LexLab Analysis Scaffolding

Matt Wagers

1/29/2021

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Session Options

Setting up libraries and options. This tutorial uses libraries from the `tidyverse` family so install that library.

```
knitr::opts_chunk$set(echo = TRUE)

library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v stringr 1.4.0
## v tidyr   1.1.0     v forcats 0.5.0
## v readr   1.3.1
## -- Conflicts ------------------------------------------------------------------------------------
## x tidyr::extract()   masks magrittr::extract()
## x stats::filter()    masks dplyr::filter()
## x stats::lag()       masks dplyr::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```
filter <- dplyr::filter
```

Import Data from Ibex

The script `LexDec.js` has an unusually simple structure: each line contains information about a single trial[1]. We can use the function `read_csv` to import these data.

In this function, we first create a vector containing column names. Many of these are not of use to us, or contain null or redundant info.

[1] This isn't the "norm" with Ibex, which lets each controller in your experiment write a line per trial. However, the template `LexDec.js` only used one controller: OnlineJudgment. There wasn't even a form to collect demographic information or debriefing info.

```
# define column names [indicated by comments in datafile]
OnlineJudgment.cdef <- c("time","IPMD5","controller", "item","element","type","group","wnum","word","RT

raw_results.tbl <- readr::read_csv("sample_results", comment = "#", col_names = OnlineJudgment.cdef)
```

```
## Parsed with column specification:
## cols(
##   time = col_double(),
##   IPMD5 = col_character(),
##   controller = col_character(),
##   item = col_double(),
##   element = col_double(),
##   type = col_character(),
##   group = col_character(),
##   wnum = col_double(),
##   word = col_character(),
##   RT = col_double(),
##   key = col_character(),
##   newline = col_logical(),
##   stim = col_character()
## )
```

```
head(raw_results.tbl)
```

```
## # A tibble: 6 x 13
##     time IPMD5 controller  item element type  group  wnum word     RT key
##    <dbl> <chr> <chr>      <dbl>   <dbl> <chr> <chr> <dbl> <chr> <dbl> <chr>
## 1 1.61e9 7872~ OnlineJud~    20       0 word~ NULL      1 CINA~   976 K
## 2 1.61e9 7872~ OnlineJud~    19       0 word~ NULL      1 GLEE~  1201 K
## 3 1.61e9 7872~ OnlineJud~     8       0 word~ NULL      1 TWEL~   605 S
## 4 1.61e9 7872~ OnlineJud~    14       0 word~ NULL      1 GILA~   538 K
## 5 1.61e9 7872~ OnlineJud~     7       0 word~ NULL      1 GROW~   528 S
## 6 1.61e9 7872~ OnlineJud~     4       0 word~ NULL      1 GHOU~   549 S
## # ... with 2 more variables: newline <lgl>, stim <chr>
```

Let's clean up our columns and drop those we don't need.

```
# Combining time and IPMD5 gives us a way of creating unique participant identifier
# the following function 'pastes' those two columns together, and then hashes them to create an anonymo

createUniqueParticipantIdentifier <- function(ibex.tbl){
  ibex.tbl$participant <- sapply(paste(ibex.tbl$time,
                                       ibex.tbl$IPMD5),
                                 digest::digest, algo="md5")
  return(ibex.tbl)
}

# Apply the CUPI function and then deselect the columns we don't need
deselect_cols.vec <- c("time", "IPMD5", "controller", "element", "group", "wnum", "newline", "stim")

compact_results.tbl <- raw_results.tbl %>%
  createUniqueParticipantIdentifier %>%
  select(-all_of(deselect_cols.vec)) %>%
  rename(LDT = RT)
```

```
# For good measure, we've also renamed the "RT" column to "LDT" to remind us what it is ... "lexical de
head(compact_results.tbl)
```

```
## # A tibble: 6 x 6
##    item type         word     LDT key   participant
##   <dbl> <chr>        <chr>  <dbl> <chr> <chr>
## 1    20 words-fake   CINALS   976 K     81b69145a3b2f05a0dbef0e5e50292ee
## 2    19 words-fake   GLEEDS  1201 K     81b69145a3b2f05a0dbef0e5e50292ee
## 3     8 words-hifreq TWELVE   605 S     81b69145a3b2f05a0dbef0e5e50292ee
## 4    14 words-fake   GILATE   538 K     81b69145a3b2f05a0dbef0e5e50292ee
## 5     7 words-hifreq GROWTH   528 S     81b69145a3b2f05a0dbef0e5e50292ee
## 6     4 words-lofreq GHOULS   549 S     81b69145a3b2f05a0dbef0e5e50292ee
```

