tsp_complexity.R

hamze

2020-01-21

```
############################################################
############################################################
##################Type, Scale, Prominence###################
############################################################
############################################################

#####################################################
#Installing the packages
#install.packages("pastecs")
#install.packages("ggplot2")
#install.packages("dplyr")
#####################################################
#set workspace to this folder
setwd("D:/Work/IJGIS/R-scripts")
#####################################################
library(pastecs)
```

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

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
library(dplyr)
```

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:pastecs':
##
##     first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
############################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")
###################################FUCTIONS#####################################

fun.complexity.swq = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
    val = questions[i, 2]
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[!is.na(answer)]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
  return(comp)
}

fun.complexity.swqt = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
    val = questions[i, 2]
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[answer != ""]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
  return(comp)
}


fun.complexity.dwq = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
```

```
    question = questions[i, 2:length(questions)]
    question = question[!is.na(question)]
    val = min(question)
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[!is.na(answer)]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
  return(comp)
}


fun.complexity.dwq2 = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
    question = questions[i, 2:length(questions)]
    question = question[!is.na(question)]
    val = max(question)
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[!is.na(answer)]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
  return(comp)
}


fun.complexity.dwqt2 = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
    question = questions[i, ]
    question = question[question != ""]
    val = question[length(question)]
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[answer != ""]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
  return(comp)
}

fun.complexity.dwqt = function (questions, answers) {
  comp = matrix(nrow = length(questions[,1]), ncol = 3, data = 0)
  for (i in 1:length(questions[,1])) {
    id = questions[i, 1]
    val = questions[i, 2]
    answer = answers[answers$V1 == id, 2:13]
    answer <- answer[answer != ""]
    complexity = length(answer)
    comp[i,] = c (id, val, complexity)
  }
```

```
  return(comp)
}


fun.extract.ncomplex.ids = function(questions, n) {
  validIds = c()
  counter = 0
  for (i in 1:length(questions[,1])) {
    qVals = questions[i, 2:5]
    if (length(qVals[!is.na(qVals)]) == n) {
      counter= counter + 1
      validIds[counter] = questions[i, 1]
    }
  }
  return (validIds)
}

fun.extract.ncomplex.type.ids = function(questions, n) {
  validIds = c()
  counter = 0
  for (i in 1:length(questions[,1])) {
    qVals = questions[i, 2:5]
    if (length(qVals[qVals != ""]) == n) {
      counter= counter + 1
      validIds[counter] = questions[i, 1]
    }
  }
  return (validIds)
}

fun.extract.ncomplex.ids = function(questions, n) {
  validIds = c()
  counter = 0
  for (i in 1:length(questions[,1])) {
    qVals = questions[i, 2:length(questions)]
    if (length(qVals[!is.na(qVals)]) == n) {
      counter= counter + 1
      validIds[counter] = questions[i, 1]
    }
  }
  return (validIds)
}


fun.agg.stats = function (df) {
  colnames(df) <- c("val", "complexity")
  df$complexity <- as.numeric(df$complexity)
  result = df %>% group_by(df[,1]) %>% summarize(count=n(), min=min(complexity), median=median(complexi
  return (result)
}


fun.translate.types = function (question_complexity, type_kb) {
```

```
  type_generic_complexity <- as.data.frame(question_complexity, stringsAsFactors=FALSE)
  for (i in 1:length(type_kb$Type_code)) {
    type_generic_complexity[type_generic_complexity$V2 == type_kb$Q_Type_Code[i], 2] = type_kb$ID[i]
  }
  type_generic_complexity[type_generic_complexity$V2 == "Q-", 2] = 10#for unknown!!
  return(type_generic_complexity)
}

fun.reverse.translate.type.generic = function (aag_type_comp, type_kb) {
  result <- as.data.frame(aag_type_comp)
  for (i in 1:length(result[,1])) {
    if (result[i,1] == 10) {
      result[i, 1] = "UNK"
    } else {
    result[i, 1] = unique(type_kb[type_kb$ID == result[i, 1], 2])
    }
  }
  return (result)
}

fun.shared.ids = function (scale_questions, prominence_questions, type_questions) {
  sharedIds = intersect(as.numeric(scale_questions$V1), as.numeric(prominence_questions$V1))
  sharedIds = intersect(sharedIds, as.numeric(type_questions$V1))
  return(sharedIds)
}


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

############################READING DATA#########################################
q_scale_all <- read.table("../sequences/scale-nf-Q-int.txt",
                          header = FALSE, sep = " ",
                          col.names = paste0("V",seq_len(4)), fill = TRUE)
a_scale_all <-  read.table("../sequences/scale-nf-A-int.txt",
                           header = FALSE, sep = " ",
                           col.names = paste0("V",seq_len(13)), fill = TRUE)

scale_swq_ids <- fun.extract.ncomplex.ids(q_scale_all, n = 1)

q_scale_swq <- q_scale_all[q_scale_all$V1 %in% scale_swq_ids, ]
a_scale_swq <- a_scale_all[a_scale_all$V1 %in% scale_swq_ids, ]

q_scale_dwq <- q_scale_all[!q_scale_all$V1 %in% scale_swq_ids, ]
a_scale_dwq <- a_scale_all[!a_scale_all$V1 %in% scale_swq_ids, ]


q_prom_all <- read.table("../sequences/prominence-nf-Q-int.txt",
                         header = FALSE, sep = " ",
                         col.names = paste0("V",seq_len(5)), fill = TRUE)
a_prom_all <-  read.table("../sequences/prominence-nf-A-int.txt",
                          header = FALSE, sep = " ",
                          col.names = paste0("V",seq_len(15)), fill = TRUE)
```

```
prom_swq_ids <- fun.extract.ncomplex.ids(q_prom_all, n = 1)

q_prom_swq <- q_prom_all[q_prom_all$V1 %in% prom_swq_ids, ]
a_prom_swq <- a_prom_all[a_prom_all$V1 %in% prom_swq_ids, ]

q_prom_dwq <- q_prom_all[!q_prom_all$V1 %in% prom_swq_ids, ]
a_prom_dwq <- a_prom_all[!a_prom_all$V1 %in% prom_swq_ids, ]


type_kb <- read.csv("../gazetteers/type_geonames.csv", header=TRUE, stringsAsFactors = FALSE)
colnames(type_kb) <- c("ID", "Generic_Code", "Type_code", "Q_Type_Code", "Desc_short")
q_type_all <- read.table("../sequences/type-nf-Q.txt",
                         header = FALSE, sep = " ",
                         col.names = paste0("V",seq_len(5)), fill = TRUE, stringsAsFactors=FALSE)
