Analysis Markdown: Critics : Study of TV Pilots

A. L. S.

The following outlines the analysis, which was performed using R. Each section below corresponds to code for the corresponding section in the paper. The `dplyr, reshape2, xtable, gridExtra, ggplot` libraries are used liberally throughout.

Introduction

First, the data is read in from its *csv file. The first few rows are displayed below.

```
x<-read.csv("Critics.csv")
head(x)
```

```
##               Show Critic Acting Curiosity Script Characters Originality
## 1   Masters of Sex   Anne      7         9      7          7          10
## 2       Getting On   Anne      7         3      5          5           9
## 3 Better Call Saul   Anne      8        10      7          7           8
## 4      Broadchurch   Anne     10        10      9          9           3
## 5      Enlightened   Anne      7         6      7          6           7
## 6       The Affair   Anne      6         8      7          8           7
```

Next, a total score called "Overall" is computed for each row.

```
x<-x %>% mutate(Overall=apply(x[,3:7],1,sum))
```

Relative weights are assigned to a vector, p, normalized, and then used to calculate the weighted total score, to be used in Section 3.2. This data frame is then melted in order for different manipulations. The first few rows of the melted data frame, xmelt, are shown, as well as a preview of its structure, where Critic takes on two values and Feature is one of the five features.

```
p<-c(1,2,1,1,.5) # relative weights for Section 3.2
p<-p/sum(p) # normalized weights
x$OverallWeighted<-sapply(lapply(1:nrow(x),function(i) x[i,3:7]*p),sum)
xmelt<-melt(x,id=c("Show","Critic"))
names(xmelt)[3:4]<-c("Feature","Score")
head(xmelt)
```

```
##               Show Critic Feature Score
## 1   Masters of Sex   Anne  Acting     7
## 2       Getting On   Anne  Acting     7
## 3 Better Call Saul   Anne  Acting     8
## 4      Broadchurch   Anne  Acting    10
## 5      Enlightened   Anne  Acting     7
## 6       The Affair   Anne  Acting     6
```

```
str(xmelt)
```

```
## 'data.frame':    196 obs. of  4 variables:
##  $ Show   : Factor w/ 14 levels "Better Call Saul",..: 9 6 1 2 4 11 7 3 12 13 ...
##  $ Critic : Factor w/ 2 levels "Anne","Avery": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Feature: Factor w/ 7 levels "Acting","Curiosity",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Score  : num  7 7 8 10 7 6 8 3 8 8 ...
```

Figure 1 is produced by the following code.

```
pdf(file="Critics_overall.pdf",width=15)
go<-ggplot(filter(xmelt,Feature=="Overall"), aes(x=Show,y=Score,fill=Critic))
go<-go+geom_bar(col="black",stat="identity",position="dodge",width=.75)
go<-go+theme(text = element_text(size=14),
        axis.text.x=element_text(angle=45,hjust=1),axis.title.y=element_text(vjust=1.3))
go<-go+scale_fill_manual(values=c("indianred1","lemonchiffon1"),
        labels=paste(levels(xmelt$Critic),": ",
        round(filter(sdf,Feature=="Overall")$m,1)," +/- ",
        round(filter(sdf,Feature=="Overall")$s,1),sep=""))
go<-go+labs(x="TV Pilot", y="Total Score (out of 50)")
go
dev.off()
```

This produces the box plot in Figure 2. Factor levels are reversed so they appear in alphabetical order from the top down.

```
xmeltrev<-xmelt
xmeltrev$Show<-with(xmeltrev,factor(Show,levels=rev(levels(Show))))
pdf(file="Critics_boxplot.pdf")
gobox<-ggplot(filter(xmeltrev,Feature!="Overall"), aes(x=Show,y=Score,fill=Critic))
gobox<-gobox+geom_boxplot(width=0.5)+theme_bw()+coord_flip()
gobox<-gobox+scale_fill_manual(values=c("indianred1","lemonchiffon1"))
gobox
dev.off()
```

Hypothesis Testing to Investigate Differences in Critics

The following vectors are formed for ease of use in the hypothesis tests. Each vector isolates the critic and only uses the total score, Overall. A sample of Avery's total scores is printed.

```
AnneO<-x[x$Critic=="Anne","Overall"] #
AveryO<-x[x$Critic=="Avery","Overall"]
nav<-length(AveryO) # number of shows for Avery
nanne<-length(AnneO) # number of shows for Anne
AveryO
```

```
##  [1] 46 39 47 48 41 47 34 35 41 46 38 44 43 47
```

