# acoustic_analysis

"

```
cur_exp = "exp2"
features = c("duration", "meanIntensity", "meanpit")
# info = c('participant','verb','condition', 'word', 'word_num')
info = c('participant','item_id','location_condition', 'word', 'word_num')
bRemove_outliers = 0
# I have experimented with removing outliers, it doesn't have much effect on duration, some people with
```

**This the analysis for exp2. The parameters of all exps can be seen at https://github.com/Xinzhu-Fang/prosody_study_exp/blob/master/tAll_exps.csv.**

**The trial-by-trial design of this exp can be seen at https://github.com/Xinzhu-Fang/prosody_study_exp/blob/master/exp2/tAll_trials.csv**

```
tAll_trials = read.csv(file.path('..', cur_exp, 'tAll_trials.csv'))


df0 = read.csv(paste0('measure_', cur_exp, '.csv'), header = T)
df0$location_condition = NA
df0$item_id = NA

for (iR in 1:nrow(df0)){
  df0$location_condition[iR] =  as.character(tAll_trials[tAll_trials$trial_id == df0$trialId[iR],'locati
  df0$item_id[iR] =  as.character(tAll_trials[tAll_trials$trial_id == df0$trialId[iR],'filler_or_item_id
  df0$present_num[iR] = as.numeric(rownames(tAll_trials[tAll_trials$trial_id == df0$trialId[iR],]))
}

df1 = df0[startsWith(df0$item_id, "item"),]

# df0 = read.csv("measure_nonrhyming_84total_60No_24Yes_20181210.csv", header  = T)
# df0 = transform(df0,trialId=as.numeric(trialId))
# sort(df0$trialId, decreasing = FALSE)
# colnamesC(df1)

df2 = df1[df1$word != 'sp',]
# code for word_num
df2 <- df2 %>%
  dplyr::group_by(participant, trialId) %>%
  # dplyr::group_by(participant, question, trialId) %>%
  dplyr::mutate(word_num=1:dplyr::n()) %>%
  dplyr::select(c(info, features))
```

```
## Adding missing grouping variables: `trialId`
```