a_type_all <-  read.table("../sequences/type-nf-A.txt",
                          header = FALSE, sep = " ",
                          col.names = paste0("V",seq_len(15)), fill = TRUE, stringsAsFactors=FALSE)

type_swq_ids <- fun.extract.ncomplex.type.ids(q_type_all, n = 1)

q_type_swq <- q_type_all[q_type_all$V1 %in% type_swq_ids, ]
a_type_swq <- a_type_all[a_type_all$V1 %in% type_swq_ids, ]

q_type_dwq <- q_type_all[!q_type_all$V1 %in% type_swq_ids, ]
a_type_dwq <- a_type_all[!a_type_all$V1 %in% type_swq_ids, ]

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

############################TRANSFORMATION####################################
scale_swq_comp <- as.data.frame(fun.complexity.swq(q_scale_swq, a_scale_swq))
scale_dwq_comp <- as.data.frame(fun.complexity.dwq(q_scale_dwq, a_scale_dwq)) #min
scale_dwq_comp2 <- as.data.frame(fun.complexity.dwq2(q_scale_dwq, a_scale_dwq)) #max
scale_all_comp <- as.data.frame(fun.complexity.dwq(q_scale_all, a_scale_all)) #min

prom_swq_comp <- as.data.frame(fun.complexity.swq(q_prom_swq, a_prom_swq))
prom_dwq_comp <- as.data.frame(fun.complexity.dwq(q_prom_dwq, a_prom_dwq)) #min
prom_dwq_comp2 <- as.data.frame(fun.complexity.dwq2(q_prom_dwq, a_prom_dwq)) #max
prom_all_comp <- as.data.frame(fun.complexity.dwq(q_prom_all, a_prom_all)) #min


type_swq_comp <- as.data.frame(
  fun.complexity.swqt(q_type_swq, a_type_swq), stringsAsFactors=FALSE)
type_dwq_comp <- as.data.frame(
  fun.complexity.dwqt(q_type_dwq, a_type_dwq), stringsAsFactors=FALSE) #first
type_dwq_comp2 <- as.data.frame(
  fun.complexity.dwqt2(q_type_dwq, a_type_dwq), stringsAsFactors=FALSE) #last

type_all_comp <- as.data.frame(
  fun.complexity.dwqt(q_type_all, a_type_all), stringsAsFactors=FALSE) #first

type_agg_swq_comp <- fun.translate.types(type_swq_comp, type_kb)
type_agg_dwq_comp <- fun.translate.types(type_dwq_comp, type_kb)#first
type_agg_dwq_comp2 <- fun.translate.types(type_dwq_comp2, type_kb)#last
```

```
type_agg_all_comp <- fun.translate.types(type_all_comp, type_kb)#first
##############################################################################

#####################SIMPLE STATISTICS#########################################
shared_ids_all <- fun.shared.ids(q_scale_all, q_prom_all, q_type_all) #3932 all shared ones
shared_ids_swq <- fun.shared.ids(q_scale_swq, q_prom_swq, q_type_swq) #1983 shared swqs
shared_ids_dwq <- fun.shared.ids(q_scale_dwq, q_prom_dwq, q_type_dwq) #1949 shared dwqs

q_type_shared_swq <- q_type_swq[as.numeric(q_type_swq$V1) %in% shared_ids_swq, ]
a_type_shared_swq <- a_type_swq[as.numeric(a_type_swq$V1) %in% shared_ids_swq, ]
q_type_shared_dwq <- q_type_dwq[as.numeric(q_type_dwq$V1) %in% shared_ids_dwq, ]
a_type_shared_dwq <- a_type_dwq[as.numeric(a_type_dwq$V1) %in% shared_ids_dwq, ]
type_swq_comp_shared <- as.data.frame(
  fun.complexity.swqt(q_type_shared_swq, a_type_shared_swq), stringsAsFactors=FALSE)
type_dwq_comp_shared <- as.data.frame(
  fun.complexity.dwqt(q_type_shared_dwq, a_type_shared_dwq), stringsAsFactors=FALSE) #first

stat_type_swq_shared <- stat.desc(as.numeric(type_swq_comp_shared$V3))[4:14]
stat_type_dwq_shared <- stat.desc(as.numeric(type_dwq_comp_shared$V3))[4:14]
summary(as.numeric(type_swq_comp_shared$V3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.292   3.000  12.000
```

```
summary(as.numeric(type_dwq_comp_shared$V3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.912   2.000  12.000
```

```
#CONCLUSION: DWQ is less complex compared to SWQ:: based on mean, and variance! (scale)


#TYPE as the most complete one!
stat_type_swq <- stat.desc(as.numeric(type_swq_comp$V3))[4:14]
stat_type_dwq <- stat.desc(as.numeric(type_dwq_comp$V3))[4:14]
#CONCLUSION: DWQ is less complex compared to SWQ:: based on mean, and variance! (scale)
summary(as.numeric(type_swq_comp$V3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.206   3.000  12.000
```

```
summary(as.numeric(type_dwq_comp$V3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    2.00    1.88    2.00   12.00
```

```
swqVsdwq = matrix(ncol = 12, nrow = 6, data = 0)
colnames(swqVsdwq) <- c("type", names(stat_type_dwq))
swqVsdwq[1, 1] = as.character("Shared Sequences")
swqVsdwq[2, 1] = as.character("SWQ")
```

```
swqVsdwq[2, 2:12] = stat_type_swq_shared
swqVsdwq[3, 1] = as.character("DWQ")
swqVsdwq[3, 2:12] = stat_type_dwq_shared
swqVsdwq[4, 1] = as.character("Type Sequences")
swqVsdwq[5, 1] = as.character("SWQ")
swqVsdwq[5, 2:12] = stat_type_swq
swqVsdwq[6, 1] = as.character("DWQ")
swqVsdwq[6, 2:12] = stat_type_dwq
swqVsdwq
```

```
##      type               min max  range sum    median mean
## [1,] "Shared Sequences" "0" "0"  "0"   "0"    "0"    "0"
## [2,] "SWQ"             "1" "12" "11"  "4350" "2"    "2.29188619599579"
## [3,] "DWQ"             "1" "12" "11"  "3556" "2"    "1.91182795698925"
## [4,] "Type Sequences"  "0" "0"  "0"   "0"    "0"    "0"
## [5,] "SWQ"             "1" "12" "11"  "7098" "2"    "2.20571783716594"
## [6,] "DWQ"             "1" "12" "11"  "5429" "2"    "1.88049878766886"
##      SE.mean              CI.mean.0.95         var
## [1,] "0"                  "0"                  "0"
## [2,] "0.0303124017227229" "0.05944914626826"   "1.7439615431831"
## [3,] "0.0239147831818559" "0.0469026509477319" "1.06376534962143"
## [4,] "0"                  "0"                  "0"
## [5,] "0.0226986702291327" "0.0445053207289192" "1.65800894989001"
## [6,] "0.0195355858903169" "0.0383051094951428" "1.10179212811703"
##      std.dev            coef.var
## [1,] "0"                "0"
## [2,] "1.3205913611648"  "0.576202851377193"
## [3,] "1.03139000849409" "0.539478463385545"
## [4,] "0"                "0"
## [5,] "1.28763696354602" "0.583772294828273"
## [6,] "1.04966286402684" "0.558183217617516"
```

```
write.csv(file = "result/swqVsdwq.csv", x = swqVsdwq, row.names = F)
################################################################################

#######################AGGREGATED STATISTICS#####################################
names_agg <- c("value", "count", "min", "median", "mean", "max")

agg_stats_scale_all <- fun.agg.stats(scale_all_comp[,2:3])
names(agg_stats_scale_all) <- names_agg
agg_stats_scale_all
```

```
## # A tibble: 8 x 6
##   value count   min median  mean   max
##   <dbl> <int> <dbl>  <dbl> <dbl> <dbl>
## 1     3    14     1    1    1.5      3
## 2     4   216     1    2    1.78     5
## 3     5   523     1    2    1.81    10
## 4     6  1905     1    2    1.85    12
## 5     7   613     1    1    1.66     7
## 6     8   378     1    2    1.71     6
## 7     9   110     1    1    1.95     8
## 8    10     8     1    2.5  3.38     9
```

```
write.csv(file = "result/scale_all.csv", x = agg_stats_scale_all, row.names = F)

agg_stats_scale_swq <- fun.agg.stats(scale_swq_comp[,2:3])
agg_stats_scale_dwq <- fun.agg.stats(scale_dwq_comp[,2:3])
agg_stats_scale_dwq2 <- fun.agg.stats(scale_dwq_comp2[,2:3])


agg_stats_prom_all <- fun.agg.stats(prom_all_comp[,2:3])
names(agg_stats_prom_all) <- names_agg
agg_stats_prom_all
```

