# Analyze burst spectra

*Emily Cibelli (emily.cibelli@gmail.com)*

*August 8, 2018*

## Introduction and setup

The context for this analysis is reported in: Cibelli (2015). Aspects of articulatory and perceptual cues in novel phoneme acquisition. UC Berkeley doctoral dissertation.

The goal of this analysis is to analyze place of articulation cues in the production of Hindi stop consonants by native English speakers by loooking at spectral information in stop bursts. Do speakers learn to produce the dental/retroflex contrast?

Input: a data file containing spectral information from the midpoint of initial stop bursts produced by learners at four sessions in a training study. (The data was extracted with Keith Johnson's script get_burst_spectra.prl.)

### Workspace setup

```
library(lme4)          # run mixed effects models
```

```
## Loading required package: Matrix
```

```
library(RePsychLing)  # evaluate zero-variance random effects, Bates et al. (2015)
library(lattice)      # basic plotting package for by-group comparisons
library(ggplot2)      # more flexible plotting package
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(gridExtra)    # allows layout of multiple ggplots on one image
library(MASS)         # lda

workingDir = "C:/Users/esc642/Box/dissertation/productionPaper/productionData"
setwd(workingDir)
```

### Lattice plot parameters

```
font.settings <- list( font = 1, cex =1.1) # , #fontfamily = "sansserif")
my.theme <- list(
  box.umbrella = list(col = c("black"), lwd =2),
  box.rectangle = list(col = c("black"), lwd = 2), #fill = c("blue", "red")),
  box.dot = list(col = c("black"), pch = 16, cex=1),
  plot.symbol   = list(col = 1, pch= 1, cex = 1), #outlier size and color
  par.xlab.text = font.settings,
  par.ylab.text = font.settings,
  par.sub.text = font.settings,
  axis.text = font.settings,
  #strip.shingle=list(col=c("black","white")),
  superpose.symbol=list(fill=c("gray","white")), # boxplots
  #superpose.fill=list(col=c("black","white")),
```

```
  superpose.polygon=list(col=c("gray","white")), # legend
  strip.background = list(col = c("white")),
  axis.line = list(lwd = 2),
  strip.border = list(lwd = 2),
  layout.heights=list(strip=2)
  #layout.heights = list(right.padding = -20)
)
```

### Multiplot function

For plotting multiple ggplots **saved to a list** in a single image.

Source: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/

```
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))
```

```
      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

## Read in data

One data file exists per session (pre-test, post-test, production training, post-prod - reported in paper as re-test).

### Description of data structure

- **stimulus:** one of 96 unique stimuli
- **subj:** participant number
- **center:** ? not used here
- **ampratio:** amplitude ratio
- **Hzpeak:** dB diff between the amplitude of most prominent peak and the F2 amplitude
- **centroid:** centroid frequency/center of gravity (1st spectram moment)
- **sd:** standard deviation (2nd spectral moment)
- **skew:** skewness, asymmetry of the spectrum (3rd spectral moment); 0 = equal distrib, pos skew = low-freq shifted; neg skew = high-freq shifted
- **kurtosis:** "peakedness" of the distribution; bigger = sharper/more peaks (4th s.m.)
- **barkpeak - barkkurtosis:** bark-normalized versions of the above
- **dur:** duration, in seconds
- **session:** pre-test, post-test, prod. training, post-prod/re-test

```
# pre-test: prior to any training
preTestDF = read.delim(sprintf("%s/burst_spectrum_preTest.txt", workingDir), head = T)
preTestDF$session = "preTest"

# post-test: after peception training
postTestDF = read.delim(sprintf("%s/burst_spectrum_postTest.txt", workingDir), head = T)
postTestDF$session = "postTest"

# production training: during articulatory training
prodTrainingDF = read.delim(sprintf("%s/burst_spectrum_prodTrain.txt", workingDir), head = T)
prodTrainingDF$session = "prodTraining"

# post-prod/re-test: after production training
postProdDF = read.delim(sprintf("%s/burst_spectrum_postProd.txt", workingDir), head = T)
postProdDF$session = "postProd"

# combine sessions
df = rbind(preTestDF, postTestDF, prodTrainingDF, postProdDF)
df$session = as.factor(as.character(df$session))
df$session = factor(df$session, levels = c("preTest", "postTest",
                                           "prodTraining", "postProd"))

# data structure
head(df)
```

```
##   stimulus subj    center  ampratio   Hzpeak centroid      sd  skew
```

```
## 1  da4_new  128 1.5477924 -9.570408 2468.750 4834.311 1475.09 -0.29
## 2     dha5  128 0.8196439 -8.033277 2234.375 4541.573 1499.73 -0.05
## 3      ti9  126 0.8202385  5.358996 2468.750 4559.803 1466.61  0.03
## 4     rtu9  124 0.6429103 -3.246865 2312.500 4485.377 1506.85  0.08
## 5     rda9  106 1.3247722 -6.828682 2906.250 4446.064 1430.13  0.12
## 6     dhu2  119 0.5164527  4.376290 2562.500 4578.806 1542.57  0.04
##   kurtosis barkpeak barkcentroid barksd barkskew barkkurtosis   dur
## 1     1.94    14.31       18.521  2.353   -0.658        2.260 0.011
## 2     1.91    13.64       18.035  2.464   -0.423        1.980 0.011
## 3     1.93    14.31       18.090  2.358   -0.354        2.030 0.004
## 4     1.86    13.87       17.948  2.447   -0.274        1.885 0.011
## 5     2.00    15.42       17.929  2.333   -0.286        2.017 0.006
## 6     1.76    14.56       18.088  2.470   -0.297        1.855 0.016
##   session
## 1 preTest
## 2 preTest
## 3 preTest
## 4 preTest
## 5 preTest
## 6 preTest
```

## Clean up data, simple feature extraction

Add information about stimuli (vowels, consonants, voicing, place of articulation.) Drop one speaker who was missing one session's data.

```
# Subject 124 has post-test production data missing, so drop their data
df = df[df$subj != 124,]

# Add stimulus information columns:

# pull out vowels (a, i, u)
df$vowel = ""
df[grepl("i", df$stimulus, perl = TRUE),]$vowel = "i"
df[grepl("a", df$stimulus, perl = TRUE),]$vowel = "a"
df[grepl("u", df$stimulus, perl = TRUE),]$vowel = "u"
df$vowel = as.factor(as.character(df$vowel))

# pull out consonants (d, t, dh, th, rd, rt, rth, rdh)
df$consonant = gsub("[aiu1234567890_newRPNC]", "", df$stimulus)
df$consonant = as.factor(as.character(df$consonant))

# place of articulation (dental or retroflex)
df$place = "dental"
df[grepl("r", df$stimulus, perl = TRUE),]$place = "retroflex"
df$place = as.factor(as.character(df$place))

# voicing
df$voicing = ""
df[grepl("dh", df$stimulus, perl = TRUE),]$voicing = "breathy"
df[grepl("th", df$stimulus, perl = TRUE),]$voicing = "aspirated"
df[grepl("du", df$stimulus, perl = TRUE),]$voicing = "voiced"
df[grepl("di", df$stimulus, perl = TRUE),]$voicing = "voiced"
df[grepl("da", df$stimulus, perl = TRUE),]$voicing = "voiced"
```

```
df[grepl("tu", df$stimulus, perl = TRUE),]$voicing = "voiceless"
df[grepl("ti", df$stimulus, perl = TRUE),]$voicing = "voiceless"
df[grepl("ta", df$stimulus, perl = TRUE),]$voicing = "voiceless"
df$voicing = as.factor(as.character(df$voicing))
```

## Outlier removal

Define outliers here as having an extreme value on any one of the four spectral moments (centroid, sd, skew, or kurtosis).

Also drop any values with a duration > 50 ms (these are likely transcription errors).

```
# Find extreme centroid, skew, sd, or kurtosis values (z > 3)
dfSaver = df
df$centroidZ = scale(df$centroid)
df$skewZ = scale(df$skew)
df$sdZ = scale(df$sd)
df$kurtosisZ = scale(df$kurtosis)

# Combine stimuli and subject to ID unique productions
df$stimSubj = as.character(interaction(df$stimulus, df$subj, sep = "."))

