This report is automatically generated with the R package **knitr** (version `1.23`) .

```
########################################
# L@S20 CODE
# Author: Ha Nguyen
########################################

########################################
# OVERVIEW
# The overall goal of the paper is to:
# * Examine the feasibility of crowdsourcing in scoring educational assessments; and
# * Evaluate whether individuals engaged in the crowdsourcing tasks learn from the experiences
# Individuals participating in the crowdsourcing task completed a pre-test on their science domain understanding,
# and then were randomly divided into 3 conditions:
# * A control group that just watched a video about relevant content
# * A group who GRADED the assessments
# * A group who GRADED the assessments and EXPLAINED the scores they gave
# They then filled out a post-test containing similar content with the pre-test
# Methodologies:
# * Evaluate the reliability of the crowdsourced scores (by calculating inter-rater reliability among the crowdworkers and
#   comparing scores to 'ground truth' scores by experts using correlations and RMSE, i.e., deviation from 'ground truth')
# * Fit linear regression models to compare post-test results, accounting for pre-test scores, science interests, and the treatment conditions
########################################

########################################
# LOAD DATA
# Load libraries
if (!require("pacman")) install.packages("pacman")
library(pacman)
pacman::p_load("psych", # data description + internal reliability Cronbach's alpha
               "ggplot2", "ggpubr", "RColorBrewer", # plotting
               "irr", # inter-rater reliability
               "mltools") # calculate rmse
# Load dataset
# setwd() enter path to data
# cc_score <- read.csv("cc_score.csv")
# head(cc_score, 3)

# Data format
# Each essay set consisted of 3 different student assignments. Each assignment was graded on 2 dimensions: Element, Causal Coherence
#ID condition essay element1 causal1 element2 causal2 element3 causal3
#1                  1     1        3       1        3       0        3
#2 21352073         1     1        2       1        1       1        2
#3 48123954         1     1        3       2        3       0        2

# We have 5 essay sets for workers to grade
essay1 <- subset(cc_score, essay==1)
essay2 <- subset(cc_score, essay==2)
essay3 <- subset(cc_score, essay==3)
essay4 <- subset(cc_score, essay==4)
essay5 <- subset(cc_score, essay==5)

# Notes: In this experiment, we had 2 conditions for crowdworkers (grade vs. grade + explain).
# We calculated the reliability statistics for the overall group and for each condition
essay1_g1 <- subset(essay1, condition==1)
essay2_g1 <- subset(essay2, condition==1)
essay3_g1 <- subset(essay3, condition==1)
essay4_g1 <- subset(essay4, condition==1)
essay5_g1 <- subset(essay5, condition==1)

# Group 2
essay1_g2 <- subset(essay1, condition==2)
essay2_g2 <- subset(essay2, condition==2)
essay3_g2 <- subset(essay3, condition==2)
essay4_g2 <- subset(essay4, condition==2)
essay5_g2 <- subset(essay5, condition==2)
########################################
# PART I: IS CROWDSOURCING RELIABLE?
# Calculate Krippendorff’s alpha (i.e., inter-rater agreement among workers on each essay)
# Create list of all data frames; run the Krippendorff's alpha function through the list
essaySet = list(essay1, essay2, essay3, essay5, essay5)
lapply(essaySet, function(x) kripp.alpha(as.matrix(x[4:9]), method="ordinal"))
```

```
## [[1]]
##  Krippendorff's alpha
##
##  Subjects = 6
##    Raters = 67
##     alpha = 0.241
##
## [[2]]
##  Krippendorff's alpha
##
##  Subjects = 6
##    Raters = 17
##     alpha = 0.594
##
## [[3]]
##  Krippendorff's alpha
##
##  Subjects = 6
##    Raters = 18
##     alpha = 0.627
##
## [[4]]
##  Krippendorff's alpha
##
##  Subjects = 6
##    Raters = 12
##     alpha = 0.618
##
## [[5]]
```

```
##  Krippendorff's alpha
##
##  Subjects = 6
##    Raters = 12
##     alpha = 0.618
```

```r
# Cronbach alpha (i.e., internal reliability)
lapply(essaySet, function(x) alpha(x[4:9]))
```

```
## [[1]]
## element1  causal1 element2  causal2 element3  causal3
##       NA       NA       NA       NA       NA       NA
##
## [[2]]
## element1  causal1 element2  causal2 element3  causal3
##       NA       NA       NA       NA       NA       NA
##
## [[3]]
## element1  causal1 element2  causal2 element3  causal3
##       NA       NA       NA       NA       NA       NA
##
## [[4]]
## element1  causal1 element2  causal2 element3  causal3
##       NA       NA       NA       NA       NA       NA
##
## [[5]]
## element1  causal1 element2  causal2 element3  causal3
##       NA       NA       NA       NA       NA       NA
```

