'Strand_Specific_Effect_Size_Extraction.Rmd

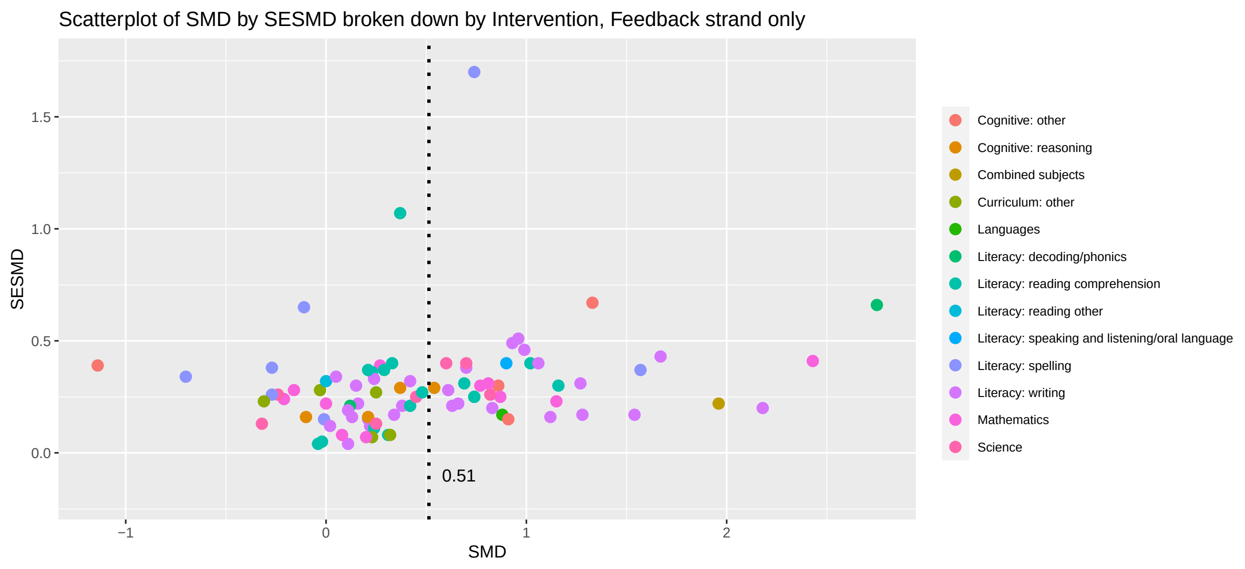
Jonathan Reardon

```
# import necessary libraries
import json
from collections import Counter
from pprint import pprint
from matplotlib import pyplot as plt
import pandas as pd
plt.style.use('ggplot')

# import dataset
with open('/home/jon/json/Batch1.json') as f:
    data=json.load(f)

####################################################
### GET STRAND LABELS AND KEYS FROM TOP OUTER LAYER
####################################################

def get_strand_info():
    '''
    a function that returns
    a dict containing strand labels
    and corresponding attribute ids
    '''
    strands={}
    for counter, element in enumerate(data["CodeSets"][0]["Attributes"]["AttributesList"]):
        attribute_name=(data["CodeSets"][0]["Attributes"]["AttributesList"][counter]["AttributeName"])
        attribute_id=(data["CodeSets"][0]["Attributes"]["AttributesList"][counter]["AttributeId"])
        strands.update( {attribute_id:attribute_name} )
    return strands

######################################
### DISPLAY STRAND SUMMARY INFORMATION
######################################

def get_strand_summary():
    '''
    A function that produces a basic
    summary of strand study counts
    and a graph to display them
    '''
    global counts, strand_title
    strand_overview=[]
    for element in range(len(data["References"])):
        for key, value in strands.items():
            for section in range(len(data["References"][element]["Codes"])):
                if key == data["References"][element]["Codes"][section]["AttributeId"]:
                    a=(data["References"][element]["ItemId"])
                    b=(data["References"][element]["Title"])
                    strand_overview.append([value, key, a, b])

    strand_title=[]
    for element in strand_overview:
        strand_title.append(element[0])

    counts = Counter(strand_title)
    pprint(counts)

#######################################
### GET THE ID FOR OUR STRAND OF CHOICE
#######################################

def get_strand_value(strand_label):
    '''
    A function that takes in a
    strand name and returns
    the strand ID
    '''
    for key, value in strands.items():
        if value == strand_label:
            return key, value

######################################################
###  GET EFFECT SIZE INFO FROM STRAND SPECIFIC STUDIES
######################################################

def get_data(strand_key, strand_value, outcome_choice):
    '''
    A function that accepts a strand id and a variable of
    interest and returns a list of that id and the variable
    values.
