data\_orgmarkdown

R uses <- for variable assignment. The variable "subjects" is being assigned a list of numbers (vector) from 501 to 551. The variable "data\_path" is being assigned the location of the data set. Then, setwd is used to set the working directory.

subjects <- c(501:551)  
data\_path <- "/Volumes/home/RAs/Azara/R\_dataorg/data/"  
setwd(data\_path)

In R, functions come as part of a package, so you must make sure the necessary packages are installed. The # before the function, in this case, renders it a comment rather than a piece of code. This is because you don't want to install packages multiple times, but it is here for demonstrative purposes. If you don't know what packages you need, Google it! In the following example 'plyr' is the name of the package:

#install.packages('plyr')  
#install.packages('plotrix')

Use "library" to specify which packages you are using. This will direct R to the functions you want to use.

library(plyr)

## Warning: package 'plyr' was built under R version 3.2.5

library(reshape)

##   
## Attaching package: 'reshape'

## The following objects are masked from 'package:plyr':  
##   
## rename, round\_any

library(reshape2)

##   
## Attaching package: 'reshape2'

## The following objects are masked from 'package:reshape':  
##   
## colsplit, melt, recast

library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:reshape2':  
##   
## dcast, melt

## The following object is masked from 'package:reshape':  
##   
## melt

library(plotrix)

## Warning: package 'plotrix' was built under R version 3.2.5

recog\_file="/recog.csv"

This is a for statement that runs 51 times (once for each subject). The variable "sub\_num" is the subject number. The data set we are considering has quite a few empty spaces that we do not want to include in our analysis. We use the function "paste0" to concatenate everything without any spaces in between, and we assign that to the variable "path". We then assign a variable "recog" to read the .csv file into R. "recog$X <- NULL" deletes the extra column. The if statement says that if the loop is on its first run, put the output from the for statement into the variable "recog\_all". If the loop is on any other run, bind the subsequent data frames by rows. Note: if you wanted to bind the data frames by columns, you would use "cbind" instead of "rbind".

for (sub in 1:length(subjects)) {  
 sub\_num <- subjects[sub]  
 path <- paste0(data\_path,sub\_num,recog\_file)  
 recog <- read.csv(path)  
 recog$X <- NULL # Remove mystery column "X"  
 if (sub==1){  
 recog\_all <- recog  
 } else {  
 recog\_all <- rbind(recog\_all,recog)  
 }  
}

The "subset" function is used to select the data in the specified columns and place them in the recog\_all variable. In the other option,"which" is used to select observations in recog\_all.

# if you want a bunch of columns and you know the column headers, then you can do it as:  
recog\_all <- subset(recog\_all,select=-c(preTrial\_recog\_acc,preTrial\_source\_acc,preStimTrial\_recog\_acc,preStimTrial\_source\_acc))  
 # OR  
recog\_all[,-which(names(recog\_all) %in% c("preTrial\_recog\_acc","preTrial\_source\_acc","preStimTrial\_recog\_acc","preStimTrial\_source\_acc"))]

## data frame with 0 columns and 19992 rows

Straightforward way to delete columns:

# if you want to delete a bunch of columns, and you know the column numbers, then you can do it as:  
recog\_all <- recog\_all[-c(19:22)] # if you want to delete columns 19 through 22.

Creating a new column called "recog\_all$delay" for the inter-stimulus interval; the 1\* turns false to zero and true to one.

# Add a new column for ISI  
recog\_all$delay <- 1\*(recog\_all$iti==4) # short ISI = 0; long ISI = 1

Another way to add a new column where responses are either zero or one (new or old).

# Add a new column for old/new judgments (whether subjects judged the stim as old or new)  
recog\_all$OldResp[recog\_all$recog\_resp\_bin==1] <- 0  
recog\_all$OldResp[recog\_all$recog\_resp\_bin>1] <- 1  
# ^^ can also be written as: recog\_all$OldResp <- 1\*(recog\_all$recog\_resp\_bin>1)

# convert old\_num values into 1's and 0's (currently, old=1 & new=2)  
recog\_all$old\_num <- 1\*(recog\_all$old\_num==1) # now, old=1 & new=0

In data set, we had "nan" when a participant missed a response; we do not want these to be counted in our analysis, so we have to make them "NA" instead.

# correcting psychopy errors\* (actually, my errors in psychopy!)  
recog\_all$recog\_resp[recog\_all$recog\_resp=='nan'] <- NA # psychopy was treating 'nan' as a text!  
recog\_all$recog\_resp\_bin[is.na(recog\_all$recog\_resp)] <- NA # psychopy had coded the response bins for non-responses as 0 instead of NA

# changing a bunch of columns into factors  
names <- c('sub\_num','old\_num','trial\_num','block\_num','stim\_num','word\_num','enc1\_trial','preWord','iti','next\_iti','delay')  
recog\_all[,names] <- lapply(recog\_all[,names],factor)  
# recog\_all$sub\_num <- as.factor(recog\_all$sub\_num)  
  
# if you want to rename a column name  
names(recog\_all)[names(recog\_all)=="enc1\_trial"] <- 'enc\_trial'  
# if you know the column number, ^ can be easily done as:  
names(recog\_all)[9] <- 'enc\_trial'

# rename function  
setnames(recog\_all,old=c("iti","next\_iti"),new=c("isi","next\_isi")) # data.table library  
# OR  
stats <- rename(recog\_all,replace=c("iti"="isi","next\_iti"="next\_isi")) # plyr library

# if you want to centralize reaction time  
recog\_all <- ddply(recog\_all, c("sub\_num"), transform, RT\_cent = scale(recog\_RT))

#### OVERALL ####  
rt <- cast(recog\_all,value="recog\_RT", sub\_num~old\_num, mean,na.rm=T)  
rt\_stim <- cast(recog\_all,value="recog\_RT", sub\_num~stim\_num, mean,na.rm=T, margins=T)

# dprime, criteria, hits, FA  
stats <- cast(recog\_all,value="OldResp", sub\_num~old\_num, mean,na.rm=T)  
names(stats) <- c('subject','FA','hits')

# Changing FA for rows which contain 0 FA to 1/197 (expecting the next (197th) trial to be a false alarm)  
no\_FA <- which(stats$FA==0)  
for (i in no\_FA) stats[i,2] <- 1/197

# Changing FA for rows which contain 0 FA to 1/197 (expecting the next (197th) trial to be a false alarm)  
no\_FA <- which(stats$FA==0)  
for (i in no\_FA) stats[i,2] <- 1/197

# Changing hits for rows which contain 1 hit to 196/197 (expecting the next (197th) trial to be a miss)  
perfect\_hits <- which(stats$hits==1)  
for (i in perfect\_hits) stats[i,3] <- 196/197

mean(stats$dprime)

## Warning in mean.default(stats$dprime): argument is not numeric or logical:  
## returning NA

## [1] NA

std.error(stats$dprime)

## Warning in is.na(x): is.na() applied to non-(list or vector) of type 'NULL'

## [1] NA

mean(stats$criteria)

## Warning in mean.default(stats$criteria): argument is not numeric or  
## logical: returning NA

## [1] NA

std.error(stats$criteria)

## Warning in is.na(x): is.na() applied to non-(list or vector) of type 'NULL'

## [1] NA

mean(stats$hits)

## [1] 0.905558

std.error(stats$hits)

## [1] 0.01039161

mean(stats$FA)

## [1] 0.07508273

std.error(stats$FA)

## [1] 0.01740739