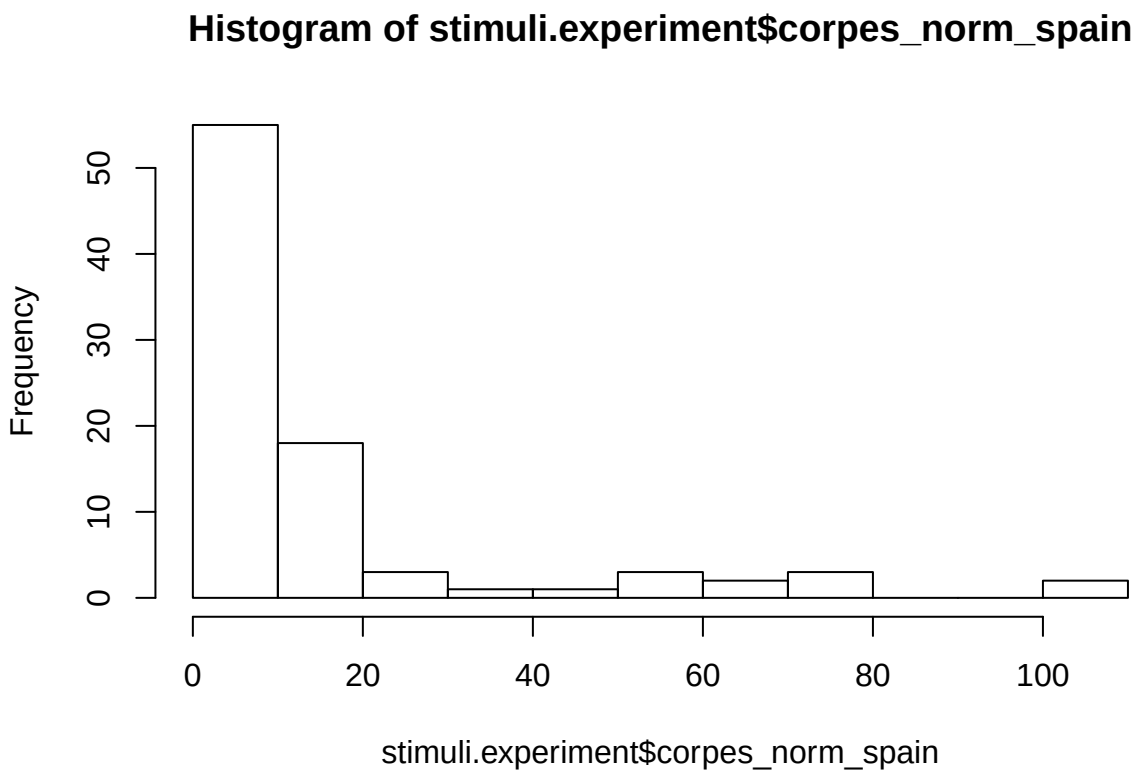
# StimuliStatistics

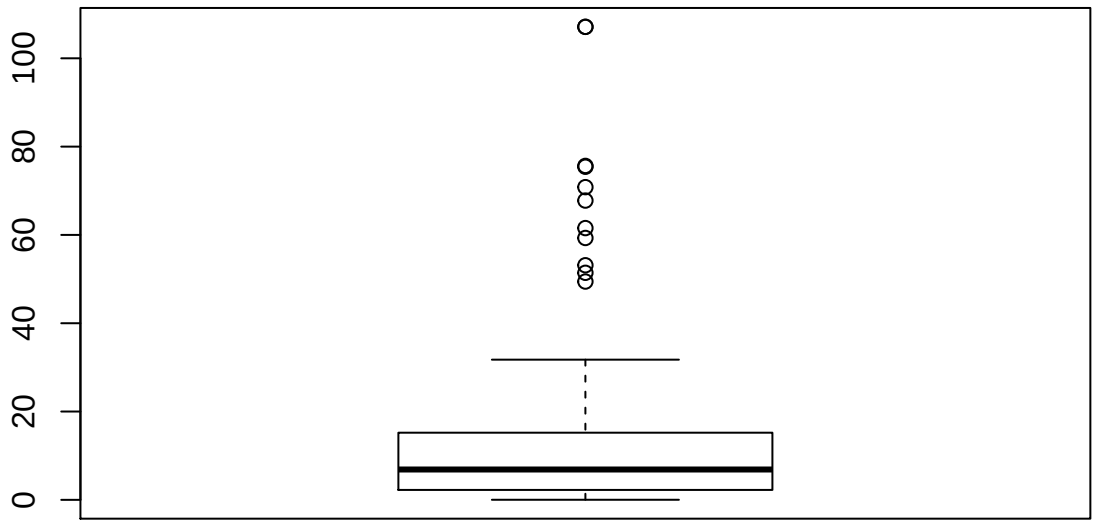
```
library("pacman")
pacman::p_load("dplyr","tidyr","readxl")
## we are going to do bootstrap tp the amount of letters of the word. for this I am going to use the bo
```

```
stimuli.all <- read.csv2("StimuliProposal_SpeechExperiment.csv",sep = ",")# load file
stimuli.all$sound <- as.character(stimuli.all$sound)
stimuli <- na.omit(as.data.frame(stimuli.all)) # delete NAs
stimuli <- stimuli %>% dplyr::select(stimuli, category, condition, sound, corpes_norm_spain)# select va
stimuli$stimuli <- as.character(stimuli$stimuli) #change variables detected automatically as factor, to
stimuli$category <- as.character(stimuli$category) #change variables detected automatically as factor,
stimuli.objects <- filter(stimuli, category=="objects")
stimuli.food <- filter(stimuli, category=="food")
stimuli.experiment <- rbind(stimuli.objects,stimuli.food)# this are the stimui we are going to use
#making a factor variable to make analysis easier
stimuli.experiment$sem.cat <- as.factor(stimuli.experiment$category)
```

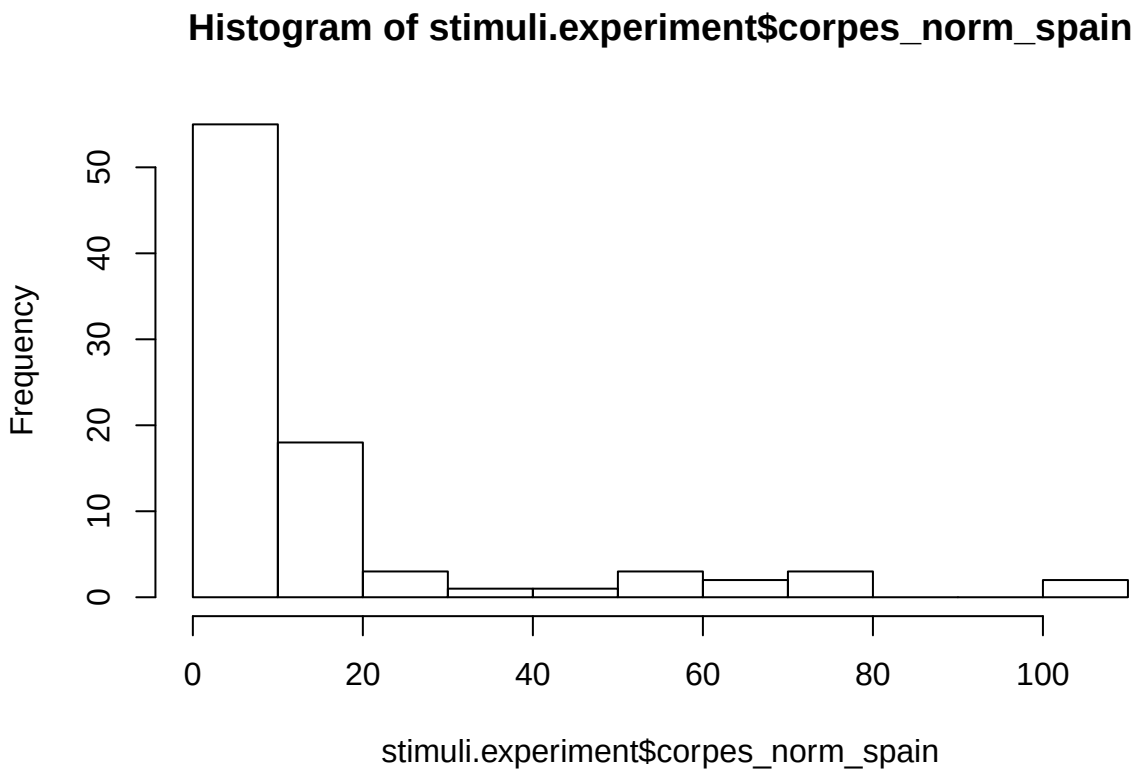
```
#plotting histogram and bosplot of distribution of general normalized frecuency
# making plot objects
general.hist <- hist(stimuli.experiment$corpes_norm_spain)
```



```r
general.boxplot <- boxplot(stimuli.experiment$corpes_norm_spain)
```

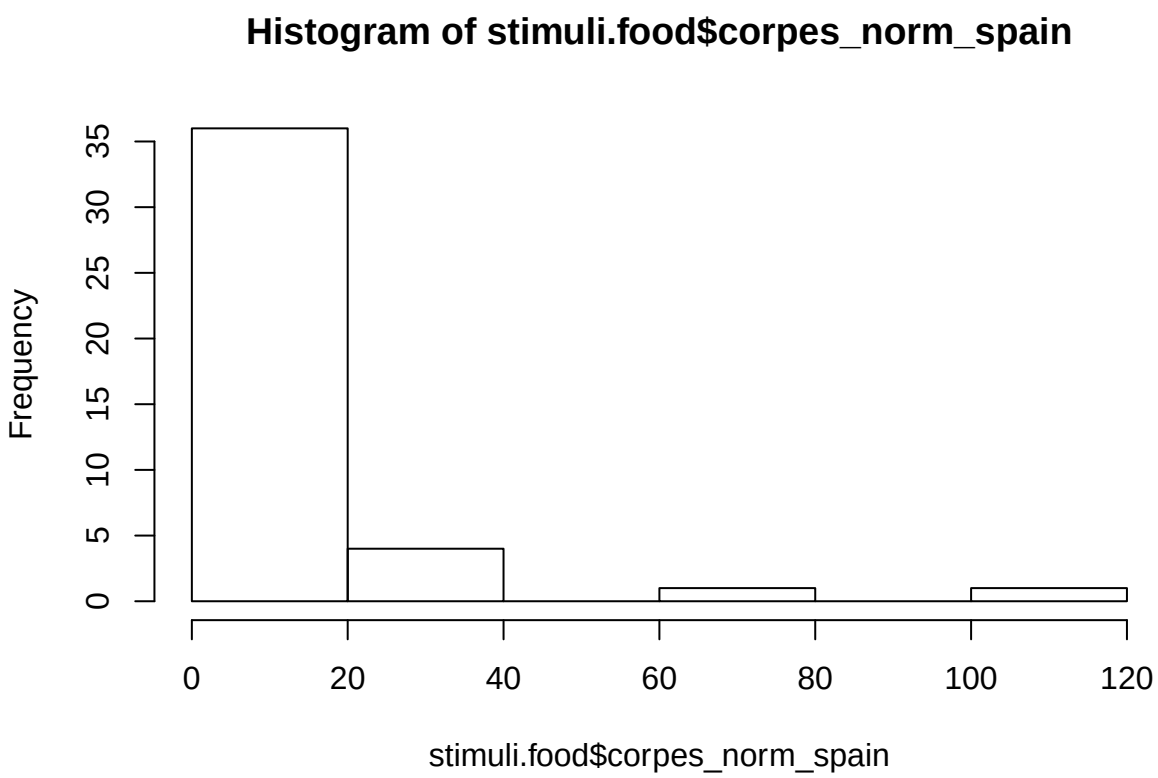



```r
####### actually plotting
plot(general.hist)
```

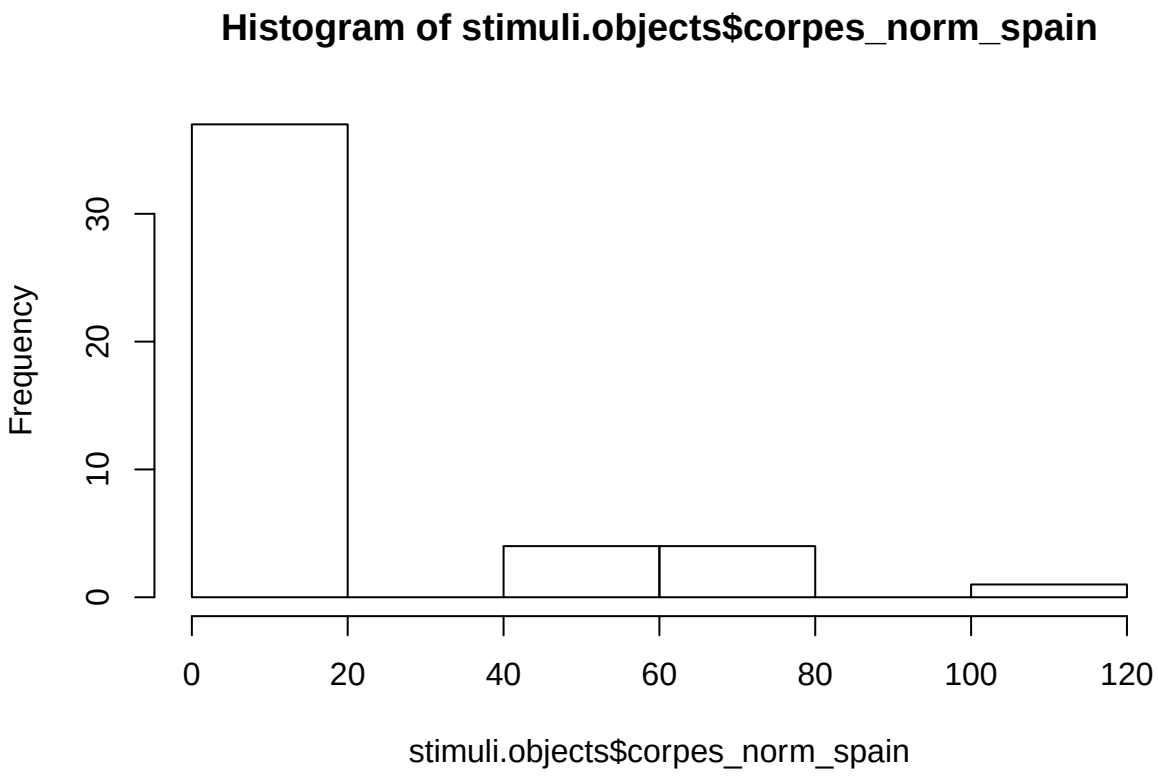



```
#general.boxplot
```

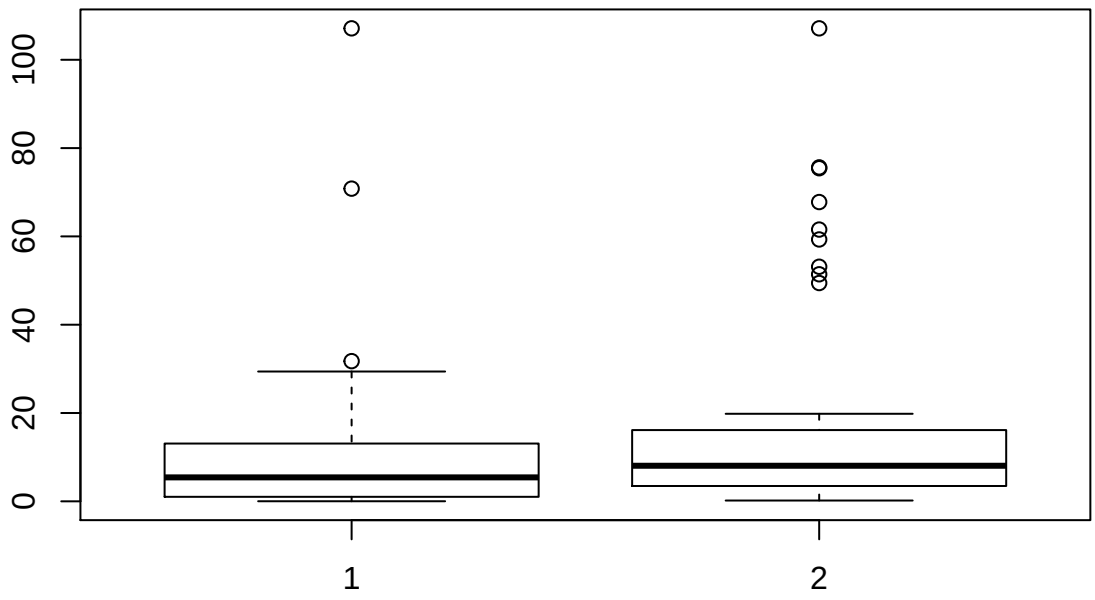
```
#plotting individual semantic categories
hist.food <- hist(stimuli.food$corpes_norm_spain)
```



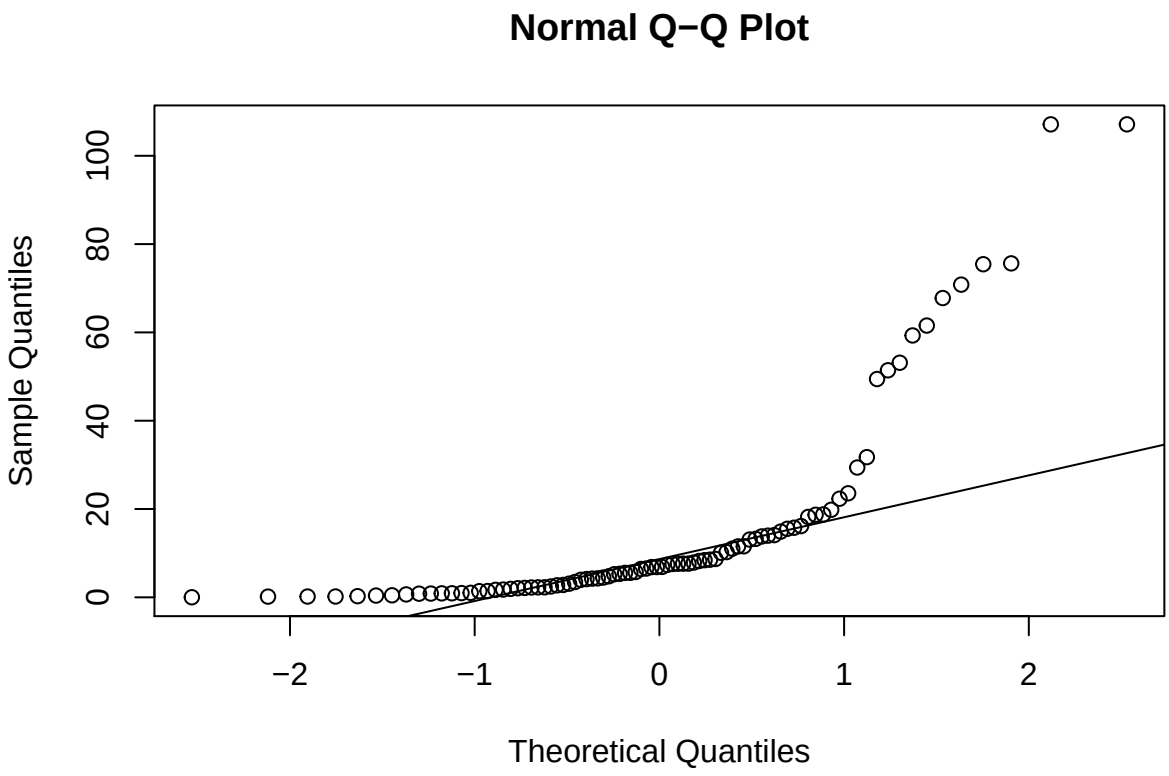

```
hist.object <- hist(stimuli.objects$corpes_norm_spain)