First, the test for unpaired sets with unequal variances. Both outputs from the `t.test` function are shown as well as the direct calculation.

```r
# not paired, unequal variances
t.test(AveryO,AnneO,var.equal=F)
```

```
##
##  Welch Two Sample t-test
##
## data:  AveryO and AnneO
## t = 3.9535, df = 21.583, p-value = 0.0006957
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   4.47711 14.38003
## sample estimates:
## mean of x mean of y
##  42.57143  33.14286
```

```r
s<-sqrt(sd(AveryO)^2/nav+sd(AnneO)^2/nanne) # standard error
tstat<-(mean(AveryO)-mean(AnneO))/s # t-statistic
dfree<-(sd(AveryO)^2/nav+sd(AnneO)^2/nanne)^2/
        (sd(AveryO)^4/nav^2/(nav-1)+sd(AnneO)^4/nanne^2/(nanne-1)) # degrees of freedom
pval<-pt(tstat,dfree,lower.tail=F)*2 # p-value
cint<-(mean(AveryO)-mean(AnneO))+c(-1,1)*qt(.975,dfree)*s # confidence interval
dfree;tstat;pval;cint;
```

```
## [1] 21.58295
```

```
## [1] 3.953497
```

```
## [1] 0.0006957321
```

```
## [1]  4.47711 14.38003
```

Next, the unpaired test with equal variances.

```r
# not paired, equal variances
t.test(AveryO,AnneO,var.equal=T)
```

```
##
##  Two Sample t-test
##
## data:  AveryO and AnneO
## t = 3.9535, df = 26, p-value = 0.0005277
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   4.526404 14.330739
## sample estimates:
## mean of x mean of y
##  42.57143  33.14286
```

```
s<-sqrt(((nav-1)*sd(AveryO)^2+(nanne-1)*sd(AnneO)^2)/(nanne+nav-2) *
        (1/nanne+1/nav)) # combined variance
tstat<-(mean(AveryO)-mean(AnneO))/s # t-statistic
pval<-pt(tstat,nav+nanne-2,lower.tail=F)*2
cint<-(mean(AveryO)-mean(AnneO))+c(-1,1)*qt(.975,nav+nanne-2)*s # confidence interval
dfree;tstat;pval;cint;
```

```
## [1] 21.58295
```

```
## [1] 3.953497
```

```
## [1] 0.000527722
```

```
## [1]  4.526404 14.330739
```

The last hypothesis test is for paired sets.

```
#paired
t.test(AveryO,AnneO,paired=T)
```

```
##
##  Paired t-test
##
## data:  AveryO and AnneO
## t = 5.5811, df = 13, p-value = 8.905e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.778889 13.078254
## sample estimates:
## mean of the differences
##                9.428571
```

```
s<-sd(AveryO-AnneO)/sqrt(nanne)
tstat<-mean(AveryO-AnneO)/s # paired
pval<-pt(tstat,nanne-1,lower.tail=F)*2
cint<-mean(AveryO-AnneO)+c(-1,1)*qt(.975,nanne-1)*s # paired
dfree;tstat;pval;cint;
```

```
## [1] 21.58295
```

```
## [1] 5.581085
```

```
## [1] 8.90456e-05
```

```
## [1]  5.778889 13.078254
```

Distribution of Scores for Features

Next, the means, standard deviations, and fractional uncertainties are calculated for each feature. Then the dat is reshaped (dcast) for Table 1.

```
sdf<-as.data.frame(summarise(group_by(xmelt,Critic,Feature),
        m=mean(Score),s=sd(Score),frac=s/m))
sdf1<-dcast(sdf,Feature~Critic,value.var="m")
sdf2<-dcast(sdf,Feature~Critic,value.var="s")
names(sdf1)[2:3]<-c("Anne_mean","Avery_mean")
names(sdf2)[2:3]<-c("Anne_sd","Avery_sd")
sdfdcast<-cbind(sdf1,select(sdf2,-Feature))
sdfdcast<-sdfdcast[,c(1,2,4,3,5)]
```

