4VCcueweights

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

rm(list = ls())  
  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(reshape2)  
library(plyr)

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

library(ggplot2)  
library(gplots)

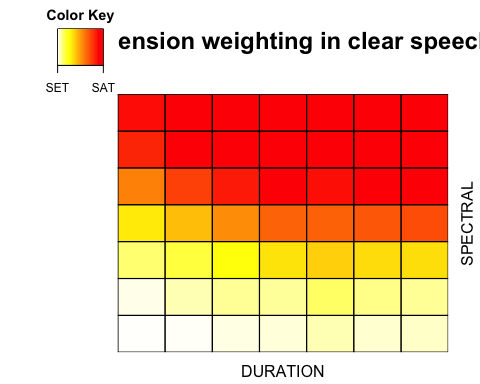
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

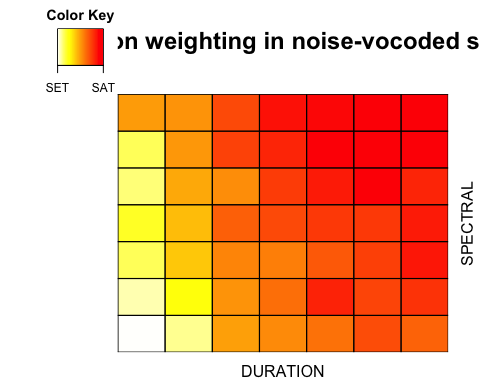
################set your working directory#########################  
setwd("/Users/charleswu/Google Drive/HoltLab/VocodingProj/4ChannelCueWeights") #mac  
##setwd("C:/Users/charl/Google Drive/HoltLab/VocodingProj/4ChannelCueWeights") #PC  
  
######################################  
#####Five general steps in this code#############  
####1. read in the data#########  
####2. select and sort the variables that you re interested in  
####3.construct the matrices for plotting. Note these mat  
###rices are not cue weights, but only proportion of responses for visualization  
####4.construct the matrices for individual cue weights and plot them  
####5.calculate overall cue weights  
######################################  
###########step 1###################  
ddd <- read.csv("Data\_final.csv") ###read in the data in the folder  
ddd$Subject <- as.factor(ddd$Subject)  
ddd$Block <- as.factor(ddd$Block)  
ddd$duration <- as.factor(ddd$duration)  
ddd$spec <- as.factor(ddd$spec) ###set these factors to categorical because it's generally good for plotting since some plotting functions will assign different color values to different levels of the categorical variable but for a continuous variable, the function might only use colors that are continuous in the color space. Later, however, you would need to change them(such as duratoin and spec) back to continuous variable for doing linear regression to calculate the cue weights (or you could just re-read the data in...)  
  
#####step 2##########  
d\_select <- select(ddd, Subject, Block, response=ImageDisplay2.RESP, StimType, StimCategory, StimSubType, duration, spec)  
d\_order <- d\_select[order(d\_select$Subject), ]  
summary(as.factor(d\_order$Subject))

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18 19   
## 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784   
## 20 21 22 23 24   
## 784 784 784 784 784

clear <- ddply(subset(d\_order, StimCategory == "Clear"), ~Subject + Block + duration + spec, summarise, perSAT = sum(response == "m")/length(response))  
  
noise <- ddply(subset(d\_order, StimCategory == "FourChannel"), ~Subject + Block + duration + spec, summarise, perSAT = sum(response == "m")/length(response))  
  
########step 3##########  
clearMatrix <- NULL#####use an empty matrix and fill it in  
#plot the heatmap for the clear first  
for (durValue in 1:7) {  
   
 clearMatrixRow <- c(mean(subset(clear, spec==1 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==2 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==3 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==4 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==5 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==6 & duration==durValue)$perSAT),  
 mean(subset(clear, spec==7 & duration==durValue)$perSAT))  
   
 clearMatrix <- rbind(clearMatrix, clearMatrixRow)  
}  
  
colnames(clearMatrix) <- c("spec1","spec2","spec3","spec4","spec5","spec6","spec7")  
rownames(clearMatrix) <- c("dur1","dur2","dur3","dur4","dur5","dur6","dur7")  
  
clearMatrix <- t(clearMatrix)#this is different from the noisematrix below because it's filled by dur, which is represented by rows. We think representing dur by columns is a better visualization so transpose the matrix  
  
  
##draw the heatmap in Ran's way  
##will have to make the matrix again to accomodate for the heatmap function  
  
clearMatrix <- NULL  
  
for (specValue in 7:1) {  
   
 meanMatrixRow <- c( mean(subset(clear, duration==1 & spec==specValue)$perSAT) ,  
 mean(subset(clear, duration==2 & spec==specValue)$perSAT),  
 mean(subset(clear, duration==3 & spec==specValue)$perSAT),  
 mean(subset(clear, duration==4 & spec==specValue)$perSAT),  
 mean(subset(clear, duration==5 & spec==specValue)$perSAT),  
 mean(subset(clear, duration==6 & spec==specValue)$perSAT),  
 mean(subset(clear, duration==7 & spec==specValue)$perSAT))  
   
 clearMatrix <- rbind(clearMatrix, meanMatrixRow)  
   