# How many extreme values?
extremeCentroid =df[abs(df$centroidZ)>3,]$stimSubj
length(extremeCentroid)
```

```
## [1] 13
```

```
extremeSD = df[abs(df$sdZ)>3,]$stimSubj
length(extremeSD)
```

```
## [1] 53
```

```
extremeSkew =df[abs(df$skewZ)>3,]$stimSubj
length(extremeSkew)
```

```
## [1] 20
```

```
extremeKurtosis =df[abs(df$kurtosisZ)>3,]$stimSubj
length(extremeKurtosis)
```

```
## [1] 65
```

```
# Find all unique productions with at least one extreme value
allExtreme = unique(c(extremeCentroid, extremeSD, extremeSkew, extremeKurtosis))
length(allExtreme)
```

```
## [1] 95
```

```
# Remove points with extreme values
df = df[!(df$stimSubj %in% allExtreme),]

# How much of the data is retained?
nrow(df)/nrow(dfSaver)
```

```
## [1] 0.9474221
```

```r
# cut durations > 50 ms
dfSaver2 = df
df = df[df$dur < 0.051,]
nrow(df)/nrow(dfSaver2)
```

```
## [1] 0.9943121
```

```r
# Total number of data points retained
nrow(df)
```

```
## [1] 6468
```

## Read in and add background data

Data was collected on proficiency in second (L2), third (L3), etc. languages. Scores are based on a 4-point rating skill for 4 skills (reading, writing, speaking, and understanding.)

```r
bkgdData = read.csv(sprintf("%s/backgroundData.csv", workingDir), head = T)

# Restrict to subjects in the long study (subject number in the 100s)
bkgdData = bkgdData[bkgdData$subjNum %in% unique(df$subj),]

# transform NA values for L2 or L3 to 0
# (only 1 subject reported an L4 or L5, so we'll ignore those columns)
otherLangMetrics = c("speakL2", "readL2", "writeL2",
                     "understandL2", "speakL3", "readL3",
                     "writeL3", "understandL3")
for (i in otherLangMetrics) {
    for (j in 1:nrow(bkgdData)) {
        if (is.na(bkgdData[j,i]))
            {bkgdData[j,i] = 0 }
                }}

# Calculate L2 average
bkgdData$L2average = 0
for (i in 1:nrow(bkgdData)) {
    bkgdData$L2average[i] = mean(c(bkgdData$readL2[i], bkgdData$writeL2[i],
                                   bkgdData$speakL2[i], bkgdData$understandL2[i])) }

# Calculate L3 average
bkgdData$L3average = 0
for (i in 1:nrow(bkgdData)) {
    bkgdData$L3average[i] = mean(c(bkgdData$readL3[i], bkgdData$writeL3[i], bkgdData$speakL3[i], bkgdDa

# what L2s and L3s are in the dataset?
table(factor(bkgdData[!(is.na(bkgdData$L2)),]$L2))
```

```
##
##       ASL Cantonese   Chinese    French     Latin  Mandarin   Spanish
##         1         1         1         2         2         2         7
```

```r
table(factor(bkgdData[!(is.na(bkgdData$L3)),]$L3))
```

```
##
##   French   German Mandarin  Russian  Spanish
```

```
##        2        2        1        2        2
```

```
# Merge with main data set
L2L3av = bkgdData[,c("subjNum", "L2average", "L3average")]
df = merge(df, L2L3av, by.x = "subj", by.y = "subjNum")
```

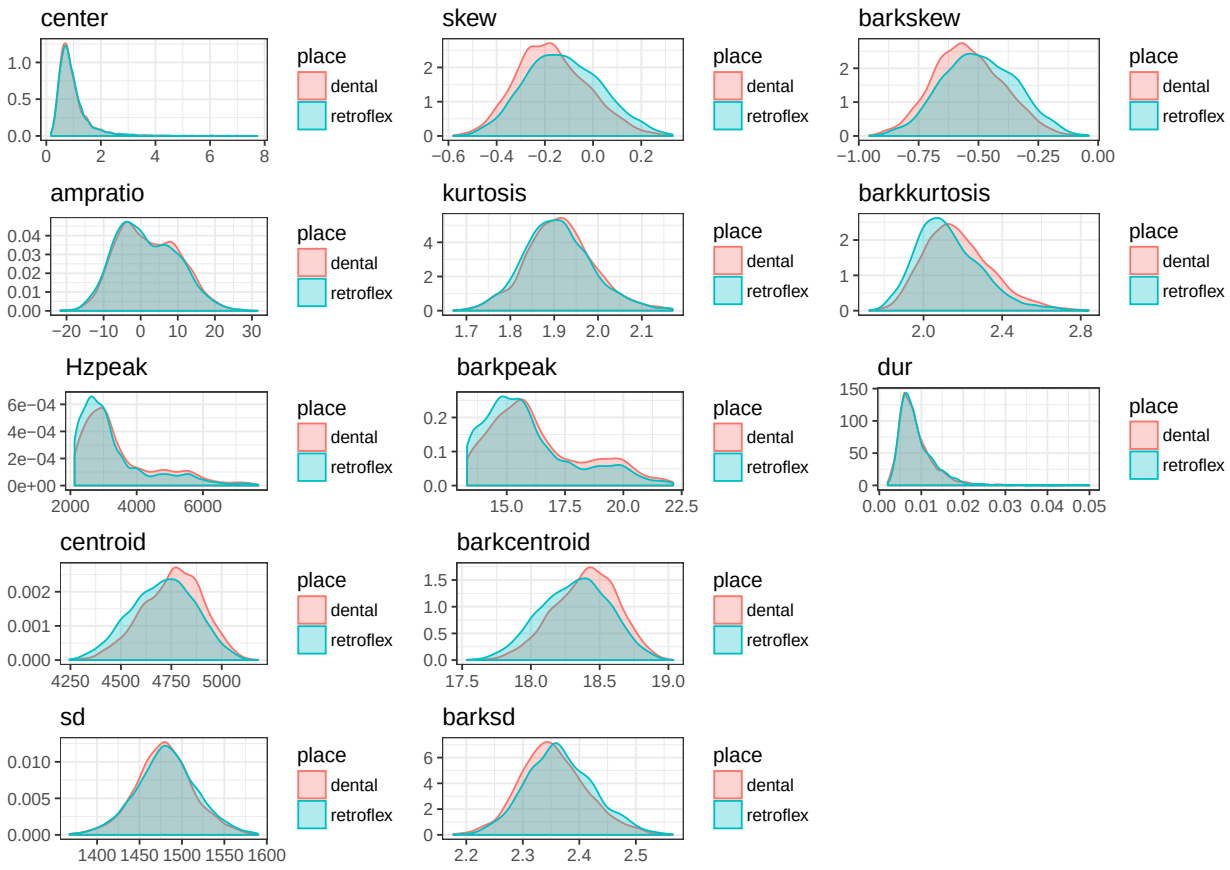
## Plot differences of each metric

Compare dental and retroflex consonants on several measures. What metrics seem to distinguish the two places of articulation?

```
# Some set-up

metrics = c("center", "ampratio", "Hzpeak", "centroid", "sd", "skew",
            "kurtosis", "barkpeak", "barkcentroid", "barksd", "barkskew",
            "barkkurtosis", "dur")

boxLab = as.factor(c("dental", "retroflex"))

