Inspecting How Swapping Sounds Affects Word Recognition

```
library(dplyr)
library(ggplot2)
library(purrr)
```

```
weighted_average <-function(df) {
  total_frequency=sum(10^(df$Frequency))
  return(sum(df$Confusability*(10^df$Frequency))/total_frequency)
}
# Goes through a list, reading csv files that each have information about
#   how easy it is to confuse ~40,000 English wordforms given a particular
#   phonetic scenario (generated from an computational experiment I was
#   running in Python). Then, it saves the language-wide information for each
#   scenario into a single cell on that scenario's row
read_swap_list_file <- function(swap_list_filename,
                                filename_prefix,
                                filename_suffix,
                                baseline_row,
                                consonants,
                                vowels) {
  read.table(swap_list_filename) %>%
    tbl_df() %>%
    mutate(V1=as.character(V1),
           Phones=purrr::map(V1,~strsplit(.,",")),
           filename=purrr::map(Phones,
                               ~paste0(filename_prefix,
                                       paste_every_other(.[[1]]),
                                       filename_suffix)),
           Type=factor(purrr::map(Phones,
                                  ~determine_type(assign_phones(.[[1]], consonants, vowels))) %>%
                         simplify())) %>%
    mutate(data=purrr::map(filename,~read.csv(.))) %>%
    bind_rows(baseline_row)
