Computer Assignment 8 - Multiple Testing with Jelly Beans

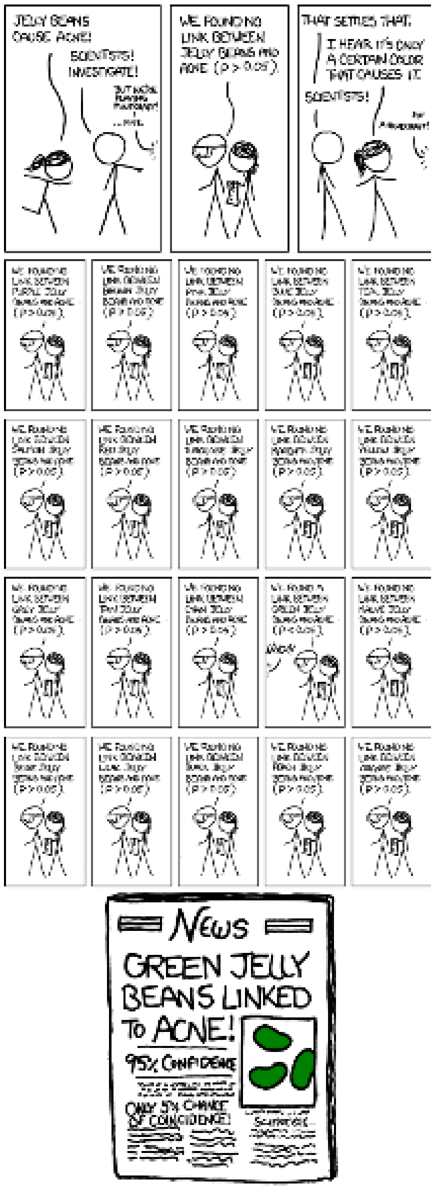
Machine Learning, Spring 2020

Rui Li

We will begin by reiterating a point made earlier in the semester about testing multiple hypotheses. Often, scientists in the field are looking for associations that seem unlikely to occur by chance. The issue with this is if one looks at enough associations then an association is bound to occur by chance rather than by genuine causality. Recall the metaphor about a needle in a haystack: finding a needle in a haystack might be amazing, but if you had checked 1000 haystacks perhaps this is less amazing…

The Problem with P-values?

To get an idea of what we are about to simulate, consider the following comic.


JELLY BEANS CAUSE ACNE!
SCIENTISTS! INVESTIGATE!
BUT WE'RE PLAYING MINECRAFT!
...FINE.
WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE ($p > 0.05$).
THAT SETTLES THAT.
I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.
SCIENTISTS!
WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN GREY JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE ($p > 0.05$).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE ($p < 0.05$).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE ($p > 0.05$).
NEWS
GREEN JELLY BEANS LINKED TO ACNE!
95% CONFIDENCE
ONLY 5% CHANCE OF COINCIDENCE!

Consider the set up of this experiment. You have a bunch of people with or without acne and you further know whether or not they have eaten a jelly bean of a particular color. We can easily simulate this in R:

```
# DON'T EDIT THIS CODE
set.seed(13)
has_acne = rbinom(100, 1, 0.4)
ate_red_jb = rbinom(100, 1, 0.6)
our_data = data.frame(HasAcne = has_acne, AteRedJB = ate_red_jb)
head(our_data)

##   HasAcne AteRedJB
## 1       1        0
## 2       0        0
## 3       0        0
## 4       0        0
## 5       1        1
## 6       0        1

summary(our_data)

##     HasAcne        AteRedJB
##  Min.   :0.00   Min.   :0.00
##  1st Qu.:0.00   1st Qu.:0.00
##  Median :0.00   Median :1.00
##  Mean   :0.37   Mean   :0.58
##  3rd Qu.:1.00   3rd Qu.:1.00
##  Max.   :1.00   Max.   :1.00
```