This outputs the latex code for Table 1.

```
print(xtable(sdfdcast),include.rownames=F)
```

This produces Figure 3.

```
pdf(file="Critics_dist.pdf",width=11)
g<-ggplot(xmelt,aes(x=Score,fill=Critic))
g<-g+geom_histogram(color="black",binwidth=1)
g<-g+facet_wrap(~Feature,scales="free")
g<-g+labs(y="Number of Counts")+theme_bw()
g<-g+scale_fill_manual(values=c("darkseagreen1","seagreen4"))
g
dev.off()
```

Next is the code for Figure 4. Because of the different ranges for the x-axis, this plot is formed in two steps. First is for all features separately, then for the total score (Overall). These plots are then knitted together with `grid.arrange`.

```
cols<-brewer.pal(8,"Blues")
pal<-colorRampPalette(cols)
g1<-ggplot(filter(xmelt,Feature!="Overall"),aes(x=Score,fill=Feature))
g1<-g1+geom_histogram(color="black",binwidth=1,breaks=seq(0,10,1))
g1<-g1+facet_wrap(Critic~Feature,nrow=2,ncol=5,scales="free_y")
g1<-g1+labs(y="Frequency")+theme_bw()
g1<-g1+theme(axis.title.x=element_text(hjust=.65))
g1<-g1+scale_fill_manual(values=pal(5),guide=F)
g1<-g1+scale_x_continuous(breaks=seq(0,10,2))
g1<-g1+scale_y_continuous(breaks=seq(0,14,2))

g1a<-ggplot(filter(xmelt,Feature=="Overall"),aes(x=Score,fill=Feature))
g1a<-g1a+geom_histogram(color="black",binwidth=1)
g1a<-g1a+facet_wrap(Critic~Feature,nrow=2,ncol=1,scales="free_y")
g1a<-g1a+labs(y="",x="")+theme_bw()
g1a<-g1a+scale_fill_manual(values="mediumorchid",guide=F)
g1a<-g1a+scale_y_continuous(breaks=seq(0,3,1))

pdf(file="Critics_splitCritic.pdf",width=13)
grid.arrange(g1,g1a,nrow=1,widths=c(3.2,.8))
dev.off()
```

In order to calculate the p-values for the matrix of hypothesis tests, the data is separated by critic and only scores for individual features are kept. Then the feature pairs are looped over, each time calculating the

p-values with `t.test`, and adding it to a data frame for each critic. The resulting data frame for Anne is shown.

```
xAnne<-x %>% filter(Critic=="Anne") %>% select(-Overall,-OverallWeighted,-Show,-Critic)
xAvery<-x %>% filter(Critic=="Avery") %>% select(-Overall,-OverallWeighted,-Show,-Critic)

featAnne<-data.frame()
featAvery<-data.frame()
for (i in 1:5){
   for (j in 1:5){
     featAnne<-rbind(featAnne,cbind(names(xAnne)[i],
                names(xAnne)[j],t.test(xAnne[,i],xAnne[,j])$p.val))
     featAvery<-rbind(featAvery,cbind(names(xAvery)[i],
                names(xAvery)[j],t.test(xAvery[,i],xAvery[,j])$p.val))
   }
}
featAnne
```

```
##             V1          V2                V3
## 1       Acting      Acting                 1
## 2       Acting   Curiosity 0.775495236395331
## 3       Acting      Script  0.46196400870266
## 4       Acting  Characters 0.641290778802611
## 5       Acting Originality 0.233993427544372
## 6    Curiosity      Acting 0.775495236395331
## 7    Curiosity   Curiosity                 1
## 8    Curiosity      Script 0.725488953311352
## 9    Curiosity  Characters 0.925237293921935
## 10   Curiosity Originality 0.446892069881033
## 11      Script      Acting  0.46196400870266
## 12      Script   Curiosity 0.725488953311352
## 13      Script      Script                 1
## 14      Script  Characters 0.755537576835056
## 15      Script Originality 0.645599251321352
## 16  Characters      Acting 0.641290778802611
## 17  Characters   Curiosity 0.925237293921935
## 18  Characters      Script 0.755537576835056
## 19  Characters  Characters                 1
## 20  Characters Originality 0.430697017771925
## 21 Originality      Acting 0.233993427544372
## 22 Originality   Curiosity 0.446892069881033
## 23 Originality      Script 0.645599251321352
## 24 Originality  Characters 0.430697017771925
## 25 Originality Originality                 1
```