```




```
# side by side boxplots
joint.boxplot <- boxplot(stimuli.food$corpes_norm_spain, stimuli.objects$corpes_norm_spain)
```




```
# checking for normality
qqnorm(stimuli.experiment$corpes_norm_spain)
qqline(stimuli.experiment$corpes_norm_spain)
```




```
#taking outliers out . Fromo graphical inspection it looks that for food the treshold > 60 and for obje
## food
stimuli.food$outlier<-as.character(c(stimuli.food$corpes_norm_spain > 25))
stimuli.food.filtered <- dplyr::filter(stimuli.food, outlier=="FALSE")
mean(stimuli.food.filtered$corpes_norm_spain)
```

```
## [1] 6.456053
```

```
median(stimuli.food.filtered$corpes_norm_spain)
```

```
## [1] 4.855
```

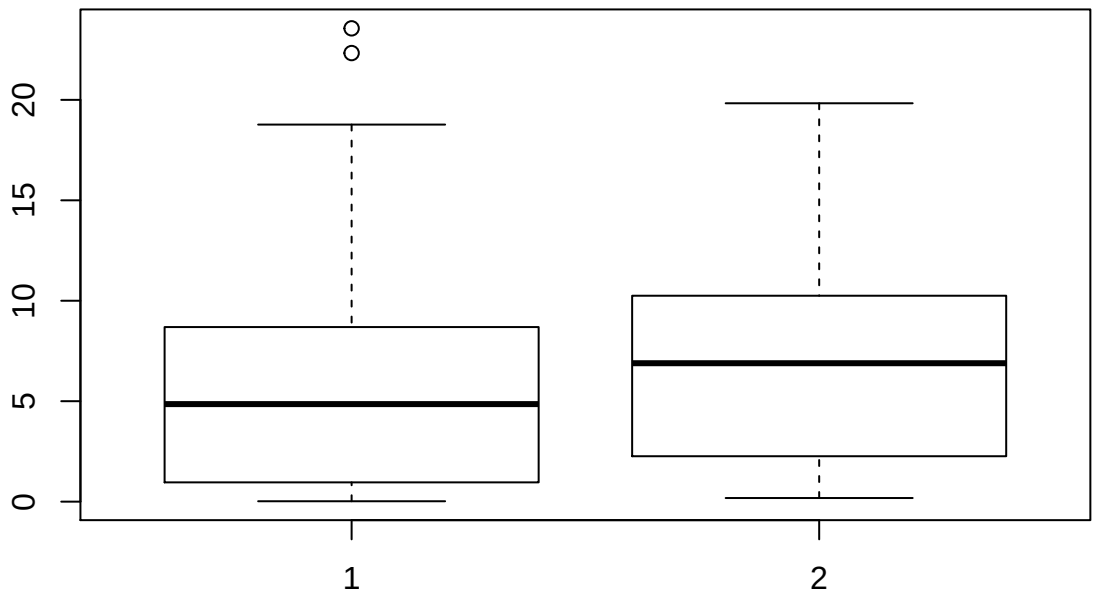
```
## objects
stimuli.objects$outlier <- as.character(c(stimuli.objects$corpes_norm_spain > 25))
stimuli.objects.filtered <- dplyr::filter(stimuli.objects, outlier == "FALSE")
mean(stimuli.objects.filtered$corpes_norm_spain)
```

```
## [1] 7.264324
```

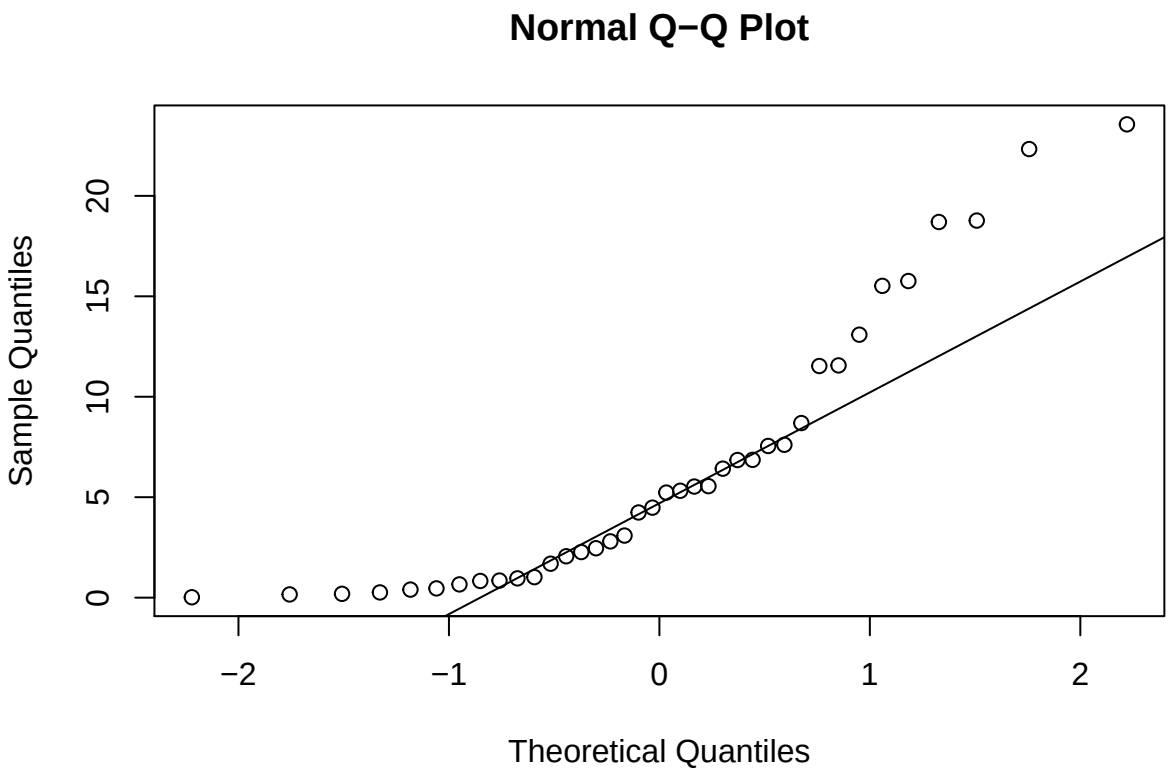
```
median(stimuli.objects.filtered$corpes_norm_spain)
```

```
## [1] 6.89
```

```
####### replotting boxplots
joint.boxplot.filtered <- boxplot(stimuli.food.filtered$corpes_norm_spain, stimuli.objects.filtered$cor
```



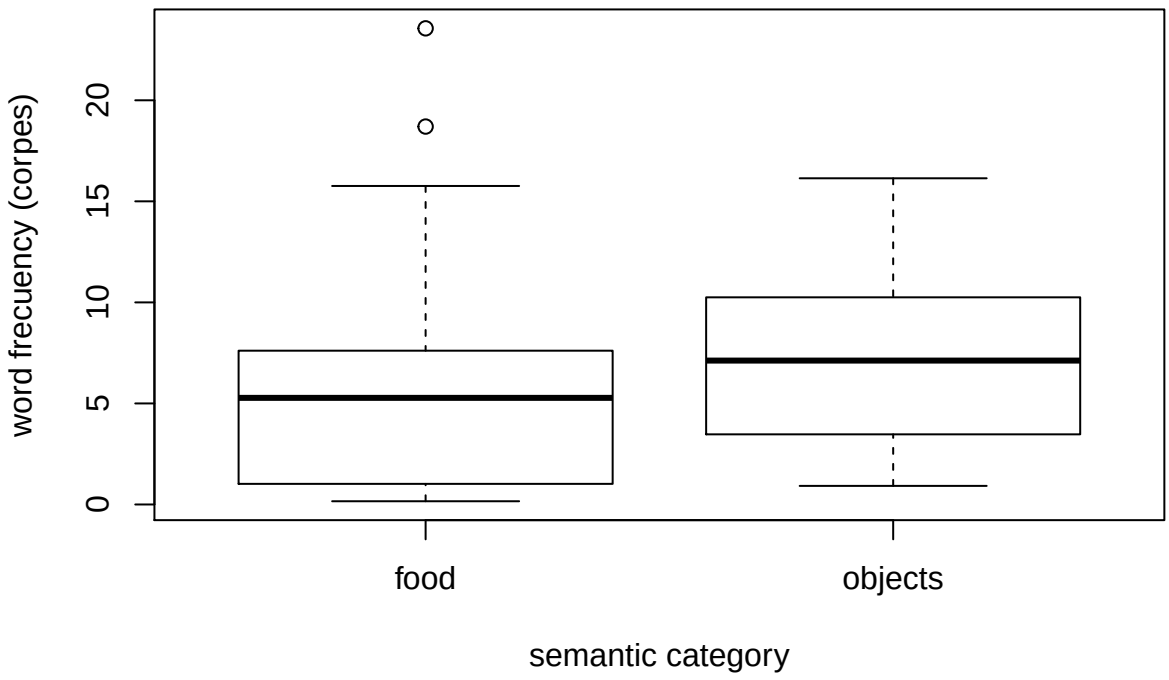

```
####### replotting qqplots
qqnorm(stimuli.food.filtered$corpes_norm_spain)
qqline(stimuli.food.filtered$corpes_norm_spain)
```




```
# making contingency tables to see how many word we are going to replace
#creating counting variable first
#-- food
stimuli.food.filtered$element <- c("word")
food.contingency <- as.data.frame(
  table(
    stimuli.food.filtered$element, stimuli.food.filtered$sound))
names(food.contingency) <- c("stimuli.Level","sound","Freq")