    '''
    outcome_studies=[]

    # iterate over each section of 'references'
    for section in range(len(data["References"])):
        # iterate over each study within each section of 'references'
        for study in range(len(data["References"][section]["Codes"])):
            # check each study to see if strand id is present
            if strand_id[0] == data["References"][section]["Codes"][study]["AttributeId"]:
                if "Outcomes" in data["References"][section]:
                    if data["References"][section]["Outcomes"][0]["OutcomeText"] == outcome_choice:
                        outcome_id=((data["References"][section]["Outcomes"][0]["OutcomeId"]))
                        outcome_type=(data["References"][section]["Outcomes"][0]["ShortTitle"])
                        outcome_text=(data["References"][section]["Outcomes"][0]["OutcomeText"])
                        SMD=(data["References"][section]["Outcomes"][0]["SMD"])
                        SESMD=(data["References"][section]["Outcomes"][0]["SESMD"])
                        year=(data["References"][section]["Year"])
                        intervention=(data["References"][section]["Outcomes"][0]["InterventionText"])
                        outcome_studies.append([strand_key, strand_value, outcome_id, outcome_text, outc

    # display number of studies found within selected strand
    print('Number of studies within strand {}: {}'.format(strand_value, len(outcome_studies)), "\n")

    pd.set_option('display.max_rows', 15)
    pd.set_option('display.max_columns', 15)

    # convert data list to pandas dataframe for viewing
    df_primary = pd.DataFrame(outcome_studies, columns=['AttributeId', 'Strand', 'OutcomeId', 'OutcomeTy

    # round effect sizes to two decimal points
    df_primary.loc[:, "SMD"] = df_primary["SMD"].astype(float).round(2)
    df_primary.loc[:, "SESMD"] = df_primary["SESMD"].astype(float).round(2)

    return df_primary

strands = get_strand_info()
get_strand_summary()
```

```
strand_id = get_strand_value("Feedback")
feedback = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
feedback_df <- data.frame(py$feedback)

feedback_df$Intervention <- as.character(feedback_df$Intervention)
feedback_df$Intervention[feedback_df$Intervention==""] <- NA
feedback_df$Intervention <- as.factor(feedback_df$Intervention)

feedback_mean_SMD <- mean(feedback_df$SMD, na.rm=TRUE)
feedback_mean_SESMD <- mean(feedback_df$SESMD, na.rm=TRUE)

feedback_mean_SMD
feedback_mean_SESMD

View(feedback_df)
```

```
ggplot(data=subset(feedback_df, !is.na(Intervention)), aes(SMD, SESMD, color=Intervention)) +
    geom_point(alpha=1, na.rm=TRUE, size=3) +
    theme_grey() +
    geom_vline(xintercept=feedback_mean_SMD, linetype="dotted", color="black", size=1) +
    theme(legend.title = element_text(color = "black", size = 10),
          legend.text = element_text(color = "black", size = 8)) +
    theme(legend.position="right") +