```r
# The above are illustrative analyses for OVERALL group // For individual conditions, replace essay1 with essay1_g1 etc.

# Deviation from expert scores
# Input expert scores = scores for the essays assigned by a group of 5 researchers and domain experts (i.e., science teachers)
set1 <- c(3, 1, 1, 0, 3, 2)
set2 <- c(3, 1, 2, 1, 2, 1)
set3 <- c(2, 1, 3, 1, 2, 0)
set4 <- c(4, 0, 3, 1, 2, 0)
set5 <- c(3, 1, 3, 1,3,  1)

rDf <- data.frame(r = numeric())
# Set 1; output as list with each entry = 1 participant
apply(essay1_g1[4:9], 1, function(row) {
  rbind(rDf, rmse(row, set1))
})
```

```
## $`1`
##   X0.912870929175277
## 1          0.9128709
##
## $`2`
##   X0.707106781186548
## 1          0.7071068
##
## $`3`
##   X1.08012344973464
## 1          1.080123
##
## $`4`
##   X0.408248290463863
## 1          0.4082483
##
## $`5`
##   X1.47196014438797
## 1           1.47196
##
## $`6`
##   X0.912870929175277
## 1          0.9128709
##
## $`7`
##   X0.707106781186548
## 1          0.7071068
##
## $`8`
##   X1.08012344973464
## 1          1.080123
##
## $`9`
##   X1.35400640077266
## 1          1.354006
##
## $`10`
##   X1.35400640077266
## 1          1.354006
##
## $`11`
##   X1
## 1  1
##
## $`12`
##   X1.22474487139159
## 1          1.224745
##
## $`13`
##   X0.912870929175277
## 1          0.9128709
##
## $`14`
##   X0.707106781186548
## 1          0.7071068
##
## $`15`
```

```
##   X1.15470053837925
## 1          1.154701
##
## $`16`
##   X1.08012344973464
## 1          1.080123
##
## $`17`
##   X1.58113883008419
## 1          1.581139
##
## $`18`
##   X0.577350269189626
## 1          0.5773503
##
## $`19`
##   X0.707106781186548
## 1          0.7071068
##
## $`20`
##   X0.912870929175277
## 1          0.9128709
##
## $`21`
##   X0.912870929175277
## 1          0.9128709
##
## $`22`
##   X0.912870929175277
## 1          0.9128709
##
## $`23`
##   X1.35400640077266
## 1          1.354006
##
## $`24`
##   X0.912870929175277
## 1          0.9128709
##
## $`25`
##   X0.912870929175277
## 1          0.9128709
##
## $`26`
##   X1.35400640077266
## 1          1.354006
##
## $`27`
##   X0.816496580927726
## 1          0.8164966
##
## $`51`
##   X1.08012344973464
## 1          1.080123
##
## $`52`
##   X0.816496580927726
## 1          0.8164966
##
## $`53`
##   X1.52752523165195
## 1          1.527525
##
## $`54`
##   X0.577350269189626
## 1          0.5773503
##
## $`55`
##   X1.29099444873581
## 1          1.290994
##
## $`60`
##   X1.4142135623731
## 1         1.414214
##
## $`61`
##   X0.707106781186548
## 1          0.7071068
##
## $`62`
##   X0.577350269189626
## 1          0.5773503
##
## $`63`
##   X0.816496580927726
## 1          0.8164966
```