metricsClean = c("Center", "Amplitude ratio", "Hz peak", "Centroid",
                 "Standard deviation", "Skewness", "Kurtosis",
                    "Bark peak", "Bark centroid", "Bark standard deviation",
                 "Bark skewness", "Bark kurtosis", "Duration")
metrics2 = as.data.frame(cbind(metrics, metricsClean))
```

### Plot 1: collapsed across style

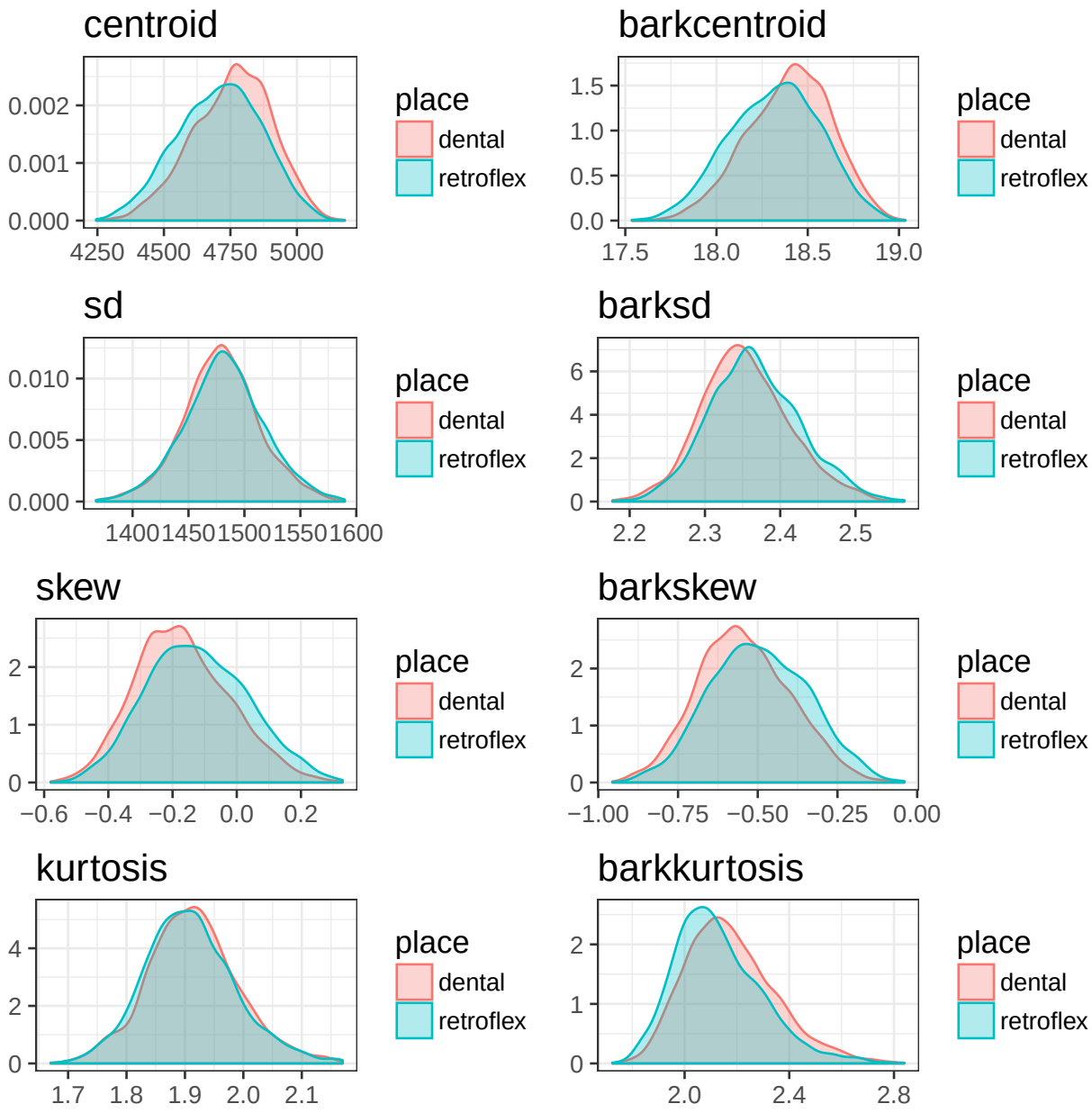
```
plotList1 = list()
for (i in metrics) {
  metricOrder = which(metrics == i)
  plotList1[[metricOrder]] = ggplot(df, aes_string(x = i)) +
    geom_density(aes(color = place, fill = place, group = place), alpha = 0.3) +
    theme_bw(base_size = 12) +
    ggtitle(i) +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank())
}

multiplot(plotlist = plotList1, cols = 3)
```




**Plot 2: collapsed across style, just the metrics reported in Forrest et al. 1988**

Forrest, K., Weismer, G., Milenkovic, P., & Dougall, R. N. (1988). Statistical analysis of word-initial voiceless obstruents: preliminary data. The Journal of the Acoustical Society of America, 84(1), 115-123.

```
plotList2 = list()
metricsForrest = metrics[c(4:7, 9:12)]

for (i in metricsForrest) {
  metricOrder = which(metricsForrest == i)
  plotList2[[metricOrder]] = ggplot(df, aes_string(x = i)) +
    geom_density(aes(color = place, fill = place, group = place), alpha = 0.3) +
    theme_bw(base_size = 14) +
    ggtitle(i) +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank())
}

multiplot(plotlist = plotList2, cols = 2)
```

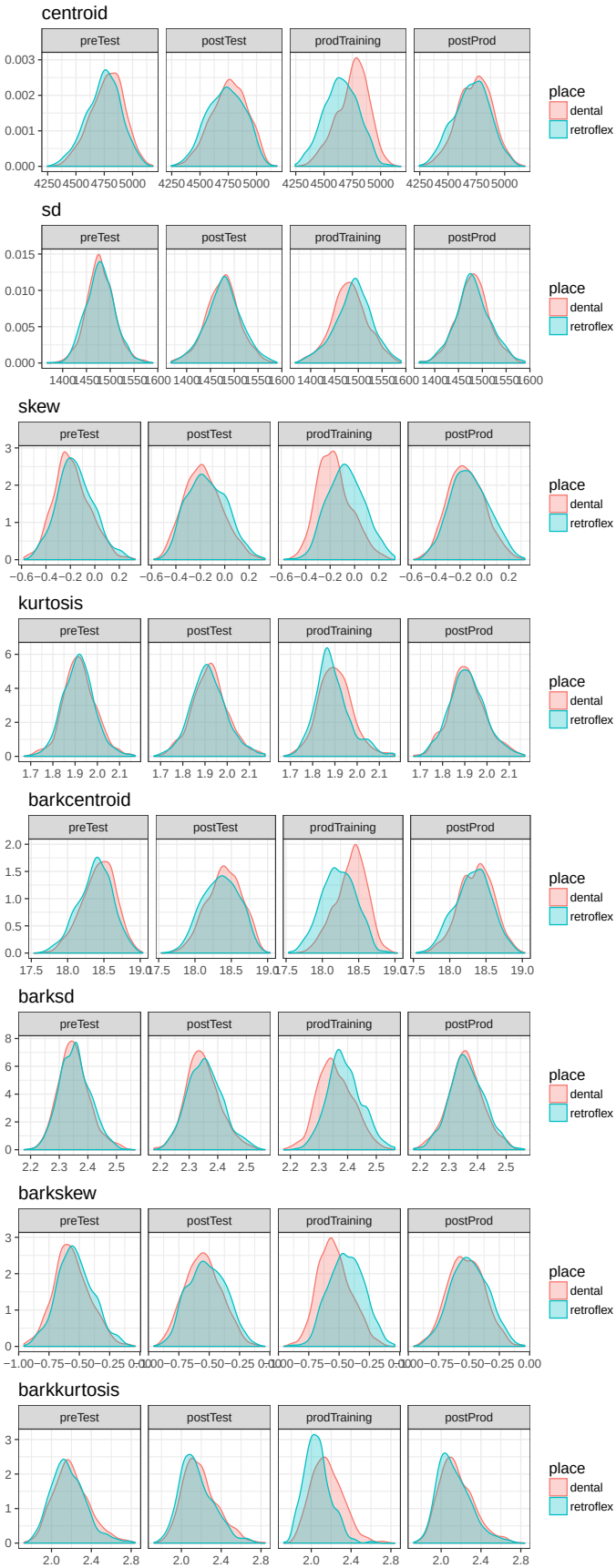



**Plot 3: By-session panels, Forrest metrics**

Compare performance in each session; do dentral and retroflex tokens become more distinguished in later sessions?

```
plotList3 = list()

for (i in metricsForrest) {
  metricOrder = which(metricsForrest == i)
  plotList3[[metricOrder]] = ggplot(df, aes_string(x = i)) +
    geom_density(aes(color = place, fill = place, group = place), alpha = 0.3) +
    facet_wrap(~session, ncol = 4) +
    theme_bw(base_size = 14) +
    ggtitle(i) +
```

```
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank())
}

multiplot(plotlist = plotList3, cols = 1)
```


centroid
preTest
postTest
prodTraining
postProd
0.003
0.002
0.001
0.000
4250 4500 4750 5000
place
dental
retroflex
sd
0.015
0.010
0.005
0.000
1400 1450 1500 1550 1600
skew
3
2
1
0
−0.6 −0.4 −0.2 0.0 0.2
kurtosis
6
4
2
0
1.7 1.8 1.9 2.0 2.1
barkcentroid
2.0
1.5
1.0
0.5
0.0
17.5 18.0 18.5 19.0
barksd
8
6
4
2
0
2.2 2.3 2.4 2.5
barkskew
−1.00 −0.75 −0.50 −0.25 0.00
barkkurtosis
2.0 2.4 2.8

## Summary of modeling strategy ————————————————-

**Step 1:** Use logistic regression to identify the most reliable/significant predictors of place of articulation.

**Step 2:** Run LDA classification based only on reliable predictors.

**Step 3:** Use mixed-effects logistic regression (GLMM) to compare classification across sessions. (Does classification accuracy improve/decline between sessions?)

**Step 4:** Examine classification performance by-subject.

## Step 1: Logistic regression to determine best predictors ————————-