    guides(fill=guide_legend(nrow=5, byrow=TRUE)) +
    theme(legend.title=element_blank()) +
    annotate(geom="text", x=feedback_mean_SMD+.15, y=-.1, label=round(feedback_mean_SMD, 2), color="blac
    ylim(-0.2, 1.75) +
    ggtitle("Scatterplot of SMD by SESMD broken down by Intervention, Feedback strand only")
```

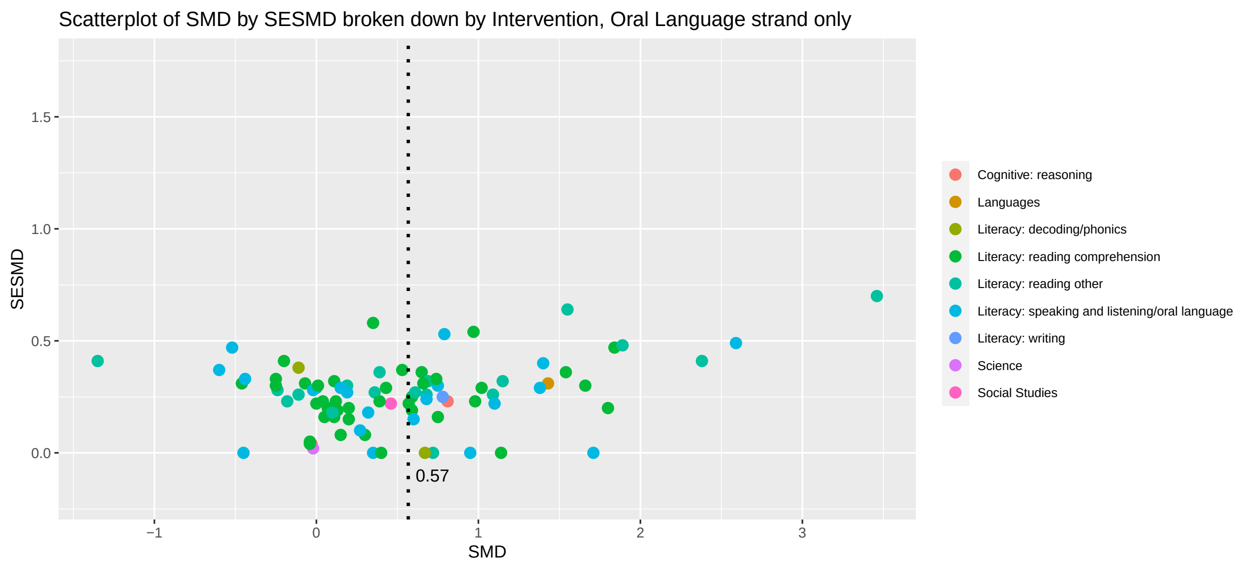


```
strand_id = get_strand_value("Oral language interventions")
oral_lang = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
print(oral_lang.head(10))
```

```
oral_lang_df <- data.frame(py$oral_lang)

oral_lang_df$Intervention <- as.character(oral_lang_df$Intervention)
oral_lang_df$Intervention[oral_lang_df$Intervention==""] <- NA
oral_lang_df$Intervention <- as.factor(oral_lang_df$Intervention)

oral_lang_mean_SMD <- mean(oral_lang_df$SMD, na.rm=TRUE)
oral_lang_mean_SESMD <- mean(oral_lang_df$SESMD, na.rm=TRUE)

oral_lang_mean_SMD
oral_lang_mean_SESMD

View(oral_lang_df)
```

```
ggplot(data=subset(oral_lang_df, !is.na(Intervention)), aes(SMD, SESMD, color=Intervention)) +
    geom_point(alpha=1, na.rm=TRUE, size=3) +
    theme_grey() +
    geom_vline(xintercept=oral_lang_mean_SMD, linetype="dotted", color="black", size=1) +
    theme(legend.title = element_text(color = "black", size = 10),
          legend.text = element_text(color = "black", size = 8)) +
    theme(legend.position="right") +
    guides(fill=guide_legend(nrow=5, byrow=TRUE)) +
    theme(legend.title=element_blank()) +