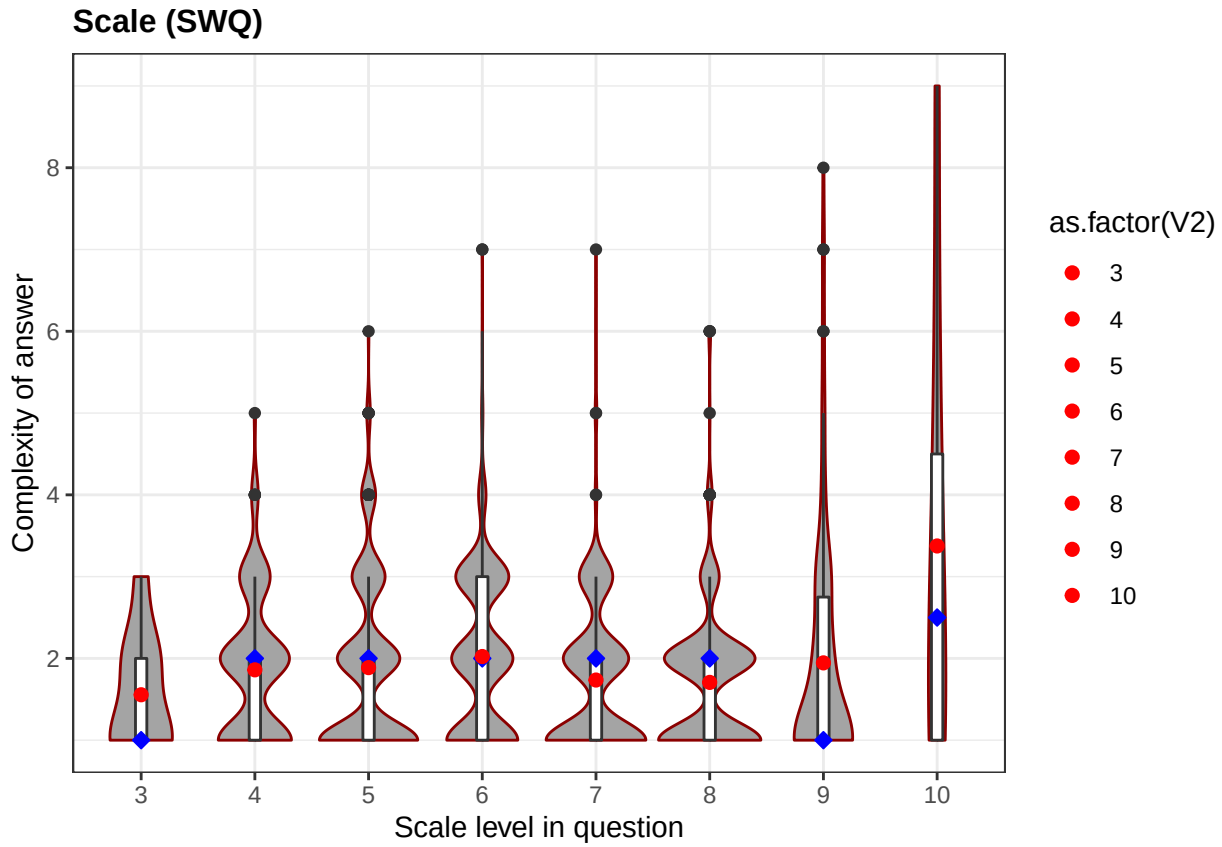
```
## # A tibble: 7 x 6
##   value count   min median  mean   max
##   <dbl> <int> <dbl>  <dbl> <dbl> <dbl>
## 1     1   483     1      2  1.95    12
## 2     2  1702     1      2  2.04     9
## 3     3  1809     1      2  2.03    10
## 4     4   871     1      2  2.01    12
## 5     5   473     1      2  1.91     8
## 6     6   436     1      2  1.98    10
## 7     7   122     1      2  1.89     7
```

```
write.csv(file = "result/prominence_all.csv", x = agg_stats_prom_all, row.names = F)


agg_stats_prom_swq <- fun.agg.stats(prom_swq_comp[,2:3])
agg_stats_prom_dwq <- fun.agg.stats(prom_dwq_comp[,2:3])
agg_stats_prom_dwq2 <- fun.agg.stats(prom_dwq_comp2[,2:3])


agg_stats_type_all <- fun.agg.stats(type_all_comp[,2:3])
names(agg_stats_type_all) <- names_agg
agg_stats_type_all
```