Check distribution of trials

Before proceeding to any analysis of the dependent variables of interest, we should always check that our scripts were functioning as we wanted them to.

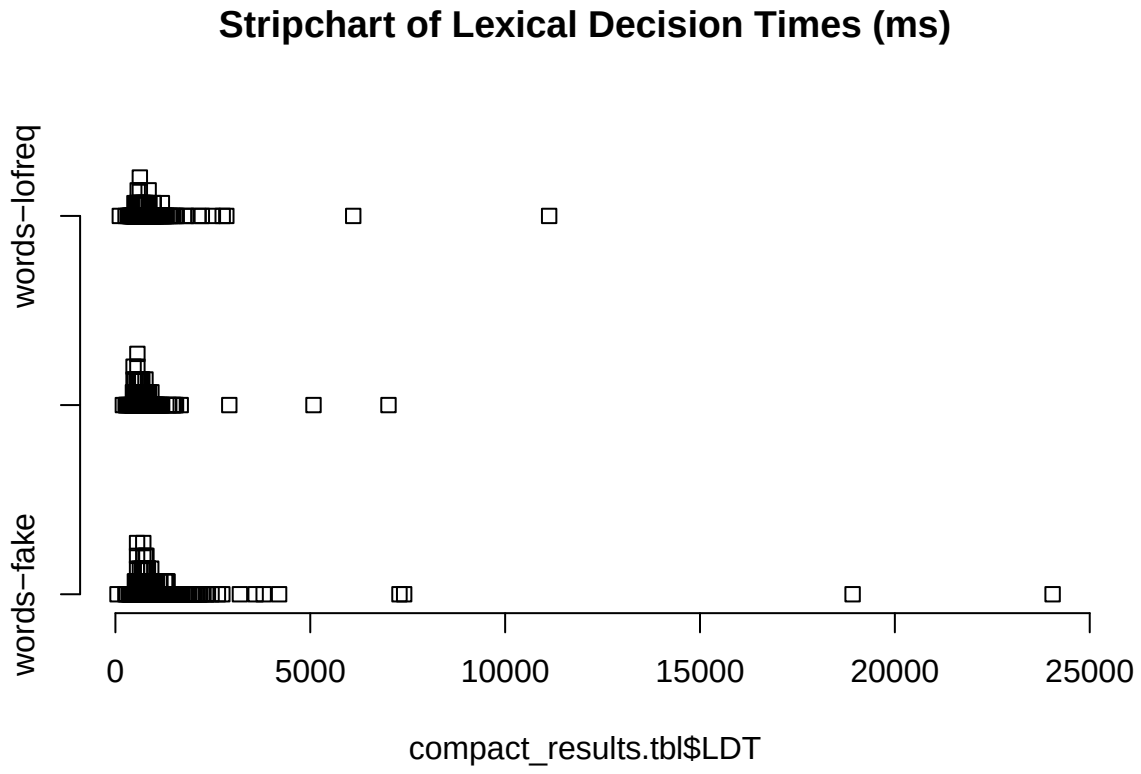
Minimally, we want to check that we collected the right distribution of trials per participant and per condition. We can also check the distribution of RTs and overall "correctness."

```
# use the table command to create a contingency table, and the `%$%` operator to "expose" the columns o
cond_by_participant.table <- compact_results.tbl %$% table(participant, type)

# add margins and print to check
cond_by_participant.table %>% addmargins()
```