stimuli.food.filtered$category <- as.factor(stimuli.food.filtered$category)
stimuli.food.filtered$mean <- mean(stimuli.food.filtered$corpes_norm_spain)
stimuli.food.filtered$median <- median(stimuli.food.filtered$corpes_norm_spain)
#-- object
stimuli.objects.filtered$element <- c("word")
object.contingency <- as.data.frame(
  table(
    stimuli.objects.filtered$element,
    stimuli.objects.filtered$sound))
names(object.contingency) <- c("stimuli.Level","sound","Freq")
stimuli.objects.filtered$category <- as.factor(stimuli.objects.filtered$category)
stimuli.objects.filtered$mean <- mean(stimuli.objects.filtered$corpes_norm_spain)
stimuli.objects.filtered$median <- median(stimuli.objects.filtered$corpes_norm_spain)
### merging filtered stimuli
stimuli.experiment.filtered <- rbind(stimuli.food.filtered,stimuli.objects.filtered)
#stimuli.experiment.filtered$sem.cat <- as.factor(stimuli.experiment.filtered$category)
### saving data as .csv to manually eliminate stimuli
#write.csv(stimuli.experiment.filtered,"stimuli_speechExperiment.csv", sep=",", row.names = F)
```

```r
#making statistics with the data we have and looking to replace stimuli if needed
stimuli.experiment.filtered <-  read_excel("stimuli_speechExperiment.xlsx")
stimuli.experiment.filtered <- na.omit(stimuli.experiment.filtered)
#######
##
## We are reloading stimuli.objects.filtered and stimul.food.filtered
##
#######
stimuli.experiment.filtered$category <-as.character(stimuli.experiment.filtered$category)
# counting characters
stimuli.experiment.filtered$n.char <- nchar(as.character(stimuli.experiment.filtered$stimuli))
#
stimuli.objects.filtered <- filter(stimuli.experiment.filtered, category=="objects")
stimuli.food.filtered <- filter(stimuli.experiment.filtered, category=="food")