```
apply(essay1_g2[4:9], 1, function(row) {
  rbind(rDf, rmse(row, set1))
})
```

```
## $`28`
##   X1.29099444873581
## 1          1.290994
##
## $`29`
##   X1.35400640077266
## 1          1.354006
##
## $`30`
##   X1.52752523165195
## 1          1.527525
##
```

```
## $`31`
##   X1.29099444873581
## 1          1.290994
##
## $`32`
##   X1.82574185835055
## 1          1.825742
##
## $`33`
##   X1.29099444873581
## 1          1.290994
##
## $`34`
##   X1.73205080756888
## 1          1.732051
##
## $`35`
##   X1.47196014438797
## 1           1.47196
##
## $`36`
##   X1.68325082306035
## 1          1.683251
##
## $`37`
##   X1.22474487139159
## 1          1.224745
##
## $`38`
##   X1.73205080756888
## 1          1.732051
##
## $`39`
##   X1.29099444873581
## 1          1.290994
##
## $`40`
##   X1.22474487139159
## 1          1.224745
##
## $`41`
##   X1.52752523165195
## 1          1.527525
##
## $`42`
##   X1.29099444873581
## 1          1.290994
##
## $`43`
##   X1.58113883008419
## 1          1.581139
##
## $`44`
##   X1.68325082306035
## 1          1.683251
##
## $`45`
##   X1.47196014438797
## 1           1.47196
##
## $`46`
##   X1.68325082306035
## 1          1.683251
##
## $`47`
##   X1.47196014438797
## 1           1.47196
##
## $`48`
##   X1.29099444873581
## 1          1.290994
##
## $`49`
##   X1.68325082306035
## 1          1.683251
##
## $`50`
##   X1.68325082306035
## 1          1.683251
##
## $`56`
##   X1.82574185835055
## 1          1.825742
##
## $`57`
##   X1.35400640077266
## 1          1.354006
##
## $`58`
##   X1.77951304200522
## 1          1.779513
##
## $`59`
##   X1.58113883008419
## 1          1.581139
##
## $`64`
##   X1.15470053837925
## 1          1.154701
##
## $`65`
##   X1.77951304200522
## 1          1.779513
##
## $`66`
```

```
##   X1.08012344973464
## 1          1.080123
##
## $`67`
##   X1
## 1  1
```

```
# Set 2
apply(essay2_g1[4:9], 1, function(row) {
  rmse(row, set2)
})
```

```
##        73        74        75        76        77        78        79        80        81
## 0.8164966 0.5773503 0.4082483 0.5773503 0.5773503 0.8164966 0.8164966 0.5773503 0.7071068
```

```
apply(essay2_g2[4:9], 1, function(row) {
  rmse(row, set2)
})
```

```
##        68        69        70        71        72        82        83        84
## 1.4142136 0.8164966 1.2247449 1.0801234 0.7071068 0.5773503 1.1547005 0.8164966
```

```
# Set 3
apply(essay3_g1[4:9], 1, function(row) {
  rmse(row, set3)
})
```

```