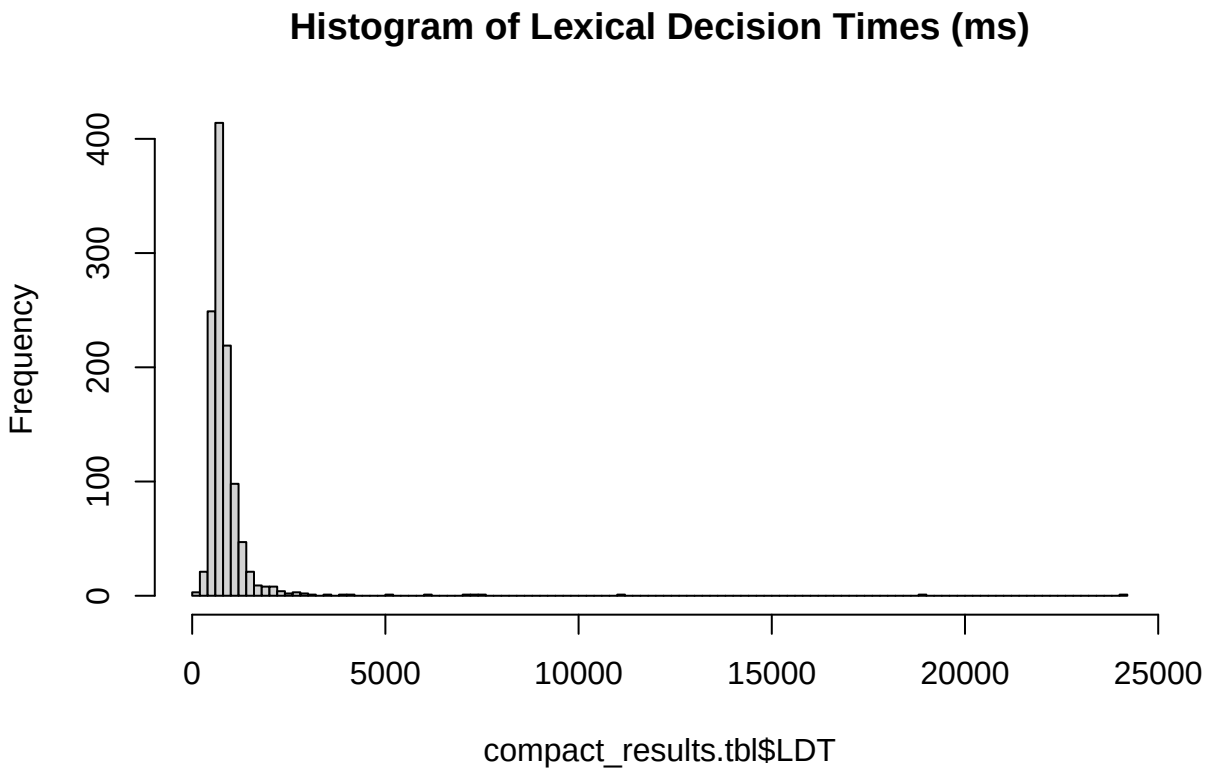
```
##                                 type
## participant                      words-fake words-hifreq words-lofreq  Sum
##   05aad3e7517cedfac820a536eaad19f4         10           5           5   20
##   075748ad8e962b20eaff4c6c931473f4         10           5           5   20
##   151459555623832a536c5e7c7da3c4aa         10           5           5   20
##   1768d687d489747280cb3bd741f6ffcb         10           5           5   20
##   191873e39bbcda2cf37d7ad9da510ab2         10           5           5   20
##   38194cc9bf1c89c2a5fefe945a6d580a         10           5           5   20
##   38342fd8f483a6d595cfdb8ee4b39a2e         10           5           5   20
##   39d58183cd9e5615659a888cbbdbece3         10           5           5   20
##   3ee637a61fc4cb7edbe652d91a054798         10           5           5   20
##   42bceca97a7ae6f8b49c228dda7d98f3         10           5           5   20
##   4fe5c113773932790be2adb659a1f049         10           5           5   20
##   5134f0c73ae296805b40d80faeb3bae9         10           5           5   20
##   5b117d5bf0f5e151bbe3d6d4828b4565         10           5           5   20
##   5bfccc939a8a00d9adbdd3076988aae2         10           5           5   20
##   5e32e9acbc42d9f0ec9244b330023614         10           5           5   20
##   613b934f5854913977294bec0099dd83         10           5           5   20
##   6291bc47c720ee7a35eafb9c42fc12b3         10           5           5   20
##   65e419502ee950acabf666b5805421f9         10           5           5   20
##   663fb2ea856253ce88cfa76f2de76fab         10           5           5   20
##   6dbdefdacd954b5ce8a5c448645e0bde         10           5           5   20
##   6f82a2c2c28815148090e597f7d228be         10           5           5   20
##   705f7d692cb8b631912d59eb91e77f38         10           5           5   20
##   73cbe555b9796a2f341ee417b8d4b5ed         10           5           5   20
##   76a21b19c2e6a9473830e1170e870c46         10           5           5   20
##   77420d4c1964028f25840ec3df50351a         10           5           5   20
```

```