kruskal.food <- kruskal.test(stimuli.objects.filtered$corpes_norm_spain, stimuli.objects.filtered$condi
kruskal.nfood <- kruskal.test(stimuli.objects.filtered$n.char, stimuli.objects.filtered$condition)

kruskal.objects <- kruskal.test(stimuli.objects.filtered$corpes_norm_spain, stimuli.objects.filtered$co
kruskal.nobjects <- kruskal.test(stimuli.objects.filtered$n.char, stimuli.objects.filtered$condition)

stimuli.experiment.filtered$sem.cat <- as.factor(stimuli.experiment.filtered$category)

#wilcox.test(codistrpes_norm_spain~sem.cat, data= stimuli.experiment.filtered)
############
```

```r
boxplot(stimuli.food.filtered$corpes_norm_spain,stimuli.objects.filtered$corpes_norm_spain, xlab="seman
```




```
#deciding how to make non word stimuli. We will do:
#-Count the amount of characters in our words
#-determine the mean and the SD of the character length distribution
#-sample the amount of characters from the afforementioned distribution (bootstraping) -> boot library
#- convert amount of characters to actual n(X) strings

############################################################
set.seed(7877)# for replicating purposes of sampling
n_targetcontrolwords <- 10 # for run
word_sample <- as.data.frame( replicate(1000,sample(stimuli.experiment.filtered$n.char,n_targetcontrolw
sample_indexes <-sample(c(1:1000))
n_sessions <- 1 # amount of desired run sessions
sample_indexes <- sample_indexes[1:n_sessions]

## note: Sample_indexes is a list of n=>1, we need to transform integers to factors for R to effectivel
word_sample <- word_sample[,as.factor(sample_indexes)]
## note: As we may have several columns depending on the runs,
#        we are using tidr::gather to rbind them. Then we use #        the second column, because the o
#        column name, value
word_sample <- tidyr::gather(as.data.frame(word_sample))[2]