**29 workers and 779 trials are included in this analysis.**

```
# write.csv(df2,'newdf.csv')
# code for getting Nth instance of question
# nthdf <- df1 %>%
```

```r
#   group_by(participant,Verb, question, condition, word_num) %>%
#   mutate(Appearance=1:n())
#write.csv(nthdf,'nthdf.csv')

# subsetting it to relevant Nth appearance
# workingdf <- nthdf %>%
#   filter (Appearance == 2)
#
# write.csv(workingdf,'workingdf2.csv')

normalize_data = function(df, remove_outliers){
  for(col_name in features){
    if(!is.numeric(df[[col_name]])){
      df[[col_name]] = as.numeric(df[[col_name]])
    }
    df[[col_name]] = scale(df[[col_name]])
    # there is surge of na after the first colling of the above line. tested by print(sum(is.na(df_Agen
    # print(sum(is.na(df_Agent)))

  }
  for(col_name in features){

    if(remove_outliers){
      df = df[df[[col_name]]>-2 & df[[col_name]]<2,]
      # print(sum(is.na(df_Agent)))

    }
  }
  return(df)
}

process_data_with_yes = function(df){

  if(cur_exp == "exp4"){
    df_Agent = df[(df$location_condition=='Agent' | df$location_condition=='Control') & df$word_num=='2
  # df_Agent inheri row hum from df

    df_Verb = df[(df$location_condition=='Agent'| df$location_condition=='Control') & df$word_num=='4',]

    df_Patient = df[(df$location_condition=='Agent'| df$location_condition=='Control') & df$word_num=='5
  } else{
    df_Agent = df[(df$location_condition=='Agent' | df$location_condition=='Control') & df$word_num=='2
  # df_Agent inheri row hum from df

    df_Verb = df[(df$location_condition=='Verb'| df$location_condition=='Control') & df$word_num=='4',]

    df_Patient = df[(df$location_condition=='Patient'| df$location_condition=='Control') & df$word_num=

  }


  # print(sum(is.na(df_Agent)))
```

```
  # relevant_columns = c('participant','verb','condition','duration','meanIntensity','meanpit')
  # df_Agent = df_Agent[relevant_columns]
  # df_Verb = df_Verb[relevant_columns]
  # df_Patient = df_Patient[relevant_columns]
  print(sum(is.na(df[df$word != 'sp',])))
  # df1[(df1$meanpit == '--undefined--') && (df1$word != 'sp'),]
  # it seems that the only undefined is meanpitch for sp

  # print(df_Verb)

  df_Verb = normalize_data(df_Verb, bRemove_outliers)
  df_Agent = normalize_data(df_Agent, bRemove_outliers)
  df_Patient = normalize_data(df_Patient, bRemove_outliers)
  # print(sum(is.na(df_Agent)))



  # return(list(df_Agent_duration, df_Agent_meanIntensity, df_Agent_meanpit, df_Patient_duration, df_Pa
  return(list(df_Verb, df_Agent, df_Patient))
}

process_data_without_yes = function(df){
  df_Agent = df[ df$location_condition!='Control' & df$word_num=='2',]
  # df_Agent inheri row hum from df

  df_Verb = df[ df$location_condition!='Control' & df$word_num=='4',]


  df_Patient = df[ df$location_condition!='Control' & df$word_num=='5',]

  df_Agent$location_condition = mapvalues(df_Agent$location_condition, from=c("Patient", "Verb"), to=c(

  df_Verb$location_condition = mapvalues(df_Verb$location_condition, from=c("Agent", "Patient"), to=c('

  df_Patient$location_condition = mapvalues(df_Patient$location_condition, from=c("Agent", "Verb"), to=

  # print(sum(is.na(df_Agent)))

  # relevant_columns = c('participant','verb','condition','duration','meanIntensity','meanpit')
  # df_Agent = df_Agent[relevant_columns]
  # df_Verb = df_Verb[relevant_columns]
  # df_Patient = df_Patient[relevant_columns]
  print(sum(is.na(df[df$word != 'sp',])))
  # df1[(df1$meanpit == '--undefined--') && (df1$word != 'sp'),]
  # it seems that the only undefined is meanpitch for sp

  # print(df_Verb)

  df_Verb = normalize_data(df_Verb, bRemove_outliers)
  df_Agent = normalize_data(df_Agent, bRemove_outliers)
  df_Patient = normalize_data(df_Patient, bRemove_outliers)
  # print(sum(is.na(df_Agent)))
```