##    7b37305caba93ad4cbc765d2905ad108         10           5           5   20
##    7d0ec1ae4919f150d391adb0b7aaed99         10           5           5   20
##    7f4130e2c2fe830c0e2678f06071164f         10           5           5   20
##    81b69145a3b2f05a0dbef0e5e50292ee         10           5           5   20
##    825a7f74bf672915174a5892721509f5         10           5           5   20
##    83603cea027f3bd0f5c16c0dfb1aa2fe         10           5           5   20
##    867165a8258ed283227355ed8ed71f1a         10           5           5   20
##    896477ed534bd0aac4ac171133160eac         10           5           5   20
##    8a3ff4729132afd39c0a2b305d813b9d         10           5           5   20
##    8d20be5db0b8fa1babe1d0fa81bafbad         10           5           5   20
##    8f12883d8496cb4650381b4eeefecbb8         10           5           5   20
##    9276f18dde096bfff8496ab0c83018e1         10           5           5   20
##    936f2e609ed1e8bc96a106691f6af3ec         10           5           5   20
##    a25c17412016f8549a6bd512ee7264e8         10           5           5   20
##    a73c6f75bfa6489524d9df4bb760d856         10           5           5   20
##    b0fb6a32968bd579677bc8be8c45455b         10           5           5   20
##    b2095da4e0f34b660ffd2d0240f177ce         10           5           5   20
##    bb265f895b38880cba7d3317170afe60         10           5           5   20
##    bc15f12d36ebf429a5b66cf64a24b322         10           5           5   20
##    d37f25406442423b53a59a7d368640d8         10           5           5   20
##    d44f6879f81ea517bfda796185446564         10           5           5   20
##    d5a8806736792c20fa31c2270982ea30         10           5           5   20
##    d71fc82f11304a0d91ac95a394a381df         10           5           5   20
##    e7476636290c0440cdbd425cc3e576a7         10           5           5   20
##    e86337e53a065624069465405418412c         10           5           5   20
##    eb198c187a621f23edb1b145a7c52f4e         10           5           5   20
##    ee944e7ae9a2b8ff9822ec271c07426b         10           5           5   20
##    f18c7348832bf439061e1a95ef1d1d85         10           5           5   20
##    f21e44f4323780b7a2ba4d0697ead9ae         10           5           5   20
##    f9a3161c8b45b265413a7f7e39f2744c         10           5           5   20
##    fd7a30b4467e19796a6eda34028045e6         10           5           5   20
##    Sum                                     560         280         280 1120
```

```r
# plot the distribution of LDTs
# many ways to plot

# stripchart method
stripchart(compact_results.tbl$LDT ~ compact_results.tbl$type, frame.plot=FALSE, method="stack", main="
```