}


# This is just turn the phones into the right format for the filename
paste_every_other <- function(l) {
  purrr::map2(l[c(TRUE,FALSE)],
              l[c(FALSE,TRUE)],
              ~paste0(.x,"_to_",.y)) %>%
    purrr::simplify() %>%
    purrr::reduce(~paste0(.x,"_and_",.y))
}
# Unused
df_every_other <- function(l) {
  data.frame(
    FirstPhone=l[c(TRUE,FALSE)],
    SecondPhones=l[c(FALSE,TRUE)]
```

```
  )
}

# Says if a phone is a consonant or vowel
assign_phones <- function(phones, consonants, vowels) {
  ifelse(phones %in% consonants,"C",
  ifelse(phones %in% vowels, "V",
         NA))
}
determine_type <- function(l) {
  if ("V" %in% l) {
    if ("C" %in% l) {
      "Mixed"
    } else {
      "Vowels"
    }
  } else {
    "Consonants"
  }
}
```

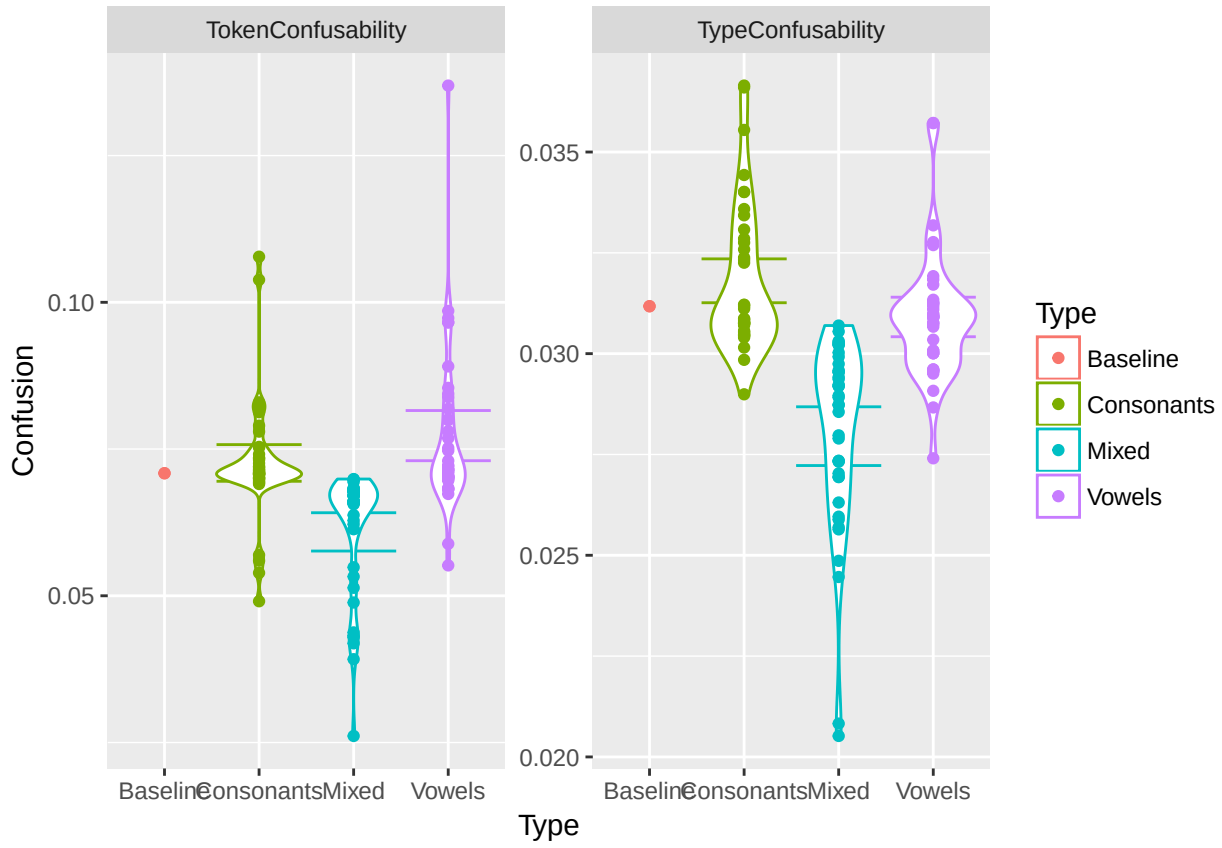
```
project_path="~/Documents/workspace2/Confusability/src/"
cm_file =         paste0(project_path, "r_readable_cm.csv")
cm_file_low_snr =         paste0(project_path, "r_readable_cm_low_snr.csv")
sonority_file = paste0(project_path, "phone_sonority.csv")
lexicon_file =    paste0(project_path, "SwappingPhonemes/lexicon_no_ao_cutler_only.csv")
swap_list_file  = paste0(project_path, "swap_list.txt")
filename_prefix_high_snr = paste0(project_path, "SwappingPhonemes/cmcl_talk_no_merging_high_snr_")
filename_prefix_low_snr = paste0(project_path, "SwappingPhonemes/cmcl_talk_no_merging_low_snr_")
filename_suffix="_no_ao_cutler_only.csv"
consonants=c("P","B","T","D","K","G","JH","CH","F","V","TH","DH","S","Z","SH","HH","M","N","L","R","NG"