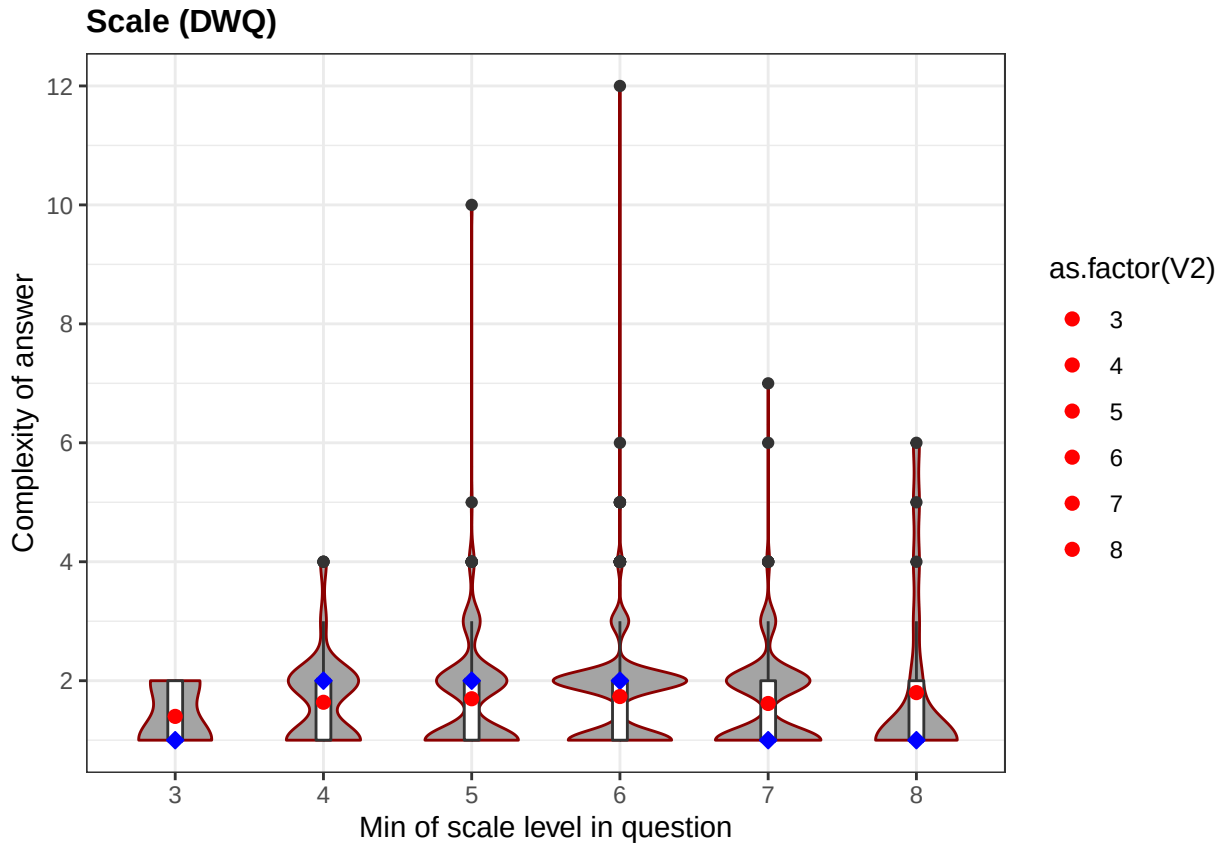
```
## # A tibble: 187 x 6
##    value   count   min median  mean   max
##    <chr>   <int> <dbl>  <dbl> <dbl> <dbl>
##  1 Q-        480     1    2    2.02     6
##  2 Q-ADM1    424     1    2    1.91     8
##  3 Q-ADM1H     3     1    2    2        3
##  4 Q-ADM2    797     1    2    1.84     9
##  5 Q-ADM2H     2     1    2    2        3
##  6 Q-ADM3    209     1    2    2.32     7
##  7 Q-ADM4    170     1    2    2.15     7
##  8 Q-ADM4H     2     2    2.5  2.5      3
##  9 Q-ADM5      7     1    2    1.57     2
## 10 Q-ADMD     37     1    2    2.03     4
## # ... with 177 more rows
```

```
write.csv(file = "result/type_all.csv", x = agg_stats_type_all, row.names = F)
```