##        85        86        87        88        89        95        96        97        98
## 0.5773503 0.4082483 0.5773503 0.5773503 0.5773503 0.5773503 0.9128709 1.0000000 0.7071068
```

```
apply(essay3_g2[4:9], 1, function(row) {
  rmse(row, set3)
})
```

```
##        90        91        92        93        94        99       100       101       102
## 1.2247449 0.5773503 0.5773503 0.9128709 1.3540064 0.4082483 1.2247449 0.5773503 0.4082483
```

```
# Set 4
apply(essay4_g1[4:9], 1, function(row) {
  rmse(row, set4)
})
```

```
##      108      109      110      111      112
## 1.080123 1.080123 1.000000 1.290994 1.080123
```

```
apply(essay4_g2[4:9], 1, function(row) {
  rmse(row, set4)
})
```

```
##      103      104      105      106      107
## 1.527525 1.154701 1.000000 1.527525 1.080123
```

```
# Set 5
apply(essay5_g1[4:9], 1, function(row) {
  rmse(row, set5)
})
```

```
##       113       114       115       116       117       118
## 0.5773503 0.7071068 0.9128709 0.8164966 0.7071068 0.8164966
```

```
apply(essay5_g2[4:9], 1, function(row) {
  rmse(row, set5)
})
```

```
##       119       120       121       122       123       124
## 0.8164966 0.0000000 0.7071068 0.7071068 0.5773503 0.7071068
```

```
# Paste output to a spreadsheet called "RMSE.csv"; formatted as follows:
# 1 = grade only; 2 = grade explain
# essay condition       rmse
# 1     grade only      0.9128709
# 1     grade only      0.7071068
# 1     grade only      1.080123

# Load RMSE data to plot
# rmse_plot <- read.csv("RMSE.csv")
head(rmse_plot, 3)
```

```
##   essay  condition      rmse
## 1     1 grade only 0.9128709
## 2     1 grade only 0.7071068
## 3     1 grade only 1.0801230
```

```
# Describe by conditions
psych::describeBy(rmse_plot$rmse, rmse_plot$condition)
```

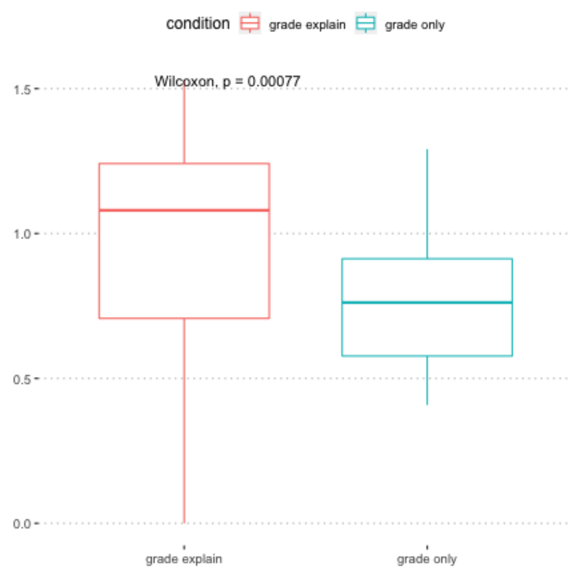
```
##
##  Descriptive statistics by group
```

```
## group: grade explain
##    vars  n mean   sd median trimmed  mad min  max range  skew kurtosis   se
## X1    1 40 0.99 0.36   1.08    1.01 0.39   0 1.53  1.53 -0.54    -0.35 0.06
## ------------------------------------------------------------------
## group: grade only
##    vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
## X1    1 50 0.78 0.21   0.76    0.77 0.25 0.41 1.29  0.88 0.23    -0.68 0.03
```

```
# Wilcoxon test to see if the two conditions differed by RMSE
wilcox.test(rmse_plot$rmse~rmse_plot$condition)
```

```
##
##      Wilcoxon rank sum test with continuity correction
##
## data:  rmse_plot$rmse by rmse_plot$condition
## W = 1411.5, p-value = 0.0007723
## alternative hypothesis: true location shift is not equal to 0
```