vowels=c('IH1', 'IY1', 'EY1', 'AY1', 'OY1', 'EH1', 'AE1', 'ER1', 'UH1', 'UW1', 'AW1', 'OW1', 'AO1', 'AH
```

```
# Loads two uniphone confusion matrices, at 'High' and 'Low' signal-to-noise ratios
confusion_matrix <- bind_rows("High"=read.csv(cm_file, quote="'"),
                              "Low"=read.csv(cm_file_low_snr, quote="'"),
                              .id="SNR")
# Puts them into a tidier format
d.cm <- confusion_matrix %>%
  tidyr::gather("StimPhone","Confusion",-HeardPhone,-SNR)

# Reads the lexicon
lexicon <- read.csv(lexicon_file) %>%
  # Removes the word-initial "Q-"
  mutate(Word=gsub("Q-","",Word),
         Phone=Word) %>%
  # Breaks each word down into its constituent phones
  tidyr::separate_rows(Phone,sep="-")

sonority_df <- read.csv(sonority_file)

# Loads baseline (non-altered) language-wide confusabilities
baseline_row <- data.frame(Type=c("Baseline")) %>%
  tbl_df() %>%
```

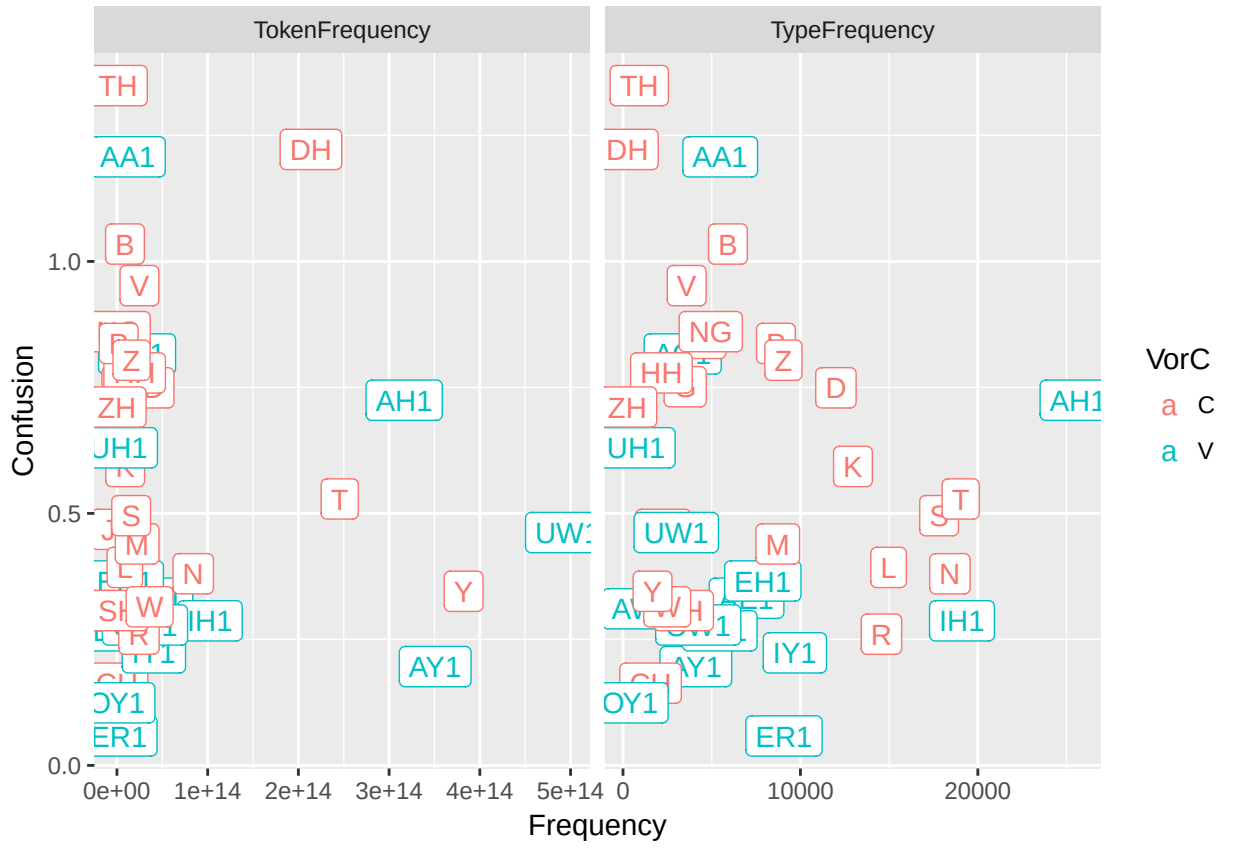
```
  # Saves all the confusability information as a single cell
  mutate(data=purrr::map(Type,
                         ~read.csv(paste0(filename_prefix_high_snr,
                                          "baseline",
                                          filename_suffix))))
# Does the same thing as above, but for the low SNR conditions
baseline_row_low_snr <- data.frame(Type=c("Baseline")) %>%
  tbl_df() %>%
  mutate(data=purrr::map(Type,~read.csv(paste0(filename_prefix_low_snr, "baseline",filename_suffix))))