names(word_sample) <- c("num") #we are renaming our remaining
#### Creating stimuli
word_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(word_sample[,1])){word_sample
word_sample$wordControl[i] <- strrep("x",word_sample$num[i])}
```

```
#

## load sentences
SentencesSubject <- read_excel("SentencesSubject.xlsx")
# s1
SentencesSubject$nfirst <- nchar(as.character(SentencesSubject$S1))
# s2
## we already have our control "targets"
# s3
SentencesSubject$nthird <- nchar(as.character(SentencesSubject$S3))
# s4
SentencesSubject$nfourth <- nchar(as.character(SentencesSubject$S4))
# s5
SentencesSubject$nfifth <- nchar(as.character(SentencesSubject$S5))
# s6
SentencesSubject$nsixth <- nchar(as.character(SentencesSubject$S6))
## make replaements from sample
n_targetcontrolsentences <- 10 # for run
# s1
ns1_sample <- as.data.frame( replicate(1000,sample(SentencesSubject$nfirst,n_targetcontrolsentences, rep
# s2
## we done this in previous lines (word condition)
# s3
ns3_sample <- as.data.frame( replicate(1000,sample(SentencesSubject$nthird,n_targetcontrolsentences, rep
# s4
ns4_sample <- as.data.frame( replicate(1000,sample(SentencesSubject$nfourth,n_targetcontrolsentences, re
# s5
ns5_sample <- as.data.frame( replicate(1000,sample(SentencesSubject$nfifth,n_targetcontrolsentences, rep
# s6
ns6_sample <- as.data.frame( replicate(1000,sample(SentencesSubject$nsixth,n_targetcontrolsentences, rep
## we are using the same indexes as in the word control condition

############## create words
## note: Sample_indexes is a list of n=>1, we need to transform integers to factors for R to effectivel
###
# ns1
ns1_sample <- ns1_sample[,as.factor(sample_indexes)]
ns1_sample <- tidyr::gather(as.data.frame(ns1_sample))[2]