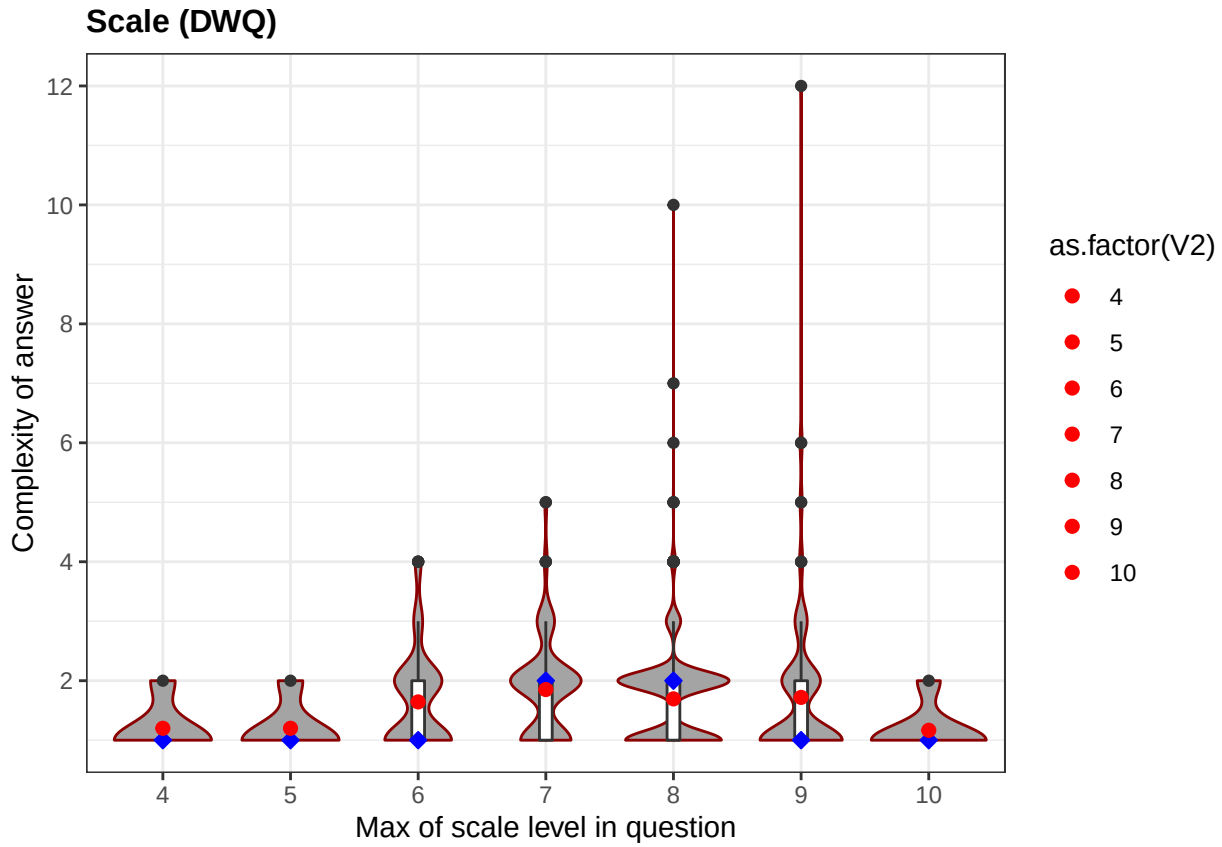
```
agg_stats_generic_type_all <- fun.agg.stats(type_agg_all_comp[,2:3])
names(agg_stats_generic_type_all) <- names_agg
write.csv(file = "result/gtype_all.csv", x = agg_stats_generic_type_all, row.names = F)


agg_ngstats_type_swq <- fun.agg.stats(type_swq_comp[,2:3])
agg_ngstats_type_dwq <- fun.agg.stats(type_dwq_comp[,2:3])
agg_ngstats_type_dwq2 <- fun.agg.stats(type_dwq_comp2[,2:3])


type_agg_swq_comp[,2] <- as.numeric(type_agg_swq_comp[,2])
type_agg_swq_comp[,3] <- as.numeric(type_agg_swq_comp[,3])
agg_stats_type_swq <- fun.agg.stats(type_agg_swq_comp[,2:3])

type_agg_dwq_comp[,2] <- as.numeric(type_agg_dwq_comp[,2])
type_agg_dwq_comp[,3] <- as.numeric(type_agg_dwq_comp[,3])
agg_stats_type_dwq <- fun.agg.stats(type_agg_dwq_comp[,2:3])

type_agg_dwq_comp2[,2] <- as.numeric(type_agg_dwq_comp2[,2])
type_agg_dwq_comp2[,3] <- as.numeric(type_agg_dwq_comp2[,3])
agg_stats_type_dwq2 <- fun.agg.stats(type_agg_dwq_comp2[,2:3])
##############################################################################

#######################VIOLIN PLOTS#############################################
ggplot(scale_swq_comp, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") +
  geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Scale (SWQ)",x="Scale level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



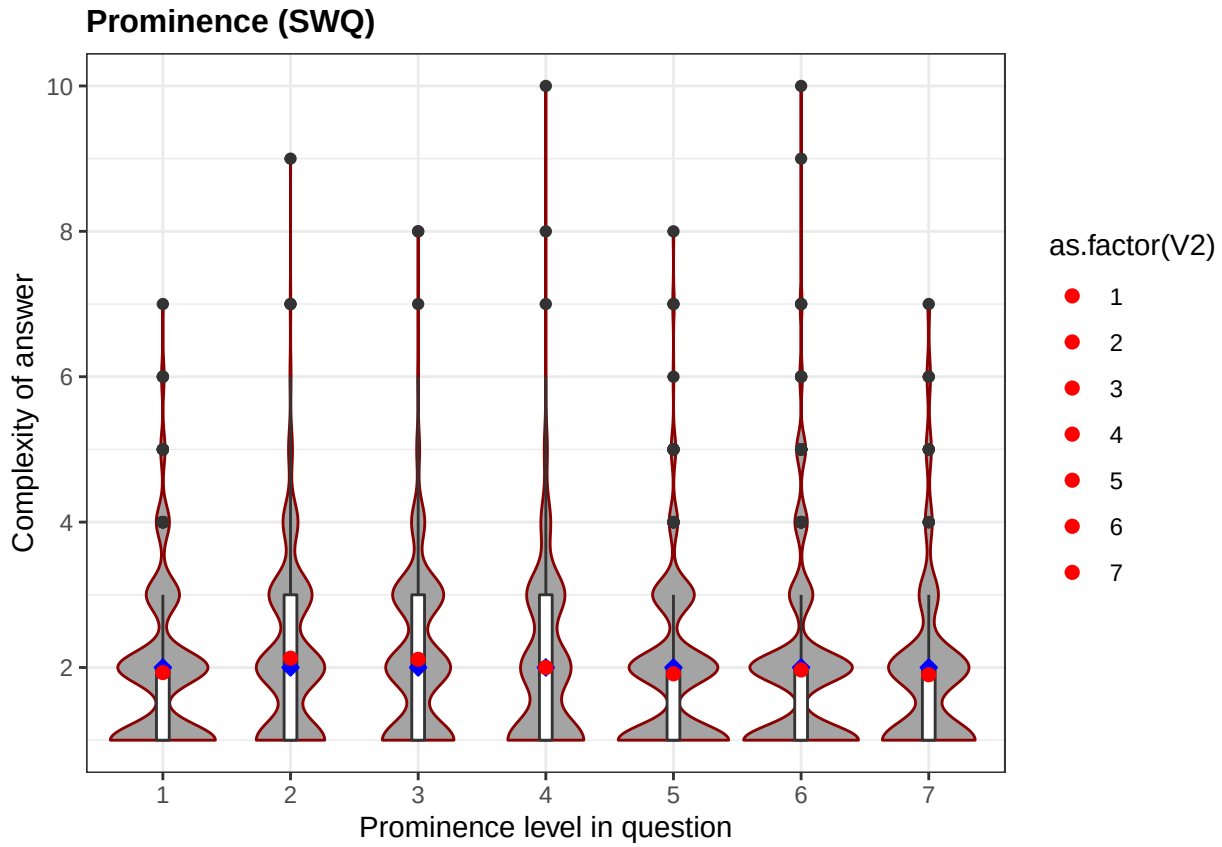