# Reads in all the confusabilities for every word in English,
#   for every possible swapping of phones, storing the language-wide
#   confusabilities as a cell in each row, with there being one row per swap
high_snr_swaps <- read_swap_list_file(swap_list_file,
                                      filename_prefix_high_snr,
                                      filename_suffix,
                                      baseline_row,
                                      consonants,vowels)
low_snr_swaps <- read_swap_list_file(swap_list_file,
                                      filename_prefix_low_snr,
                                      filename_suffix,
                                      baseline_row_low_snr,
                                      consonants,vowels)
```

```
# Calculates the Weighted and Unweighted mean language-wide confusability for each row
high_snr_swaps <- high_snr_swaps %>%
  mutate(UnWeightedMean = purrr::map(data,~mean(1-.$Confusability)) %>% purrr::simplify(),
         WeightedMean = purrr::map(data,~1-weighted_average(.)) %>% purrr::simplify())

low_snr_swaps <- low_snr_swaps %>%
  mutate(UnWeightedMean = purrr::map(data,~mean(1-.$Confusability)) %>% purrr::simplify(),
         WeightedMean = purrr::map(data,~1-weighted_average(.)) %>% purrr::simplify())
```

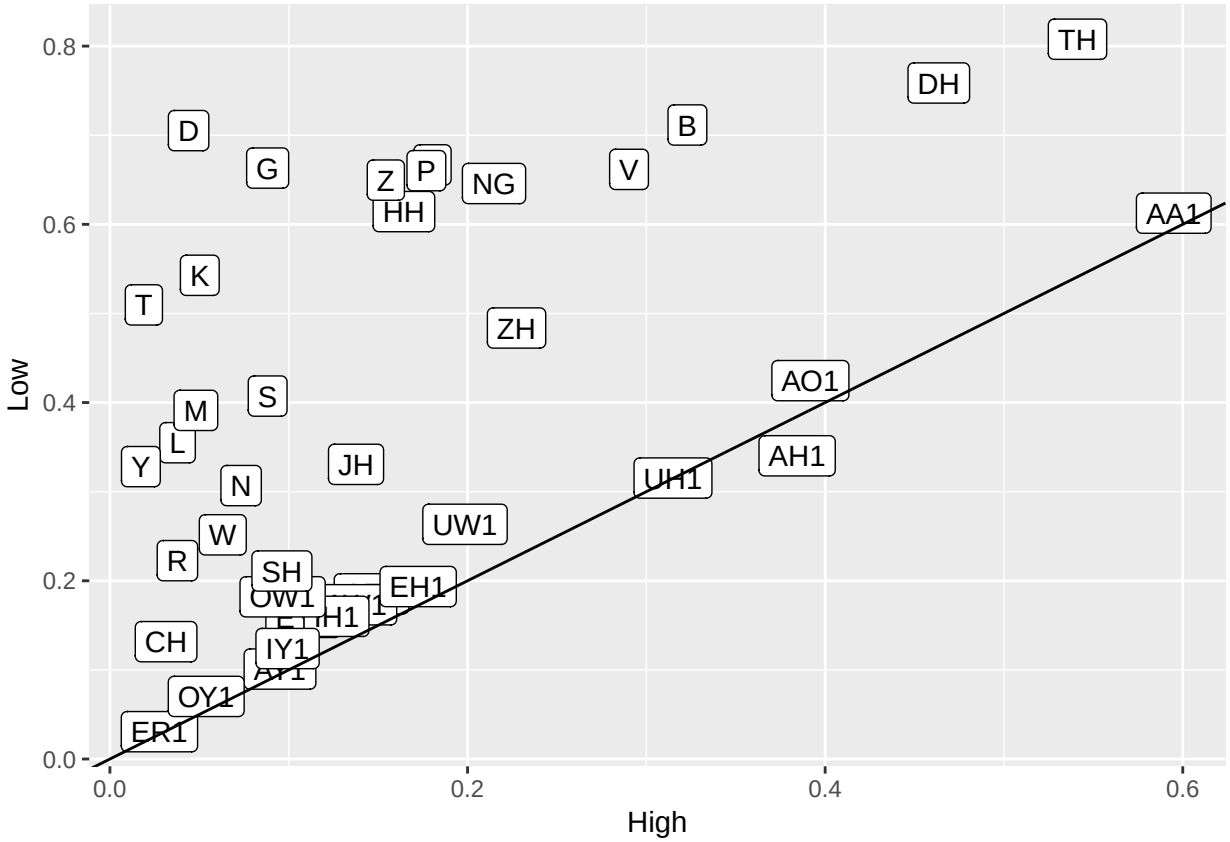
```
high_snr_swaps %>%
  rename(TokenConfusability=WeightedMean,
         TypeConfusability=UnWeightedMean) %>%
  tidyr::gather(ConfusionType,Confusion,c(TokenConfusability,TypeConfusability)) %>%