```
# Plot a boxplots to compare the RMSE of the 2 conditions.
ggpubr::ggboxplot(rmse_plot,
          y="rmse", x="condition", color="condition") +
  stat_compare_means() +
  theme_pubclean() +
  theme(axis.title.x=element_blank(),
        axis.title.y=element_blank())
```




```
# Plot histogram with vertical lines indicating the Mean of each condition
# First calculate the mean by condition
mu2 <- plyr::ddply(rmse_plot, "condition", summarise, grp.mean=mean(rmse))
```

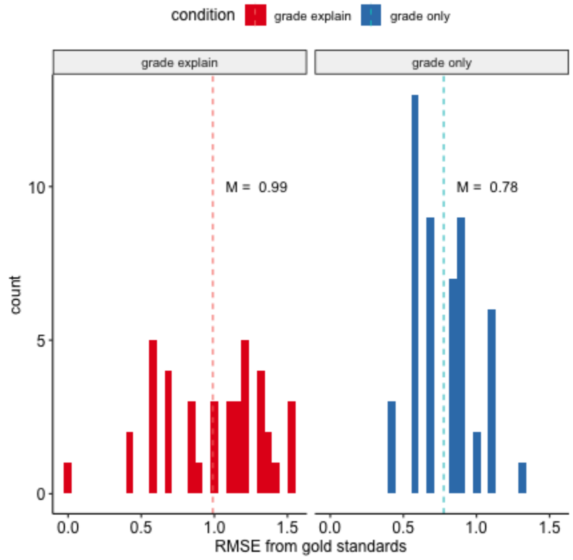
```
## Error in llply(.data = .data, .fun = .fun, ..., .progress = .progress, : object 'summarise' not found
```

```
head(mu2)
```

```
##       condition  grp.mean
## 1 grade explain 0.9897752
## 2    grade only 0.7759461
```

```
ggplot(rmse_plot, aes(x=rmse, fill=condition)) +
  scale_x_continuous(name = "RMSE from gold standards") +
  geom_histogram(position="identity") +
  facet_grid(cols=vars(condition)) +
  scale_fill_brewer(palette="Set1") +
  #geom_density(alpha = .5) +
  geom_vline(data=mu2, aes(xintercept=grp.mean, color=condition), # add vertical lines = mean; color by conditions
             linetype="dashed") +
  geom_text(data=mu2, aes(x=grp.mean+.3, y = 10, label=paste("M = ", round(grp.mean, 2)))) +
  theme_pubr()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#######################################

#######################################
# PART II: DID CROWDWORKERS LEARN FROM THE TASKS?
# Load in data = posttest scores for individuals after the crowdsourcing tasks
# cc_merge <- read.csv("cc_ks_merge3_nofraud.csv")
# Data format:
#ID condition numbLinks evidence causal sum1 elements numbLinks2 system sum2 sum3 science1 science2 science3 science4 science5 science6
#1 38075803         1         1         1      0    2         1           1      0    2    4         2         2         3         3         2         3
#2 89898934         1         3         1      0    4         1           1      0    2    6         3         3         3         3         3         2
#3 54184919         1         0         0      1    1         1           2      0    3    4         3         4         3         3         3         2
#sum_science c_interact c_balance c_total
#1          15          14          1      15
#2          17          16          2      18
#3          18          16          1      17

# Data consists of the participant ID, condition that they are in (no crowdsourcing, grade only, and grade and explain),
# their post-test, which consisted of 2 questions. Each qustion consisted of several elements (e.g., number of links they mentioned in their answers, comp
# their science interest (science1 - science 6)
# their pretest scores (c_total)

# Create a variable "condition2", that is, participants who participated in crowdsourcing (grade + grade explain) vs. did not
cc_merge$condition2 <- ifelse(cc_merge$condition==1 | cc_merge$condition==2, 1, 0)

# Change variables to factor variables
cc_merge$condition <- as.factor(cc_merge$condition)
cc_merge$condition2 <- as.factor(cc_merge$condition2)

# Get sum scores
cc_merge$sumPosttest = cc_merge$numbLinks+cc_merge$causal+cc_merge$numbLinks2+cc_merge$system

# Was there a difference in the sum scores?
pairwise.wilcox.test(cc_merge$sumPosttest, cc_merge$condition) #3 conditions
```

```
##
##      Pairwise comparisons using Wilcoxon rank sum test
##
## data:  cc_merge$sumPosttest and cc_merge$condition
##
##   0     1
## 1 0.037 -
## 2 0.319 0.307
##
## P value adjustment method: holm
```

```
wilcox.test(cc_merge$sumPosttest~cc_merge$condition2) #2 conditions
```

```
##
##      Wilcoxon rank sum test with continuity correction
##
## data:  cc_merge$sumPosttest by cc_merge$condition2
## W = 3252.5, p-value = 0.03482
## alternative hypothesis: true location shift is not equal to 0
```

```
# Linear regression models
# Predicting posttest scores, accounting for conditions + pretest + science interest sum scores. All variables are standardized.