Here is our simple random sample of 100 people in which we know their acne and red-jelly-bean-eating status. It is significant, and indeed quite important, to note that each of these variables were generated **randomly and independently** of one another. Therefore, we know something that researchers in the field do not normally know: that in this case red-jelly-bean-eating should have **no effect** on the acne of an individual. Let's show this using one of our logistic regression models.

```
m1 <- glm(HasAcne ~ AteRedJB, data = our_data, family = binomial('logit'))
summary(m1)

##
## Call:
## glm(formula = HasAcne ~ AteRedJB, family = binomial("logit"),
##     data = our_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0620  -1.0620  -0.8203   1.2974   1.5829
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.9163     0.3416  -2.683   0.0073 **
## AteRedJB      0.6387     0.4324   1.477   0.1397
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 131.79  on 99  degrees of freedom
## Residual deviance: 129.55  on 98  degrees of freedom
## AIC: 133.55
##
## Number of Fisher Scoring iterations: 4
```

Examine the `Coefficients` section, specifically the column `Pr(>|z|)`, which is the column of p-values. Here we see the p-value derived using the test $H_0: \beta_i = 0$ where $i \in \{0,1\}$ and β_0, β_1 are the coefficients for the `Intercept` and our `AteRedJB` variable, respectively. Notice that we got what we expected! The p-value for the test $H_0: \beta_1 = 0$ was above our usual 0.05 threshold so we therefore **fail to reject** the statement that red-jelly-bean-eating is not significant.

This is all well and good! Statistics works!

But wait. What if we repeated this experiment over and over again? During each experiment we will take a new random sample of people and examine a new color of Jelly Bean. We will record whether or not the jelly-bean-eating for each new experiment is significant. Indeed,

```
# DO NOT EDIT THIS CODE
set.seed(10)
colors = c("Purple", "Brown", "Green", "Pink", "Blue", "Teal", "Salmon", "Red
", "Turquoise",
           "Magenta", "Yellow", "Grey", "Tan", "Cyan", "Mauve", "Beige",
           "Lilac", "Black", "Peach", "Orange")
for(index in 1:20){
  has_acne = rbinom(100, 1, 0.4)
  ate_color_jb = rbinom(100, 1, 0.6)
  our_data = data.frame(HasAcne = has_acne, AteJB = ate_color_jb)
  fit <- glm(HasAcne ~ AteJB, data = our_data, family = binomial('logit'))
  p_value = coef(summary(fit))[,"Pr(>|z|)"][2]
  print(paste(colors[index], ": ",p_value < 0.05, sep = ""))
}

## [1] "Purple: FALSE"
## [1] "Brown: FALSE"
## [1] "Green: TRUE"
## [1] "Pink: FALSE"
## [1] "Blue: FALSE"
## [1] "Teal: FALSE"
## [1] "Salmon: FALSE"
## [1] "Red: FALSE"
## [1] "Turquoise: FALSE"
## [1] "Magenta: FALSE"
## [1] "Yellow: FALSE"
## [1] "Grey: FALSE"
## [1] "Tan: FALSE"
## [1] "Cyan: FALSE"
## [1] "Mauve: FALSE"
## [1] "Beige: FALSE"
## [1] "Lilac: FALSE"
## [1] "Black: FALSE"
## [1] "Peach: FALSE"
## [1] "Orange: FALSE"
```

So, were they right? Does eating green Jelly Beans cause acne?

Questions

The purpose of this exercise is to introduce you to the multiple comparisons problem and thereby hopefully serve as a warning to those who want to use statistics in practice in the future. Note that since this is primarily a written exercise, complete sentences and proper grammar are expected.

1. Read pages 67-68 in Gareth James' book. Based on what is written here, and any other (credible) resource that you can find, provide a full definition of a p-value.

```
The p-value is the probability of obtaining results as extreme as the observe
d results of a statistical hypothesis test, assuming that the null hypothesis
 is correct. The p-value is used as an alternative to rejection points to pro
vide the smallest level of significance at which the null hypothesis would be
 rejected. A smaller p-value means that there is stronger evidence in favor o
f the alternative hypothesis.
```

2. In what situation might a p-value be inaccurate?

```
There are several situations may having misleading p-value.

1. If the variability between groups is similar to that within groups, then t
he means are likely to differ only because of random error.

2. The sample number is small and very closed to the random error number, whi
ch means p-value < 0.05 is caused by accident.

3. The p-value is based on a false null hypothesis.
```

