BF_Analysis_final

Ana Baciero

27/5/2021

Data analysis

```
library("papaja")
library(tidyverse)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.15.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(RColorBrewer)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(dplyr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
RawdataBF <- miceadds::load.Rdata2( filename="RawdataBF.RData")
CleanBF <- miceadds::load.Rdata2( filename="CleanBF.RData")
BrailleDotsTable <- miceadds::load.Rdata2( filename="BrailleDotsTable.RData")

CleanBF%>%
  summarize(n_distinct(SubjectID))
```

```
##   n_distinct(SubjectID)
## 1                    24
```

```
CleanBF%>%
  group_by(Pairs)%>%
  summarise(meanRT = mean(RT),
            meanAcc = mean(Accuracy)) -> x
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
cor(x$meanRT, x$meanAcc)
```

```
## [1] -0.499411
```

1. DISTANCE MATRICES (or confusion matrices)

Symmetrical matrix in which (ij)th element is the measure of distinction between the (i)th and the (j)th object. For dissimilarity matrices, the closer the measure is to 0, the lesser the two elements differ from each other (more confusable). The diagonal elements are usually equal to zero (or not considered) - i.e. the distinction between an object and itself is postulated as zero.

```
Different<- CleanBF%>%
  filter(Rubric == "n")

matrix.acc<-tapply(Different$Accuracy, list(Different$IT1, Different$IT2),mean)%>%
  round(3)

    # Can we make it symmetric?

    Acc.Order1 <- Different%>%
      filter(Order==1)%>%
      group_by(ItNumber,Pairs)%>%
      summarise(MAcc1 = mean(Accuracy))%>%
      ungroup()
```

```
## `summarise()` regrouping output by 'ItNumber' (override with `.groups` argument)
```

```
    Acc.Order2 <-Different%>%
      filter(Order==2)%>%
      group_by(ItNumber,Pairs)%>%
```

```
      summarise(MAcc2 = mean(Accuracy))%>%
      ungroup()
```

```
## `summarise()` regrouping output by 'ItNumber' (override with `.groups` argument)
```

```
    Acc.orders <- bind_cols(Acc.Order1,Acc.Order2)%>%
      mutate(difference = MAcc1 - MAcc2,
             OrderMatters = if_else(difference > (abs(.05)),"T",""))
```

```
## New names:
## * ItNumber -> ItNumber...1
## * Pairs -> Pairs...2
## * ItNumber -> ItNumber...4
## * Pairs -> Pairs...5
```

```
    t.test(Acc.orders$MAcc1,Acc.orders$MAcc2, paired = T)
```

```
##
##  Paired t-test
##
## data:  Acc.orders$MAcc1 and Acc.orders$MAcc2
## t = 0.76602, df = 324, p-value = 0.4442
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.003255397  0.007407061
## sample estimates:
## mean of the differences
##             0.002075832
```

```
    ## YES

    ## for those whose acc differ in more than 5%:
    n_acc<-Acc.orders%>%
      filter(OrderMatters=="T") #33/325 differ in more than 5% accuracy.


sym.acc.matrix <- (matrix.acc + t(matrix.acc))/2
kable(round(sym.acc.matrix,3))
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a	NA	1.000	0.905	0.959	0.963	0.981	1.000	1.000	0.938	1.000	0.943	0.979	0.940	1.000	0.962
b	1.000	NA	0.945	0.979	0.980	0.961	0.866	0.916	0.983	0.978	0.960	0.808	0.979	1.000	0.962
c	0.905	0.945	NA	0.980	1.000	1.000	1.000	0.979	0.981	0.963	0.980	0.981	1.000	1.000	1.000
d	0.959	0.979	0.980	NA	0.901	0.957	0.960	0.900	1.000	1.000	1.000	1.000	1.000	0.981	0.943
e	0.963	0.980	1.000	0.901	NA	1.000	1.000	0.960	1.000	0.981	1.000	0.963	1.000	1.000	1.000
f	0.981	0.961	1.000	0.957	1.000	NA	0.824	0.901	0.981	0.921	1.000	0.960	0.978	1.000	0.981
g	1.000	0.866	1.000	0.960	1.000	0.824	NA	0.942	1.000	0.944	1.000	1.000	1.000	1.000	1.000
h	1.000	0.916	0.979	0.900	0.960	0.901	0.942	NA	0.980	0.914	1.000	1.000	1.000	1.000	0.982
i	0.938	0.983	0.981	1.000	1.000	0.981	1.000	0.980	NA	0.744	1.000	0.982	1.000	1.000	1.000
j	1.000	0.978	0.963	1.000	0.981	0.921	0.944	0.914	0.744	NA	0.980	1.000	0.961	1.000	0.960
k	0.943	0.960	0.980	1.000	1.000	1.000	1.000	1.000	1.000	0.980	NA	0.980	0.940	0.980	0.961
l	0.979	0.808	0.981	1.000	0.963	0.960	1.000	1.000	0.982	1.000	0.980	NA	1.000	1.000	1.000
m	0.940	0.979	1.000	1.000	1.000	0.978	1.000	1.000	1.000	0.961	0.940	1.000	NA	0.979	1.000
n	1.000	1.000	1.000	0.981	1.000	1.000	1.000	1.000	1.000	1.000	0.980	1.000	0.979	NA	0.701
o	0.962	0.962	1.000	0.943	1.000	0.981	1.000	0.982	1.000	0.960	0.961	1.000	1.000	0.701	NA
p	0.980	0.979	1.000	1.000	1.000	0.960	0.962	1.000	0.982	0.962	0.981	0.940	0.942	0.980	1.000
q	1.000	0.981	0.980	1.000	1.000	0.960	0.940	0.962	0.979	1.000	1.000	0.940	0.979	0.960	1.000
r	0.940	1.000	1.000	0.981	0.979	1.000	0.959	0.881	1.000	1.000	1.000	0.881	0.945	0.981	0.982
s	1.000	0.961	1.000	1.000	0.961	0.962	0.960	1.000	0.920	0.978	1.000	0.981	1.000	0.960	0.980
t	1.000	1.000	1.000	0.981	1.000	0.923	1.000	1.000	0.980	0.941	1.000	0.925	1.000	0.981	0.979
u	0.979	1.000	1.000	1.000	1.000	1.000	0.940	1.000	0.982	0.982	0.871	0.960	0.900	1.000	0.980
v	1.000	0.982	0.981	0.981	0.961	1.000	0.958	0.921	1.000	0.980	1.000	0.885	0.959	1.000	1.000
w	0.980	0.982	0.980	0.980	0.980	0.982	0.945	0.959	0.980	0.960	1.000	0.980	0.981	0.904	0.963
x	0.959	0.960	0.982	1.000	0.982	0.982	1.000	1.000	0.961	0.980	0.941	0.981	0.778	0.981	1.000
y	1.000	1.000	0.961	0.980	1.000	1.000	0.980	0.979	0.961	1.000	0.982	0.980	0.963	0.704	0.979
z	0.962	0.981	0.960	1.000	1.000	0.982	1.000	1.000	1.000	1.000	0.980	1.000	1.000	0.720	0.764

```
CorrectDifferent.norm<- Different%>%
  filter(Accuracy == 1)%>%
  mutate(RTms = RT*1000)%>%
  group_by(SubjectID)%>%
  mutate(MRT_subj = mean(RTms))%>%
  ungroup()%>%
  mutate(normTime = RTms/MRT_subj)
#Normalizing RTs so mean RT = 1 per subject (as in Courrieu et al., 2004 & Wiley et al., 2016)


matrix.rt<-tapply(CorrectDifferent.norm$normTime, list(CorrectDifferent.norm$IT1, CorrectDifferent.norm$
  round(3)

    # Can we make it symmetric?

    RT.Order1 <- CorrectDifferent.norm%>%
      filter(Order==1)%>%
      group_by(ItNumber,Pairs)%>%
      summarise(MNRT1 = mean(normTime))%>%
      ungroup()
```

```
## `summarise()` regrouping output by 'ItNumber' (override with `.groups` argument)
```

```
    RT.Order2 <- CorrectDifferent.norm%>%
      filter(Order==2)%>%
      group_by(ItNumber,Pairs)%>%
```

```
      summarise(MNRT2 = mean(normTime))%>%
      ungroup()
```

```
## `summarise()` regrouping output by 'ItNumber' (override with `.groups` argument)
```

```
    RT.orders <- bind_cols(RT.Order1,RT.Order2)%>%
      mutate(difference = MNRT1 - MNRT2,
             OrderMatters = if_else(difference > (abs(.05)),"T",""))
```

```
## New names:
## * ItNumber -> ItNumber...1
## * Pairs -> Pairs...2
## * ItNumber -> ItNumber...4
## * Pairs -> Pairs...5
```

```
    t.test(RT.orders$MNRT1,RT.orders$MNRT2, paired = T)
```

```
##
##  Paired t-test
##
## data:  RT.orders$MNRT1 and RT.orders$MNRT2
## t = -0.96583, df = 324, p-value = 0.3349
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.011512175  0.003930686
## sample estimates:
## mean of the differences
##            -0.003790745
```