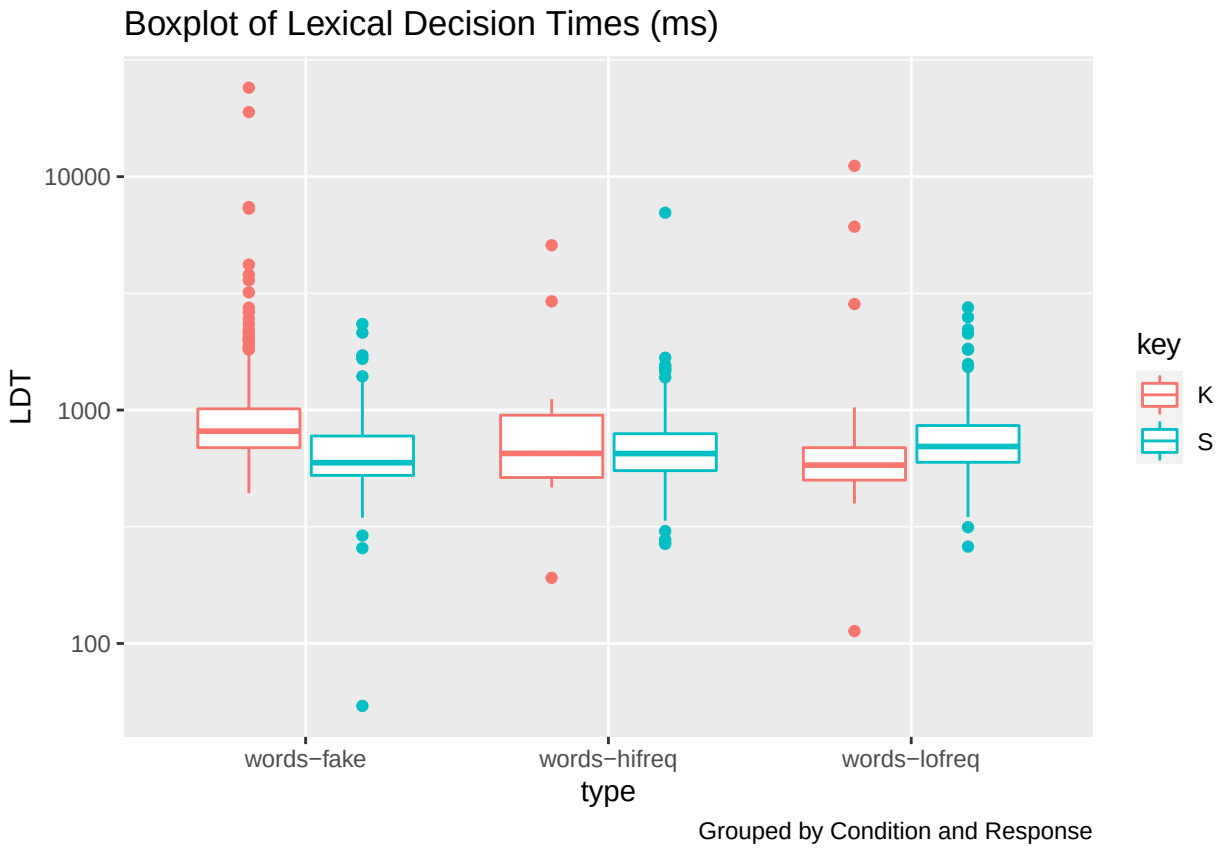



```
# histogram
hist(compact_results.tbl$LDT, breaks=100, main="Histogram of Lexical Decision Times (ms)")
```




```
# ggplot boxplot with log-10 axes
compact_results.tbl %>%
```

```
ggplot(aes(x=type, y=LDT)) + geom_boxplot(aes(col=key)) + scale_y_log10() +
labs(title="Boxplot of Lexical Decision Times (ms)", caption="Grouped by Condition and Response")
```