3. Do green Jelly Beans cause acne? Your answer to this question should explain either why the above experiment validates your conclusion or why the above results may be misleading.

```
Green Jelly Beans does not cause acne. Given 20 different colors, it's possib
le that the test generates a false positive, because there could still be a 1
 in 20 chance that this result was purely a statistical fluke.
```

4. If you checked 100 colors of Jelly Beans (rather than 20), how many would you expect to have a statistically significant effect on acne-having? Why?

```
According to the statistical result, there could be a 5% false positive. Thus,
 there may have 5 statistically significant effect out of 100 colors on acne-
having, which are all false positive.
```

5. Multiple testing is actually a well-known problem in Statistics. A common solution is to do a **Bonferroni Correction**. There are many resources online on how to do this. Research the **Bonferroni Correction** and repeat the above experience using it (and using the same `set.seed()`). Discuss the results.

```
set.seed(10)
colors = c("Purple", "Brown", "Green", "Pink", "Blue", "Teal", "Salmon", "Red
", "Turquoise",
           "Magenta", "Yellow", "Grey", "Tan", "Cyan", "Mauve", "Beige",
           "Lilac", "Black", "Peach", "Orange")
p_org = c()
for(index in 1:20){
  has_acne = rbinom(100, 1, 0.4)
  ate_color_jb = rbinom(100, 1, 0.6)
  our_data = data.frame(HasAcne = has_acne, AteJB = ate_color_jb)
  fit <- glm(HasAcne ~ AteJB, data = our_data, family = binomial('logit'))
  p_value = coef(summary(fit))[,"Pr(>|z|)"][2]

  #Bonferroni Correction
  p_org = c(p_value,p_org)
}

  p_adj =p.adjust(p_org, method = "bonferroni")
  for (index in 1:20) {
    print(paste(colors[index], ": ",p_adj[index]<0.05, sep = ""))
  }

## [1] "Purple: FALSE"
## [1] "Brown: FALSE"
## [1] "Green: FALSE"
## [1] "Pink: FALSE"
## [1] "Blue: FALSE"
## [1] "Teal: FALSE"
## [1] "Salmon: FALSE"
## [1] "Red: FALSE"
## [1] "Turquoise: FALSE"
## [1] "Magenta: FALSE"
## [1] "Yellow: FALSE"
## [1] "Grey: FALSE"
## [1] "Tan: FALSE"
## [1] "Cyan: FALSE"
## [1] "Mauve: FALSE"
## [1] "Beige: FALSE"
## [1] "Lilac: FALSE"
## [1] "Black: FALSE"
## [1] "Peach: FALSE"
## [1] "Orange: FALSE"
```

```
According to the result, All Jelly Beans does not cause acne. p_adjust does n
ot show any significant value.
```

Model Assumptions

We have spent a lot of time this semester looking at different supervised and unsupervised methods for analyzing data and performing classification. There is a saying in statistics: **every dog has its day**. This is to say that there is always a situation where a certain method will out perform all the others and will be, in a sense, the "right" method for a given problem. There is an art to picking the right tool in your toolbox for a given application, so the next few problems (on this and the next computer assignment) will try to build your intuition on when to use certain methods. There are some questions in this next exercise that are similar to previous exercises. This is to give you extra practice and to help motivate what we will try to do next week.

The Perfect Data for LDA

Recall that in the case of LDA is we assume that the marginal class densities are both normal and have the same variance. This is, for the binary classification problem, we have that:

$$ \text{Data Classified as 0} \sim \mathcal{N}(\mu_0, \Sigma),\\ \text{Data Classified as 1} \sim \mathcal{N}(\mu_1, \Sigma). $$

```r
# Make sure to install this package if you do not have it:
library(mvtnorm)
my_sigma = t(matrix(c(1:25), ncol=5)) %*% matrix(c(1:25), ncol=5)
data_0 = rmvnorm(250, mean = rep(1, times = 5), sigma = my_sigma)
data_1 = rmvnorm(250, mean = rep(-1, times = 5), sigma = my_sigma)
labels = c(rep(0, times = 250), rep(1, times = 250))
names = c("variable1", "variable2", "variable3", "variable4", "variable5", "l
abels")
full_data = rbind(data_0, data_1)
full_data = cbind(full_data, labels)
full_data = as.data.frame(full_data)
names(full_data) = names
```