```
    ## YES


sym.rt.matrix <- (matrix.rt + t(matrix.rt))/2
kable(round(sym.rt.matrix,3))
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a	NA	0.946	1.097	0.969	0.952	1.044	0.995	0.961	1.035	0.954	0.998	1.015	0.992	0.962	0.966
b	0.946	NA	1.021	1.006	0.991	1.020	1.004	1.019	1.001	1.002	0.949	1.043	0.922	1.000	0.979
c	1.097	1.021	NA	1.028	0.946	0.960	0.956	0.964	0.965	0.984	0.976	0.958	0.941	0.958	0.944
d	0.969	1.006	1.028	NA	1.128	1.099	0.988	1.018	0.984	1.046	1.034	1.002	0.970	0.966	0.956
e	0.952	0.991	0.946	1.128	NA	0.986	1.137	1.010	1.055	0.993	0.962	0.943	0.965	0.947	1.026
f	1.044	1.020	0.960	1.099	0.986	NA	1.083	1.063	0.994	1.084	1.006	1.007	0.950	0.996	0.954
g	0.995	1.004	0.956	0.988	1.137	1.083	NA	1.060	0.972	1.010	0.925	1.011	0.974	0.966	0.991
h	0.961	1.019	0.964	1.018	1.010	1.063	1.060	NA	0.998	1.129	0.945	0.991	0.940	0.998	0.987
i	1.035	1.001	0.965	0.984	1.055	0.994	0.972	0.998	NA	1.106	1.007	0.945	0.992	0.953	0.972
j	0.954	1.002	0.984	1.046	0.993	1.084	1.010	1.129	1.106	NA	0.935	0.974	0.905	1.008	0.982
k	0.998	0.949	0.976	1.034	0.962	1.006	0.925	0.945	1.007	0.935	NA	0.980	1.059	1.038	1.029
l	1.015	1.043	0.958	1.002	0.943	1.007	1.011	0.991	0.945	0.974	0.980	NA	1.006	0.949	0.965
m	0.992	0.922	0.941	0.970	0.965	0.950	0.974	0.940	0.992	0.905	1.059	1.006	NA	0.993	0.998
n	0.962	1.000	0.958	0.966	0.947	0.996	0.966	0.998	0.953	1.008	1.038	0.949	0.993	NA	1.172
o	0.966	0.979	0.944	0.956	1.026	0.954	0.991	0.987	0.972	0.982	1.029	0.965	0.998	1.172	NA
p	0.946	0.994	0.953	0.956	0.930	0.942	1.043	0.988	1.006	0.917	1.000	1.055	0.992	1.043	0.988
q	1.145	0.970	1.022	0.944	0.979	1.084	1.088	0.958	0.950	0.972	0.954	0.979	0.974	1.019	1.035
r	0.942	0.999	0.984	0.984	0.994	1.006	1.029	1.008	0.982	0.992	1.009	1.063	0.986	1.038	1.014
s	0.982	1.000	0.929	0.966	0.968	1.070	0.975	1.008	1.034	0.991	0.988	0.974	0.990	1.001	1.202
t	1.008	0.987	0.966	0.998	0.960	0.968	0.994	0.981	0.979	0.976	0.959	0.995	0.990	1.082	0.981
u	0.991	0.930	0.976	0.923	0.941	0.978	0.978	0.958	0.950	0.976	1.228	1.029	1.094	1.032	1.008
v	0.951	0.954	0.974	0.970	0.932	0.954	0.962	0.996	0.944	0.982	0.976	1.014	0.965	0.986	0.970
w	0.968	0.969	0.977	0.999	1.026	0.993	0.967	1.039	0.959	1.006	1.036	0.998	0.970	1.018	1.059
x	0.944	0.969	0.944	0.993	0.967	0.966	0.972	0.988	0.970	0.957	1.000	0.995	1.070	1.034	1.029
y	1.068	0.969	0.913	0.984	0.908	0.978	1.083	0.962	0.966	0.972	0.993	0.992	1.030	1.068	1.058
z	0.947	0.975	0.961	0.963	0.978	0.979	0.989	1.002	0.998	0.968	0.988	0.979	0.964	1.151	1.286

```
#Transform similarity matrix it into a dissimilarity matrix:

diss.rt.matrix <- 1/sym.rt.matrix
kable(round(diss.rt.matrix,3))
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a	NA	1.057	0.912	1.032	1.050	0.958	1.005	1.041	0.966	1.048	1.002	0.985	1.008	1.039	1.036
b	1.057	NA	0.979	0.995	1.009	0.980	0.996	0.981	0.999	0.998	1.054	0.959	1.085	1.001	1.021
c	0.912	0.979	NA	0.973	1.058	1.042	1.046	1.037	1.036	1.016	1.025	1.044	1.063	1.044	1.059
d	1.032	0.995	0.973	NA	0.886	0.910	1.012	0.983	1.016	0.956	0.967	0.998	1.030	1.035	1.047
e	1.050	1.009	1.058	0.886	NA	1.014	0.880	0.990	0.948	1.007	1.040	1.060	1.036	1.057	0.974
f	0.958	0.980	1.042	0.910	1.014	NA	0.923	0.941	1.007	0.922	0.995	0.993	1.053	1.004	1.048
g	1.005	0.996	1.046	1.012	0.880	0.923	NA	0.943	1.028	0.990	1.081	0.989	1.027	1.036	1.009
h	1.041	0.981	1.037	0.983	0.990	0.941	0.943	NA	1.002	0.886	1.059	1.009	1.064	1.002	1.013
i	0.966	0.999	1.036	1.016	0.948	1.007	1.028	1.002	NA	0.904	0.993	1.059	1.009	1.049	1.028
j	1.048	0.998	1.016	0.956	1.007	0.922	0.990	0.886	0.904	NA	1.070	1.026	1.104	0.992	1.018
k	1.002	1.054	1.025	0.967	1.040	0.995	1.081	1.059	0.993	1.070	NA	1.021	0.944	0.963	0.972
l	0.985	0.959	1.044	0.998	1.060	0.993	0.989	1.009	1.059	1.026	1.021	NA	0.995	1.054	1.036
m	1.008	1.085	1.063	1.030	1.036	1.053	1.027	1.064	1.009	1.104	0.944	0.995	NA	1.007	1.002
n	1.039	1.001	1.044	1.035	1.057	1.004	1.036	1.002	1.049	0.992	0.963	1.054	1.007	NA	0.854
o	1.036	1.021	1.059	1.047	0.974	1.048	1.009	1.013	1.028	1.018	0.972	1.036	1.002	0.854	NA
p	1.058	1.006	1.050	1.045	1.075	1.062	0.959	1.012	0.995	1.091	1.000	0.947	1.008	0.959	1.012
q	0.873	1.031	0.979	1.060	1.021	0.922	0.919	1.043	1.053	1.029	1.048	1.021	1.027	0.981	0.966
r	1.062	1.001	1.016	1.016	1.006	0.994	0.972	0.992	1.018	1.008	0.991	0.940	1.014	0.964	0.987
s	1.018	1.000	1.076	1.035	1.033	0.934	1.026	0.992	0.968	1.009	1.012	1.027	1.010	0.999	0.832
t	0.992	1.013	1.036	1.002	1.041	1.033	1.007	1.019	1.021	1.024	1.043	1.005	1.010	0.925	1.019
u	1.009	1.075	1.025	1.083	1.063	1.022	1.023	1.044	1.053	1.025	0.815	0.972	0.914	0.969	0.992
v	1.052	1.048	1.027	1.030	1.073	1.049	1.040	1.004	1.060	1.018	1.024	0.986	1.036	1.014	1.031
w	1.034	1.032	1.024	1.001	0.974	1.007	1.034	0.962	1.043	0.994	0.965	1.003	1.030	0.983	0.944
x	1.059	1.032	1.060	1.007	1.034	1.035	1.029	1.012	1.030	1.045	1.001	1.005	0.935	0.967	0.972
y	0.936	1.032	1.095	1.016	1.101	1.022	0.923	1.040	1.035	1.028	1.007	1.008	0.970	0.936	0.946
z	1.056	1.026	1.041	1.038	1.022	1.021	1.011	0.998	1.002	1.034	1.012	1.021	1.038	0.869	0.778

2. HIERARCHICAL CLUSTERING

- Cluster: A collection of data points that exhibit two key features:
  - Similarity of points within cluster
  - Dissimilarity of points between clusters
- Hierarchical clustering is a technique that allows us to find what are the clusters within a group of people/objects. It has two approaches:
  - Agglomerative: bottom-up approach in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
  - Divisive: the reverse of the agglomerative algorithm as it is a top-down approach.
- The steps of clustering hierarchically (Johnson, 1967) :
  - Start with a matrix of pairwise distances between objects
  - Begin by assigning each item to its own cluster, so that if you have N items, you now have N clusters.
  - Find the closest (least distant) pair of items and merge them into a single cluster, so that now you have one less cluster.
    - *You can quantify how distant/similar two object's vectors are to each other with a variety of metrics*
      - Euclidean Distance: Square-root of the sum of the squared differences between each characteristic (usual)

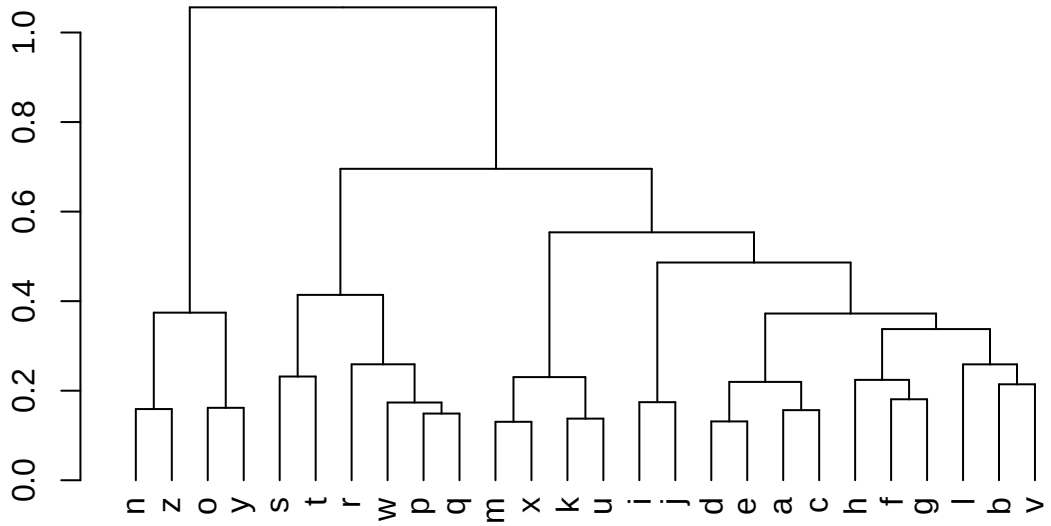
        - City Block Distance: Sum of the absolute differences between each characteristic in two vectors
        - ...
    - Look at the distances between each cluster (including the newly created cluster), and merge the two most similar into a single cluster
        * *There are different rules/methods for determining how similar an object is to a cluster, when determining if it can join*
            - Single linkage: join to the cluster that has the most similar object
            - Complete linkage: join to the cluster based on the worst case
            - Average linkage: join to cluster based on average distance
            - Ward's method: join to cluster that minimize variance within and maximize variance between clusters
    - Repeat steps 2 and 3 until all items are clustered into a single cluster with N items.

Then, inspect the dendrogram (ie., visual representation of which objects are in the same group, and when they merged) to examine what the distinct groupings