```
# Center predictor variables
df$centroidC = scale(df$centroid, center = T)
df$sdC = scale(df$sd, center = T)
df$HzpeakC = scale(df$Hzpeak, center = T)
df$skewC = scale(df$skew, center = T)
df$kurtosisC = scale(df$kurtosis, center = T)
df$ampratioC = scale(df$ampratio, center = T)
df$barksdC = scale(df$barksd, center = T)
df$barkskewC = scale(df$barkskew, center = T)
df$barkkurtosisC = scale(df$barkkurtosis, center = T)
df$barkcentroidC = scale(df$barkcentroid, center = T)
df$L2averageC = scale(df$L2average, center = T)

# Model place to find optimal combination of predictors.
# remove predictors that do not significantly contribute to model fit
LDA.glm = glm(place ~ centroidC + sdC + skewC + kurtosisC + HzpeakC + ampratioC,
              data = df, family = "binomial")
summary(LDA.glm)
```

```
##
## Call:
## glm(formula = place ~ centroidC + sdC + skewC + kurtosisC + HzpeakC +
##     ampratioC, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7576  -1.1154  -0.9174   1.1854   1.5761
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.0267892  0.0252949  -1.059    0.290
## centroidC   -0.5442920  0.1268972  -4.289 1.79e-05 ***
## sdC          0.3494039  0.0512891   6.812 9.60e-12 ***
## skewC       -0.1414772  0.1221624  -1.158    0.247
## kurtosisC    0.2642565  0.0529675   4.989 6.07e-07 ***
## HzpeakC     -0.0004857  0.0337046  -0.014    0.989
## ampratioC    0.0298589  0.0268602   1.112    0.266
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 8965.2  on 6467  degrees of freedom
## Residual deviance: 8752.8  on 6461  degrees of freedom
## AIC: 8766.8
##
## Number of Fisher Scoring iterations: 4
```

```
# remove HzpeakC (Smallest estimate, biggest p-value)
LDA.glm2 = glm(place ~ centroidC + sdC + skewC + kurtosisC  + ampratioC,
                   data = df, family = "binomial")
summary(LDA.glm2)
```

```
##
## Call:
## glm(formula = place ~ centroidC + sdC + skewC + kurtosisC + ampratioC,
##     family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7576  -1.1153  -0.9173   1.1854   1.5764
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02679    0.02529  -1.059    0.290
## centroidC   -0.54471    0.12348  -4.411 1.03e-05 ***
## sdC          0.34941    0.05129   6.813 9.55e-12 ***
## skewC       -0.14165    0.12157  -1.165    0.244
## kurtosisC    0.26409    0.05172   5.106 3.29e-07 ***
## ampratioC    0.02985    0.02686   1.112    0.266
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8965.2  on 6467  degrees of freedom
## Residual deviance: 8752.8  on 6462  degrees of freedom
## AIC: 8764.8
##
## Number of Fisher Scoring iterations: 4
```

```
# Remove skewC
LDA.glm3 = glm(place ~ centroidC + sdC + kurtosisC  + ampratioC,
                   data = df, family = "binomial")
summary(LDA.glm3)
```

```
##
## Call:
## glm(formula = place ~ centroidC + sdC + kurtosisC + ampratioC,
##     family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7651  -1.1120  -0.9256   1.1856   1.5568
##
## Coefficients:
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02657    0.02529  -1.050    0.293
## centroidC   -0.40518    0.02975 -13.618  < 2e-16 ***
## sdC          0.34472    0.05116   6.738 1.61e-11 ***
## kurtosisC    0.26268    0.05177   5.074 3.90e-07 ***
## ampratioC    0.02856    0.02683   1.064    0.287
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8965.2  on 6467  degrees of freedom
## Residual deviance: 8754.2  on 6463  degrees of freedom
## AIC: 8764.2
##
## Number of Fisher Scoring iterations: 4
```

```
# Remove ampratioC
LDA.glm4 = glm(place ~ centroidC + sdC + kurtosisC,
                   data = df, family = "binomial")
summary(LDA.glm4)
```

```
##
## Call:
## glm(formula = place ~ centroidC + sdC + kurtosisC, family = "binomial",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7589  -1.1160  -0.9182   1.1851   1.5776
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02679    0.02529  -1.059     0.29
## centroidC   -0.39792    0.02893 -13.754  < 2e-16 ***
## sdC          0.34737    0.05109   6.799 1.05e-11 ***
## kurtosisC    0.26917    0.05141   5.236 1.64e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8965.2  on 6467  degrees of freedom
## Residual deviance: 8755.3  on 6464  degrees of freedom
## AIC: 8763.3
##
## Number of Fisher Scoring iterations: 4
```

```
# Final model: place ~ centroid + sd + kurtosis
# Similar to Forrest et al. (1988), but with centroid instead of skew
```

## Step 2A: LDA classification across all sessions ————————————-

**Fit a model to data across all sessions**

```r
# Fit LDA model
ldaFit = lda(place ~ sdC + centroidC + kurtosisC, data = df, CV = T, na.action = "na.omit")

# Constuct a table to compare classification to true place of articulation labels
ldaTab = table(df$place, ldaFit$class)

# Accuracy by place
diag(prop.table(ldaTab, 1))
```

```
##    dental retroflex
## 0.6723560 0.4759962
```

```r
# Overall accuracy
sum(diag(prop.table(ldaTab))) # Overall accuracy
```

```
## [1] 0.575603
```

### Permutation test to compare accuracy to chance

Logic: would performance at this accuracy level be expected by chance? Test by scrambling labels and data and re-running classification over the permuted data

**Rough model for this code:** http://mukatira.net/4436_HOME/BioStat/www.bigre.ulb.ac.be/Users/jvanheld/statistics_bioinformatics/R-files/LDA_permutation_tests.R

**See also:** Combrisson, E., & Jerbi, K. (2015). Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy. *Journal of neuroscience methods*, 250, 126-136.

```r
# permute the data 1000 times
nrep = 1000

# create a vector to hold each permutation's classification accuracy
permHolderAll = rep(0, nrep)

# for each permutation:
for (i in 1:nrep) {
  # print progress every 50 perumtations - useful when interactive, suppress for markdown
  #if (i %% 50 == 0) {
  #  timeKeeper = sprintf("Rep %s", i)
  #  print(timeKeeper)}
  # create a data frame with just the data values (no labels)
  dfPermut = df[,c("sdC", "centroidC", "kurtosisC")]
  # scramble the place of articulation labels
  randomSet = sample(df$place)
  # attach the permuted labels to the new data frame
  dfPermut$randomSet = randomSet
  # run classification on the permuted data set
  random.lda = lda(randomSet ~ sdC + centroidC + kurtosisC,
                   data = dfPermut, CV = T, na.action = "na.omit")
  # table comparing true labels to classification of permuted labels
  randomTab = table(df$place, random.lda$class)
  # how good was classification of the permuted data? save to holder
  permHolderAll[i] = sum(diag(prop.table(randomTab)))
}
```

**Calculate a p-value of sorts:** How many permuted classifications were equal to or more accurate than the real test? If fewer than 5%, we can say that the original classification was reliable

```
length(permHolderAll[permHolderAll >= sum(diag(prop.table(ldaTab)))])/nrep
```

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

## Step 2B: LDA classification by session ————————-

Run the same LDA analysis as in step 2A, but by session. This will demonstrate the relative ability to classify tokens into dental and retroflex at each stage in the training study.