}  
  
heatmap.2(clearMatrix, dendrogram="none", Rowv=FALSE, Colv=FALSE, col=rev(heat.colors(256)), keysize=1.3, trace='none', density.info="none", labRow=c("","","","","","",""), labCol=c("","","","","","",""), sepwidth=c(0.005,0.005), margins=c(2,2), sepcolor="Black", colsep=0:ncol(clearMatrix), rowsep=0:nrow(clearMatrix), xlab="DURATION", ylab="SPECTRAL", main = "Dimension weighting in clear speech", key.xlab = NA,  
 key.xtickfun = function() {  
 breaks = pretty(parent.frame()$breaks)  
 breaks = breaks[c(1,length(breaks))]  
 list(at = parent.frame()$scale01(breaks),  
 labels = c("SET", "SAT"))  
 })



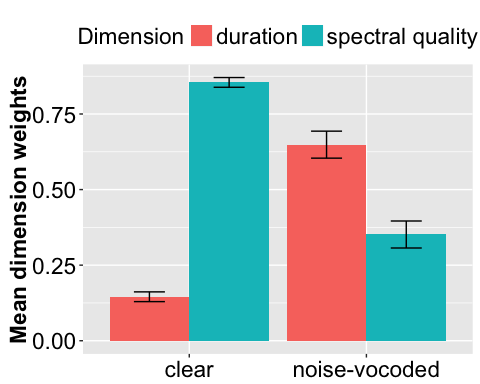
###do the same thing for data in the noise-vocoded condition  
noiseMatrix <- NULL  
  
for (specValue in 7:1) {  
   
 meanMatrixRow <- c( mean(subset(noise, duration==1 & spec==specValue)$perSAT) ,  
 mean(subset(noise, duration==2 & spec==specValue)$perSAT),  
 mean(subset(noise, duration==3 & spec==specValue)$perSAT),  
 mean(subset(noise, duration==4 & spec==specValue)$perSAT),  
 mean(subset(noise, duration==5 & spec==specValue)$perSAT),  
 mean(subset(noise, duration==6 & spec==specValue)$perSAT),  
 mean(subset(noise, duration==7 & spec==specValue)$perSAT))  
   
 noiseMatrix <- rbind(noiseMatrix, meanMatrixRow)  
   
}  
  
heatmap.2(noiseMatrix, dendrogram="none", Rowv=FALSE, Colv=FALSE, col=rev(heat.colors(256)), keysize=1.3, trace='none', density.info="none", labRow=c("","","","","","",""), labCol=c("","","","","","",""), sepwidth=c(0.005,0.005), margins=c(2,2), sepcolor="Black", colsep=0:ncol(clearMatrix), rowsep=0:nrow(clearMatrix), xlab="DURATION", ylab="SPECTRAL", main = "Dimension weighting in noise-vocoded speech", key.xlab = NA,  
 key.xtickfun = function() {  
 breaks = pretty(parent.frame()$breaks)  
 breaks = breaks[c(1,length(breaks))]  
 list(at = parent.frame()$scale01(breaks),  
 labels = c("SET", "SAT"))  
 })



########step 4########  
###IMPORTANT: When you do this analysis, make sure that the variables are not factors and they have to be continuous variables! Otherwise the correlation coefficients will come out wrong!!!!  
###for the cue weights, calculate individual cue weights first  
###and then take the average to represent the overal cue weights  
  
##here I will just re-read the data for simplicity reasons, even though this might not be the smartest idea... but it works :) ###  
rm(list = ls())  
ddd <- read.csv("Data\_final.csv")  
d\_select <- select(ddd, Subject, Block, response=ImageDisplay2.RESP, StimType, StimCategory, StimSubType, duration, spec)  
d\_order <- d\_select[order(d\_select$Subject), ]  
summary(as.factor(d\_order$Subject))

