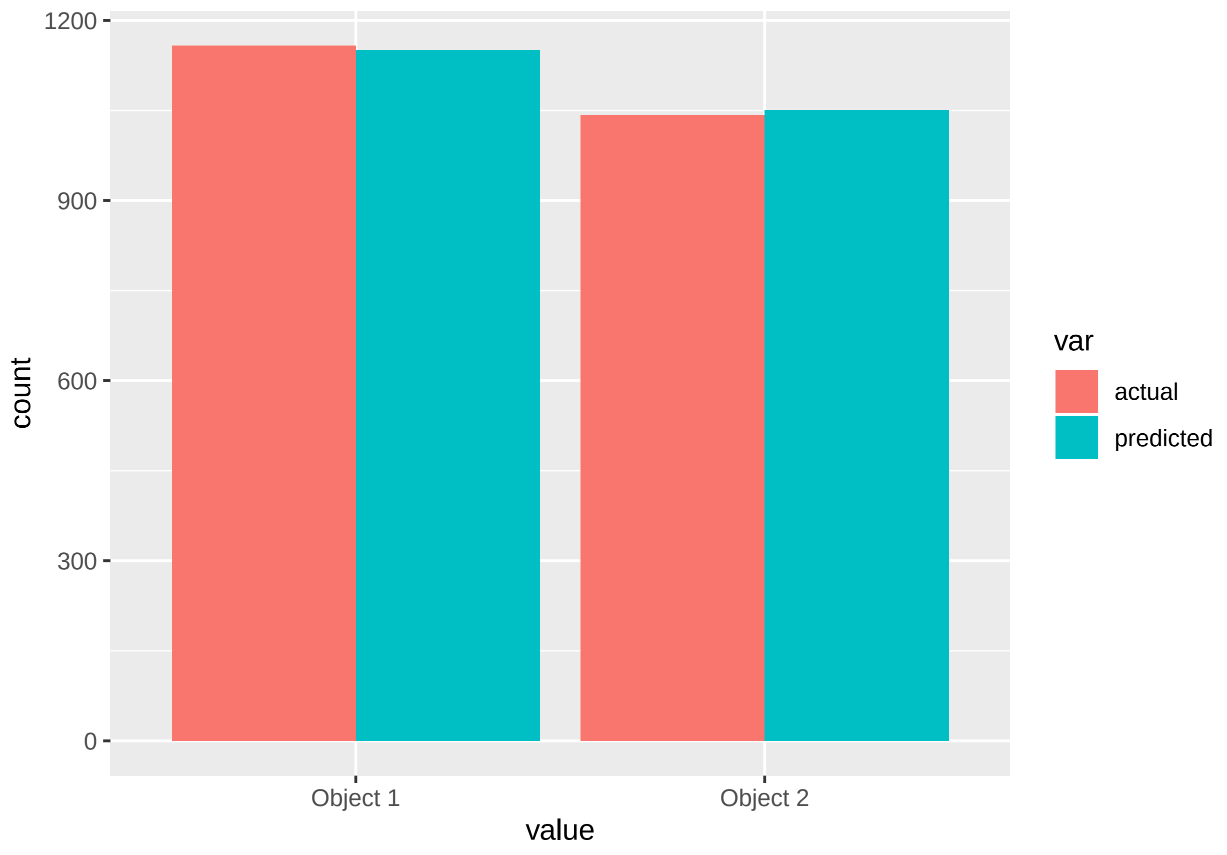
The accuracy of the model for confidence is `{r} percent(accuracy)`. We can also compare the distribution of the prediction with the original test dataset, as shown in the figure above.

Model for decision

```
prediction = predict(mod.decision.seperate, newdata = dat.test, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df2 = data.frame(actual = dat.test$InfDN, predicted = prediction)
df2_long = gather(df2, key = var, value = value, actual, predicted)
table(df2$actual,df2$predicted)
```

```
##
##              Object 1 Object 2
##    Object 1      943      215
##    Object 2      207      835
```

```
ggplot(data = df2_long, aes(x = value, fill = var)) +
  geom_bar(position = position_dodge())
```



```
accuracy = sum(df2$actual == df2$predicted)/ nrow(df2)
percent(accuracy)
```

```
## [1] "80.8%"
```

The accuracy of the model for decision is `{r} percent(accuracy)`.