```
# ACCURACY

## Euclidian method (because of previous articles):

dist.res.acc<-dist(sym.acc.matrix, method = "euclidian")

        ## What linkage method to use

        ### define linkage methods

        m <- c( "average", "single", "complete", "ward")
        names(m) <- c( "average", "single", "complete", "ward")

        ### function to compute agglomerative coefficient:

        ac1 <- function(x) {
          cluster::agnes(dist.res.acc, method = x)$ac
        }

        ### calculate agglomerative coefficient for each clustering linkage method

        sapply(m, ac1)
```

```
##   average    single  complete      ward
## 0.6372010 0.5147231 0.7165576 0.8335939
```

```
        ## we'll use ward's method (Note that agnes(*, method="ward") corresponds to hclust(*, "ward.D2


dend.acc<- hclust(dist.res.acc, method = "ward.D2")%>%
  as.dendrogram()

plot(dend.acc)
```



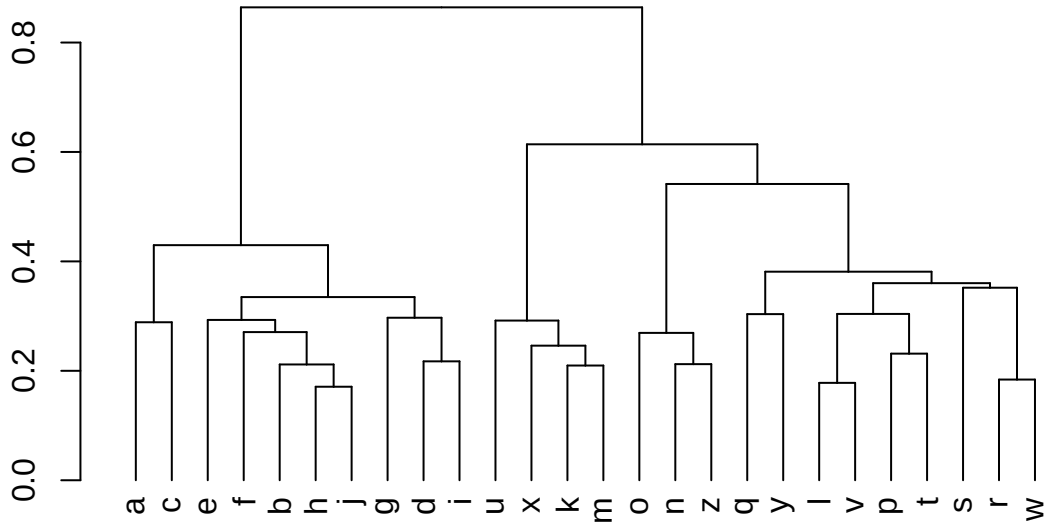

```
order.hclust(hclust(dist.res.acc, method = "ward.D2"))
```

```
##  [1] 14 26 15 25 19 20 18 23 16 17 13 24 11 21  9 10  4  5  1  3  8  6  7 12  2
## [26] 22
```

```
# RT

dist.res.rt<-dist(diss.rt.matrix, method = "euclidian")

        ## What linkage method to use

        ac2 <- function(x) {
          cluster::agnes(dist.res.rt, method = x)$ac
        }


        sapply(m, ac2)
```

```
##   average    single  complete      ward
## 0.3875329 0.2089485 0.5362902 0.7230720
```

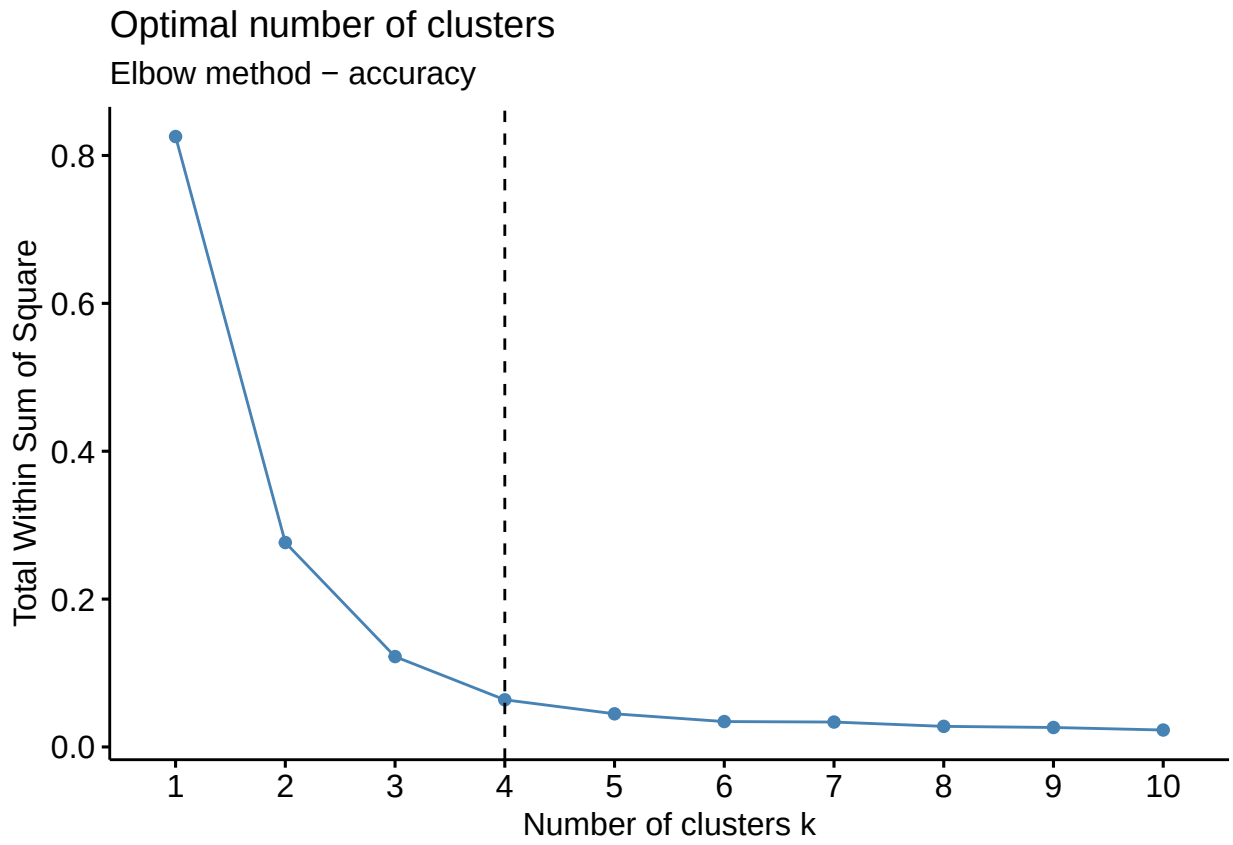
```
        ## we'll use ward's method


dend.rt<- hclust(dist.res.rt, method = "ward.D2")%>%
  as.dendrogram()

plot(dend.rt)
```



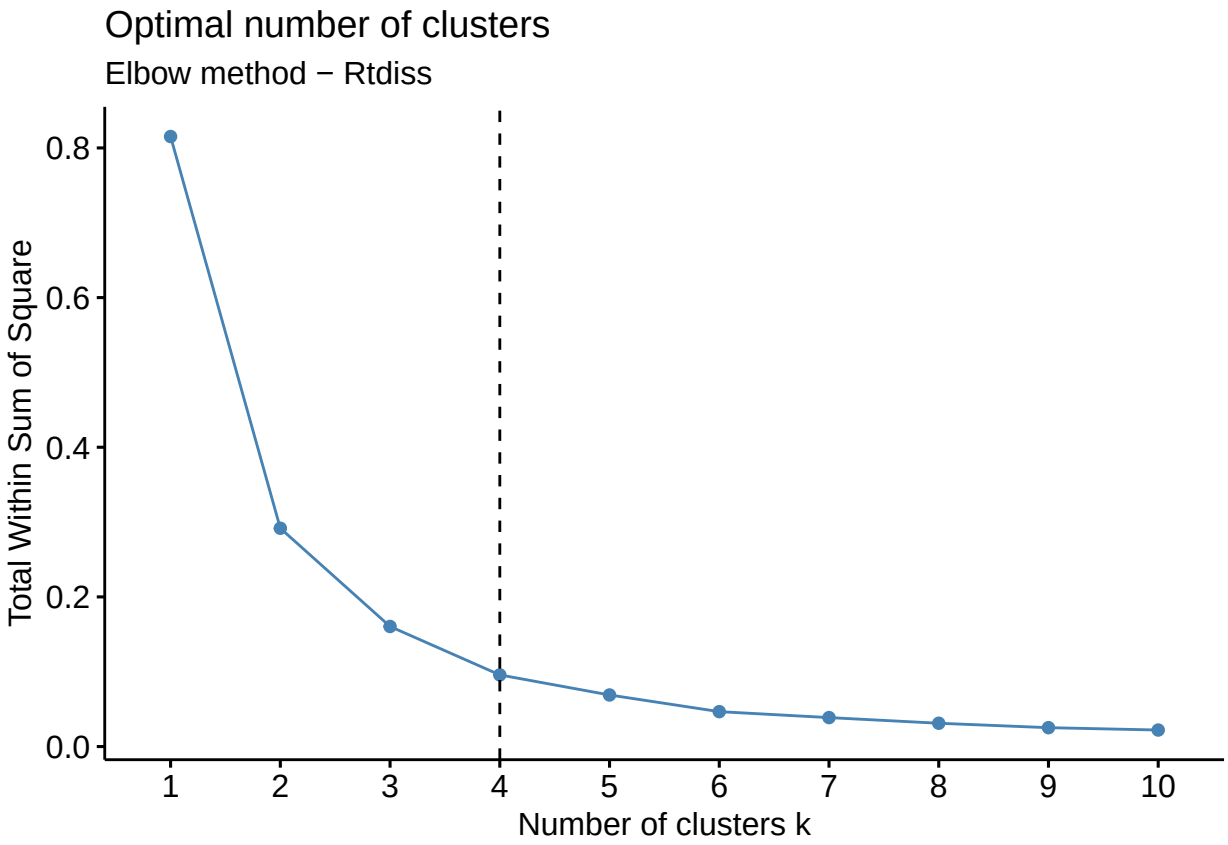

```
## Optimal number of clusters

sym.acc.matrix1<- sym.acc.matrix%>%
  gdata::lowerTriangle()%>%
  as_data_frame()
```