```
# specify the number of repetitions to run for each permutation test
nrep = 1000
```

### Pre-test

```
ldaFitpreTest = lda(place ~ centroidC + sdC + kurtosisC,
                    data = df[df$session == "preTest",], CV = T, na.action = "na.omit")
ldaFitpreTestTab = table(df[df$session == "preTest",]$place, ldaFitpreTest$class)
diag(prop.table(ldaFitpreTestTab, 1)) # By-PoA accuracy
```

```
##    dental retroflex
## 0.6548348 0.4166667
```

```
sum(diag(prop.table(ldaFitpreTestTab))) # Overall accuracy
```

```
## [1] 0.5385097
```

```
# Extract class from ldaFit for cross-session comparison (Step 3)
ldaClassPreTest = ldaFitpreTest$class
```

```
# Permutation test
permHolderpreTest = rep(0, nrep)
for (i in 1:nrep) {
  #if (i %% 50 == 0) {
  #  timeKeeper = sprintf("Rep %s", i)
  #  print(timeKeeper)}
  dfPermut = df[df$session == "preTest",c("sdC", "centroidC", "kurtosisC")]
  randomSet = sample(df[df$session == "preTest",]$place)
  dfPermut$randomSet = randomSet
  random.lda.preTest = lda(randomSet ~ sdC + centroidC + kurtosisC,
                    data = dfPermut, CV = T, na.action = "na.omit")
  randomTabpreTest = table(df[df$session == "preTest",]$place, random.lda.preTest$class)
  permHolderpreTest[i] = sum(diag(prop.table(randomTabpreTest)))
}

# p-value:
length(permHolderpreTest[permHolderpreTest >=
                           sum(diag(prop.table(ldaFitpreTestTab)))])/nrep
```

```
## [1] 0.034
```

### Post-test

```
ldaFitPostTest = lda(place ~ centroidC + sdC + kurtosisC,
                     data = df[df$session == "postTest",], CV = T, na.action = "na.omit")
ldaFitPostTestTab = table(df[df$session == "postTest",]$place, ldaFitPostTest$class)
diag(prop.table(ldaFitPostTestTab, 1)) # By-PoA accuracy
```

```
##    dental retroflex
## 0.6198547 0.4766585
```

```
sum(diag(prop.table(ldaFitPostTestTab))) # Overall accuracy
```

```
## [1] 0.5487805
```

```
# Extract class from ldaFit for cross-session comparison (Step 3)
ldaClassPostTest = ldaFitPostTest$class
```

```
# permutation test
permHolderpostTest = rep(0, nrep)
for (i in 1:nrep) {
  #if (i %% 50 == 0) {
  #  timeKeeper = sprintf("Rep %s", i)
  #  print(timeKeeper)}
  dfPermut = df[df$session == "postTest",c("sdC", "centroidC", "kurtosisC")]
  randomSet = sample(df[df$session == "postTest",]$place)
  dfPermut$randomSet = randomSet
  random.lda.postTest = lda(randomSet ~ sdC + centroidC + kurtosisC,
                    data = dfPermut, CV = T, na.action = "na.omit")
  randomTabpostTest = table(df[df$session == "postTest",]$place, random.lda.postTest$class)
  permHolderpostTest[i] = sum(diag(prop.table(randomTabpostTest)))
}

# p-value:
length(permHolderpostTest[permHolderpostTest >=
                            sum(diag(prop.table(ldaFitPostTestTab)))])/nrep
```

```
## [1] 0.033
```

### Prod. training

```
ldaFitProdTrain = lda(place ~ centroidC + sdC + kurtosisC,
                      data = df[df$session == "prodTraining",],
                      CV = T, na.action = "na.omit")
ldaFitProdTrainTab = table(df[df$session == "prodTraining",]$place, ldaFitProdTrain$class)
diag(prop.table(ldaFitProdTrainTab, 1)) # By-PoA accuracy
```

```
##    dental retroflex
## 0.7015834 0.5975000
```

```
sum(diag(prop.table(ldaFitProdTrainTab))) # Overall accuracy
```

```
## [1] 0.6502159
```

```
# Extract class from ldaFit for cross-session comparison (step 3)
ldaClassProdTrain = ldaFitProdTrain$class
```

```r
# permutation test
permHolderprodTrain = rep(0, nrep)
for (i in 1:nrep) {
  #if (i %% 50 == 0) {
  #  timeKeeper = sprintf("Rep %s", i)
  #  print(timeKeeper)}
  dfPermut = df[df$session == "prodTraining",c("sdC", "centroidC", "kurtosisC")]
  randomSet = sample(df[df$session == "prodTraining",]$place)
  dfPermut$randomSet = randomSet
  random.lda.prodTrain = lda(randomSet ~ sdC + centroidC + kurtosisC,
                  data = dfPermut, CV = T, na.action = "na.omit")
  randomTabprodTrain = table(df[df$session == "prodTraining",]$place, random.lda.prodTrain$class)
  permHolderprodTrain[i] = sum(diag(prop.table(randomTabprodTrain)))
}

# p-value:
length(permHolderprodTrain[permHolderprodTrain >=
                             sum(diag(prop.table(ldaFitProdTrainTab)))])/nrep
```

```
## [1] 0.011
```

### Post-prod (re-test)

```r
ldaFitPostProd = lda(place ~ centroidC + sdC + kurtosisC, data = df[df$session == "postProd",], CV = T,
ldaFitPostProdTab = table(df[df$session == "postProd",]$place, ldaFitPostProd$class)
diag(prop.table(ldaFitPostProdTab, 1)) # By-PoA accuracy
```

```
##    dental retroflex
## 0.6744186 0.3896595
```

```r
sum(diag(prop.table(ldaFitPostProdTab))) # Overall accuracy
```

```
## [1] 0.5341615
```

```r
# Extract class from ldaFit for cross-session comparison (Step 3)
ldaClassPostProd = ldaFitPostProd$class
```

```r
# permutation test
permHolderpostProd = rep(0, nrep)
for (i in 1:nrep) {
  #if (i %% 50 == 0) {
  #  timeKeeper = sprintf("Rep %s", i)
  #  print(timeKeeper)}
  dfPermut = df[df$session == "postProd",c("sdC", "centroidC", "kurtosisC")]
  randomSet = sample(df[df$session == "postProd",]$place)
  dfPermut$randomSet = randomSet
  random.lda.postProd = lda(randomSet ~ sdC + centroidC + kurtosisC,
                  data = dfPermut, CV = T, na.action = "na.omit")
  randomTabpostProd = table(df[df$session == "postProd",]$place, random.lda.postProd$class)
  permHolderpostProd[i] = sum(diag(prop.table(randomTabpostProd)))
}

# p-value:
length(permHolderpostProd[permHolderpostProd >= sum(diag(prop.table(ldaFitPostProdTab)))])/nrep
```

```
## [1] 0.119
```

## Step 3: GLMM to compare classification across sessions

Comparisons cannot be made between sessions on the basis of permutation tests. This code takes the classifications from each session LDA, codes them as correct or incorrect, and then runs a logistic regression to determine if classification accuracy changes over the course of the four sessions.

### Create data frame for GLMM

```
# First, create a vector of the actual labels
# Split by session to ensure that we preserve the same order as used in the LDAs
trueClass = c(as.character(df[df$session == "preTest",]$place),
              as.character(df[df$session == "postTest",]$place),
              as.character(df[df$session == "prodTraining",]$place),
              as.character(df[df$session == "postProd",]$place))

# Make sure we have all the data
length(trueClass) == nrow(df)
```

```
## [1] TRUE
```

```
# Create a vector of LDA classifications from the by-session models
ldaClass = c(as.character(ldaClassPreTest),
             as.character(ldaClassPostTest),
             as.character(ldaClassProdTrain),
             as.character(ldaClassPostProd))

# Make sure the lengths match
length(ldaClass) == length(trueClass)
```

```
## [1] TRUE
```

```
# Create a vector of session codes
sessionCode = c(rep("preTest", length(ldaClassPreTest)),
            rep("postTest", length(ldaClassPostTest)),
            rep("prodTrain", length(ldaClassProdTrain)),
            rep("postProd", length(ldaClassPostProd)))

# Create a vector of subject IDs
subjectCode =  c(df[df$session == "preTest",]$subj,
                 df[df$session == "postTest",]$subj,
                 df[df$session == "prodTraining",]$subj,
                 df[df$session == "postProd",]$subj)

# Combine all data into a data frame
logRegDF = data.frame(subjectCode, sessionCode, trueClass, ldaClass)

# Create accuracy
logRegDF$accuracy = logRegDF$ldaClass == logRegDF$trueClass

# reorder sessionCode
logRegDF$sessionCode = factor(logRegDF$sessionCode,
```

```
                                levels =
                                  c("preTest", "postTest", "prodTrain", "postProd"))
```

### Reverse-helmert numeric coding for session

See: http://pidgin.ucsd.edu/pipermail/r-lang/2016-September/000886.html

Relevant paper (pg. 4): http://arxiv.org/pdf/1405.2094v1.pdf

divide by 2, 3, and 4 (# of things included in comparison) to equal 0