  ggplot(aes(x=Type,y=Confusion,color=Type)) %>%
   zplyr::errorbars() +
  facet_wrap(~ConfusionType,scales="free_y") +
  geom_violin() +
  geom_point()
```



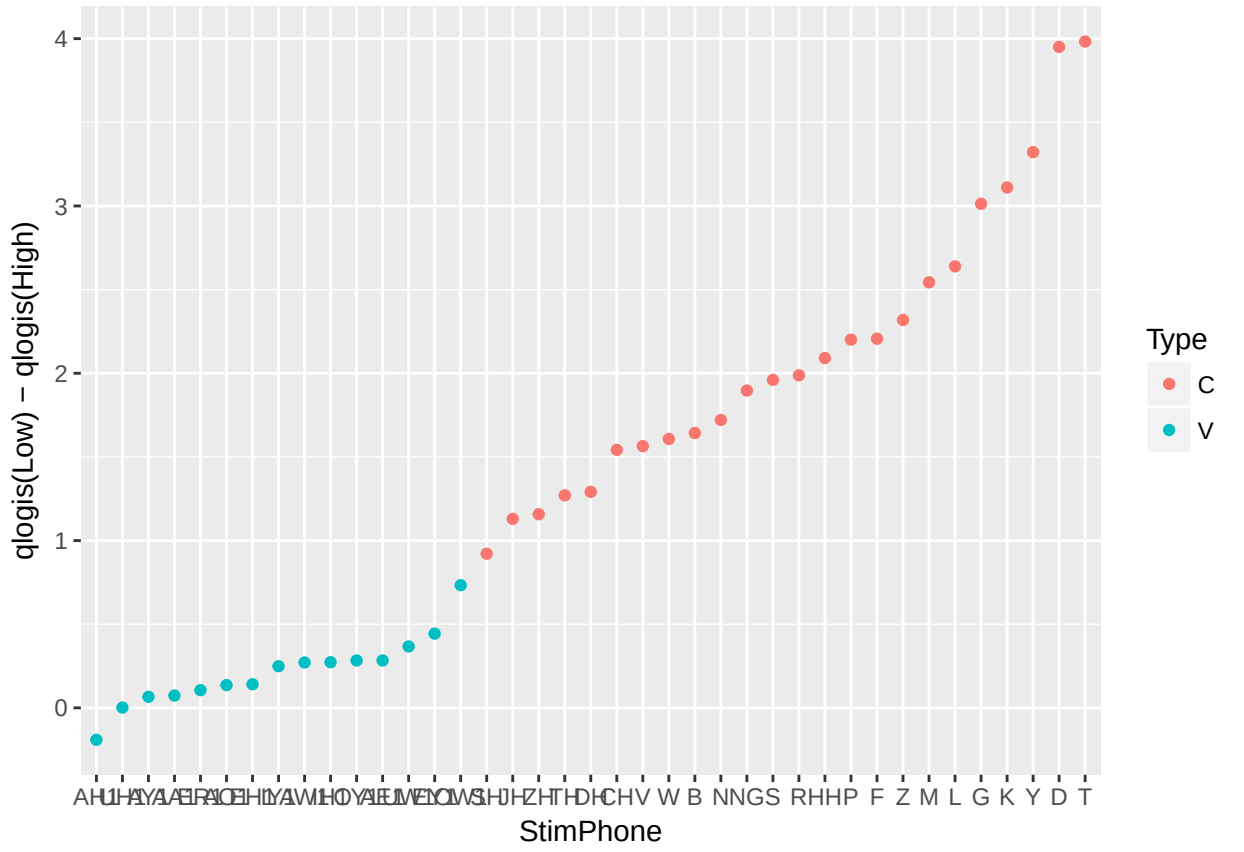

```
lexicon %>% group_by(Phone) %>%
  summarize(TokenFrequency=sum(10^Frequency),
            TypeFrequency=n(),
            VorC=assign_phones(first(Phone),consonants,vowels)) %>%
  left_join(d.cm %>% group_by(StimPhone) %>%
              summarise(Confusion=sum(Confusion[HeardPhone!=StimPhone])) %>%
              rename(Phone=StimPhone),
            by="Phone") %>%
  tidyr::gather(FreqType,Frequency,c(TokenFrequency,TypeFrequency)) %>%
  ggplot(aes(x=Frequency,y=Confusion,color=VorC,label=Phone)) +
  facet_wrap(~FreqType,scales="free_x") +
  geom_point() +
  geom_label()
```



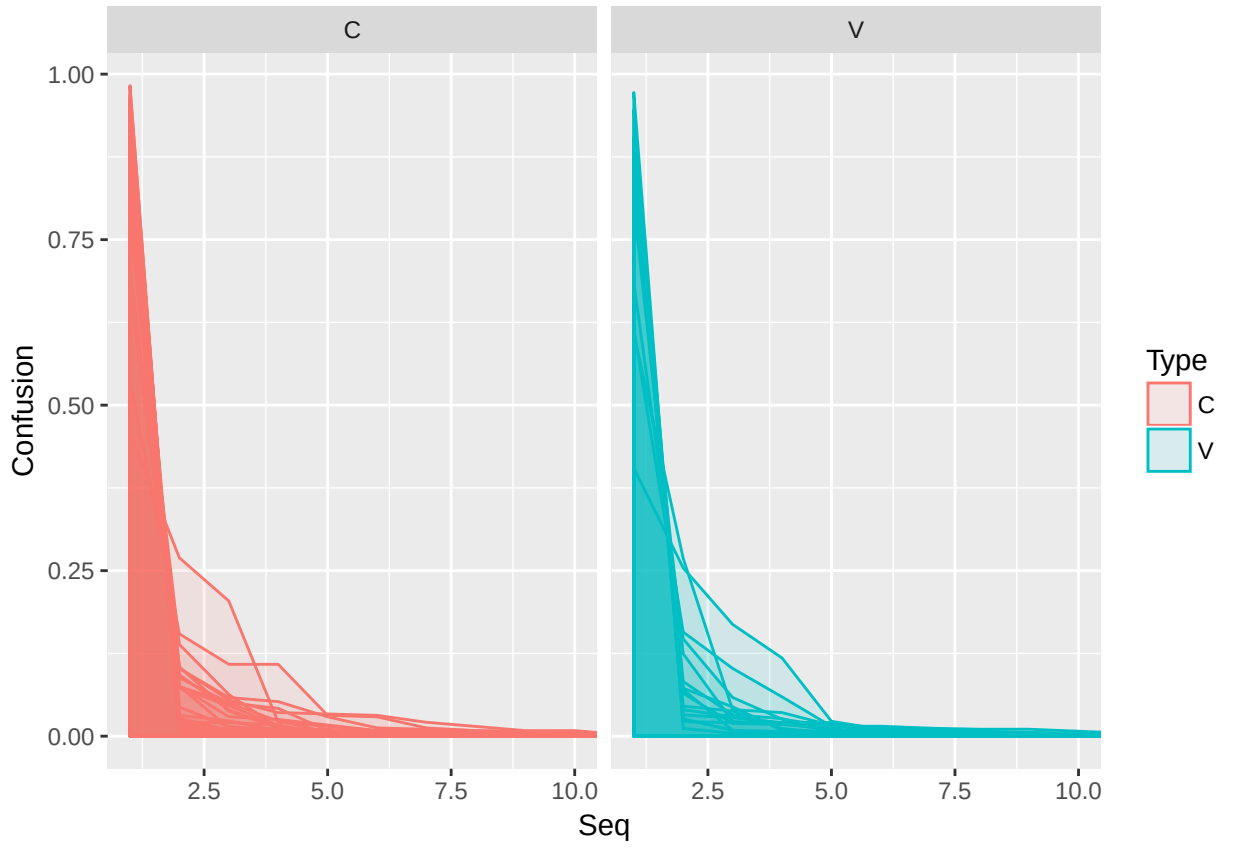

```
d.cm %>% group_by(SNR,StimPhone) %>%
  summarise(Confusion=sum(Confusion[HeardPhone!=StimPhone])) %>%
  tidyr::spread(SNR,Confusion) %>%
  ggplot(aes(x=High,y=Low,label=StimPhone)) + geom_label() +
  geom_abline(intercept = 0,slope=1)
```



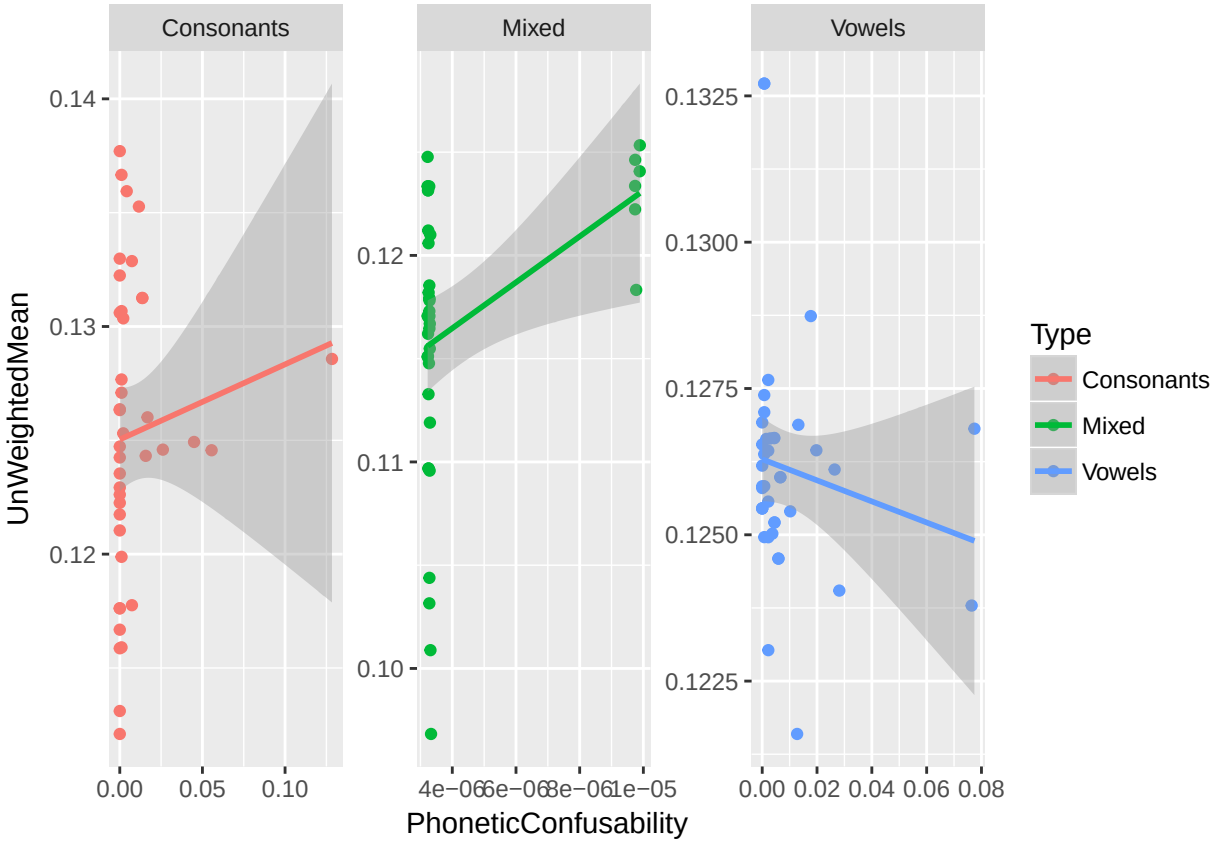

```
d.cm %>% group_by(SNR,StimPhone) %>%
  summarise(Confusion=sum(Confusion[HeardPhone!=StimPhone])) %>%
  tidyr::spread(SNR,Confusion) %>%
  arrange(plogis(qlogis(Low)-qlogis(High))) %>%
  mutate(StimPhone=factor(StimPhone,
                          levels=unique(StimPhone)),
         Type=assign_phones(StimPhone,consonants,vowels)) %>%