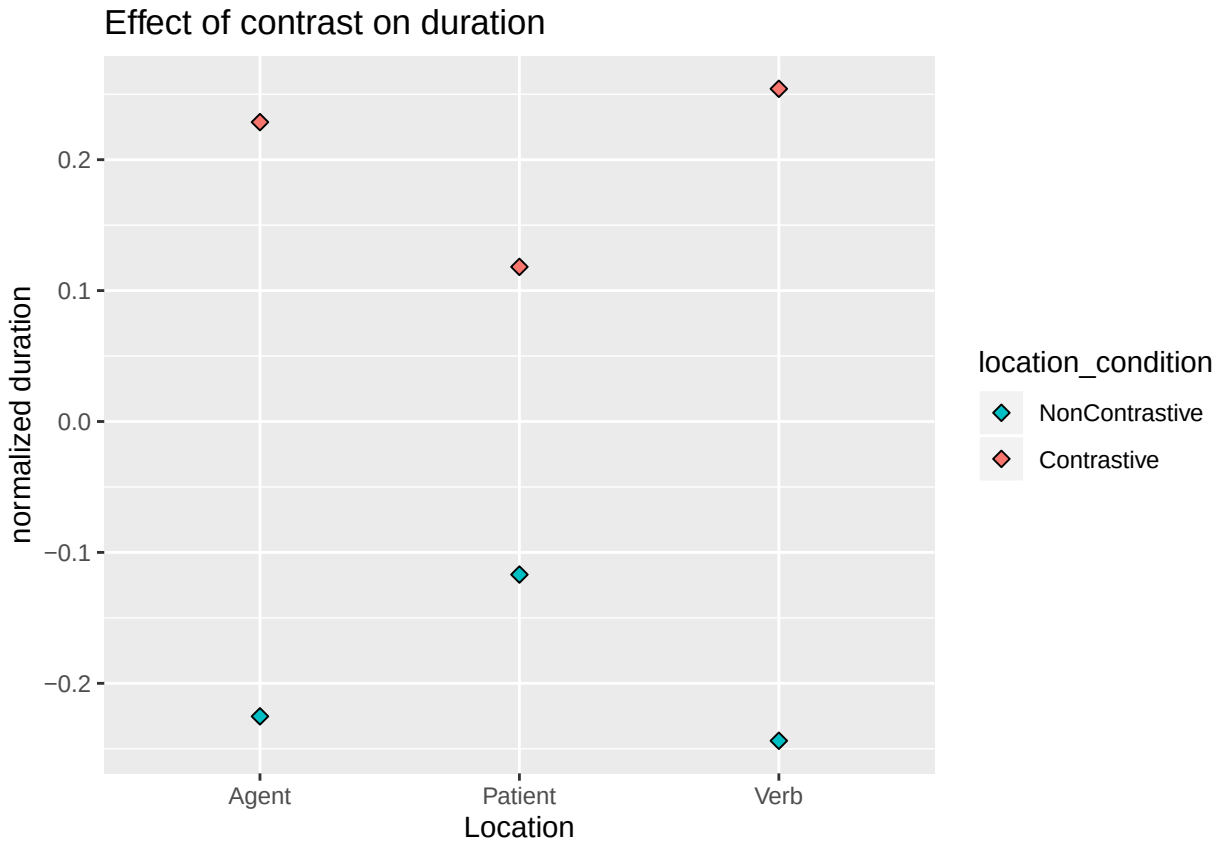
```
  # return(list(df_Agent_duration, df_Agent_meanIntensity, df_Agent_meanpit, df_Patient_duration, df_Pa
  return(list(df_Verb, df_Agent, df_Patient))
}



c(df_Verb, df_Agent, df_Patient) %<-% process_data_with_yes(df2)
```