ns1_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(ns1_sample[,1])){ns1_sample$wordControl[i] <- strrep("x",ns1_sample$value[i])}

###
# ns3
ns3_sample <- ns3_sample[,as.factor(sample_indexes)]
ns3_sample <- tidyr::gather(as.data.frame(ns3_sample))[2]
ns3_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(ns3_sample[,1])){ns3_sample$wordControl[i] <- strrep("x",ns3_sample$value[i])}
###
# ns4
ns4_sample <- ns4_sample[,as.factor(sample_indexes)]
ns4_sample <- tidyr::gather(as.data.frame(ns4_sample))[2]
```

```
ns4_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(ns4_sample[,1])){ns4_sample$wordControl[i] <- strrep("x",ns4_sample$value[i])}
###
# ns5
ns5_sample <- ns5_sample[,as.factor(sample_indexes)]
ns5_sample <- tidyr::gather(as.data.frame(ns5_sample))[2]
ns5_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(ns5_sample[,1])){ns5_sample$wordControl[i] <- strrep("x",ns5_sample$value[i])}
###
# ns6
ns6_sample <- ns6_sample[,as.factor(sample_indexes)]
ns6_sample <- tidyr::gather(as.data.frame(ns6_sample))[2]
ns6_sample$wordControl <- c("")#empty string variable
#repeat x required n times
for (i in 1:length(ns6_sample[,1])){ns6_sample$wordControl[i] <- strrep("x",ns6_sample$value[i])}
```

```
## We are going to use this chunk of code to merge all the data we already have and put it into a gener
namesdf <- names(stimuli.experiment.filtered)
namesdf <- namesdf[1:10]
stimuli.experiment.filtered <- stimuli.experiment.filtered[,as.factor(namesdf)]
as
```

```
## function (object, Class, strict = TRUE, ext = possibleExtends(thisClass,
##     Class))
## {
##     thisClass <- .class1(object)
##     if (.identC(thisClass, Class) || .identC(Class, "ANY"))
##         return(object)
##     where <- .classEnv(thisClass, mustFind = FALSE)
##     coerceFun <- getGeneric("coerce", where = where)