In order to manipulate these data frames, they are reorganized by `dcast` and converted to numbers. Then code is run to determine if any of the pairs have a p-value that is less than the significance level of 0.05. The minimum value for each critic is also calculated. This could have been calculated directly from the data frames above, instead of melting the dcasted data frame, but this is just an illustration of how data can be reshaped.

```
featAnne<-dcast(featAnne,V1~V2)
```

```
## Using V3 as value column: use value.var to override.
```

```
featAnne[,-1]<-apply(featAnne[,-1],c(1,2),as.numeric)
featAvery<-dcast(featAvery,V1~V2)
```

```
## Using V3 as value column: use value.var to override.
```

```
featAvery[,-1]<-apply(featAvery[,-1],c(1,2),as.numeric)
sum(apply(featAnne[,-1],1,function(i) any(i < 0.05)))
```

```
## [1] 0
```

```
sum(apply(featAvery[,-1],1,function(i) any(i < 0.05)))
```

```
## [1] 2
```

```
min(melt(featAnne)$value)
```

```
## Using V1 as id variables
```

```
## [1] 0.2339934
```

```
min(melt(featAvery)$value)
```

```
## Using V1 as id variables
```

```
## [1] 0.03478625
```

There are 0 values for Anne and 2 values for Avery that are below 0.05. (Avery's values are degenerate, as in they came from the same pair of features). The minimum value for Avery is 0.035 and 0.23 for Anne.

Figure 5 is next. The data from the data frames above first need to be converted into matrix form in order to be plotted.

```
matAnne<-as.matrix(featAnne[,-1])
matAvery<-as.matrix(featAvery[,-1])

pdf(file="Critics_pvals.pdf",width=15)
par(mfrow=c(1,2))
par(mar=c(5,5,5,5))
cols<-brewer.pal(5,"BuGn")
pal<-colorRampPalette(cols)
image.plot(matAvery,col=pal(40),xlab="Feature",ylab="Feature",axes=F)
title(main="Critic = Avery")
axis(1,at=seq(0,1,length.out=5),labels=names(featAnne)[-1])
axis(2,at=seq(0,1,length.out=5),labels=names(featAnne)[-1])

par(mar=c(5,5,5,5))
cols<-brewer.pal(10,"PuBu")
pal<-colorRampPalette(cols)
image.plot(matAnne,col=pal(40),xlab="Feature",ylab="",axes=F)
title(main="Critic = Anne")
axis(1,at=seq(0,1,length.out=5),labels=names(featAvery)[-1])
axis(2,at=seq(0,1,length.out=5),labels=names(featAvery)[-1])
```

Tallying the Scores

The top six shows are then selected. First, scores per critic are islated, arranged by total score (Overall), and then ranked. The same is done for the weighted scores. Then the data frames for each critic are merged into one, Overall.

```
rk<-seq(1:14)
Anne<-x %>% filter(Critic=="Anne") %>% select(Show,Overall,OverallWeighted) %>%
        arrange(desc(Overall)) %>% mutate(Rank=rk)
Anne<-Anne  %>%arrange(desc(OverallWeighted)) %>% mutate(RankWeighted=rk)
Avery<-x %>% filter(Critic=="Avery")%>% select(Show,Overall,OverallWeighted)%>%
        arrange(desc(Overall))%>% mutate(Rank=rk)
Avery<-Avery  %>%arrange(desc(OverallWeighted)) %>% mutate(RankWeighted=rk)
Overall<-merge(Anne,Avery,by="Show")
```