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

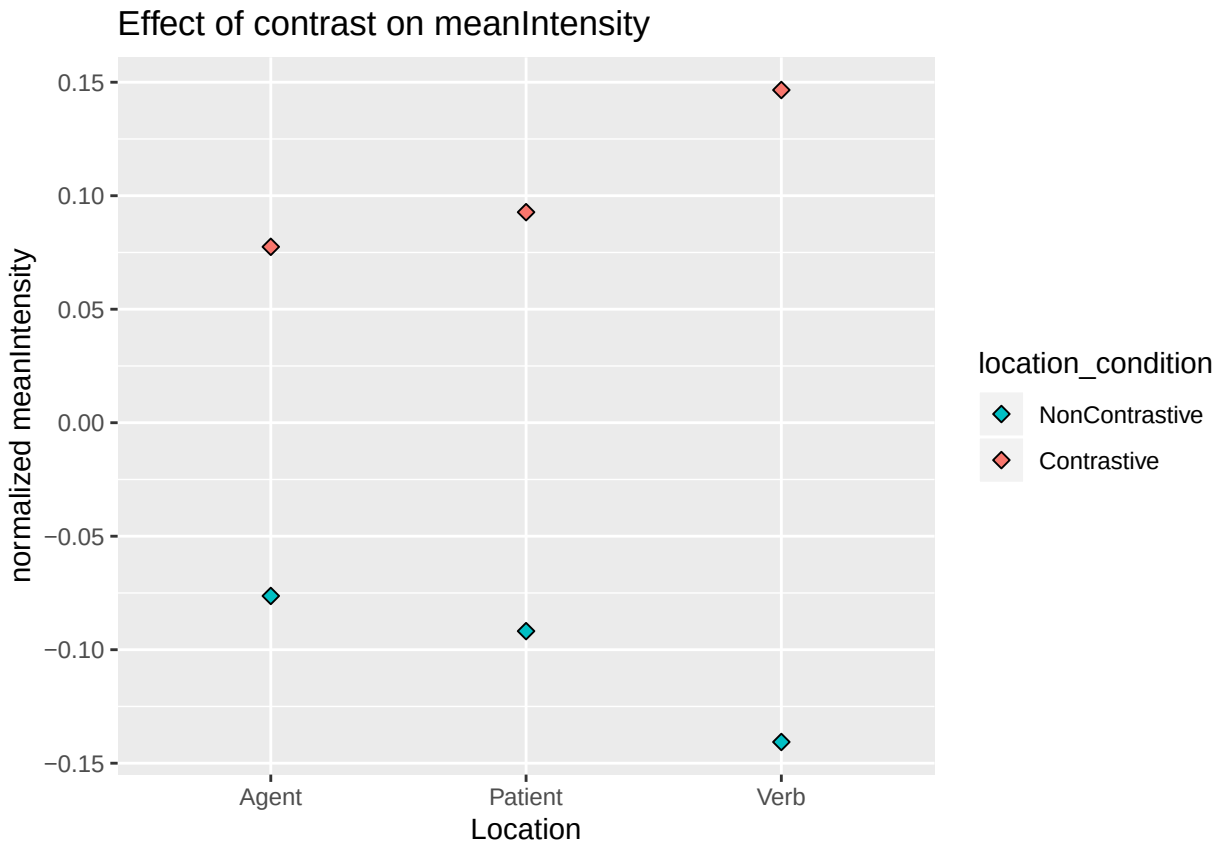
```
# c(df_Verb, df_Agent, df_Patient) %<-% process_data_without_yes(df2)



combine_datasets = function(Agent,Verb,Patient){
  Agent$condition = mapvalues(Agent$location_condition,c('Agent'),c('contrast'))
  Verb$condition = mapvalues(Verb$location_condition,c('Verb'),c('contrast'))
  Patient$condition = mapvalues(Patient$location_condition,c('Patient'),c('contrast'))

  Agent$Location = 'Agent'
  Verb$Location = 'Verb'
  Patient$Location = "Patient"

  return(rbind(Agent,Verb,Patient))
}

summarize_data = function(d, feature){
  # http://www.cookbook-r.com/Graphs/Plotting_means_and_error_bars_(ggplot2)/
  return(summarySE(d,measurevar=feature ,groupvars=c('Location','condition')))
}
# plot_data = function(d,feature, title){
#   print(ggplot(d, aes(x=Location, y=get(feature), fill=condition)) +
#           geom_bar(position=position_dodge(), stat="identity") +
#           geom_errorbar(aes(ymin=get(feature)-ci, ymax=get(feature)+ci),
#                         width=.2,
#                         position=position_dodge(.9))+
#           xlab("Location") +
#           ylab(paste0("normalized ", feature)) +
#           scale_fill_hue(name="location_condition",
#                          breaks=c("Control", "contrast"),
#                          labels=c("NonContrastive", "Contrastive")) +
#           ggtitle(title))
# }
plot_data = function(d,feature, title){
  print(ggplot(d, aes(x=Location, y=get(feature), fill=condition)) +
          geom_point(size=2, shape=23) +

          xlab("Location") +
          ylab(paste0("normalized ", feature)) +
          scale_fill_hue(name="location_condition",
                         breaks=c("Control", "contrast"),
                         labels=c("NonContrastive", "Contrastive")) +
          ggtitle(title))
```

```
}
for (iF in features){
  print(iF)

  combined_dataset = combine_datasets(df_Agent, df_Verb, df_Patient)
  summarized_dataset= summarize_data(combined_dataset, iF)


  plot_data(summarized_dataset,iF, title= paste0('Effect of contrast on ', iF))
}
```