A few things become apparent:

- RTs have very long "right tail". We didn't timeout the response, so the trial wouldn't advance until a key was pressed.
- the `key` variable isn't very helpful

The right-tail problem for RT distributions is very problematic, because there is not a good way to identify outliers[2]. We end up having to do something pretty arbitrary, more or less. Let's "slice off" a slender tail at either end: 0.5% of the smallest observations, and 0.5% of the largest observations. This is reasonably conservative, but takes care of 24 sec LDTs ...

```
# first, determine the "cutoffs"
tails <- quantile(compact_results.tbl$LDT, c(0.005, 0.995))

# then, using filter to exclude, using the `between` boolean function
trimmed_LDT.tbl <- compact_results.tbl %>%
  filter(between(LDT, tails[1], tails[2]))

# double check we excluded the intended amount of data

excluded_pct <- 100 * (1-nrow(trimmed_LDT.tbl)/nrow(compact_results.tbl))
message(paste("Excluded", round(excluded_pct,1),"% of lexical decision times"))
```

[2]Many have grappled with this! Ratcliff (1993) is a classic starting point. More recently Baayen & Milin (2010) and Lo & Andrews (2015) have pursued model-based solutions.

```
## Excluded 1.1 % of lexical decision times
```

Now let's solve the other problem: better labels for `key`

```
# To solve this problem, we first create a translation key:
# the first column is the original names of `key`, the other column gives us new names
keycode.tbl <- tibble(key = c("K", "S"),
                      judgment = c("nonword", "word"))

# We then use a join command to match all rows from the left table with matching rows in the right tabl
trimmed_LDT.tbl <- left_join(trimmed_LDT.tbl, keycode.tbl)
```

```
## Joining, by = "key"
```

```
# Use `head` to check it worked
head(trimmed_LDT.tbl)
```