```
## Warning: `as_data_frame()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
fviz_nbclust(sym.acc.matrix1, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method - accuracy")
```

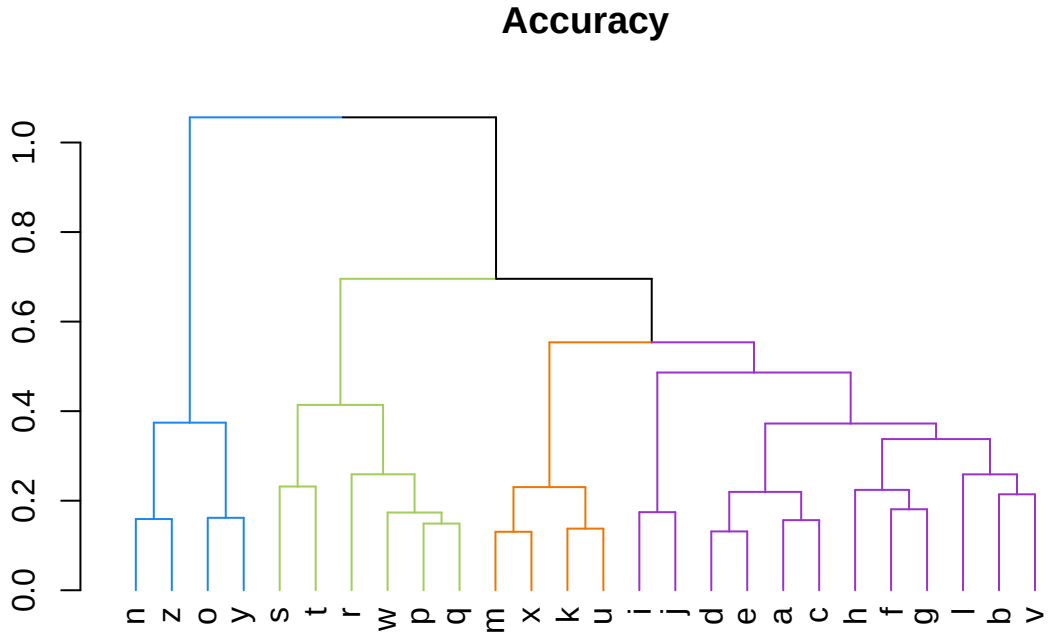



```
diss.rt.matrix1<- diss.rt.matrix%>%
  gdata::lowerTriangle()%>%
  as_data_frame()

fviz_nbclust(diss.rt.matrix1, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method - Rtdiss")
```

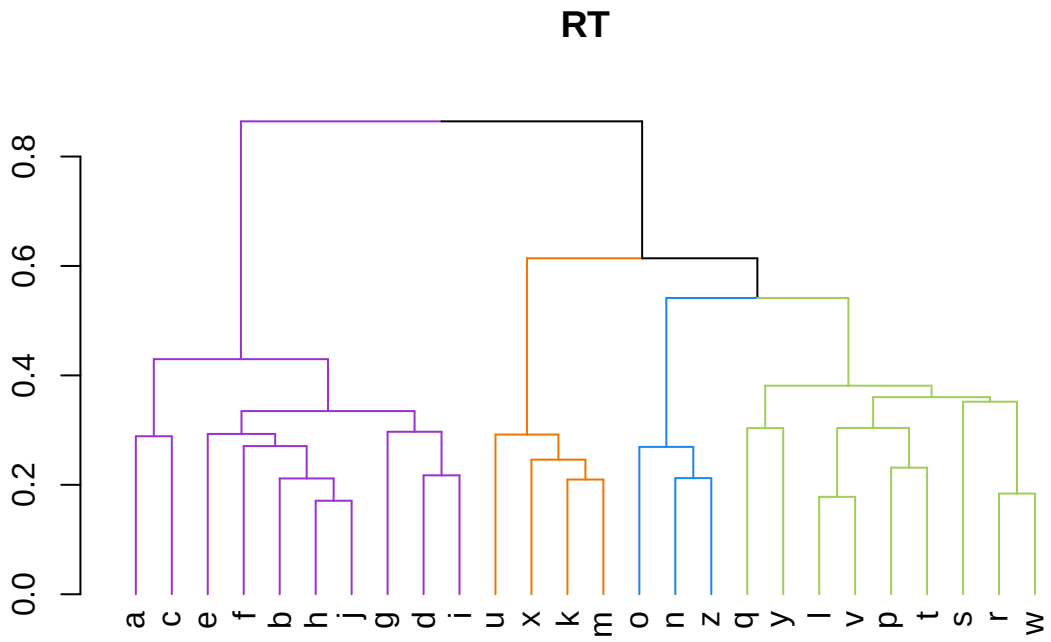



```
## 4 (both accuracy & rt)
```

```
dend.acc4<- dend.acc %>%
  set("branches_k_color", value = c( "dodgerblue2", "darkolivegreen3","darkorange2", "darkorchid3"), k =
  plot(horiz =F, main = "Accuracy")
```




```
dend.rt4<- dend.rt%>%
  set("branches_k_color", value = c("darkorchid3", "darkorange2","dodgerblue2" ,"darkolivegreen3"), k =
  plot(horiz = F, main = "RT")
```




```
cluster_assignments_acc <- cutree(hclust(dist.res.acc, method = "ward.D2"), 4)
cluster_assignments_rt <- cutree(hclust(dist.res.rt, method = "ward.D2"), 4)
```

3. MODELS (for different pairs)

pga *A form to validate the results obtained in the analyses above is to explore if the abstract features interpreted from the hierarchical clustering can accont for the data better than other reasanable alternatives. To this end, we implement three explanations of what constitute a braille feature. These three explnations fullfill the following criteria*

a) No two different letters should have all features in common b) Knowing the features, one should infer the letter. c) No feature should be the result of the combination other features.

Ideas:

a) Each position in the 2x3 matrix is a feature. That is, any two letters that share either a raised do

EXAMPLE:	Pair	Pos.1	Pos.2	Pos.3	Pos.4	Pos.5	Pos.6
	b k	2/12	0	0	2/12	2/12	2/12
	b C	2/12	0	2/12	0	2/12	2/12

b) Only raised dots are features.

EXAMPLE:	Pair	Dot.1	Dot.2	Dot.3	Dot.4	Dot.5	Dot.6
	b k	2/4	0	0	0	0	0

```
            ---------------------------------------------------------
            | b C  |  2/4  |   0   |   0   |   0   |   0   |   0   |
```

c) There are more "abstract" features, related to the shapes that different combinations of dots make. I

- Vertical lines: in column 1 (dots 12, 23, or 123) and in column 2 (dots 45, 56, or 456)
- Horizontal lines: in row 1 (dots 14), in row 2 (dots 25), and in row 3 (dots 36)
- Gap in middle line: dots 25 are not raised
- Gap in the bottom line: dots 36 are not raised (DIVIDES BASE PATTERNS (that only use the upper fou
- DOT 3 ??

```
  EXAMPLE:    | Pair | Verticalc1 | Verticalc2 | Horizontal1 | Horizontal2 | Horizontal3 | GapMiddle | G
              ---------------------------------------------------------------------------------------
              | b k  |     0      |     0      |      0      |      0      |      0      |      0    |
              ---------------------------------------------------------------------------------------
              | b c  |     0      |     0      |      0      |      0      |      0      |      0    |
```

d) There can also be some other variables:

- Mirror letters: a letter can be a mirror (by horizontal or vertical planes) of other letter (e.g.
- Number of dots raised: the more dots a letter has raised, the more complex it becomes.

** Important to note: Dot 3 should be crucial. Check it!

Dataset with all possible pairs and the dots raised in each one of their letters

```
stim.1 <- letters
stim.2 <- letters

pair <- array(dim=26*26)
type <- array(dim = 26*26)

Start<- as_data_frame(c("S"))%>%
  rename(stimuli = value)

for(i in 1:26){
  for(j in 1:26){
    if (stim.1[i] == stim.2[j]) {
      type [(i-1)*26+j] <- 0
    }
    else {
      type[(i-1)*26+j] <- 1
    }
    pair[(i-1)*26+j] <- paste(stim.1[i],stim.2[j])
  }
}

type <- as.character(type)
```

```r
pairs <- as.character(pair)

allpairs<- as.data.frame(cbind(pairs, type))

alldifferentpairs <- allpairs%>%
  filter(type== 1)%>%
  mutate(Pairs = pairs)%>%
  separate(pairs, c("IT1", "IT2"))%>%
  select(Pairs, IT1, IT2)

x<- BrailleDotsTable%>%
  select(-DotPosition)
colnames(x)<- c("IT1", "DotNumber_1",  "Dot1_1", "Dot2_1", "Dot3_1",  "Dot4_1",  "Dot5_1",  "Dot6_1")
y<- BrailleDotsTable%>%
  select(-DotPosition)
colnames(y)<- c("IT2", "DotNumber_2",  "Dot1_2", "Dot2_2", "Dot3_2",  "Dot4_2",  "Dot5_2",  "Dot6_2")

braille.features<- alldifferentpairs%>%
  merge(x, by = "IT1")%>%
  merge(y, by = "IT2")%>%
  select(Pairs, IT1, DotNumber_1, Dot1_1, Dot2_1, Dot3_1, Dot4_1, Dot5_1, Dot6_1, IT2, DotNumber_2, Dot
```