    annotate(geom="text", x=oral_lang_mean_SMD+.15, y=-.1, label=round(oral_lang_mean_SMD, 2), color="bl
    ylim(-0.2, 1.75) +
    ggtitle("Scatterplot of SMD by SESMD broken down by Intervention, Oral Language strand only")
```



```
strand_id = get_strand_value("Oral language interventions")
oral_lang = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
strand_id = get_strand_value("Feedback")
feedback = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
strand_id = get_strand_value("Peer tutoring")
peer_tut = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
strand_id = get_strand_value("Teaching assistants")
teaching_assist = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
strand_id = get_strand_value("Small group tuition")
small_group = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

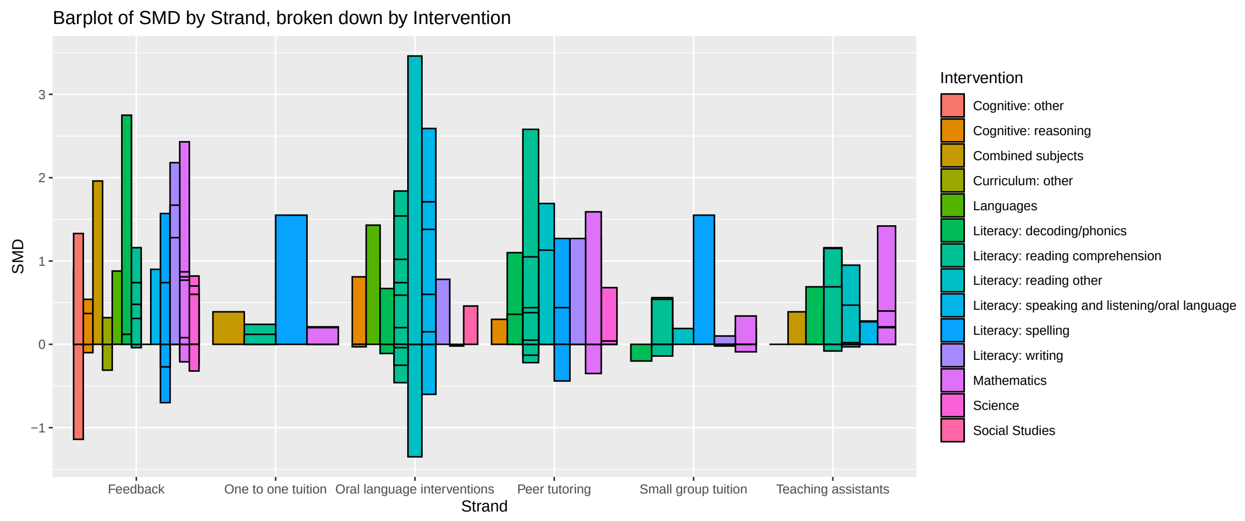
```
strand_id = get_strand_value("One to one tuition")
one_to_one = get_data(strand_id[0], strand_id[1], "Primary outcome")
```

```
master_df = pd.concat([oral_lang, feedback, peer_tut, teaching_assist, small_group, one_to_one])
master_df
```

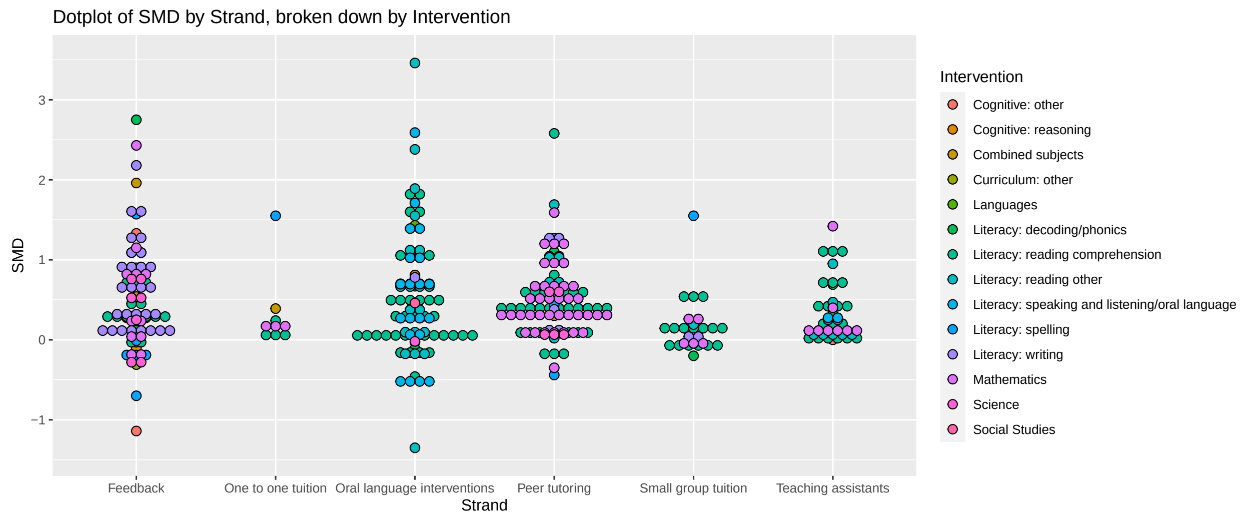
```
master_df <- data.frame(py$master_df)
```

```
## Warning in py_to_r.pandas.core.frame.DataFrame(x): index contains duplicated
## values: row names not set
```

```
master_df$Intervention[master_df$Intervention==""] <- NA

filter(master_df, !is.na(Intervention)) %>%
  ggplot(aes(fill=Intervention, y=SMD, x=Strand)) +