performance of simple memory cues only

We want to explore how good is the model if we only include simple memory cues as our predictors. ## Model for confidence

```
mod.confidence.simple = polr(confidence ~ obj1.R + obj2.R + obj1.F + obj2.F +
                               obj1.KnA + obj2.KnA, data = dat.training)
accuracy.confidence = data.frame("all cues" = 0, "recognition only" = 0,
                                 "familiarity only" = 0, "knowledge amount only" = 0)
prediction = predict(mod.confidence.simple, newdata = dat.test)
```

```
df1 = data.frame(actual = dat.test$confidence, predicted = prediction)
accuracy.confidence[1,1] = sum(df1$actual == df1$predicted)/ nrow(df1)

mod.confidence.R = polr(confidence ~ obj1.R + obj2.R, data = dat.training)
prediction = predict(mod.confidence.R, newdata = dat.test)
df1 = data.frame(actual = dat.test$confidence, predicted = prediction)
accuracy.confidence[1,2] = sum(df1$actual == df1$predicted)/ nrow(df1)

mod.confidence.F = polr(confidence ~ obj1.F + obj2.F, data = dat.training)
prediction = predict(mod.confidence.F, newdata = dat.test)
df1 = data.frame(actual = dat.test$confidence, predicted = prediction)
accuracy.confidence[1,3] = sum(df1$actual == df1$predicted)/ nrow(df1)

mod.confidence.KnA = polr(confidence ~ obj1.KnA + obj2.KnA, data = dat.training)
prediction = predict(mod.confidence.KnA, newdata = dat.test)
df1 = data.frame(actual = dat.test$confidence, predicted = prediction)
accuracy.confidence[1,4] = sum(df1$actual == df1$predicted)/ nrow(df1)

accuracy.confidence
```

```
##    all.cues recognition.only familiarity.only knowledge.amount.only
## 1 0.4113636        0.3840909        0.4018182             0.3959091
```

The accuracy of the model with all simple memory cues is about 41%. However, the accuracy of the other models with just a single simple memory cue is very close to 41%. Also, the number 41% is not far from the accuracy of our previously chosen model, which means that the simple memory cues alone themselves can do a good job on prediction.

Model for decision

```
mod.decision.simple = glm(InfDN ~ obj1.R + obj2.R + obj1.F + obj2.F +
                              obj1.KnA + obj2.KnA, data = dat.training,
                          family = binomial)
accuracy.decision = data.frame("all cues" = 0, "recognition only" = 0,
                              "familiarity only" = 0, "knowledge amount only" = 0)
prediction = predict(mod.decision.simple, newdata = dat.test, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df2 = data.frame(actual = dat.test$InfDN, predicted = prediction)
accuracy.decision[1,1] = sum(df2$actual == df2$predicted)/ nrow(df2)

mod.decision.R = glm(InfDN ~ obj1.R + obj2.R, data = dat.training,
                          family = binomial)
prediction = predict(mod.decision.R, newdata = dat.test, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df2 = data.frame(actual = dat.test$InfDN, predicted = prediction)
accuracy.decision[1,2] = sum(df2$actual == df2$predicted)/ nrow(df2)

mod.decision.F = glm(InfDN ~ obj1.F + obj2.F, data = dat.training,
                          family = binomial)
```

```r
prediction = predict(mod.decision.F, newdata = dat.test, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df2 = data.frame(actual = dat.test$InfDN, predicted = prediction)
accuracy.decision[1,3] = sum(df2$actual == df2$predicted)/ nrow(df2)

mod.decision.KnA = glm(InfDN ~ obj1.KnA + obj2.KnA, data = dat.training,
                           family = binomial)
prediction = predict(mod.decision.KnA, newdata = dat.test, type = "response")
prediction[prediction >= 0.5] = "Object 2"
prediction[prediction < 0.5] = "Object 1"
df2 = data.frame(actual = dat.test$InfDN, predicted = prediction)
accuracy.decision[1,4] = sum(df2$actual == df2$predicted)/ nrow(df2)

accuracy.decision
```

```
##    all.cues recognition.only familiarity.only knowledge.amount.only
## 1 0.7813636        0.6972727        0.7690909                  0.75
```

The performance of model with all the simple memory cues for decision is very good compared to the previous one, only about 3% less. Among these three cues, familiarity and knowledge amount are very good, achieving 77% and 75% accuracy respectively. Recognition is a bit behind, only about 70%.

Conclusion and discussion

The performance of the model for confidence on the test dataset is not ideal as it is below 50%. It might be due to the fact that the confidence itself is very subjective. We might need to get some more features, e.g. personalities of the individual other than memory cues, which is beyond our research. As for the model for decision, the performance is good, achieving about 80% accuracy.
Simple memory cues can be used as very good predictors in both models for confidence and decision, as the accuracy is not dramatically less than the full model. However, it is worth noting that the confidence data is skewed, i.e., the number of cases in some categories are significantly greater than the others. Using accuracy to evaluate the perdiction performance of models might not be the best way.