```
sessionRHC =  sapply(logRegDF$sessionCode,function(i) contr.helmert(4)[i,])
logRegDF$session1.2 = sessionRHC[1,]/2
logRegDF$session12.3 = sessionRHC[2,]/3
logRegDF$session123.4 = sessionRHC[3,]/4

head(logRegDF)
```

```
##   subjectCode sessionCode trueClass  ldaClass accuracy session1.2
## 1         101     preTest retroflex retroflex     TRUE       -0.5
## 2         101     preTest retroflex retroflex     TRUE       -0.5
## 3         101     preTest retroflex retroflex     TRUE       -0.5
## 4         101     preTest retroflex retroflex     TRUE       -0.5
## 5         101     preTest retroflex retroflex     TRUE       -0.5
## 6         101     preTest retroflex retroflex     TRUE       -0.5
##   session12.3 session123.4
## 1  -0.3333333        -0.25
## 2  -0.3333333        -0.25
## 3  -0.3333333        -0.25
## 4  -0.3333333        -0.25
## 5  -0.3333333        -0.25
## 6  -0.3333333        -0.25
```

### run GLMM

```
# glmer1: full model
# includes all fixed effects (session predictors) as by-subject random slopes
sessionLDA.glmer1 = glmer(accuracy ~ session1.2 + session12.3 + session123.4 +
                        (1 + session1.2 + session12.3 + session123.4 |subjectCode),
                        data = logRegDF,
                        family = "binomial")
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge with max|grad| = 0.00238238 (tol =
## 0.001, component 1)
```

```
# Did not converge

# glmer2: Try decorrelated slopes (||)
sessionLDA.glmer2 = glmer(accuracy ~ session1.2 + session12.3 + session123.4 +
                            (1 + session1.2 + session12.3 + session123.4 || subjectCode),
                          data = logRegDF,
                          family = "binomial")
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge with max|grad| = 0.00386656 (tol =
## 0.001, component 1)
```

```
# check for components contributing zero variance
# see Bates et al., "Parsimonious mixed models"
summary(rePCA(sessionLDA.glmer2))
```

```
## $subjectCode
## Importance of components:
##                          [,1]   [,2]      [,3]      [,4]
## Standard deviation     0.3077 0.1546 0.0003608 0.0002761
## Proportion of Variance 0.7985 0.2015 0.0000000 0.0000000
## Cumulative Proportion  0.7985 1.0000 1.0000000 1.0000000
```

```
# 2 zero variance components. which ones?

summary(sessionLDA.glmer2, corr = FALSE)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session1.2 +
##     session12.3 + session123.4 || subjectCode)
##    Data: logRegDF
##
##      AIC      BIC   logLik deviance df.resid
##   8778.0   8832.2  -4381.0   8762.0     6460
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6544 -1.0877  0.6880  0.9073  1.0430
##
## Random effects:
##  Groups        Name         Variance  Std.Dev.
##  subjectCode   (Intercept)  2.390e-02 0.1545880
##  subjectCode.1 session1.2   7.624e-08 0.0002761
##  subjectCode.2 session12.3  9.469e-02 0.3077180
##  subjectCode.3 session123.4 1.302e-07 0.0003608
## Number of obs: 6468, groups:  subjectCode, 19
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.27887    0.04365   6.389 1.67e-10 ***
## session1.2    0.04283    0.07091   0.604 0.545865
## session12.3   0.45463    0.09511   4.780 1.75e-06 ***
## session123.4 -0.19241    0.05814  -3.310 0.000934 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## convergence code: 0
## Model failed to converge with max|grad| = 0.00386656 (tol = 0.001, component 1)
```

```
# 1.2 and 123.4 slopes ~ 0

# glmer3: Remove zero variance slopes
```

```
sessionLDA.glmer3 = glmer(accuracy ~ session1.2 + session12.3 + session123.4 +
                            (1   + session12.3 ||subjectCode),
                          data = logRegDF,
                          family = "binomial")
# make sure that the removed slopes don't cause a drop in model fit
anova(sessionLDA.glmer2, sessionLDA.glmer3)
```

```
## Data: logRegDF
## Models:
## sessionLDA.glmer3: accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session12.3 ||
## sessionLDA.glmer3:     subjectCode)
## sessionLDA.glmer2: accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session1.2 +
## sessionLDA.glmer2:     session12.3 + session123.4 || subjectCode)
##                    Df  AIC    BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## sessionLDA.glmer3  6 8774 8814.6  -4381     8762
## sessionLDA.glmer2  8 8778 8832.2  -4381     8762     0      2          1
```

```
# p = n.s., prefer glmer3 to glmer2

summary(rePCA(sessionLDA.glmer3))
```

```
## $subjectCode
## Importance of components:
##                           [,1]   [,2]
## Standard deviation     0.3077 0.1546
## Proportion of Variance 0.7984 0.2016
## Cumulative Proportion  0.7984 1.0000
```

```
# no more zero variance components

# glmer4: add correlations in random effects back in
sessionLDA.glmer4 = glmer(accuracy ~ session1.2 + session12.3 + session123.4 +
                            (1   + session12.3 |subjectCode),
                          data = logRegDF,
                          family = "binomial")

anova(sessionLDA.glmer3, sessionLDA.glmer4)
```

```
## Data: logRegDF
## Models:
## sessionLDA.glmer3: accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session12.3 ||
## sessionLDA.glmer3:     subjectCode)
## sessionLDA.glmer4: accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session12.3 |
## sessionLDA.glmer4:     subjectCode)
##                    Df    AIC    BIC  logLik deviance  Chisq Chi Df
## sessionLDA.glmer3  6 8774.0 8814.6 -4381.0   8762.0
## sessionLDA.glmer4  7 8763.4 8810.8 -4374.7   8749.4 12.586      1
##                    Pr(>Chisq)
## sessionLDA.glmer3
## sessionLDA.glmer4  0.0003886 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# p = 0.00039, prefer glmer4 to glmer3

summary(sessionLDA.glmer4)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## accuracy ~ session1.2 + session12.3 + session123.4 + (1 + session12.3 |
##     subjectCode)
##    Data: logRegDF
##
##      AIC      BIC   logLik deviance df.resid
##   8763.4   8810.8  -4374.7   8749.4     6461
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.7532 -1.0845  0.6777  0.9075  1.0904
##
## Random effects:
##  Groups      Name        Variance Std.Dev. Corr
##  subjectCode (Intercept) 0.02632  0.1622
##              session12.3 0.10995  0.3316   1.00
## Number of obs: 6468, groups:  subjectCode, 19
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.28074    0.04509   6.227 4.77e-10 ***
## session1.2    0.04145    0.07061   0.587 0.557208
## session12.3   0.45984    0.09937   4.628 3.70e-06 ***
## session123.4 -0.19502    0.05815  -3.354 0.000797 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) sss1.2 ss12.3
## session1.2  -0.004
## session12.3  0.652  0.004
## sessin123.4 -0.009  0.005 -0.019
```

### Check significance of fixed effects with nested chisq

Compare model with predictor to one without. Does the inclusion of the predictor improve model fit?

```
# Matt Goldrick's lme convenience function
chiReport.func <- function(a){
  ifelse (a$"Pr(>Chisq)"[2] > .0001,return(paste("chisq(",a$"Chi Df"[2],")=",round(a$Chisq[2],2),", p =
```

```
# pre-test vs. post-test
chiReport.func(anova(sessionLDA.glmer4,
                     update(sessionLDA.glmer4 ,.~.-session1.2)))
```

```
## [1] "chisq(1)=0.34, p = 0.5572"
```

```
# prod training vs pre+post
chiReport.func(anova(sessionLDA.glmer4,
                     update(sessionLDA.glmer4 ,.~.-session12.3)))
```

```
## [1] "chisq(1)=14.65, p = 1e-04"
```

```
# retest vs. pre+post+prod
chiReport.func(anova(sessionLDA.glmer4,
                     update(sessionLDA.glmer4 ,.~.-session123.4)))
```

```
## [1] "chisq(1)=11.22, p = 8e-04"
```

**Interpretation:**

No change in accuracy from pre-test to post-test (b = 0.041, t = 0.587, p = 0.557)

Improved accuracy at production training (b = 0.460, t = 4.628, p < 0.001)

Drop in accuracy at post-production (session123.4) (b = -0.195, t= -3.354, p < 0.001)

## Step 4: By-subject classification

This section replicates the LDA classifications in steps 2 and 3 (overall and by session), but on an individual subject basis, to determine the variance in classification accuracy by-subject.

### Set up data