  geom_bar(position="dodge", stat="identity", color="black", na.rm=TRUE) +
  ggtitle("Barplot of SMD by Strand, broken down by Intervention")
```

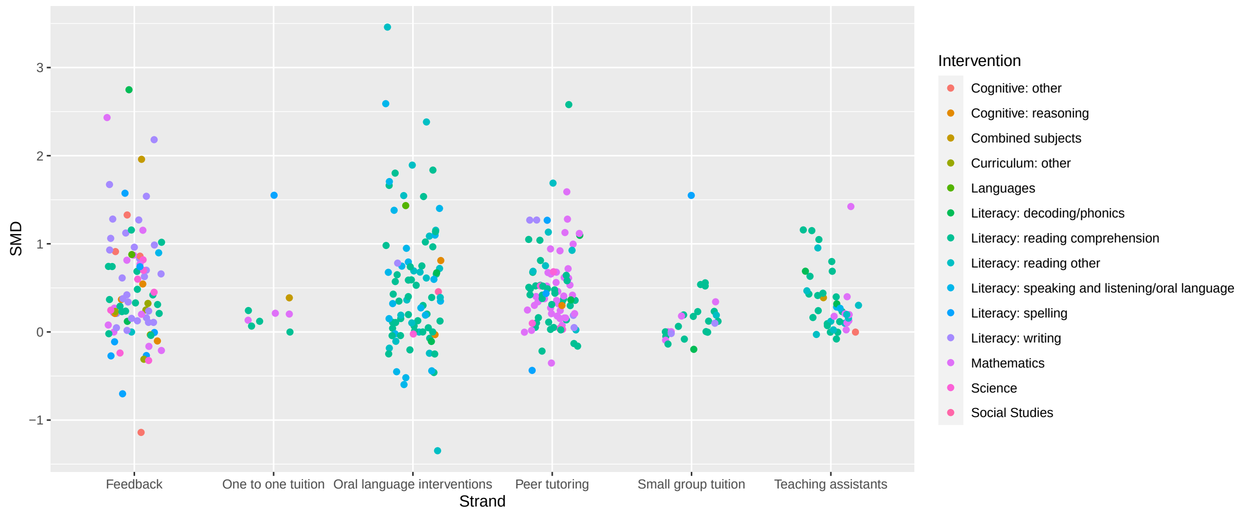


```r
filter(master_df, !is.na(Intervention)) %>%
  ggplot(aes(fill=Intervention, y=SMD, x=Strand)) +
  geom_dotplot(binaxis='y', stackdir='center', dotsize=.6, binwidth=.2, na.rm=TRUE) +
  ggtitle("Dotplot of SMD by Strand, broken down by Intervention")
```



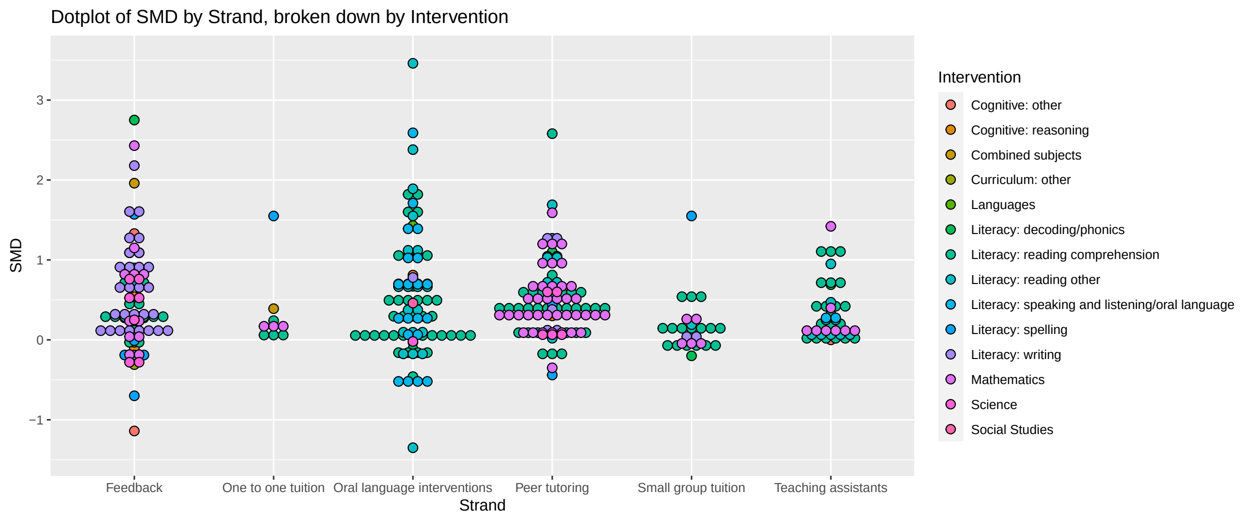
```r
filter(master_df, !is.na(Intervention)) %>%
  ggplot(aes(y=SMD, x=Strand, size=Intervention, color=Intervention)) +
  geom_jitter(shape=16, position=position_jitter(.2), size=2, shape=21)
```

```
## Warning: Duplicated aesthetics after name standardisation: shape
```



"

```r
filter(master_df, !is.na(Intervention)) %>%
  ggplot(aes(fill=Intervention, y=SMD, x=Strand)) +
  geom_dotplot(binaxis='y', stackdir='center', dotsize=.6, binwidth=.2, na.rm=TRUE) +
  ggtitle("Dotplot of SMD by Strand, broken down by Intervention")
```


Dotplot of SMD by Strand, broken down by Intervention
SMD
Feedback
One to one tuition
Oral language interventions
Peer tutoring
Small group tuition
Teaching assistants
Strand
Intervention
Cognitive: other
Cognitive: reasoning
Combined subjects
Curriculum: other
Languages
Literacy: decoding/phonics
Literacy: reading comprehension
Literacy: reading other
Literacy: speaking and listening/oral language
Literacy: spelling
Literacy: writing
Mathematics
Science
Social Studies