```
ggplot(scale_dwq_comp, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Scale (DWQ)",x="Min of scale level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



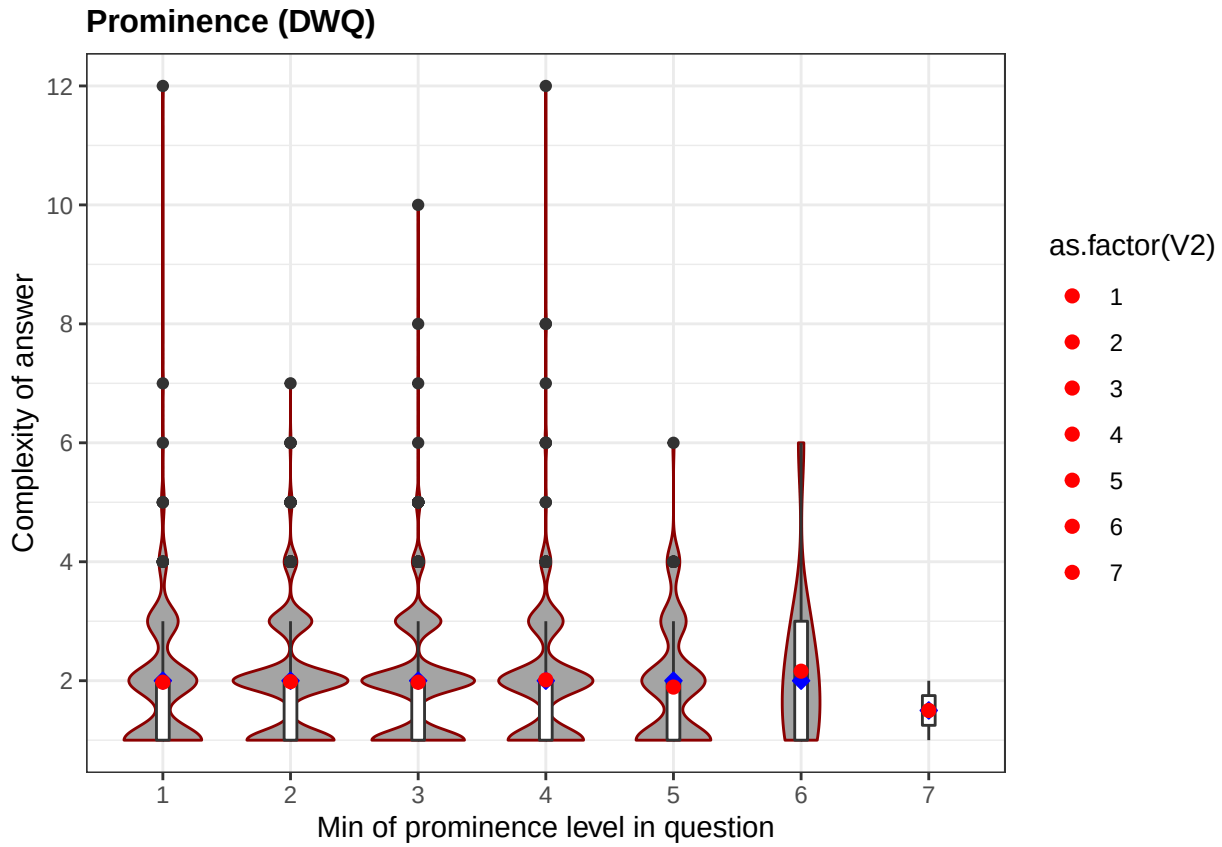

```
ggplot(scale_dwq_comp2, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Scale (DWQ)",x="Max of scale level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



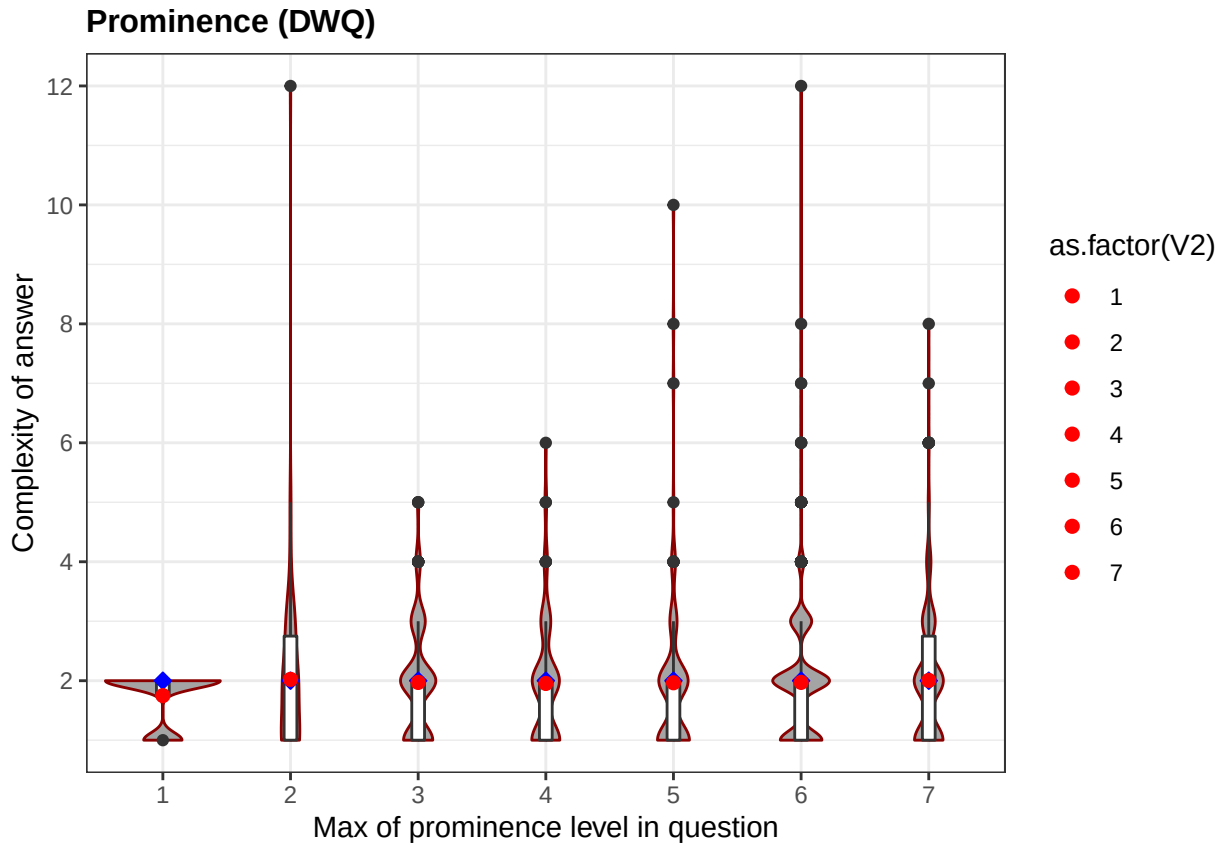