```r
# Option 1
## Each raised dot & gap is a feature: P1, P2, P3, P4, P5, P6


braille.features.1<- braille.features%>%
  group_by(Pairs)%>%
  mutate(DotNumber_1 = as.numeric(DotNumber_1),
         DotNumber_2 = as.numeric(DotNumber_2),
         TotalDots_Pair = sum(DotNumber_1, DotNumber_2),
         P1 = if_else(Dot1_1 == Dot1_2 , 2/12,0/12),
         P2 = if_else(Dot2_1 == Dot2_2 , 2/12,0/12),
         P3 = if_else(Dot3_1 == Dot3_2 , 2/12,0/12),
         P4 = if_else(Dot4_1 == Dot4_2 , 2/12,0/12),
         P5 = if_else(Dot5_1 == Dot5_2 , 2/12,0/12),
         P6 = if_else(Dot6_1 == Dot6_2 , 2/12,0/12))%>%
  ungroup()%>%
  select(Pairs, P1, P2, P3, P4, P5, P6)



# Option 2
## Each raised dot is a feature: Dot1, Dot2, Dot3, Dot4, Dot5, Dot6


braille.features.2<- braille.features%>%
  group_by(Pairs)%>%
  mutate(DotNumber_1 = as.numeric(DotNumber_1),
         DotNumber_2 = as.numeric(DotNumber_2),
         TotalDots_Pair = sum(DotNumber_1, DotNumber_2),
         Dot1 = (if_else(Dot1_1 == 1 & Dot1_2 == 1,2,0))/TotalDots_Pair,
         Dot2 = (if_else(Dot2_1 == 1 & Dot2_2 == 1,2,0))/TotalDots_Pair,
         Dot3 = (if_else(Dot3_1 == 1 & Dot3_2 == 1,2,0))/TotalDots_Pair,
```

```
        Dot4 = (if_else(Dot4_1 == 1 & Dot4_2 == 1,2,0))/TotalDots_Pair,
        Dot5 = (if_else(Dot5_1 == 1 & Dot5_2 == 1,2,0))/TotalDots_Pair,
        Dot6 = (if_else(Dot6_1 == 1 & Dot6_2 == 1,2,0))/TotalDots_Pair)%>%
  select(Pairs, Dot1, Dot2, Dot3, Dot4, Dot5, Dot6)



# Option 3
## Abstract features (x8): Verticals (V_c1, V_c2), Horizontals (H_1, H_2, H_3), Gaps (MiddleGap, Bottom


braille.features.3<- braille.features%>%
  group_by(Pairs)%>%
  mutate(Vertical1_1 = if_else(Dot1_1 == 1 & Dot2_1 == 1,1,0),
         Vertical1_2 = if_else(Dot3_1 == 1 & Dot2_1 == 1,1,0),
         Vertical1_3 = if_else(Dot1_1 == 1 & Dot2_1 == 1 & Dot3_1 == 1,1,0),
         Vertical2_1 = if_else(Dot4_1 == 1 & Dot5_1 == 1,1,0),
         Vertical2_2 = if_else(Dot5_1 == 1 & Dot6_1 == 1,1,0),
         Vertical2_3 = if_else(Dot4_1 == 1 & Dot5_1 == 1 & Dot6_1 == 1,1,0),
         VerticalsIT1_c1 = if_else(Vertical1_1 == 1 | Vertical1_2 == 1 | Vertical1_3 == 1,1,0),
         VerticalsIT1_c2 = if_else(Vertical2_1 == 1 | Vertical2_2 == 1 | Vertical2_3 == 1,1,0))%>%
  select(-Vertical1_1, -Vertical1_2, -Vertical1_3, -Vertical2_1, -Vertical2_2, -Vertical2_3)%>%
  mutate(Vertical1_1 = if_else(Dot1_2 == 1 & Dot2_2 == 1,1,0),
         Vertical1_2 = if_else(Dot3_2 == 1 & Dot2_2 == 1,1,0),
         Vertical1_3 = if_else(Dot1_2 == 1 & Dot2_2 == 1 & Dot3_2 == 1,1,0),
         Vertical2_1 = if_else(Dot4_2 == 1 & Dot5_2 == 1,1,0),
         Vertical2_2 = if_else(Dot5_2 == 1 & Dot6_2 == 1,1,0),
         Vertical2_3 = if_else(Dot4_2 == 1 & Dot5_2 == 1 & Dot6_2 == 1,1,0),
         VerticalsIT2_c1 = if_else(Vertical1_1 == 1 | Vertical1_2 == 1 | Vertical1_3 == 1,1,0),
         VerticalsIT2_c2 = if_else(Vertical2_1 == 1 | Vertical2_2 == 1 | Vertical2_3 == 1,1,0),
         TotalVerticals_IT1 = sum(VerticalsIT1_c1, VerticalsIT1_c2),
         TotalVerticals_IT2 = sum(VerticalsIT2_c1, VerticalsIT2_c2),
         SharedVertical_c1 = if_else(VerticalsIT1_c1 == 1 & VerticalsIT2_c1 == 1, 2, 0),
         SharedVertical_c2 = if_else(VerticalsIT1_c2 == 1 & VerticalsIT2_c2 == 1, 2, 0),)%>%
  select(-Vertical1_1, -Vertical1_2, -Vertical1_3, -Vertical2_1, -Vertical2_2, -Vertical2_3)%>%
  mutate(HorizontalIT1_1 = if_else(Dot1_1 == 1 & Dot4_1 == 1,1,0),
         HorizontalIT1_2 = if_else(Dot2_1 == 1 & Dot5_1 == 1,1,0),
         HorizontalIT1_3 = if_else(Dot3_1 == 1 & Dot6_1 == 1,1,0),
         HorizontalIT2_1 = if_else(Dot1_2 == 1 & Dot4_2 == 1,1,0),
         HorizontalIT2_2 = if_else(Dot2_2 == 1 & Dot5_2 == 1,1,0),
         HorizontalIT2_3 = if_else(Dot3_2 == 1 & Dot6_2 == 1,1,0),
         TotalHorizontals_IT1 = sum(HorizontalIT1_1, HorizontalIT1_2, HorizontalIT1_3),
         TotalHorizontals_IT2 = sum(HorizontalIT2_1, HorizontalIT2_2, HorizontalIT2_3),
         SharedHorizontal_1 = if_else(HorizontalIT1_1 == 1 & HorizontalIT2_1 == 1, 2, 0),
         SharedHorizontal_2 = if_else(HorizontalIT1_2 == 1 & HorizontalIT2_2 == 1, 2, 0),
         SharedHorizontal_3 = if_else(HorizontalIT1_3 == 1 & HorizontalIT2_3 == 1, 2, 0))%>%
  select(-HorizontalIT1_1, -HorizontalIT1_2, -HorizontalIT1_3, -HorizontalIT2_1, -HorizontalIT2_2, -Hori
  mutate(GapMiddle_IT1 = if_else(Dot2_1 == 0 & Dot5_1 == 0,1,0),
         GapMiddle_IT2 = if_else(Dot2_2 == 0 & Dot5_2 == 0,1,0),
         GapBottom_IT1 = if_else(Dot3_1 == 0 & Dot6_1 == 0,1,0),
         GapBottom_IT2 = if_else(Dot3_2 == 0 & Dot6_2 == 0,1,0),
         TotalGaps_IT1 = sum(GapBottom_IT1,GapMiddle_IT1),
         TotalGaps_IT2 = sum(GapBottom_IT2,GapMiddle_IT2),
```

```
        SharedGapMiddle = if_else(GapMiddle_IT1 ==1 & GapMiddle_IT2 == 1, 2, 0),
        SharedGapBottom = if_else(GapBottom_IT1 ==1 & GapBottom_IT2 == 1, 2, 0),
        SharedDot3 = if_else(Dot3_1 == 1 & Dot3_2 == 1,2,0),
        TotalFeatures_pair = sum(TotalGaps_IT1, TotalGaps_IT2, TotalHorizontals_IT1, TotalHorizontals_
        V_c1 = SharedVertical_c1/TotalFeatures_pair,
        V_c2 = SharedVertical_c2/TotalFeatures_pair,
        H_1 = SharedHorizontal_1/TotalFeatures_pair,
        H_2 = SharedHorizontal_2/TotalFeatures_pair,
        H_3 = SharedHorizontal_3/TotalFeatures_pair,
        MiddleGap = SharedGapMiddle/TotalFeatures_pair,
        BottomGap = SharedGapBottom/TotalFeatures_pair,
        Dot3 = SharedDot3/TotalFeatures_pair)%>%
  select(Pairs, V_c1, V_c2,H_1,H_2,H_3,MiddleGap,BottomGap,Dot3)


#MirrorH = if_else(Pairs == "d f" | Pairs == "f d" | Pairs == "e i" | Pairs == "i e" | Pairs == "h j" |
#MirrorV = if_else(Pairs == "m u" | Pairs == "u m" | Pairs == "n z" | Pairs == "z n" | Pairs == "p v" |
```

```
#braille.features.1 --> P1, P2, P3, P4, P5, P6
braille.features.2
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 650 x 7
## # Groups:   Pairs [650]
##    Pairs  Dot1  Dot2  Dot3  Dot4  Dot5  Dot6
##    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 k a   0.667     0     0     0     0     0
##  2 p a   0.4       0     0     0     0     0
##  3 u a   0.5       0     0     0     0     0
##  4 r a   0.4       0     0     0     0     0
##  5 s a   0         0     0     0     0     0
##  6 z a   0.4       0     0     0     0     0
##  7 b a   0.667     0     0     0     0     0
##  8 x a   0.4       0     0     0     0     0
##  9 w a   0         0     0     0     0     0
## 10 g a   0.4       0     0     0     0     0
## # ... with 640 more rows
```

```
braille.features.3
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 650 x 9
```

```
## # Groups:   Pairs [650]
##    Pairs  V_c1  V_c2   H_1   H_2   H_3 MiddleGap BottomGap  Dot3
##    <chr> <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>
##  1 k a       0     0     0     0     0     0.5       0         0
##  2 p a       0     0     0     0     0     0         0         0
##  3 u a       0     0     0     0     0     0.4       0         0
##  4 r a       0     0     0     0     0     0         0         0
##  5 s a       0     0     0     0     0     0         0         0
##  6 z a       0     0     0     0     0     0         0         0
##  7 b a       0     0     0     0     0     0         0.5       0
##  8 x a       0     0     0     0     0     0.333     0         0
##  9 w a       0     0     0     0     0     0         0         0
## 10 g a       0     0     0     0     0     0         0.286     0
## # ... with 640 more rows
```