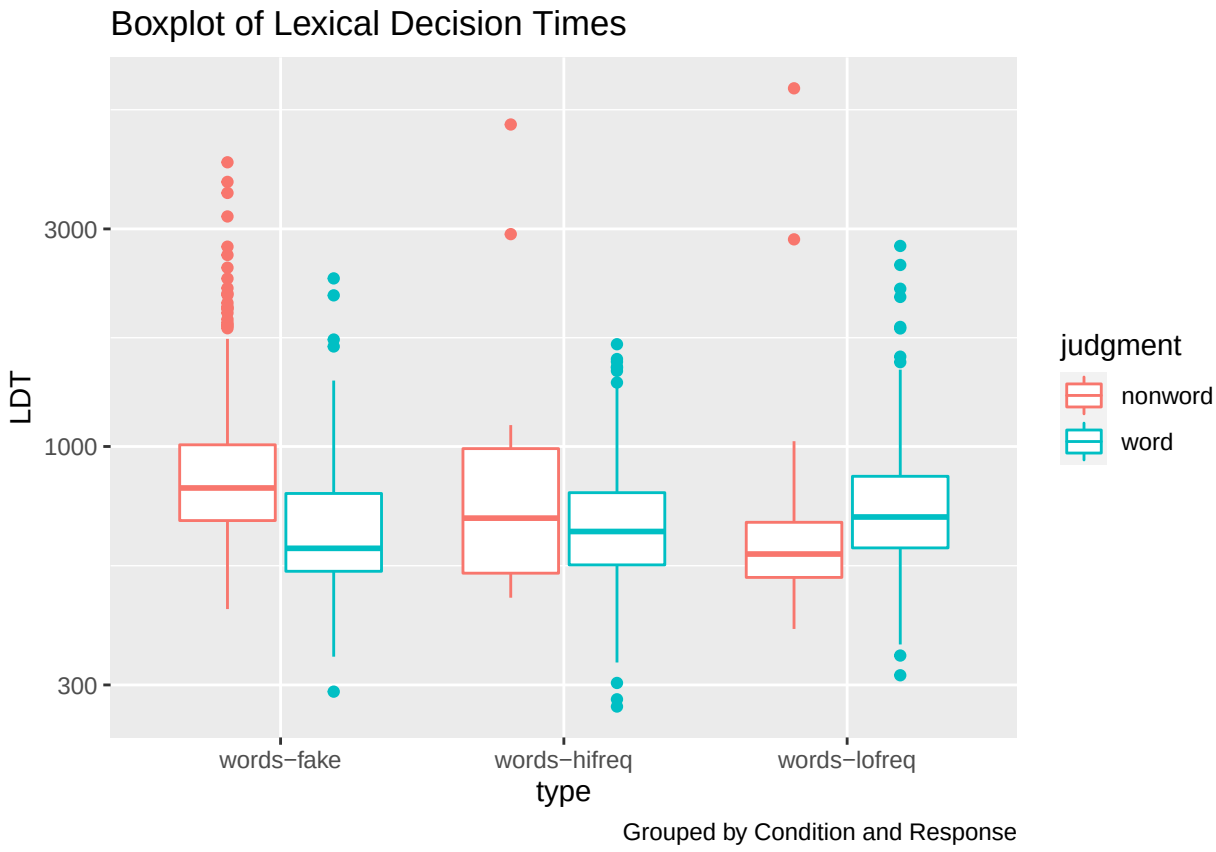
```
## # A tibble: 6 x 7
##    item type         word     LDT key   participant                      judgment
##   <dbl> <chr>        <chr>  <dbl> <chr> <chr>                            <chr>
## 1    20 words-fake   CINALS   976 K     81b69145a3b2f05a0dbef0e5e50292~ nonword
## 2    19 words-fake   GLEEDS  1201 K     81b69145a3b2f05a0dbef0e5e50292~ nonword
## 3     8 words-hifreq TWELVE   605 S     81b69145a3b2f05a0dbef0e5e50292~ word
## 4    14 words-fake   GILATE   538 K     81b69145a3b2f05a0dbef0e5e50292~ nonword
## 5     7 words-hifreq GROWTH   528 S     81b69145a3b2f05a0dbef0e5e50292~ word
## 6     4 words-lofreq GHOULS   549 S     81b69145a3b2f05a0dbef0e5e50292~ word
```

```
## You might want to use the same strategy to rename the "type" variable into two variables: "lexicalit

## We can use the -select command to get rid of the columns we know longer want ...
## ... this is destructive, so once it's run once the commands above won't work anymore
trimmed_LDT.tbl <- trimmed_LDT.tbl %>% select(-key)
head(trimmed_LDT.tbl)
```

```
## # A tibble: 6 x 6
##    item type         word     LDT participant                      judgment
##   <dbl> <chr>        <chr>  <dbl> <chr>                            <chr>
## 1    20 words-fake   CINALS   976 81b69145a3b2f05a0dbef0e5e50292ee nonword
## 2    19 words-fake   GLEEDS  1201 81b69145a3b2f05a0dbef0e5e50292ee nonword
## 3     8 words-hifreq TWELVE   605 81b69145a3b2f05a0dbef0e5e50292ee word
## 4    14 words-fake   GILATE   538 81b69145a3b2f05a0dbef0e5e50292ee nonword
## 5     7 words-hifreq GROWTH   528 81b69145a3b2f05a0dbef0e5e50292ee word
## 6     4 words-lofreq GHOULS   549 81b69145a3b2f05a0dbef0e5e50292ee word
```

With our neater dataset, let's replot our LDT range:

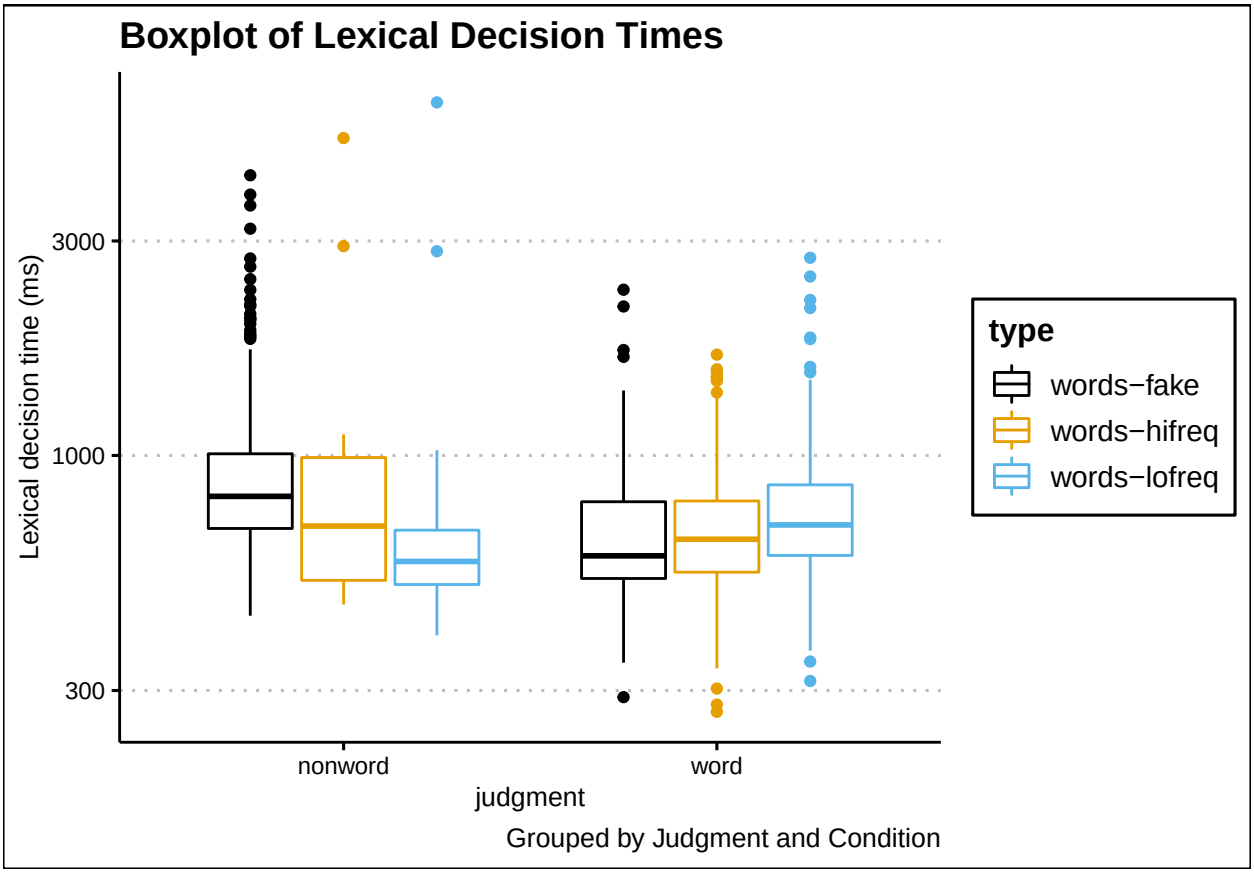
```
# ggplot boxplot with log-10 axes
trimmed_LDT.tbl %>%
  ggplot(aes(x=type, y=LDT)) + geom_boxplot(aes(col=judgment)) + scale_y_log10() +
  labs(title="Boxplot of Lexical Decision Times", caption="Grouped by Condition and Response")
```



The center line in a boxplot is the median[3]. Can you spot whether or not there's a frequency effect in the correct judgment of real words? Let's "regroup" to make it easier to spot ... observe how I swap `judgment` and `type` in the ggplot call below (with respect to the calls in the above chunks). Note I also store the result.

```
trimmed_LDT.ggp <- trimmed_LDT.tbl %>%
  ggplot(aes(x=judgment, y=LDT)) + geom_boxplot(aes(col=type)) + scale_y_log10() +
  labs(title="Boxplot of Lexical Decision Times", caption="Grouped by Judgment and Condition")

# let's "print" it with some friendlier colors & themes ... and make sure we don't forget our units!
trimmed_LDT.ggp + ggthemes::theme_clean() + ggthemes::scale_color_colorblind() + ylab("Lexical decision
```

[3]Good time to read up on boxplots!



References

Baayen, R. H., & Milin, P. (2010). Analyzing reaction times. *International Journal of Psychological Research*, *3*(2), 12–28.

Lo, S., & Andrews, S. (2015). To transform or not to transform: Using generalized linear mixed models to analyse reaction time data. *Frontiers in Psychology*, *6*, 1171. https://doi.org/10.3389/fpsyg.2015.01171

Ratcliff, R. (1993). Methods for dealing with reaction time outliers. *Psychological Bulletin*, *114*(3), 510–532.