The names of the data frame are changed and then the summed totals per critic are calculated, as well as the mean ranks for non-weighted and weighted features.

```
nms<-names(Overall)
nms<-gsub("x","Anne",nms);nms<-gsub("y","Avery",nms)
names(Overall)<-nms
Overall<-Overall %>% mutate(Total=apply(Overall[,grep("Overall\\.",nms)],1,sum),
        meanRank=apply(Overall[,grep("Rank\\.",nms)],1,mean),
        TotalWeighted=apply(Overall[,grep("OverallW",nms)],1,sum),
        meanRankWeighted=apply(Overall[,grep("RankW",nms)],1,mean))
Overall<-Overall[,c(grep("Weighted",names(Overall),invert=T),grep("Weighted",names(Overall)))]
head(Overall)
```

```
##                  Show Overall.Anne Rank.Anne Overall.Avery Rank.Avery
## 1    Better Call Saul           40         4            47          2
## 2         Broadchurch           41         2            48          1
## 3              Empire           16        14            35         13
## 4         Enlightened           33         8            41          9
## 5 Friday Night Lights           29        12            47          4
## 6          Getting On           29        11            39         11
##   Total meanRank OverallWeighted.Anne RankWeighted.Anne
## 1    87      3.0             8.363636                 3
## 2    89      1.5             9.000000                 1
## 3    51     13.5             3.363636                14
## 4    74      8.5             6.454545                 8
## 5    76      8.0             6.090909                11
## 6    68     11.0             5.000000                12
##   OverallWeighted.Avery RankWeighted.Avery TotalWeighted meanRankWeighted
## 1              9.545455                 2      17.90909              2.5
## 2              9.818182                 1      18.81818              1.0
## 3              7.181818                13      10.54545             13.5
## 4              8.181818                 9      14.63636              8.5
## 5              9.545455                 3      15.63636              7.0
## 6              7.727273                11      12.72727             11.5
```

Next, the data frame is rearranged based on total score and rank in various ways. Then the result is printed to a latex table for Table 2.

```
Overall<-select(Overall,Show,Total,meanRank,TotalWeighted,meanRankWeighted)
Overall<-Overall %>% arrange(desc(Total))
Overall<-Overall %>% arrange(meanRank)
Overall<-Overall %>% arrange(desc(TotalWeighted))
Overall<-Overall %>% arrange(meanRankWeighted)
```

```
print(xtable(Overall),include.rownames=F)
```

Outliers and Refinements to the Algorithm

The Broadchurch data is isolated, and the quantiles are calculated for each critic. This is formed into Table 3.

```
broadchurch<-xmelt %>% filter(Show=="Broadchurch")

qavery<-quantile(filter(broadchurch,Feature!="Overall" & Critic=="Avery")$Score)
qanne<-quantile(filter(broadchurch,Feature!="Overall" & Critic=="Anne")$Score)
q<-rbind(qavery,qanne)
q
```

```
##        0%      25% 50%   75% 100%
## qavery  8 9.863636  10 10.00   10
## qanne   3 9.000000   9  9.75   10
```

```
row.names(q)<-c("Avery","Anne")
print(xtable(q))
```

```
## % latex table generated in R 3.1.2 by xtable 1.7-4 package
## % Sun May 31 19:01:28 2015
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrrr}
##   \hline
##  & 0\% & 25\% & 50\% & 75\% & 100\% \\
##   \hline
## Avery & 8.00 & 9.86 & 10.00 & 10.00 & 10.00 \\
##   Anne & 3.00 & 9.00 & 9.00 & 9.75 & 10.00 \\
##    \hline
## \end{tabular}
## \end{table}
```

Results with Weights

The results for the weighted features:

```
Overall<-Overall %>% arrange(meanRankWeighted)
head(Overall)
```

```
##                   Show Total meanRank TotalWeighted meanRankWeighted
```

```
## 1         Broadchurch    89      1.5      18.81818             1.0
## 2    Better Call Saul    87      3.0      17.90909             2.5
## 3           West Wing    88      3.5      17.72727             3.5
## 4      Masters of Sex    86      4.0      17.18182             5.0
## 5          The Affair    83      4.5      16.81818             5.0
## 6 Friday Night Lights    76      8.0      15.63636             7.0
```

Correlations Between Features

The following is for Figure 6.

```
pdf(file="Critics_actingvsscript.pdf")
g4<-ggplot(x,aes(x=Acting,y=Script))
g4<-g4+geom_point(aes(color=Critic,shape=Critic),size=5,alpha=0.8)
g4<-g4+theme_bw()+geom_smooth(aes(color=Critic),method=lm)
g4<-g4+scale_color_hue(l=30)
g4
dev.off()
```

Figure 7 is built with the `pairs` function.

```
pdf(file="Critics_pairs.pdf")
pairs(select(x,-Show,-Overall,-Critic),panel=panel.smooth,col=3+(x$Critic=="Anne"))
dev.off()
```