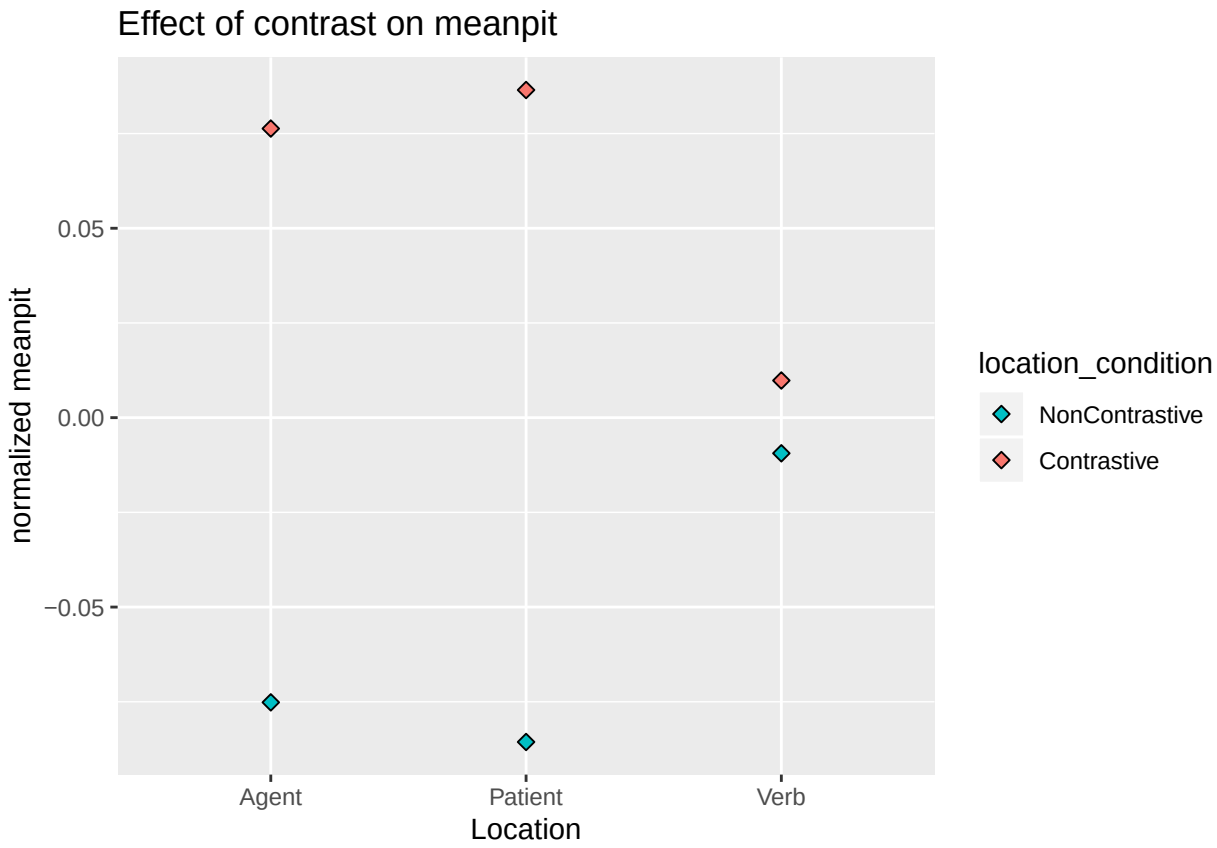
```
## [1] "duration"
```




```
## [1] "meanIntensity"
```


Effect of contrast on meanIntensity
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
normalized meanIntensity
Agent
Patient
Verb
Location
location_condition
NonContrastive
Contrastive

```
## [1] "meanpit"
```




```
run_regression = function(location,observation){
  cat("  \n###", observation, "of", location, "  \n")
  r = lmer(get(observation) ~ location_condition + (1 + location_condition|participant) + (1 + location_
  # r = lmer(get(observation) ~ location_condition  + (1 + location_condition | item_id), data=get(past
  print(summary(r))
  summary(r)
  cat("  \n")
}



# for (iF in features){
#   run_regression("Agent",iF)
#
#
#   run_regression("Patient", iF)
#
#   run_regression("Verb", iF)
#
#
#
# }
# r = lmer(get(observation) ~ condition + (1 | participant) + (1 | verb), data=df)
```