```
#CONCLUSION: 1- SWQ vs. DWQ --> White Box --> less complex!


ggplot(prom_swq_comp, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Prominence (SWQ)",x="Prominence level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



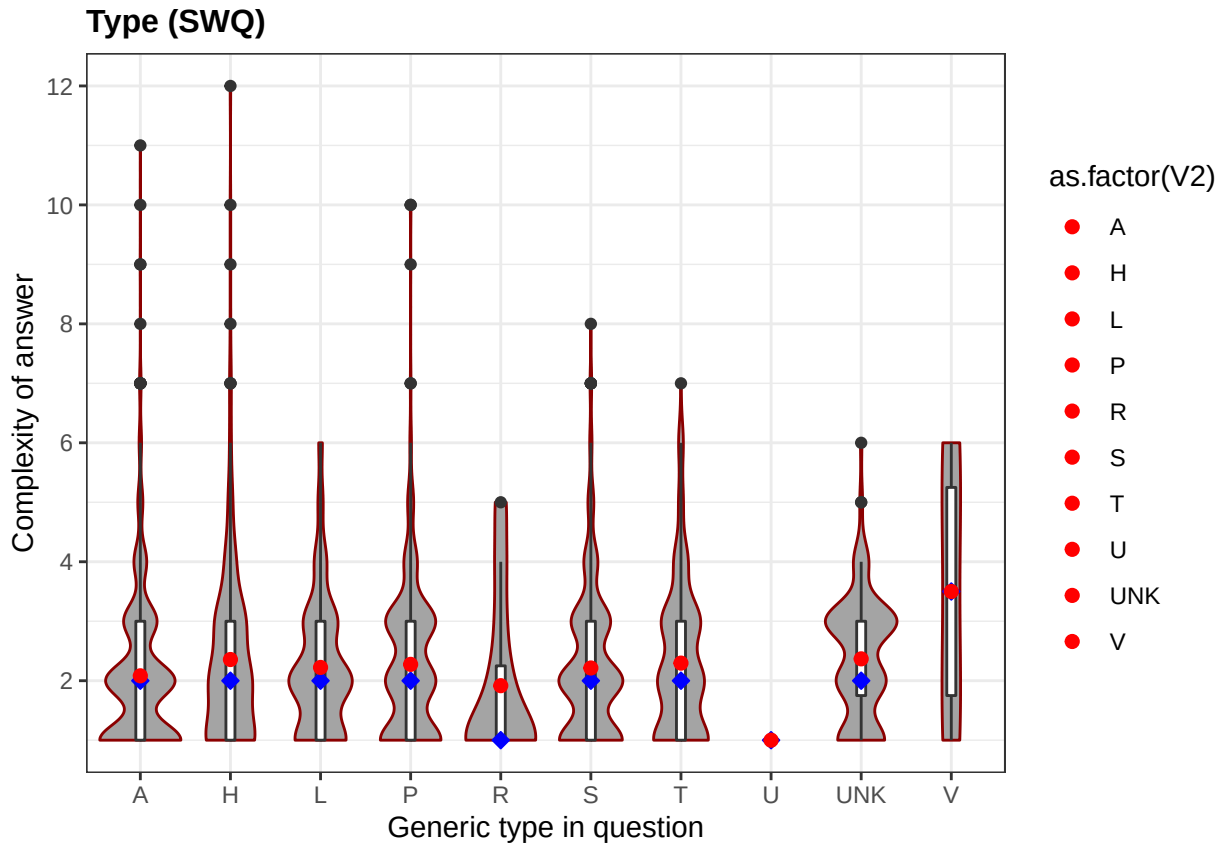

```
ggplot(prom_dwq_comp, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Prominence (DWQ)",x="Min of prominence level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



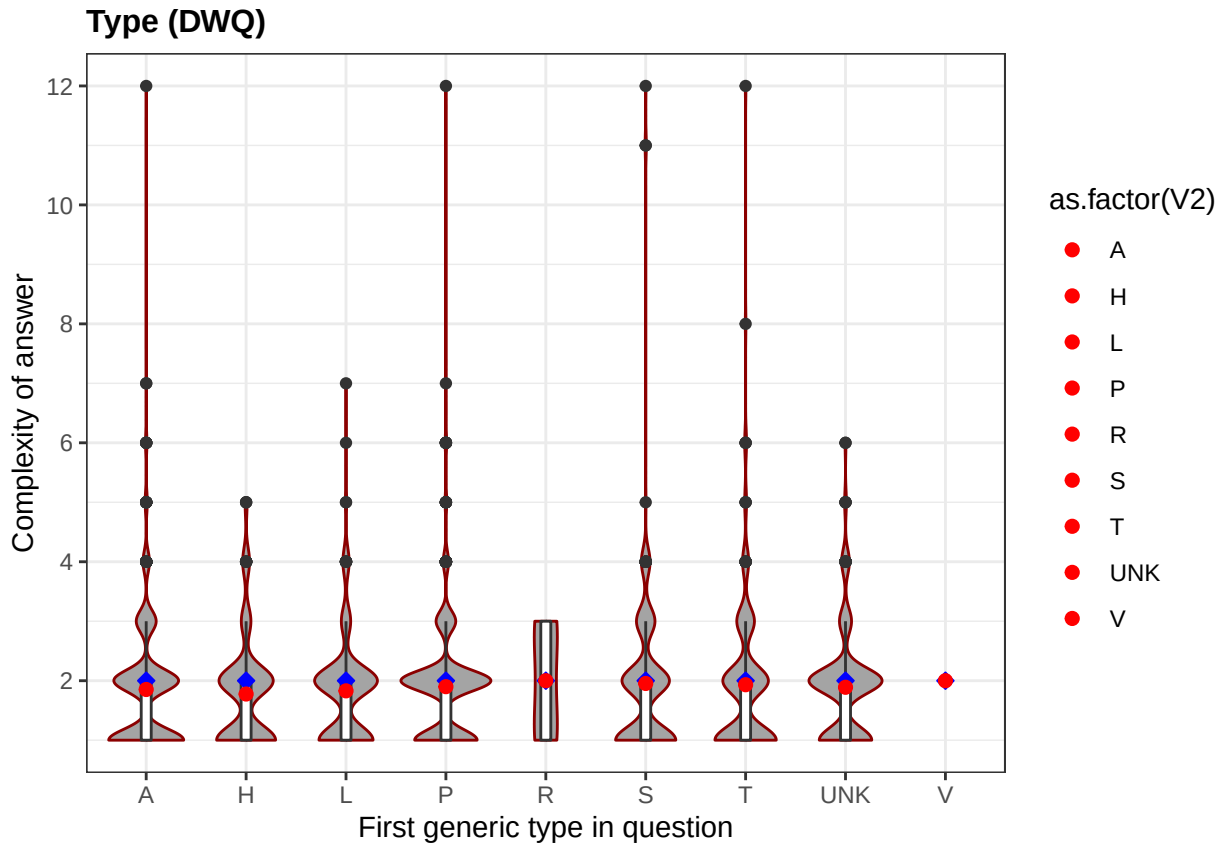