  ggplot(aes(x=StimPhone,y=qlogis(Low)-qlogis(High),color=Type)) + geom_point()
```



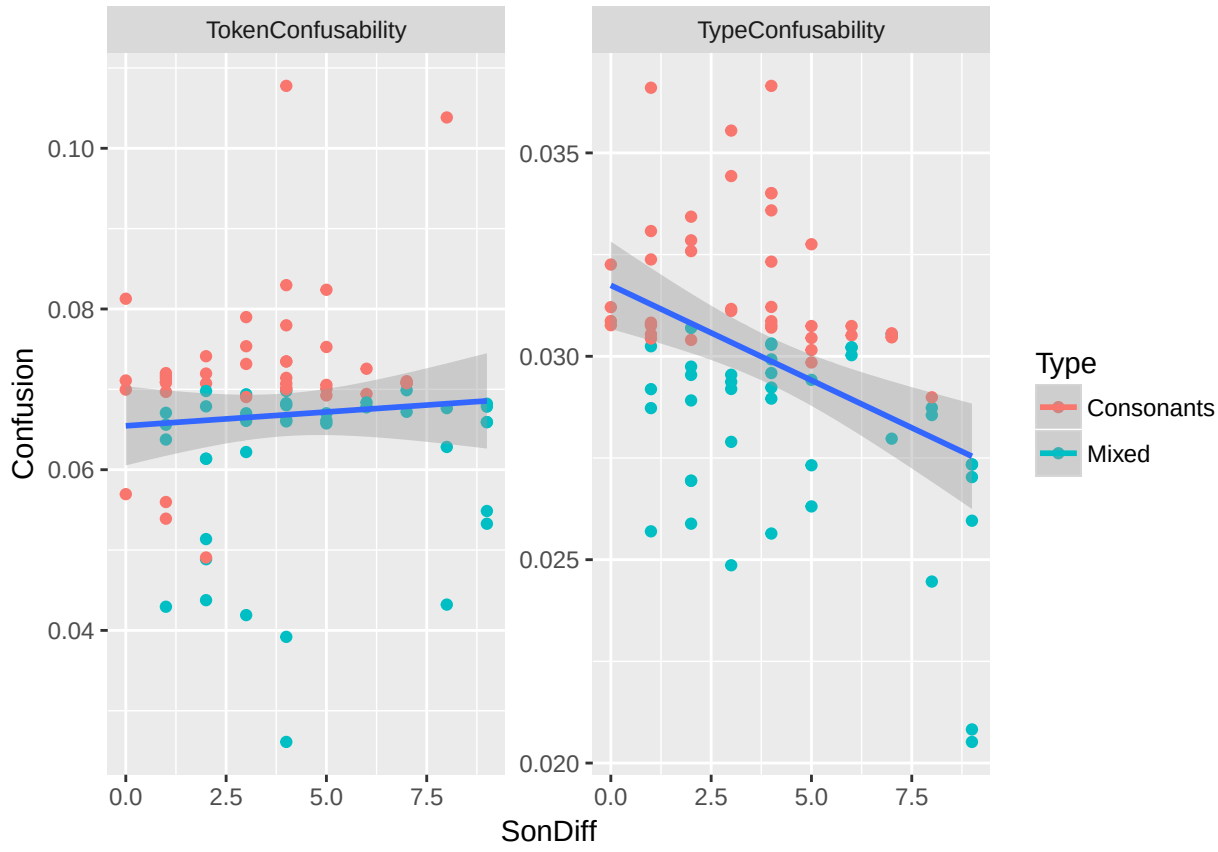

```
d.cm %>%
  group_by(StimPhone,SNR) %>%
  arrange(-Confusion) %>%
  mutate(Seq=seq_along(HeardPhone)) %>%
  ungroup() %>%
  mutate(Type=assign_phones(StimPhone,consonants,vowels)) %>%

  filter(SNR=="High" #& Type=="V"
         ) %>%
  ggplot(aes(x=Seq,y=Confusion,color=Type,fill=Type,group=StimPhone)) +
  # geom_point() +
  facet_wrap(~Type) +
  coord_cartesian(xlim=c(1,10)) +
  geom_density(stat="identity",alpha=0.1)
```



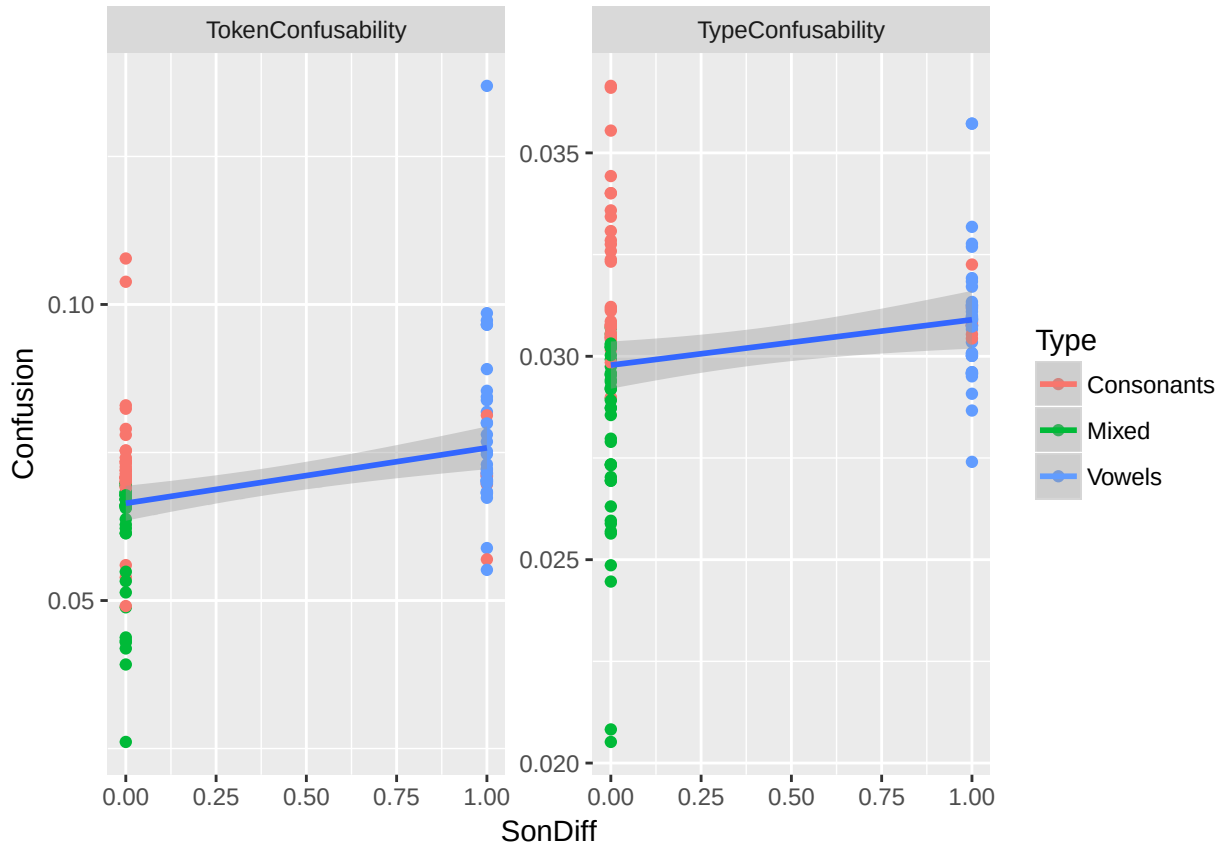

```
left_join(low_snr_swaps %>%
    mutate(PhoneAttacher=purrr::map(Phones,
                                    ~paste(sort(c(.[[1]][1],.[[1]][2])),collapse=",")) %>%
             purrr::simplify()) %>%
      filter(Type != "Baseline"), d.cm %>%
    mutate(X=paste0(HeardPhone,",",StimPhone)) %>%
    mutate(Phones=purrr::map(X,~strsplit(.,",")),
           PhoneAttacher=purrr::map(Phones,
                                    ~paste(sort(c(.[[1]][1], .[[1]][2])),collapse=",")) %>%
             purrr::simplify()) %>%
    filter(SNR=="High") %>%
    group_by(PhoneAttacher) %>%
    summarise(PhoneticConfusability=mean(Confusion)),
  by="PhoneAttacher") %>%
  # filter(PhoneticConfusability < 0.02) %>%
  ggplot(aes(x=PhoneticConfusability,
             y=UnWeightedMean,
             color=Type)) +
  geom_point() +
  facet_wrap(~Type,scales="free") +
  geom_smooth(method="lm",formula = "y ~ x")
```



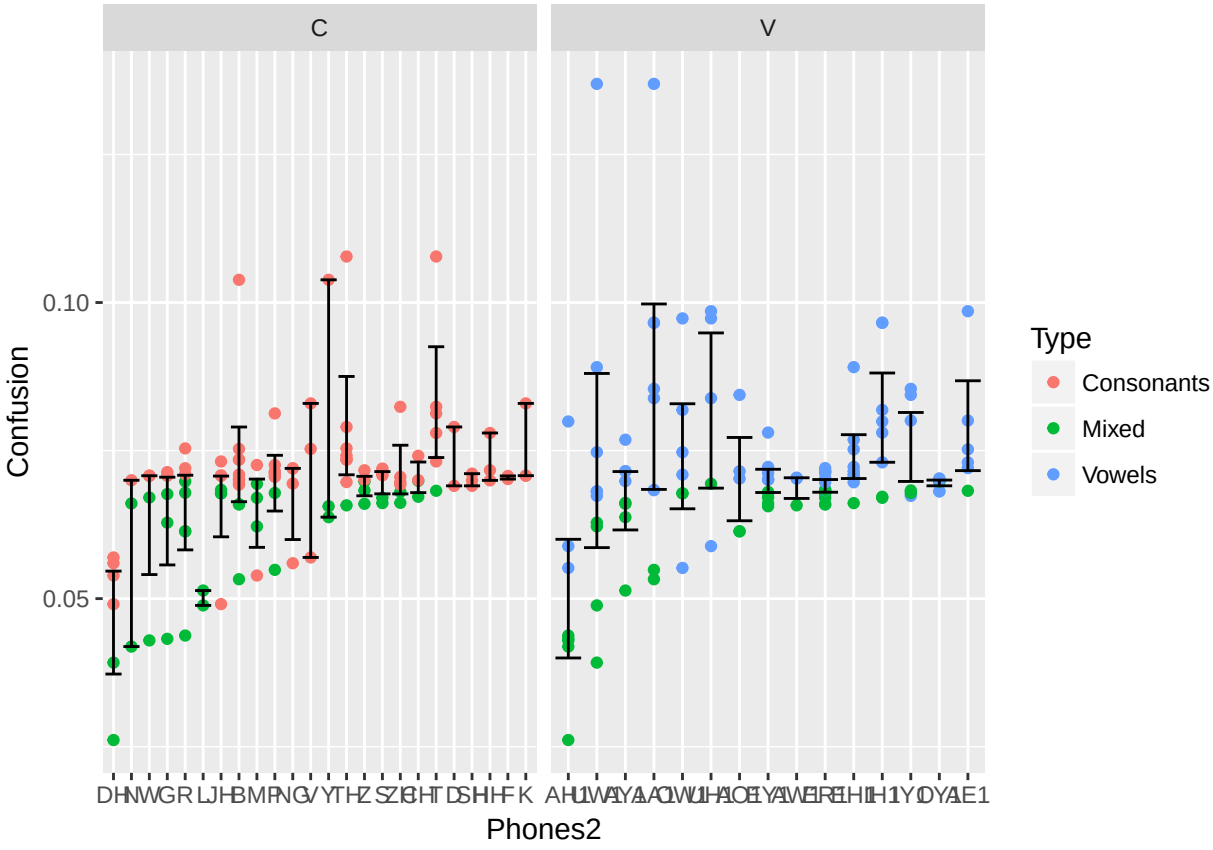

```