A crucial thing to note here is that this data **fits the assumptions we made with a LDA model exactly**. The class marginals are, indeed, normal and they do, indeed, have the same variance. Let us now examine how well an LDA model does on data like this, and compare it to another one of our models.

Questions

1) Fit an LDA model on this data and examine the model output. Does this output makes sense based off of our data? Print the linear discriminants (look at possible model outputs).

```
library(MASS)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

full_lda = lda(labels~., data = full_data)

## Warning in lda.default(x, grouping, ...): variables are collinear

lda_predictions <- full_lda %>% predict(full_data)
lda_error = sum(lda_predictions$class!=full_data$labels)/length(full_data$lab
els)
lda_error

## [1] 0.37

From the LDA model, we get 0.63 correctness, which is not a good description
of the model.
```

2) Split the data into a testing data set and a training data set. Build an LDA model on the training data. Use the training data again to predict the class labels and report the performance of your model. What is the error rate? (the proportion of times your model classified an observation incorrectly)

```
#Split the data set to testing data and training data
set.seed(13)
training_size <- round(.75 * nrow(full_data))  # training set size
indices = sample(1:nrow(full_data), training_size)
training_set <- full_data[indices,]
testing_set <- full_data[-(indices),]

train_lda = lda(labels~., data = training_set)

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
train_predictions <- train_lda %>% predict(training_set)
train_error = sum(train_predictions$class!=training_set$labels)/length(traini
ng_set$labels)
train_error

## [1] 0.3573333

The error rate is 0.357.
```

3) Now predict the class labels using the testing data and report the performance of your model. What is the error rate?

```
train_lda = lda(labels~., data = training_set)

## Warning in lda.default(x, grouping, ...): variables are collinear

test_predictions <- train_lda %>% predict(testing_set)
test_error = sum(test_predictions$class!=testing_set$labels)/length(testing_s
et$labels)
test_error

## [1] 0.408

The error rate is 0.408.
```

4) Repeat steps 2-3 using a knn model for $k \in \{1,5,11\}$. Considering only the cases where you predicted labels on your testing data, compare the error rates between the LDA model and all of the knn models.

```
#knn risk
library(class)
train_data_classifiers = as.factor(training_set$labels)
train_data_observations = training_set[,-6]
test_data_observations = testing_set[,-6]
test_data_classifiers = as.factor(testing_set$labels)

#k=1
knn.1 <-  knn(train_data_observations, test_data_observations, cl = train_dat
a_classifiers, k=1)
knn.1_risk = sum(test_data_classifiers != knn.1)/length(knn.1)
knn.1_risk

## [1] 0.392

#k=5
knn.5 <-  knn(train_data_observations, test_data_observations, cl = train_dat
a_classifiers, k=5)
knn.5_risk = sum(test_data_classifiers != knn.5)/length(knn.5)
knn.5_risk

## [1] 0.432

#k=11
knn.11 <-  knn(train_data_observations, test_data_observations, cl = train_da
ta_classifiers, k=11)
knn.11_risk = sum(test_data_classifiers != knn.11)/length(knn.11)
knn.11_risk

## [1] 0.472
```

```
From the result we notice that the error rate of LDA models are 0.37, 0.35, 0.
40 respetively, and the error rate of Knn models are 0.39, 0.43, and 0.47 res
pectively. Thus the mean error rate of LDA is smaller than that of Knn, showi
ng that LDA works better for this data.
```

Note that this is a case where LDA is supposed to perform especially well. The punchline here is that if you encounter data where the class marginals seem normal with the same variance, then LDA is a *very good* choice! For our next CA we will examine what happens when the assumptions inhernit to LDA are broken. That is, when LDA perhaps is not the best tool for the job.