##     coerceMethods <- .getMethodsTable(coerceFun, environment(coerceFun),
##         inherited = TRUE)
##     asMethod <- .quickCoerceSelect(thisClass, Class, coerceFun,
##         coerceMethods, where)
##     if (is.null(asMethod)) {
##         sig <- c(from = thisClass, to = Class)
##         asMethod <- selectMethod("coerce", sig, optional = TRUE,
##             useInherited = FALSE, fdef = coerceFun, mlist = getMethodsForDispatch(coerceFun))
##         if (is.null(asMethod)) {
##             canCache <- TRUE
##             inherited <- FALSE
##             if (is(object, Class)) {
##                 ClassDef <- getClassDef(Class, where)
##                 if (isFALSE(ext))
##                   stop(sprintf("internal problem in as(): %s is(object, \"%s\") is TRUE, but the meta
##                     dQuote(thisClass), Class), domain = NA)
##                 else if (isTRUE(ext))
##                   asMethod <- .makeAsMethod(quote(from), TRUE,
##                     Class, ClassDef, where)
##                 else {
##                   test <- ext@test
##                   asMethod <- .makeAsMethod(ext@coerce, ext@simple,
```

```
##                     Class, ClassDef, where)
##                   canCache <- (!is.function(test)) || isTRUE(body(test))
##                 }
##             }
##             if (is.null(asMethod) && extends(Class, thisClass)) {
##                 ClassDef <- getClassDef(Class, where)
##                 asMethod <- .asFromReplace(thisClass, Class,
##                   ClassDef, where)
##             }
##             if (is.null(asMethod)) {
##                 asMethod <- selectMethod("coerce", sig, optional = TRUE,
##                   c(from = TRUE, to = FALSE), fdef = coerceFun,
##                   mlist = coerceMethods)