high_snr_swaps %>%
  filter(Type!="Baseline") %>%
  rename(TokenConfusability=WeightedMean,
         TypeConfusability=UnWeightedMean) %>%
  tidyr::gather(ConfusionType,Confusion,c(TokenConfusability,TypeConfusability)) %>%
  mutate(FirstPhone=purrr::map(Phones, ~paste0(.[[1]][[1]],collapse=",")) %>%
             purrr::simplify(),
         SecondPhone=purrr::map(Phones, ~paste0(.[[1]][[2]],collapse=",")) %>%
             purrr::simplify(),
         SonDiff = purrr::map2(FirstPhone, SecondPhone,
                               ~abs(sonority_df[sonority_df$Phone==.x,]$SonLevel -
                                   sonority_df[sonority_df$Phone==.y,]$SonLevel)) %>%
           purrr::simplify()) %>%
  filter(Type!="Vowels") %>%
  ggplot(aes(x=SonDiff,y=Confusion,color=Type,group=ConfusionType)) +
  facet_wrap(~ConfusionType,scales="free") +
  geom_point() + geom_smooth(method="lm",formula=y~x)
```




```
high_snr_swaps %>%
  filter(Type!="Baseline") %>%
  rename(TokenConfusability=WeightedMean,
         TypeConfusability=UnWeightedMean) %>%
  tidyr::gather(ConfusionType,Confusion,c(TokenConfusability,TypeConfusability)) %>%
  mutate(FirstPhone=purrr::map(Phones, ~paste0(.[[1]][[1]],collapse=",")) %>%
             purrr::simplify(),
         SecondPhone=purrr::map(Phones, ~paste0(.[[1]][[2]],collapse=",")) %>%
             purrr::simplify(),
         SonDiff = purrr::map2(FirstPhone, SecondPhone,
                               ~ifelse(sonority_df[sonority_df$Phone==.x,]$SonType ==