```
library(lme4)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
############
# POSITION
############

Acc.tomodel.1 <- Different%>%
  select(-Subject, -KEY, -Rubric,-RT, -Order)%>%
  merge(braille.features.1, by = "Pairs")


acc_position_mo <- glmer(Accuracy ~ scale(P1) + scale(P2) + scale(P3) + scale(P4) + scale(P5) + scale(P
summary(acc_position_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## Accuracy ~ scale(P1) + scale(P2) + scale(P3) + scale(P4) + scale(P5) +
##     scale(P6) + (1 | SubjectID) + (1 | ItNumber)
##    Data: Acc.tomodel.1
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
##   4138.0   4207.4  -2060.0   4120.0    16460
##
```

```
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -25.5386   0.0712   0.1179   0.1858   1.2165
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.6727   0.8202
##  SubjectID (Intercept) 1.0879   1.0430
## Number of obs: 16469, groups:  ItNumber, 650; SubjectID, 24
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.372534   0.230730  18.951  < 2e-16 ***
## scale(P1)   -0.109423   0.060223  -1.817   0.0692 .
## scale(P2)   -0.619656   0.060601 -10.225  < 2e-16 ***
## scale(P3)   -0.447771   0.059401  -7.538 4.77e-14 ***
## scale(P4)   -0.245564   0.058626  -4.189 2.81e-05 ***
## scale(P5)   -0.345276   0.058835  -5.869 4.40e-09 ***
## scale(P6)    0.005531   0.058065   0.095   0.9241
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr) sc(P1) sc(P2) sc(P3) sc(P4) sc(P5)
## scale(P1) -0.008
## scale(P2) -0.096 -0.068
## scale(P3) -0.069 -0.018  0.057
## scale(P4) -0.045 -0.010  0.049  0.088
## scale(P5) -0.058  0.029  0.081  0.078  0.095
## scale(P6)  0.004  0.051 -0.027  0.009  0.039  0.060
```

```
#save(acc_position_mo, file = "acc_position_mo.RData")


############
# DOTS
############

Acc.tomodel.2 <- Different%>%
  select(-Subject, -KEY, -Rubric,-RT, -Order)%>%
  merge(braille.features.2, by = "Pairs")


acc_dots_mo <- glmer(Accuracy ~ scale(Dot1) + scale(Dot2) + scale(Dot3) + scale(Dot4) + scale(Dot5) + s
summary(acc_dots_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Accuracy ~ scale(Dot1) + scale(Dot2) + scale(Dot3) + scale(Dot4) +
##     scale(Dot5) + scale(Dot6) + (1 | SubjectID) + (1 | ItNumber)
##    Data: Acc.tomodel.2
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
```

```
##   4163.2   4232.6  -2072.6   4145.2    16460
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -26.7049   0.0719   0.1187   0.1837   1.3001
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.7616   0.8727
##  SubjectID (Intercept) 1.0801   1.0393
## Number of obs: 16469, groups:  ItNumber, 650; SubjectID, 24
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.38316    0.23435  18.703  < 2e-16 ***
## scale(Dot1) -0.49729    0.08464  -5.875 4.22e-09 ***
## scale(Dot2) -0.70886    0.07078 -10.015  < 2e-16 ***
## scale(Dot3) -0.54688    0.06667  -8.203 2.34e-16 ***
## scale(Dot4) -0.36765    0.07387  -4.977 6.46e-07 ***
## scale(Dot5) -0.47521    0.06479  -7.335 2.22e-13 ***
## scale(Dot6) -0.14307    0.06077  -2.354   0.0186 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) sc(D1) sc(D2) sc(D3) sc(D4) sc(D5)
## scale(Dot1) -0.069
## scale(Dot2) -0.118  0.493
## scale(Dot3) -0.085  0.248  0.334
## scale(Dot4) -0.055  0.500  0.215  0.276
## scale(Dot5) -0.075  0.339  0.310  0.282  0.253
## scale(Dot6) -0.029  0.167  0.199 -0.033  0.156  0.116
```

```
#save(acc_dots_mo, file = "acc_dots_mo.RData")



###########
# ABSTRACT
###########

Acc.tomodel.3 <- Different%>%
  select(-Subject, -KEY, -Rubric,-RT, -Order)%>%
  merge(braille.features.3, by = "Pairs")


acc_abstract_mo <- glmer(Accuracy ~ scale(V_c1) + scale(V_c2) + scale(H_1) + scale(H_2) + scale(H_3) + s
summary(acc_abstract_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Accuracy ~ scale(V_c1) + scale(V_c2) + scale(H_1) + scale(H_2) +
##     scale(H_3) + scale(MiddleGap) + scale(BottomGap) + scale(Dot3) +
##     (1 | SubjectID) + (1 | ItNumber)
```

```
##    Data: Acc.tomodel.3
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
##   4180.3   4265.1  -2079.1   4158.3    16458
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -18.7819   0.0723   0.1166   0.1813   1.2894
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.8834   0.9399
##  SubjectID (Intercept) 1.0692   1.0340
## Number of obs: 16469, groups:  ItNumber, 650; SubjectID, 24
##
## Fixed effects:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.39014    0.23402  18.760  < 2e-16 ***
## scale(V_c1)      -0.43499    0.05939  -7.324 2.41e-13 ***
## scale(V_c2)      -0.26915    0.06292  -4.278 1.89e-05 ***
## scale(H_1)       -0.09462    0.06543  -1.446    0.148
## scale(H_2)       -0.24780    0.05678  -4.364 1.27e-05 ***
## scale(H_3)       -0.08643    0.05889  -1.468    0.142
## scale(MiddleGap) -0.28741    0.05448  -5.276 1.32e-07 ***
## scale(BottomGap) -0.30974    0.06549  -4.729 2.25e-06 ***
## scale(Dot3)      -0.39043    0.06766  -5.771 7.89e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) s(V_1) s(V_2) s(H_1) s(H_2) s(H_3) sc(MG) sc(BG)
## scale(V_c1) -0.090
## scale(V_c2) -0.038  0.217
## scale(H_1)  -0.013  0.076 -0.090
## scale(H_2)  -0.056 -0.091 -0.280  0.146
## scale(H_3)  -0.019  0.119 -0.019  0.053  0.042
## scl(MddlGp) -0.067  0.182  0.128 -0.004  0.049 -0.006
## scl(BttmGp) -0.056  0.125  0.147  0.092  0.086  0.044  0.086
## scale(Dot3) -0.057 -0.019  0.096  0.082  0.174 -0.148  0.014  0.334
```

```
#save(acc_abstract_mo, file = "acc_abstract_mo.RData")
```

```
# Comparing Non-Nested Models:  CAREFUL! wHAT IF ONE MODEL (I.E.,MODEL3) DOESN'T HAVE THE SAME DF??

Acc.model.comparison <- data.frame(Model = c("position", "dots", "abstract"),
AIC = c(AIC(acc_position_mo), AIC(acc_dots_mo), AIC(acc_abstract_mo)),
BIC = c(BIC(acc_position_mo), BIC(acc_dots_mo), BIC(acc_abstract_mo)),
stringsAsFactors = FALSE)
kable(Acc.model.comparison)
```

Model	AIC	BIC
position	4138.009	4207.393
dots	4163.175	4232.558
abstract	4180.277	4265.079

```
#CHOOSE MODEL 1 (lower AIc & BIC)



###########

performance::r2_nakagawa(acc_position_mo)
```

```
## # R2 for Mixed Models
##
##   Conditional R2: 0.431
##      Marginal R2: 0.127
```

```
performance::r2_nakagawa(acc_position_mo, by_group = T)
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
## Level 1   |  0.000
## SubjectID |  0.525
## ItNumber  | -0.076
```

```
performance::r2_nakagawa(acc_dots_mo)
```

```
## # R2 for Mixed Models
##
##   Conditional R2: 0.433
##      Marginal R2: 0.116
```

```
performance::r2_nakagawa(acc_dots_mo, by_group = T)
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
## Level 1   |  0.000
## SubjectID |  0.463
## ItNumber  | -0.069
```

```
performance::r2_nakagawa(acc_abstract_mo)
```

```
## # R2 for Mixed Models
##
##   Conditional R2: 0.430
##      Marginal R2: 0.091
```

```
performance::r2_nakagawa(acc_abstract_mo, by_group = T)
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
```

```
## Level 1   |  0.000
## SubjectID |  0.377
## ItNumber  | -0.058
```

```
### marginal R2 describes the proportion of variance explained by the fixed factor(s) alone & condition
```

```
##QUESTION:HOW DOES ONE INTERPRET THIS??
```

```
############
# POSITION
############

RT.tomodel.1 <- CorrectDifferent.norm%>%
  select(-Subject, -KEY, -Rubric,-Accuracy, -Order)%>%
  merge(braille.features.1, by = "Pairs")