```
# Build a data frame to hold results
uniqueSubj = unique(df$subj)
acc = c("overall", "dental", "retroflex") # overall here means over both PoAs
sessionTypes = c("preTest", "postTest", "prodTraining", "postProd", "overall") # Overall here means ove

subjAccDF = expand.grid(uniqueSubj, acc, sessionTypes)
colnames(subjAccDF) = c("subj", "category", "session")
subjAccDF$accuracy = 0
subjAccDF$subj = as.factor(as.character(subjAccDF$subj))

# For loops - just the 4 sessions
uniqueSess = unique(df$session)
```

### Run overall LDA

```
for (i in uniqueSubj) {
   #Run model
   overallLDA = lda(place ~ sdC + centroidC + kurtosisC,
                    data = df[df$subj == i,],
                    CV = T, na.action = "na.omit")
    # Get accuracy table
    overallTab = table(df[df$subj == i,]$place, overallLDA$class)
    # Accuracy by place of articulation
    overallProp = diag(prop.table(overallTab, 1))
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == "overall" &
                subjAccDF$category == "dental",]$accuracy = overallProp[1]
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == "overall" &
                subjAccDF$category == "retroflex",]$accuracy = overallProp[2]
    # Overall accuracy
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == "overall" &
                subjAccDF$category == "overall",]$accuracy =
```

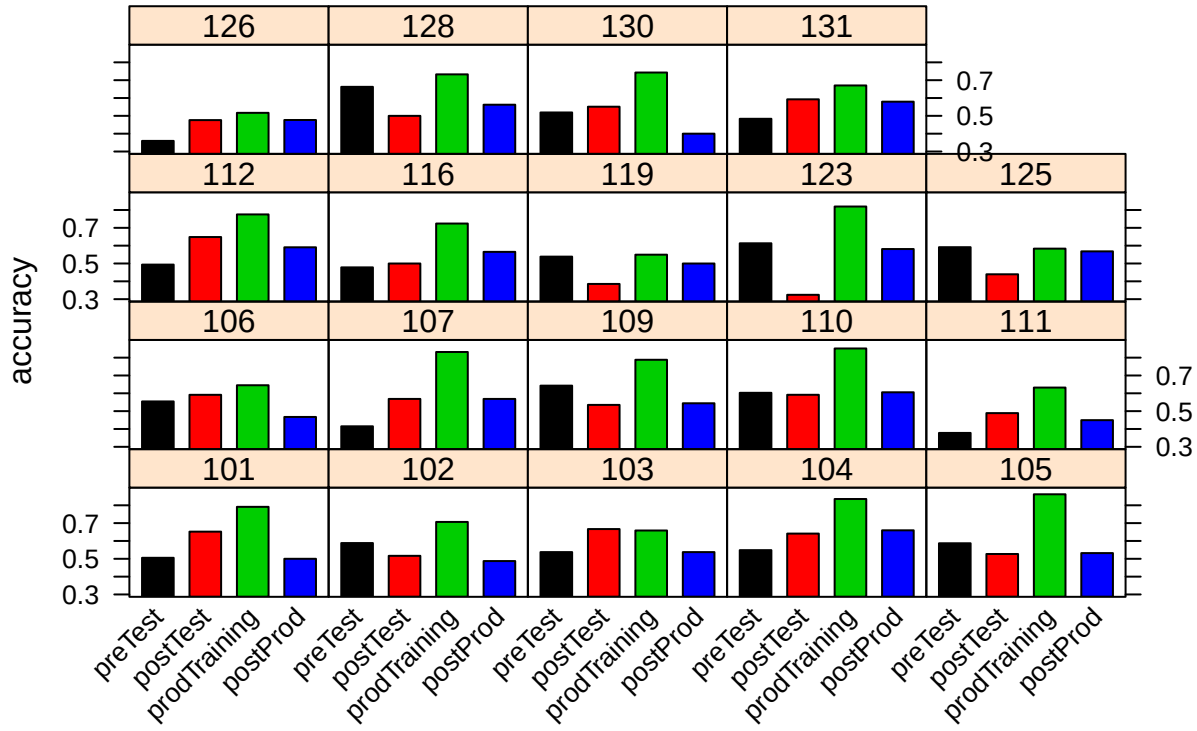
```
      sum(diag(prop.table(overallTab)))
}
```

### Run by-session LDA

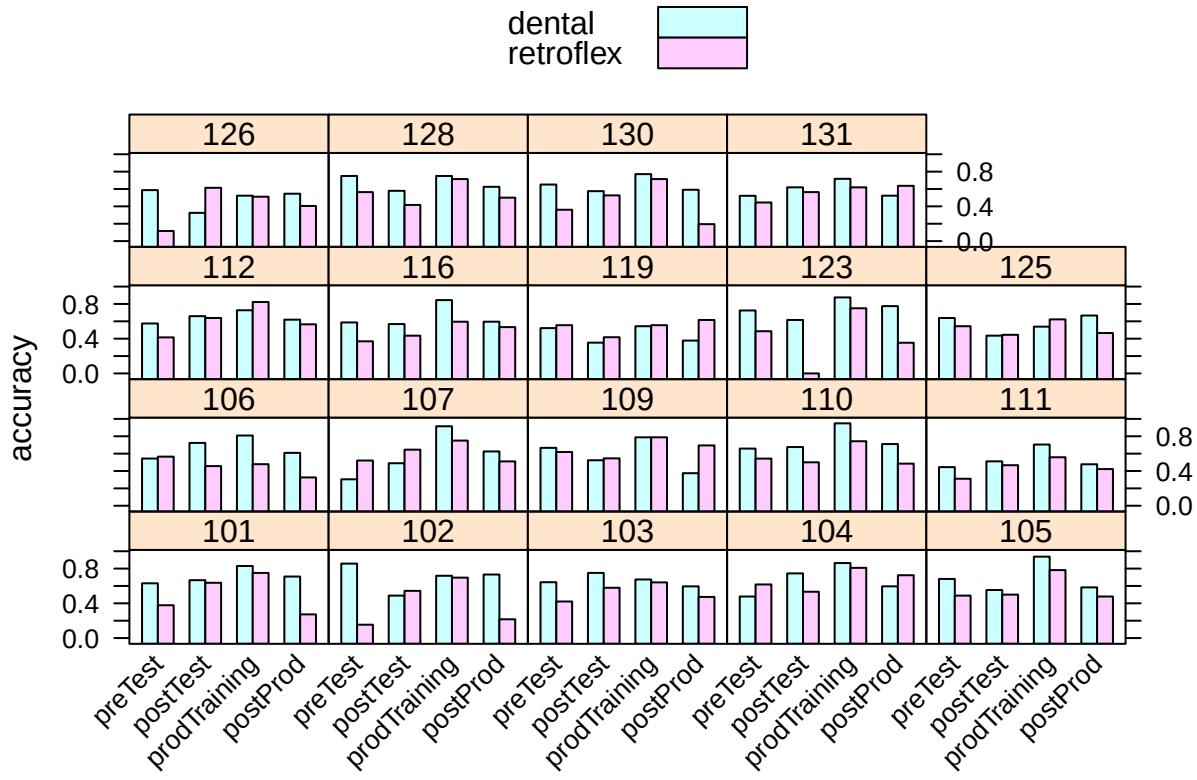
```
for (i in uniqueSubj) {
  for (j in uniqueSess) {
    # Run model
    indivLDA = lda(place ~ sdC + centroidC + kurtosisC,
                   data = df[df$subj == i & df$session == j,],
                   CV = T, na.action = "na.omit")
    # Get accuracy table
    indivTab = table(df[df$subj == i & df$session == j,]$place, indivLDA$class)
    # Accuracy by place of articulation
    poaProp = diag(prop.table(indivTab, 1))
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == j &
                subjAccDF$category == "dental",]$accuracy = poaProp[1]
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == j &
                subjAccDF$category == "retroflex",]$accuracy = poaProp[2]
    # Overall accuracy
    subjAccDF[subjAccDF$subj == i & subjAccDF$session == j &
                subjAccDF$category == "overall",]$accuracy =
      sum(diag(prop.table(indivTab)))
    rm(indivTab)
    rm(poaProp)
  }
}
```

### Visualize individual differences

```
# By subject accuracy across sessions

# Most speakers are peaking in prod training and dropping again in post-prod/re-test
barchart(accuracy ~ session | subj,
         data = subjAccDF[subjAccDF$category == "overall" &
                            subjAccDF$session != "overall",], col = c(1,2,3,4),
         scales=list(x=list(rot=45)))
```




