OrdinalAnalysis.R

hamze

2020-01-21

```
############################################################
############################################################
######################ORDINAL ANLAYSIS######################
############################################################
############################################################

#####################################################
#Installing the packages
#install.packages("plyr")
#####################################################
#set workspace to this folder
setwd("D:/Work/IJGIS/R-scripts")
#####################################################
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.5.3
```

```
############################PREPROCESSING#########################
fun.to.int <- function(file_address, result_address) {
  raw <- read.table(file = file_address, sep = ",")
  processed = data.frame(V1 = c(raw))
  processed$woQ <- gsub("Q-", "", raw$V1)
  processed$woQA <- gsub("A-", "", processed$woQ)
  write.table(processed$woQA, file=result_address,
              quote = F, sep = " ", row.names = F, col.names = F)
}
fun.to.int("../sequences/prominence-nf-Q.txt",
           "../sequences/prominence-nf-Q-int.txt")
fun.to.int("../sequences/prominence-nf-A.txt",
           "../sequences/prominence-nf-A-int.txt")
fun.to.int("../sequences/scale-nf-Q.txt",
           "../sequences/scale-nf-Q-int.txt")
fun.to.int("../sequences/scale-nf-A.txt",
           "../sequences/scale-nf-A-int.txt")
############################PROMINENNCE#########################

all_questions <- read.table("../sequences/prominence-nf-Q-int.txt",
                            header = FALSE, sep = " ",
                            col.names = paste0("V",seq_len(4)), fill = TRUE)
head(all_questions, 5)
```

```
##      V1 V2 V3 V4
## 1 32774  4 NA NA
## 2 49158  3  6 NA
## 3 49160  2 NA NA
## 4 32777  1  6 NA
## 5 49166  3 NA NA
```

```
all_answers <- read.table("../sequences/prominence-nf-A-int.txt",
                          header = FALSE, sep = " ",
                          col.names = paste0("V",seq_len(13)), fill = TRUE)
head(all_answers, 5)
```

```
##      V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13
## 1 32774  4  7  6  6 NA NA NA NA  NA  NA  NA  NA
## 2 49158  4 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 3 49160  7 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 4 32777  6 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 5 49166  2  6 NA NA NA NA NA NA  NA  NA  NA  NA
```

```
point_matrix = matrix(0, nrow = 4, ncol = 3)
range_matrix = matrix(0, nrow = 4, ncol = 3)

fun.rel1 = function (qVal, aVal) {
  if (qVal == aVal)
    return (1)
  else if (qVal > aVal)
    return (0)
  else
    return (2)
}

fun.relv1 = function (vals) {
  qVal = vals[1]
  aVal = vals[2]
  if (qVal == aVal)
    return (1)
  else if (qVal > aVal)
    return (0)
  else
    return (2)
}


fun.rel2 = function (qVal_min, qVal_max, aVal) {
  if (qVal_min > aVal)
    return (0)
  else if (qVal_max < aVal)
    return (2)
  else
    return (1)
}

fun.relv2 = function (vals) {
```

```
  qVal_min = vals[1]
  qVal_max = vals[2]
  aVal = vals[3]
  if (qVal_min > aVal)
    return (0)
  else if (qVal_max < aVal)
    return (2)
  else
    return (1)
}

for (i in 1:length(all_questions$V1)) {
  id = all_questions[i, 1]

  num_vec = as.numeric(all_questions[i,2:4])
  num_vec = num_vec[!is.na(num_vec)]

  ans_vec = all_answers[all_answers$V1 == id, 2:13]
  ans_vec = ans_vec[!is.na(ans_vec)]

  if (length(num_vec) > 0 && length(ans_vec) > 0 ) {
  min_val = min(num_vec)
  max_val = max(num_vec)


  min_ans = min(ans_vec)
  max_ans = max(ans_vec)
  median_ans = median(ans_vec)

  #if (min_val == max_val) { #point-based
  if (length(num_vec) == 1) { #SWQ

    res = colwise(fun.relv1)(rbind(
      as.data.frame(
        matrix(data = min_val, nrow = 1, ncol = length(ans_vec))), c(ans_vec)))
    point_matrix[1,1] = point_matrix[1,1] + length(which(res == 0))
    point_matrix[1,2] = point_matrix[1,2] + length(which(res == 1))
    point_matrix[1,3] = point_matrix[1,3] + length(which(res == 2))

    min_rel = fun.rel1(min_val, min_ans)
    point_matrix[2, min_rel+1] = point_matrix[2, min_rel+1] + 1

    median_rel = fun.rel1(min_val, median_ans)
    point_matrix[3, median_rel+1] = point_matrix[3, median_rel+1] + 1

    max_rel = fun.rel1(min_val, max_ans)
    point_matrix[4, max_rel+1] = point_matrix[4, max_rel+1] + 1

  } else {#DWQ #range-based
    res = colwise(fun.relv2)(rbind(as.data.frame(
      matrix(data = min_val, nrow = 1, ncol = length(ans_vec))),
      as.data.frame(matrix(data = max_val, nrow = 1, ncol = length(ans_vec))), c(ans_vec)))
    range_matrix[1,1] = range_matrix[1,1] + length(which(res == 0))
```

```
    range_matrix[1,2] = range_matrix[1,2] + length(which(res == 1))
    range_matrix[1,3] = range_matrix[1,3] + length(which(res == 2))

    min_rel = fun.rel2(min_val, max_val, min_ans)
    range_matrix[2, min_rel+1] = range_matrix[2, min_rel+1] + 1

    median_rel = fun.rel2(min_val, max_val, median_ans)
    range_matrix[3, median_rel+1] = range_matrix[3, median_rel+1] + 1

    max_rel = fun.rel2(min_val, max_val, max_ans)
    range_matrix[4, max_rel+1] = range_matrix[4, max_rel+1] + 1
  }
  }
}

df_range = as.data.frame(range_matrix)