RT_position_mo <- glmer(normTime ~ scale(P1) + scale(P2) + scale(P3) + scale(P4) + scale(P5) + scale(P6
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(RT_position_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( identity )
## Formula:
## normTime ~ scale(P1) + scale(P2) + scale(P3) + scale(P4) + scale(P5) +
##     scale(P6) + (1 | SubjectID) + (1 | ItNumber)
##    Data: RT.tomodel.1
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
## -10183.1 -10106.4   5101.6 -10203.1    15905
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.899 -0.485 -0.153  0.254 36.852
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.00252  0.0502
##  SubjectID (Intercept) 0.00000  0.0000
##  Residual              0.04997  0.2235
## Number of obs: 15915, groups:  ItNumber, 650; SubjectID, 24
##
## Fixed effects:
##              Estimate Std. Error t value Pr(>|z|)
## (Intercept) 1.0013180  0.0030557 327.689  < 2e-16 ***
## scale(P1)   0.0009118  0.0030636   0.298  0.76600
## scale(P2)   0.0127008  0.0030533   4.160 3.19e-05 ***
## scale(P3)   0.0191905  0.0030470   6.298 3.01e-10 ***
## scale(P4)   0.0003963  0.0030464   0.130  0.89649
## scale(P5)   0.0094074  0.0030498   3.085  0.00204 **
## scale(P6)   0.0029378  0.0030474   0.964  0.33503
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr) sc(P1) sc(P2) sc(P3) sc(P4) sc(P5)
## scale(P1) -0.007
## scale(P2) -0.018 -0.072
## scale(P3) -0.016 -0.016  0.028
## scale(P4) -0.004 -0.020  0.024  0.038
## scale(P5) -0.010  0.019  0.045  0.043  0.036
## scale(P6) -0.002  0.040 -0.026  0.014  0.009  0.029
## optimizer (bobyqa) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

```
############
# DOTS
############

RT.tomodel.2 <- CorrectDifferent.norm%>%
  select(-Subject, -KEY, -Rubric,-Accuracy, -Order)%>%
  merge(braille.features.2, by = "Pairs")


RT_dots_mo <- glmer(normTime ~ scale(Dot1) + scale(Dot2) + scale(Dot3) + scale(Dot4) + scale(Dot5) + sca
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(RT_dots_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( identity )
## Formula: normTime ~ scale(Dot1) + scale(Dot2) + scale(Dot3) + scale(Dot4) +
##     scale(Dot5) + scale(Dot6) + (1 | SubjectID) + (1 | ItNumber)
##    Data: RT.tomodel.2
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
## -10192.9 -10116.1   5106.4 -10212.9    15905
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.889 -0.485 -0.153  0.254 36.838
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.002451 0.04951
##  SubjectID (Intercept) 0.000000 0.00000
##  Residual              0.050029 0.22367
## Number of obs: 15915, groups:  ItNumber, 650; SubjectID, 24
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept) 1.001302   0.003009 332.799  < 2e-16 ***
## scale(Dot1) 0.006687   0.003625   1.845  0.06509 .
## scale(Dot2) 0.015083   0.003334   4.524 6.07e-06 ***
```

```
## scale(Dot3) 0.019437   0.003160   6.150 7.74e-10 ***
## scale(Dot4) 0.004457   0.003347   1.332  0.18298
## scale(Dot5) 0.017076   0.003123   5.467 4.57e-08 ***
## scale(Dot6) 0.009917   0.003052   3.250  0.00116 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) sc(D1) sc(D2) sc(D3) sc(D4) sc(D5)
## scale(Dot1) -0.013
## scale(Dot2) -0.019  0.404
## scale(Dot3) -0.014  0.162  0.211
## scale(Dot4) -0.008  0.414  0.134  0.199
## scale(Dot5) -0.014  0.244  0.181  0.188  0.152
## scale(Dot6) -0.005  0.135  0.134 -0.076  0.110  0.072
## optimizer (bobyqa) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

```
############
# ABSTRACT
############

RT.tomodel.3 <- CorrectDifferent.norm%>%
  select(-Subject, -KEY, -Rubric,-Accuracy, -Order)%>%
  merge(braille.features.3, by = "Pairs")


RT_abstract_mo <- glmer(normTime ~ scale(V_c1) + scale(V_c2) + scale(H_1) + scale(H_2) + scale(H_3) + s
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(RT_abstract_mo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( identity )
## Formula: normTime ~ scale(V_c1) + scale(V_c2) + scale(H_1) + scale(H_2) +
##     scale(H_3) + scale(MiddleGap) + scale(BottomGap) + scale(Dot3) +
##     (1 | SubjectID) + (1 | ItNumber)
##    Data: RT.tomodel.3
## Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05))
##
##      AIC      BIC   logLik deviance df.resid
## -10187.0 -10094.9   5105.5 -10211.0    15903
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.897 -0.485 -0.155  0.252 36.235
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  ItNumber  (Intercept) 0.00246  0.0496
##  SubjectID (Intercept) 0.00000  0.0000
##  Residual              0.04986  0.2233
## Number of obs: 15915, groups:  ItNumber, 650; SubjectID, 24
```

```
##
## Fixed effects:
##                  Estimate Std. Error t value Pr(>|z|)
## (Intercept)      1.001251   0.003020 331.506  < 2e-16 ***
## scale(V_c1)      0.008662   0.003067   2.824 0.004743 **
## scale(V_c2)      0.007442   0.003178   2.341 0.019207 *
## scale(H_1)       0.002969   0.003044   0.975 0.329385
## scale(H_2)       0.010055   0.003130   3.212 0.001316 **
## scale(H_3)       0.006579   0.003037   2.167 0.030259 *
## scale(MiddleGap) 0.007492   0.002986   2.509 0.012102 *
## scale(BottomGap) 0.012101   0.003121   3.877 0.000106 ***
## scale(Dot3)      0.018704   0.003196   5.852 4.85e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) s(V_1) s(V_2) s(H_1) s(H_2) s(H_3) sc(MG) sc(BG)
## scale(V_c1) -0.014
## scale(V_c2) -0.008  0.155
## scale(H_1)  -0.005  0.055 -0.097
## scale(H_2)  -0.010 -0.137 -0.283  0.113
## scale(H_3)  -0.005  0.088 -0.001  0.036  0.016
## scl(MddlGp) -0.009  0.102  0.084 -0.023  0.020 -0.009
## scl(BttmGp) -0.012  0.074  0.098  0.062  0.040  0.028  0.056
## scale(Dot3) -0.011 -0.048  0.069  0.062  0.117 -0.177  0.008  0.252
## optimizer (bobyqa) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

```
# Comparing Non-Nested Models:  CAREFUL! wHAT IF ONE MODEL (I.E.,MODEL3) DOESN'T HAVE THE SAME DF??

RT.model.comparison <- data.frame(Model = c("position", "dots", "abstract"),
AIC = c(AIC(RT_position_mo), AIC(RT_dots_mo), AIC(RT_abstract_mo)),
BIC = c(BIC(RT_position_mo), BIC(RT_dots_mo), BIC(RT_abstract_mo)),
stringsAsFactors = FALSE)
kable(RT.model.comparison)
```

Model	AIC	BIC
position	-10183.15	-10106.40
dots	-10192.90	-10116.15
abstract	-10187.03	-10094.93

```
#CHOOSE MODEL 2 (lower AIc & BIC)



############

performance::r2_nakagawa(RT_position_mo)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## Random effect variances not available. Returned R2 does not account for random effects.
```

```
## # R2 for Mixed Models
```

```
##
##   Conditional R2: NA
##      Marginal R2: 0.012
```

```
performance::r2_nakagawa(RT_position_mo, by_group = T)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
## Level 1   | -0.004
## SubjectID |  0.138
## ItNumber  |
```

```
performance::r2_nakagawa(RT_dots_mo)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## Random effect variances not available. Returned R2 does not account for random effects.
```

```
## # R2 for Mixed Models
##
##   Conditional R2: NA
##      Marginal R2: 0.013
```

```
performance::r2_nakagawa(RT_dots_mo, by_group = T)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
## Level 1   | -0.006
## SubjectID |  0.161
```

```
## ItNumber  |
```

```
performance::r2_nakagawa(RT_abstract_mo)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## Random effect variances not available. Returned R2 does not account for random effects.
```

```
## # R2 for Mixed Models
##
##   Conditional R2: NA
##      Marginal R2: 0.013
```

```
performance::r2_nakagawa(RT_abstract_mo, by_group = T)
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## boundary (singular) fit: see ?isSingular
```

```
## Warning: Can't compute random effect variances. Some variance components equal zero. Your
##   model may suffer from singulariy.
##   Solution: Respecify random structure! You may also decrease the 'tolerance'
##   level to enforce the calculation of random effect variances.
```

```
## # Explained Variance by Level
##
## Level     |     R2
## ------------------
## Level 1   | -0.002
## SubjectID |  0.158
## ItNumber  |
```