```
# Dental vs. retroflex by-subject trends over sessions

# With a few exceptions, accuracy in one category tracks accuracy in the other
barchart(accuracy ~ session | subj,
         groups = factor(category),
         data = subjAccDF[subjAccDF$category != "overall" &
                          subjAccDF$session != "overall",], auto.key = TRUE,
         scales=list(x=list(rot=45)))
```



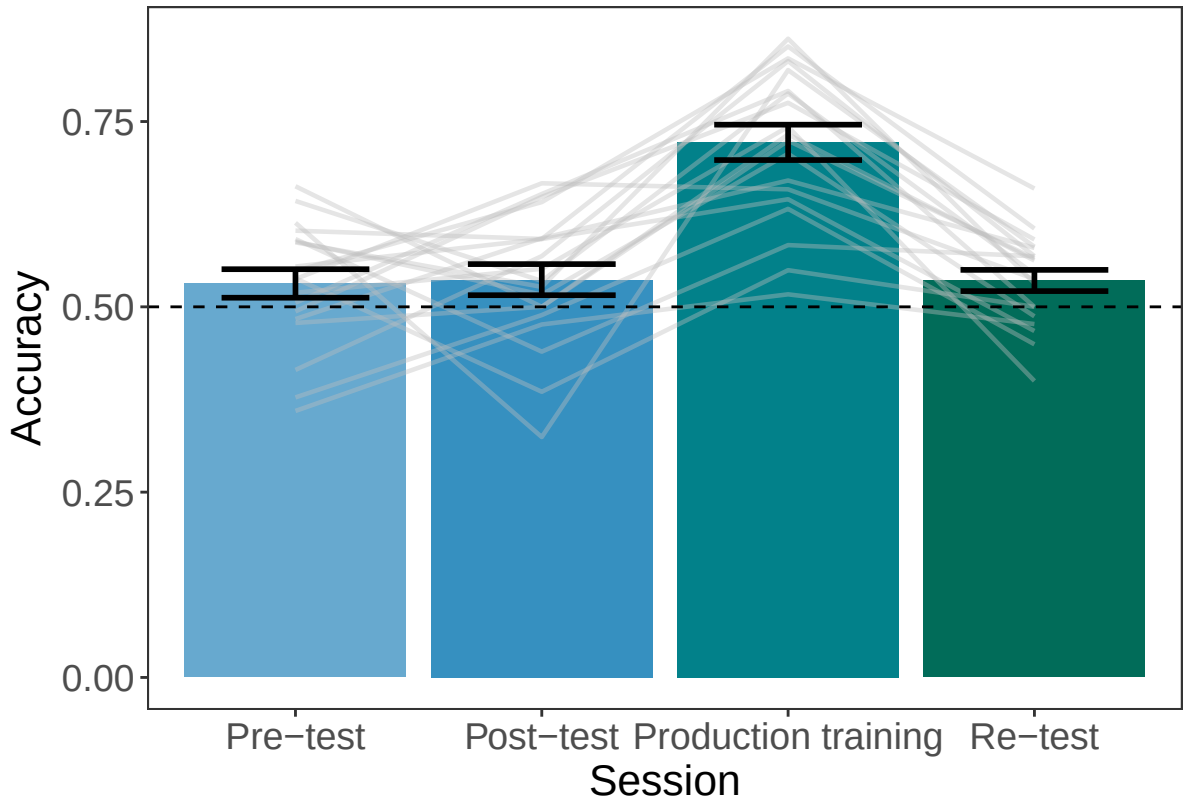

### Plot: average accuracy + subject lines

Figure 5B in manuscript.

See: http://stackoverflow.com/questions/27366615/bar-plot-of-group-means-with-lines-of-individual-results-overlaid

```
# Colors for plots
preCol = "#67a9cf"
postCol = "#3690c0"
prodCol = "#02818a"
reCol = "#016c59"

# Get place-overall acc data into a aggregated format, by session and subject
aggSubjOverall = subjAccDF[subjAccDF$category == "overall"
                           & subjAccDF$session != "overall",]
aggSubjOverall$category = NULL

# ggplot theme for all plots
my.theme = theme_bw() +
  theme(panel.grid.minor=element_blank(),        # No gridlines
        panel.grid.major=element_blank(),
        axis.title.x = element_text(size=16),
        axis.text.x  = element_text(size = 14),
        strip.text.x = element_text(size=16),
        axis.title.y = element_text(size=16),
        axis.text.y  = element_text(size = 14),
```

```
        plot.margin = unit(c(.5, .5, .5, .5), "cm"),
        plot.title = element_text(hjust = 0.5, size = 16))

bp = ggplot(aggSubjOverall) +
  stat_summary(aes(x = session, y = accuracy,
                   group = session, fill = session),
               fun.y = "mean", geom = "bar")+
  scale_fill_manual(values = c("preTest" = preCol,
                               "postTest" = postCol,
                               "prodTraining" = prodCol,
                               "postProd" = reCol))+
  geom_line(aes(x = session, y = accuracy, group = subj),
            size = 0.8, col = "grey", alpha = 0.4) +
  stat_summary(aes(x = session, y = accuracy, group = session),
               fun.data = "mean_se", geom = "errorbar",
               col = "black", width = 0.6, size = 1) +
  geom_hline(aes(yintercept=0.5), linetype = 2)

bp +
  guides(fill = FALSE) +                          # suppress legend
  my.theme +
  # X-axis
  scale_x_discrete(breaks=c("preTest", "postTest", "prodTraining", "postProd"),
                     labels=c("Pre-test", "Post-test",
                              "Production training", "Re-test"),
                     name = "Session") +
  # Y-axis
  scale_y_continuous(name = "Accuracy")
```




**Plot: Overall improvement compared to baseline (pre-test)** ————————-

```
# Convert data to wide format so that within-speaker session differences can be calculated
aggSubjWide = reshape(aggSubjOverall, direction = "wide",
                      idvar = "subj", timevar = "session")
colnames(aggSubjWide) = c("subj", "PreTest",
                          "PostTest", "ProdTraining", "ReTest")

# Calculate session difference (accuracy at session X - accuracy at session X-1)
# Add colors for values falling above or below the 0,1 line
# Those above indicate improvement in performance, those below indicate decline
aggSubjWide$postPre = aggSubjWide$PostTest - aggSubjWide$PreTest
aggSubjWide$postPreCol = "#000075" # transparent black
aggSubjWide[aggSubjWide$postPre > 0,]$postPreCol = "#228B2275" # transparent forest green
aggSubjWide[aggSubjWide$postPre < 0,]$postPreCol = "#FF000075" # transparent red

aggSubjWide$prodPre = aggSubjWide$ProdTraining - aggSubjWide$PreTest
aggSubjWide$prodPreCol = "black"
aggSubjWide[aggSubjWide$prodPre > 0,]$prodPreCol = "#228B2275" # transparent forest green
aggSubjWide[aggSubjWide$prodPre < 0,]$prodPreCol = "#FF000075" # transparent red

aggSubjWide$rePre = aggSubjWide$ReTest - aggSubjWide$PreTest
aggSubjWide$rePreCol = "black"
aggSubjWide[aggSubjWide$rePre > 0,]$rePreCol = "#228B2275" # transparent forest green
aggSubjWide[aggSubjWide$rePre < 0,]$rePreCol = "#FF000075" # transparent red
```

```r
par(mfrow = c(1,3))
par(mar = c(5,5,3,1))
plot(PostTest ~ PreTest, data = aggSubjWide,
     pch = 21, cex = 2.75, col = "black", bg = postPreCol,
     ylab = "Post-test accuracy", xlab = "Pre-test accuracy",
     cex.lab = 1.6, cex.axis = 1.6,
     ylim = c(0,1), xlim = c(0,1))
abline(0,1)
plot(ProdTraining ~ PreTest, data = aggSubjWide,
     pch = 21, cex = 2.75, col = "black", bg = prodPreCol,
     ylab = "Prod. training accuracy", xlab = "Pre-test accuracy",
     cex.lab = 1.6, cex.axis = 1.6,
     ylim = c(0,1), xlim = c(0,1))
abline(0,1)
plot(ReTest ~ PreTest, data = aggSubjWide,
     pch = 21, cex = 2.75, col = "black", bg = rePreCol,
     ylab = "Re-test accuracy", xlab = "Pre-test accuracy",
     cex.lab = 1.6, cex.axis = 1.6,
     ylim = c(0,1), xlim = c(0,1))
abline(0,1)
```