```
ggplot(prom_dwq_comp2, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Prominence (DWQ)",x="Max of prominence level in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```



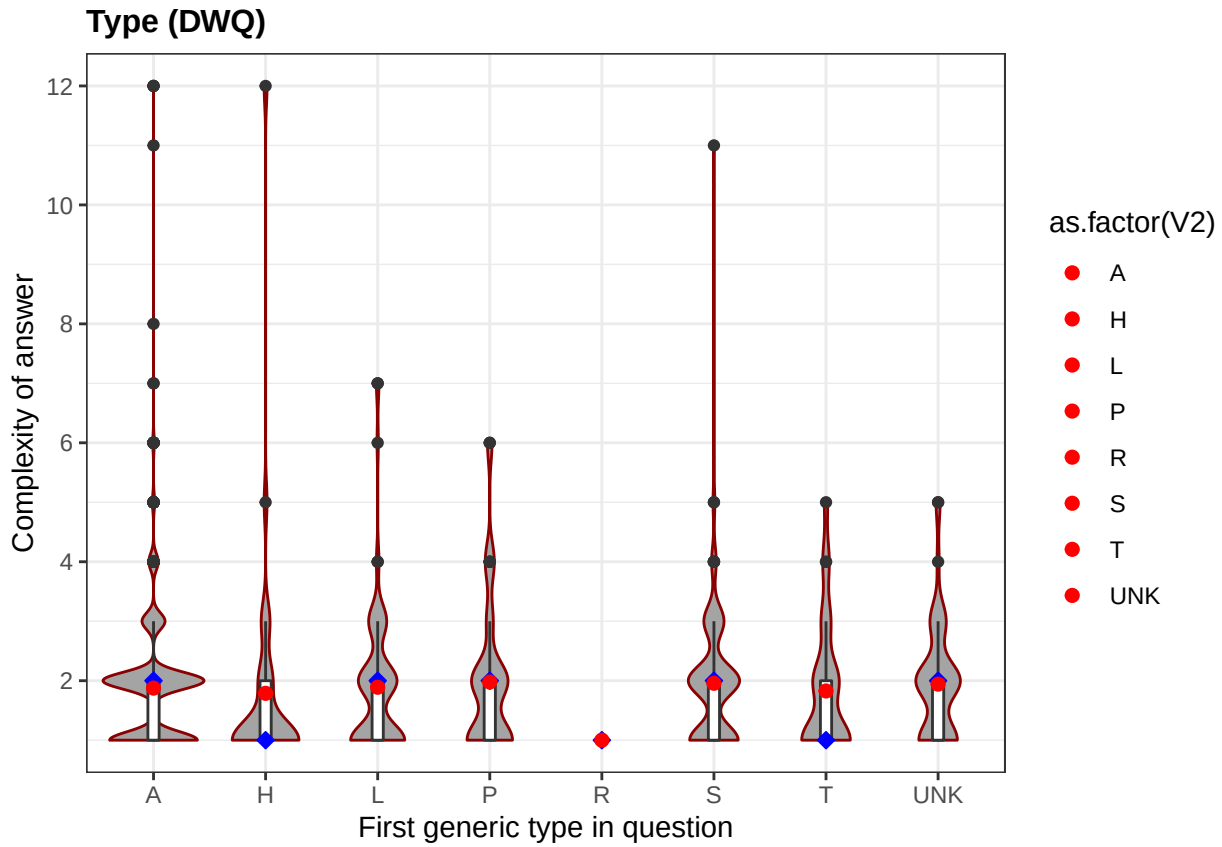

```
#CONCLUSION: 1- SWQ vs. DWQ --> White Box --> less complex!

type_swq_comp_generic <- fun.reverse.translate.type.generic(type_agg_swq_comp[,2:3], type_kb)
ggplot(type_swq_comp_generic, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Type (SWQ)",x="Generic type in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```




```
type_dwq_comp_generic <- fun.reverse.translate.type.generic(type_agg_dwq_comp[,2:3], type_kb)
ggplot(type_dwq_comp_generic, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Type (DWQ)",x="First generic type in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```




```
type_dwq_comp_generic2 <- fun.reverse.translate.type.generic(type_agg_dwq_comp2[,2:3], type_kb)
ggplot(type_dwq_comp_generic2, aes(x=as.factor(V2), y=V3, fill = as.factor(V2))) +
  geom_violin(fill='#A4A4A4', color="darkred") + geom_boxplot(width=0.1, fill="white") +
  stat_summary(fun.y=median, geom="point", shape=23, size=2, color="blue", fill="blue") +
  stat_summary(fun.y=mean, geom="point", size=2, color="red") +
  labs(title="Type (DWQ)",x="First generic type in question", y = "Complexity of answer") +
  theme_minimal() + theme_bw() +
  theme(plot.title = element_text(color = "black", size = "12", face = "bold")) +
  scale_y_continuous(breaks=seq(0,12,2))
```




```
#CONCLUSION: 1- SWQ vs. DWQ --> White Box --> less complex!
################################################################################
write.csv(file = "result/ng-swq-type-comp.csv", x = type_swq_comp)
write.csv(file = "result/ng-dwq-type-comp.csv", x = type_dwq_comp)
```