colnames(df_range) = c("lower than", "between-equal", "greater than")
rownames(df_range) = c("each_value", "min_value", "median_value", "max_value")
df_range # prominence relation between detailed where questions and their answers
```

```
##              lower than between-equal greater than
## each_value          499          3844         1353
## min_value           376          2209          290
## median_value        189          2242          444
## max_value           101          1583         1191
```

```
df_point = as.data.frame(point_matrix)
colnames(df_point) = c("lower than", "equal", "greater than")
rownames(df_point) = c("each_value", "min_value", "median_value", "max_value")
df_point # prominence relation between simple where questions and their answers
```

```
##              lower than equal greater than
## each_value         1216   779         4142
## min_value           932   441         1648
## median_value        594   302         2125
## max_value           310   245         2466
```

```
write.csv(df_range, "result/dwq_prominence.csv")
write.csv(df_point, "result/swq_prominence.csv")

############################SCALE#########################

all_questions <- read.table("../sequences/scale-nf-Q-int.txt",
                            header = FALSE, sep = " ",
                            col.names = paste0("V",seq_len(4)), fill = TRUE)
head(all_questions, 5)
```

```
##      V1 V2 V3 V4
## 1 24576  6  8 NA
## 2 40965  7  5 NA
## 3 49158  7  8 NA
## 4 24583  3 NA NA
## 5 49166  7 NA NA
```

```r
all_answers <- read.table("../sequences/scale-nf-A-int.txt",
                          header = FALSE, sep = " ",
                          col.names = paste0("V",seq_len(13)), fill = TRUE)
head(all_answers, 5)
```

```
##      V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13
## 1 24576  7  9 NA NA NA NA NA NA  NA  NA  NA  NA
## 2 40965  5 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 3 49158  7 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 4 24583  9 NA NA NA NA NA NA NA  NA  NA  NA  NA
## 5 49166  4  8 NA NA NA NA NA NA  NA  NA  NA  NA
```

```r
point_matrix = matrix(0, nrow = 4, ncol = 3)
range_matrix = matrix(0, nrow = 4, ncol = 3)


for (i in 1:length(all_questions$V1)) {
  id = all_questions[i, 1]

  num_vec = as.numeric(all_questions[i,2:4])
  num_vec = num_vec[!is.na(num_vec)]

  ans_vec = all_answers[all_answers$V1 == id, 2:13]
  ans_vec = ans_vec[!is.na(ans_vec)]

  if (length(num_vec) > 0 && length(ans_vec) > 0) {

  min_val = min(num_vec)
  max_val = max(num_vec)



  min_ans = min(ans_vec)
  max_ans = max(ans_vec)
  median_ans = median(ans_vec)

  #if (min_val == max_val) {#point-based
  if (length(num_vec) == 1) {#SWQ
    res = colwise(fun.relv1)(rbind(
      as.data.frame(
        matrix(data = min_val, nrow = 1, ncol = length(ans_vec))), c(ans_vec)))
    point_matrix[1,1] = point_matrix[1,1] + length(which(res == 0))
    point_matrix[1,2] = point_matrix[1,2] + length(which(res == 1))
    point_matrix[1,3] = point_matrix[1,3] + length(which(res == 2))

    min_rel = fun.rel1(min_val, min_ans)
    point_matrix[2, min_rel+1] = point_matrix[2, min_rel+1] + 1

    median_rel = fun.rel1(min_val, median_ans)
    point_matrix[3, median_rel+1] = point_matrix[3, median_rel+1] + 1

    max_rel = fun.rel1(min_val, max_ans)
    point_matrix[4, max_rel+1] = point_matrix[4, max_rel+1] + 1
```

```
  } else {#DWQ #range-based
    res = colwise(fun.relv2)(rbind(
      as.data.frame(
        matrix(data = min_val, nrow = 1, ncol = length(ans_vec))),
      as.data.frame(matrix(data = max_val, nrow = 1, ncol = length(ans_vec))), c(ans_vec)))
    range_matrix[1,1] = range_matrix[1,1] + length(which(res == 0))
    range_matrix[1,2] = range_matrix[1,2] + length(which(res == 1))
    range_matrix[1,3] = range_matrix[1,3] + length(which(res == 2))

    min_rel = fun.rel2(min_val, max_val, min_ans)
    range_matrix[2, min_rel+1] = range_matrix[2, min_rel+1] + 1

    median_rel = fun.rel2(min_val, max_val, median_ans)
    range_matrix[3, median_rel+1] = range_matrix[3, median_rel+1] + 1

    max_rel = fun.rel2(min_val, max_val, max_ans)
    range_matrix[4, max_rel+1] = range_matrix[4, max_rel+1] + 1
  }
  }
}

df_range = as.data.frame(range_matrix)
colnames(df_range) = c("lower than", "between-equal", "greater than")
rownames(df_range) = c("each_value", "min_value", "median_value", "max_value")
df_range # scale relation between detailed where questions and their answers
```

```
##              lower than between-equal greater than
## each_value          202          1908         1068
## min_value           174          1460          233
## median_value        113          1436          318
## max_value            68           770         1029
```

```
df_point = as.data.frame(point_matrix)
colnames(df_point) = c("lower than", "equal", "greater than")
rownames(df_point) = c("each_value", "min_value", "median_value", "max_value")
df_point # scale relation between simple where questions and their answers
```

```
##              lower than equal greater than
## each_value          602   466         2528
## min_value           481   300         1119
## median_value        275   271         1354
## max_value           202   121         1577
```

```
write.csv(df_range, "result/dwq_scale.csv")
write.csv(df_point, "result/swq_scale.csv")
```