##                 inherited <- TRUE
##             }
##             else if (canCache)
##                 asMethod <- .asCoerceMethod(asMethod, thisClass,
##                   ClassDef, FALSE, where)
##             if (is.null(asMethod))
##                 stop(gettextf("no method or default for coercing %s to %s",
##                   dQuote(thisClass), dQuote(Class)), domain = NA)
##             else if (canCache) {
##                 cacheMethod("coerce", sig, asMethod, fdef = coerceFun,
##                   inherited = inherited)
##             }
##         }
##     }
##     if (strict)
##         asMethod(object)
##     else asMethod(object, strict = FALSE)
## }
## <bytecode: 0x524a5b8>
## <environment: namespace:methods>
```

```
namessentences <- names(SentencesSubject)
namessentences <- namessentences[1:8]
SentencesSubject <- SentencesSubject[,as.factor(namessentences)]
# --words
control_new <- data.frame(word_sample$wordControl)
names(control_new) <- c("stimuli")
control_new$category <- c("control")
control_new$condition <- c("control")
control_new$sound <- c(NaN)
control_new$corpes_norm_spain <- c(NaN)
control_new$outlier <- c("FALSE")
control_new$element <- c("word_control")
control_new$mean <- c(NaN)
control_new$median <- c(NaN)
control_new$n.char <- word_sample$num
#sentences
sentences <- data.frame(word_sample$wordControl,c(6),ns1_sample$wordControl,word_sample$wordControl,ns3_
#
stimuli.experiment.filtered <- rbind(stimuli.experiment.filtered,control_new)
#
```

```r
names(sentences) <- names(SentencesSubject)
SentencesSubject <- rbind(SentencesSubject,sentences)
#### merging sentences with word stimuli by stimuli
test <- merge(stimuli.experiment.filtered,SentencesSubject,by="stimuli")
#write.csv(test,"completeStimuli_Exp1.csv",fileEncoding = "UTF-8")
```