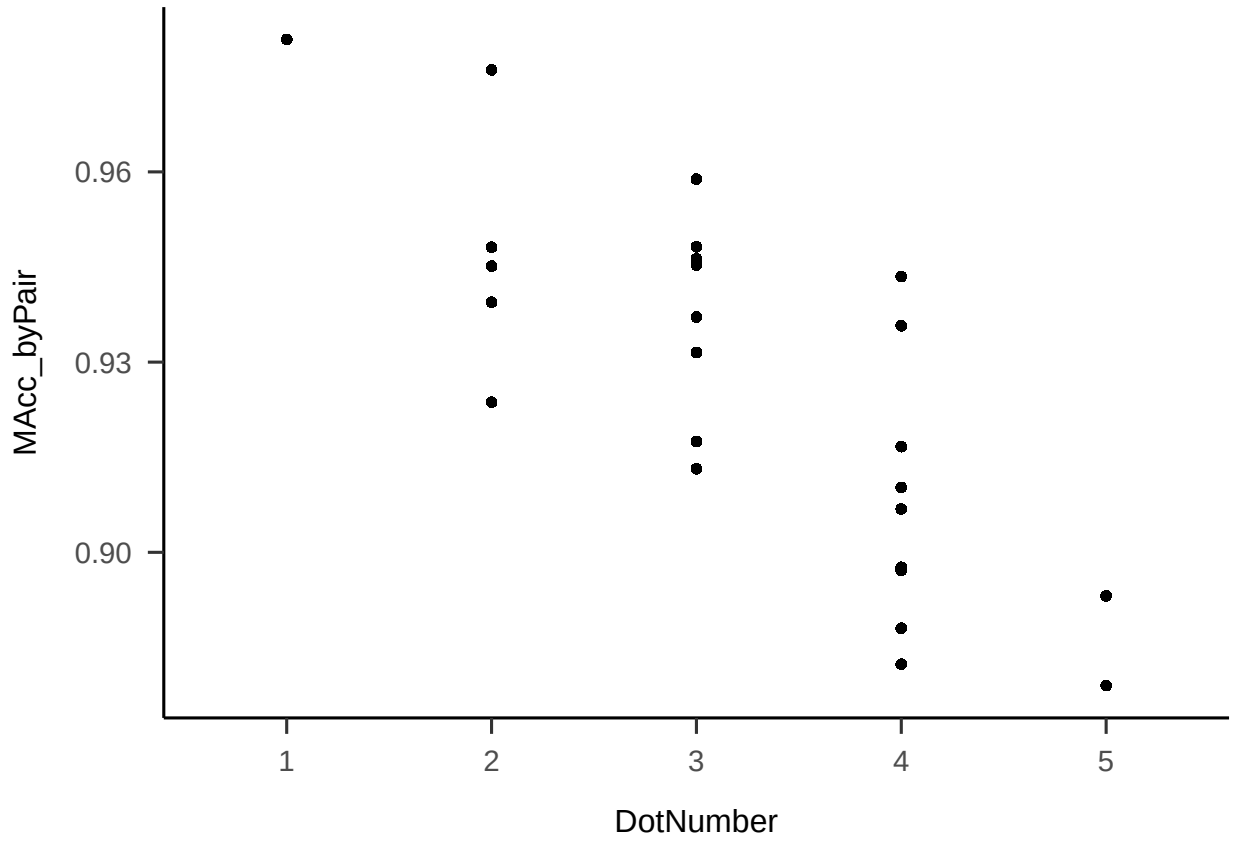
3. SAME PAIRS

FOLLOWING WILEY... "to determine whether the visual complexity of letters affects perception, [...] the visual complexity of each letter was represented by its total number of visual features [...]. Over the set of 45 letters, complexity values ranged from 4 to 15 features (mean = 7.33). The correlation between visual complexity and both average discrimination time and accuracy was calculated for each participant. The Fisher z-transformation was applied to each correlation before averaging, and then back-transformed to the Pearson's r. 95% confidence intervals are provided for each average correlation and for the difference between the groups.

```
# When "Total number of features" = DOTS

BrailleDotsTable%>%
  rename(IT1 = "Character") -> BrailleDotsTable_2

CleanBF%>%
  filter(Rubric=="m")%>%
  merge(BrailleDotsTable_2, by="IT1")%>%
  select(SubjectID, ItNumber, Pairs, IT1, IT2, DotNumber, Accuracy, RT)%>%
  mutate(RTms = RT*1000)%>%
  group_by(SubjectID)%>%
```

```
  mutate(MRT_subj = mean(RTms))%>%
  ungroup()%>%
  mutate(normTime = RTms/MRT_subj) -> same_data



#correlation between Number of Dots & RT & acc

same_data%>%
  group_by(Pairs)%>%
  mutate(MAcc_byPair = mean(Accuracy))%>%
  ungroup()-> toplot


ggplot(toplot, mapping = aes(x=DotNumber, y=MAcc_byPair)) +
  geom_point(shape = 20, size = 2) +
  theme_apa()+
  stat_smooth(method = "lm",
        col = "forestgreen",
        se = T,
        size = 1)
```

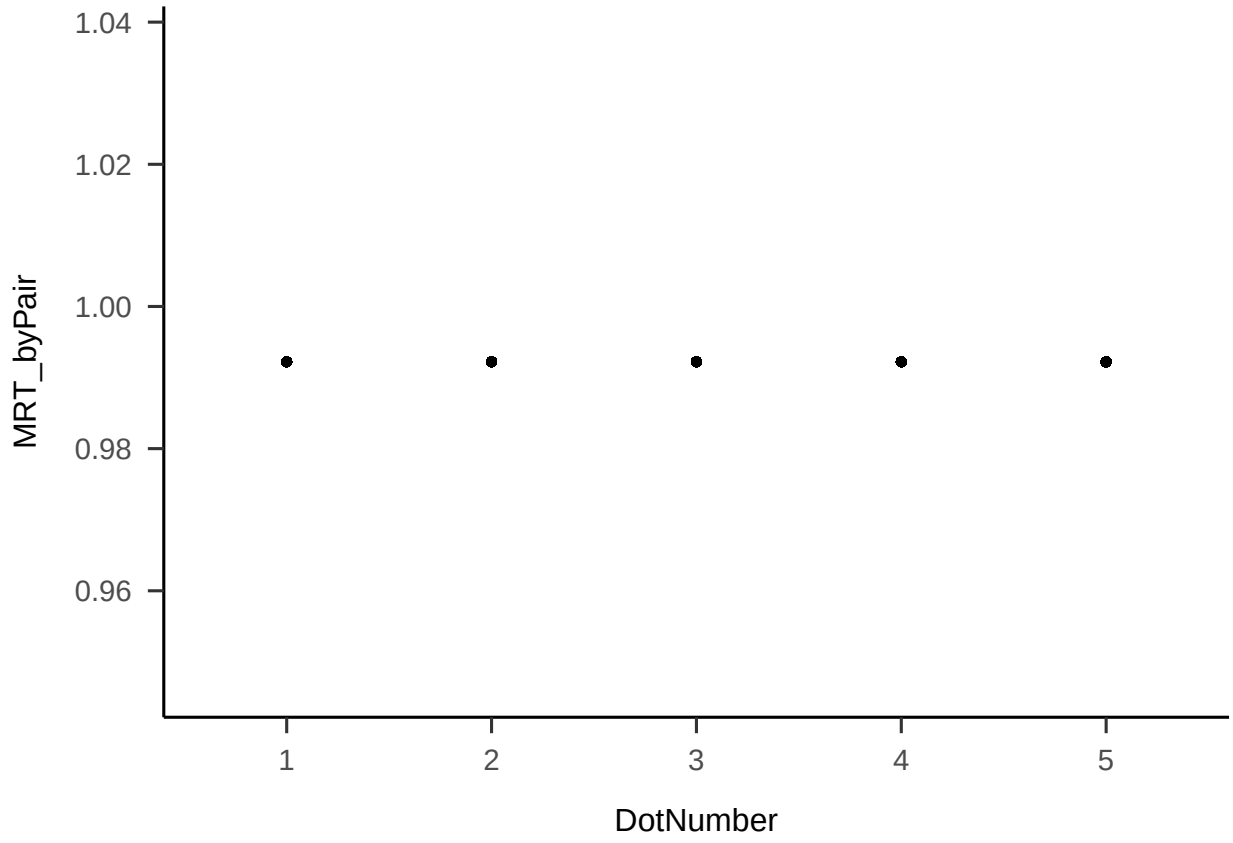
```
## `geom_smooth()` using formula 'y ~ x'
```




```
toplot%>%
  filter(Accuracy==1)%>%
```

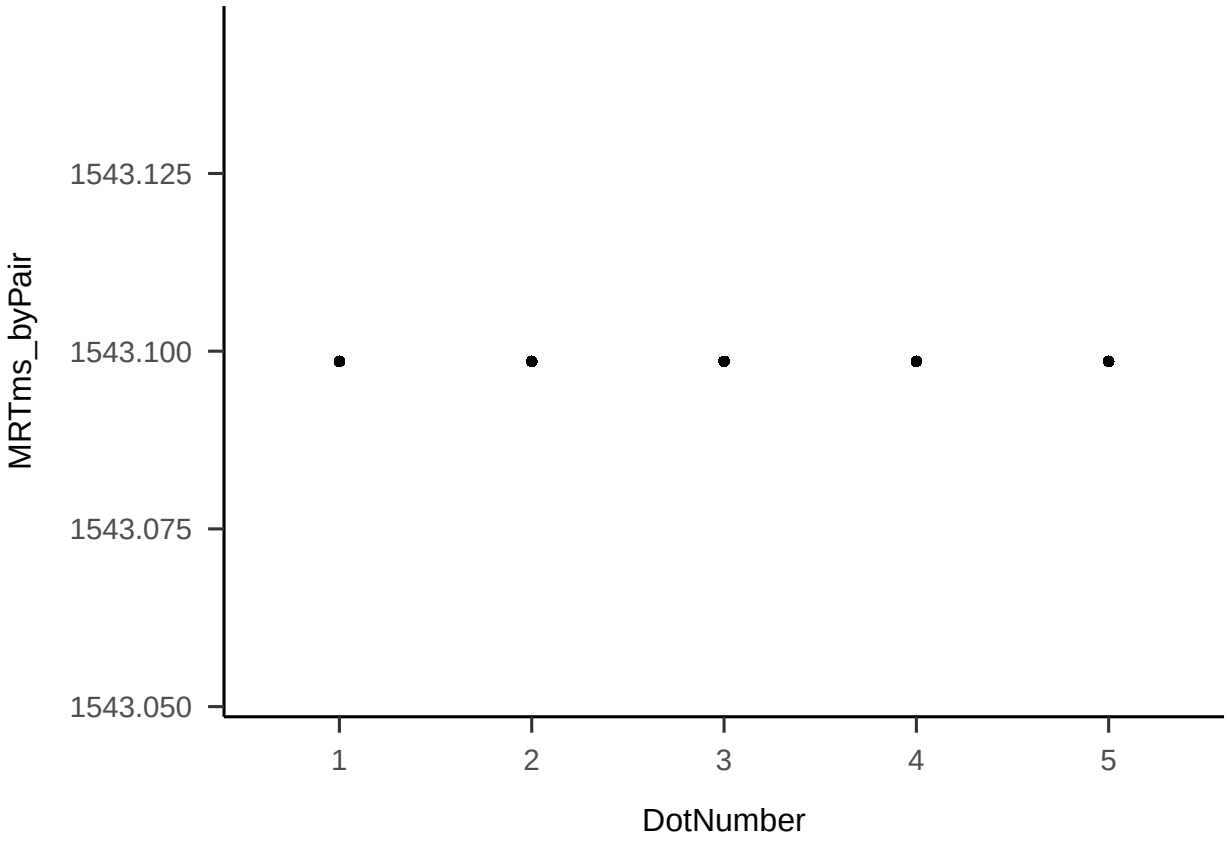
```
  mutate(MRT_byPair = mean(normTime),
         MRTms_byPair = mean(RTms)) ->toplot2

ggplot(toplot2, mapping = aes(x=DotNumber, y=MRT_byPair)) +
  geom_point(shape = 20, size = 2) +
  theme_apa()+
  stat_smooth(method = "lm",
        col = "forestgreen",
        se = T,
        size = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(toplot2, mapping = aes(x=DotNumber, y=MRTms_byPair)) +
  geom_point(shape = 20, size = 2) +
  theme_apa()+
  stat_smooth(method = "lm",
        col = "forestgreen",
        se = T,
        size = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# "The average correlation between discrimination time on correct Same Pair trials and the total number
```