                                   sonority_df[sonority_df$Phone==.y,]$SonType,1,0)) %>%
           purrr::simplify()) %>%
  # filter(Type!="Vowels") %>%
  ggplot(aes(x=SonDiff,y=Confusion,color=Type,group=ConfusionType)) +
  facet_wrap(~ConfusionType,scales="free") +
  geom_point() + geom_smooth(method="lm",formula=y~x)
```




```
high_snr_swaps %>%
  filter(Type!="Baseline") %>%
  rename(TokenConfusability=WeightedMean,
         TypeConfusability=UnWeightedMean) %>%
  tidyr::gather(ConfusionType,Confusion,c(TokenConfusability,TypeConfusability)) %>%
  mutate(Phones=purrr::map(Phones, ~paste0(.[[1]][[1]],",",.[[1]][[2]],collapse="")) %>%
             purrr::simplify(),
         Phones2=Phones) %>%
  select(-data,-filename) %>%
  tidyr::separate_rows(Phones2,sep=",") %>%
  mutate(PhoneSet = purrr::map(Phones,~strsplit(.,",")),
         OtherPhone=purrr::map2(PhoneSet, Phones2, ~.x[[1]][.x[[1]] != .y]) %>%
             purrr::simplify()) %>%
  mutate(PhType =  assign_phones(Phones2,    consonants = consonants, vowels=vowels),
         PhType2 = assign_phones(OtherPhone ,consonants = consonants, vowels=vowels)) %>%
  filter(ConfusionType=="TokenConfusability") %>%
  arrange(PhType,Confusion) %>%
  mutate(Phones2=factor(Phones2,levels=unique(Phones2))) %>%
  ggplot(aes(x=Phones2,y=Confusion,group=Phones2,color=Type))+
  geom_point() +
  stat_summary(fun.data = mean_cl_boot,geom="errorbar",color="black")+
  facet_wrap(~PhType,scales="free_x",ncol=2)
```


C
V
0.10
0.05
Confusion
Phones2
Type
Consonants
Mixed
Vowels