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18 19   
## 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784 784   
## 20 21 22 23 24   
## 784 784 784 784 784

clear <- ddply(subset(d\_order, StimCategory == "Clear"), ~Subject + Block + duration + spec, summarise, perSAT = sum(response == "m")/length(response))  
  
noise <- ddply(subset(d\_order, StimCategory == "FourChannel"), ~Subject + Block + duration + spec, summarise, perSAT = sum(response == "m")/length(response))  
  
clearCWMatrix = NULL  
for (subid in 1:24){  
 if (subid == 15) ##have to skip 15 for now because we don't have the data, only specific to my study though  
 next  
 dur <- subset(clear, Subject == subid)  
 cw <- lm(perSAT~duration + spec, dur)  
 sum <- abs(round ( as.numeric (cw$coefficients[2]), digits = 3)) + abs( round ( as.numeric (cw$coefficients[3]), digits = 3))  
 iddur <- abs(round ( as.numeric (cw$coefficients[2]), digits = 3))/sum  
 idspec <- abs(round ( as.numeric (cw$coefficients[3]), digits = 3))/sum  
 clearCWMatrix <- rbind(clearCWMatrix, c(Subject = subid, iddur, idspec))  
}  
clearCWMatrix <- data.frame (clearCWMatrix)  
colnames(clearCWMatrix) <- c("Subject","cleardurweights", "clearspecweights") ###label the column names of the matrix  
  
noiseCWMatrix = NULL  
for (subid in 1:24){  
 if (subid == 15)  
 next  
 dur <- subset(noise, Subject == subid)  
 cw <- lm(perSAT~duration + spec, dur)  
 sum <- abs(round ( as.numeric (cw$coefficients[2]), digits = 3)) + abs(round ( as.numeric (cw$coefficients[3]), digits = 3))  
 iddur <- abs(round ( as.numeric (cw$coefficients[2]), digits = 3))/sum  
 idspec <-abs(round ( as.numeric (cw$coefficients[3]), digits = 3))/sum  
 noiseCWMatrix <- rbind(noiseCWMatrix, c(Subject = subid, iddur, idspec))  
}  
noiseCWMatrix <- data.frame (noiseCWMatrix)  
colnames(noiseCWMatrix) <- c("Subject","noisedurweights", "noisespecweights") ###label the column names  
  
##plot histogram for the average cue weights with the error bars indicating standard error  
cwmat <- melt(join(clearCWMatrix, noiseCWMatrix, by = "Subject"), id.vars = "Subject") #melt the cue weights matrix by subject  
cwmat[, 4] <- rep(c("clear", "noise-vocoded"), c(length(clearCWMatrix$Subject)\*2, length(clearCWMatrix$Subject)\*2)) ##since the dimension of the matrix changes, add the labels to these extra dimensions  
cwmat[, 5] <- rep(c("duration", "spectral quality"), c(length(clearCWMatrix$Subject), length(clearCWMatrix$Subject)))  
colnames(cwmat) <- c("Subject", "whole", "value", "condition", "Dimension")  
sum <- ddply(cwmat, ~condition + Dimension, summarise, mean = mean(value), sd = sd(value), n = length(Subject), se = sd/sqrt(n)) ##create the 'sum' matrix to be plotted  
  
a = 17  
ggplot(sum, aes(x = condition, y = mean, fill = Dimension)) +  
 geom\_bar(stat = "identity", position = position\_dodge()) +  
 geom\_errorbar(aes(ymin = mean-se, ymax = mean + se), pos = position\_dodge(0.9), width = 0.35) +  
 labs(x = NULL, y = "Mean dimension weights") +  
 theme(legend.text=element\_text(size=a), legend.title=element\_text(size = a), axis.title=element\_text(size=a, face = "bold"), axis.text = element\_text(size = a, color = "Black"), legend.position = "top") ###use ggplot here to visualize the data



#########step 5#############  
noise <- ddply (subset(d\_order, StimCategory == "FourChannel"), ~Subject + duration + spec, summarise, perSAT = sum(response == "m")/length(response))  
noiselm <- lm(perSAT ~ duration + spec, noise)  
noise\_corr\_DurPre = round( as.numeric(noiselm$coefficients[2]) , digits=3)  
noise\_corr\_SpecPre = round( as.numeric(noiselm$coefficients[3]) , digits=3)  
noise\_SUM\_Pre = noise\_corr\_DurPre + noise\_corr\_SpecPre  
noise\_corr\_DurPre\_NORM = round( noise\_corr\_DurPre / noise\_SUM\_Pre , digits=3)  
noise\_corr\_SpecPre\_NORM = round( noise\_corr\_SpecPre / noise\_SUM\_Pre , digits=3)  
  
clearlm <- lm(perSAT ~ duration + spec, clear)  
clear\_corr\_DurPre = round( as.numeric(clearlm$coefficients[2]) , digits=3)  
clear\_corr\_SpecPre = round( as.numeric(clearlm$coefficients[3]) , digits=3)  
clear\_SUM\_Pre = clear\_corr\_DurPre + clear\_corr\_SpecPre  
clear\_corr\_DurPre\_NORM = round( clear\_corr\_DurPre / clear\_SUM\_Pre , digits=3)  
clear\_corr\_SpecPre\_NORM = round( clear\_corr\_SpecPre / clear\_SUM\_Pre , digits=3)  
summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00