summary(lm(scale(sumPosttest)~factor(condition) + scale(c_total) + scale(sum_science), data=cc_merge))
```

```
##
## Call:
## lm(formula = scale(sumPosttest) ~ factor(condition) + scale(c_total) +
##     scale(sum_science), data = cc_merge)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0645 -0.7262 -0.1243  0.6613  3.3198
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.13931    0.11640  -1.197  0.23321
## factor(condition)1   0.42269    0.18420   2.295  0.02309 *
```

```
## factor(condition)2  0.15400    0.18530   0.831  0.40720
## scale(c_total)      0.26113    0.07865   3.320  0.00112 **
## scale(sum_science)  0.15948    0.07876   2.025  0.04460 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9648 on 155 degrees of freedom
##   (21 observations deleted due to missingness)
## Multiple R-squared:  0.137,  Adjusted R-squared:  0.1147
## F-statistic: 6.149 on 4 and 155 DF,  p-value: 0.0001281
```

```r
summary(lm(scale(sumPosttest)~factor(condition2) + scale(c_total) + scale(sum_science), data=cc_merge))
```

```
##
## Call:
## lm(formula = scale(sumPosttest) ~ factor(condition2) + scale(c_total) +
##     scale(sum_science), data = cc_merge)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0592 -0.7319 -0.1415  0.6466  3.3204
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.13908    0.11668  -1.192  0.23509
## factor(condition2)1  0.28967    0.15484   1.871  0.06326 .
## scale(c_total)       0.26042    0.07884   3.303  0.00119 **
## scale(sum_science)   0.15309    0.07881   1.943  0.05386 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9671 on 156 degrees of freedom
##   (21 observations deleted due to missingness)
## Multiple R-squared:  0.1272, Adjusted R-squared:  0.1104
## F-statistic: 7.576 on 3 and 156 DF,  p-value: 9.183e-05
```

```r
# adjusted p-value due to multiple comparisons, using the Benjamini-Hochberg procedure
# The BH procedure: order all p-values from small to large, multiply each p-value by # of tests, and divide by the rank order
# For condition
p.adjust(c(.02, .25, .0003, .0495), method="BH") #.0400 0.2500 0.0012 0.0660
```

```
## [1] 0.0400 0.2500 0.0012 0.0660
```

```r
# For condition2
p.adjust(c(.039, .0004, .057), method="BH") #0.0570 0.0012 0.0570
```

```
## [1] 0.0570 0.0012 0.0570
```

The R session information (including the OS info, R version and all packages used):

```r
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## Random number generation:
##  RNG:     Mersenne-Twister
##  Normal:  Inversion
##  Sample:  Rounding
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] mltools_0.3.5      RColorBrewer_1.1-2 pacman_0.5.1       ggpubr_0.2.1
##  [5] magrittr_1.5       ggplot2_3.3.2      psych_1.8.12       irr_0.84.1
##  [9] lpSolve_5.6.13.3   WebPower_0.5.2     PearsonDS_1.1      lavaan_0.6-5
## [13] MASS_7.3-51.4      sjstats_0.17.5     simr_1.0.5         lme4_1.1-21
## [17] Matrix_1.2-17
##
## loaded via a namespace (and not attached):
##   [1] TH.data_1.0-10       minqa_1.2.4          colorspace_1.4-1
##   [4] ggsignif_0.5.0       ellipsis_0.3.0       rio_0.5.16
##   [7] sjlabelled_1.1.0     estimability_1.3     ergm_3.11.0
##  [10] tergm_3.6.1          rstudioapi_0.10      farver_2.0.1
##  [13] fansi_0.4.0          mvtnorm_1.0-11       codetools_0.2-16
##  [16] splines_3.6.1        mnormt_1.5-5         robustbase_0.93-5
##  [19] knitr_1.23           sjmisc_2.8.1         nloptr_1.2.1
##  [22] pROC_1.15.3          pbkrtest_0.4-8.6     broom_0.5.2
##  [25] binom_1.1-1          compiler_3.6.1       emmeans_1.4
##  [28] backports_1.1.5      assertthat_0.2.1     cli_1.1.0
##  [31] tools_3.6.1          coda_0.19-3          gtable_0.3.0
##  [34] glue_1.4.2           dplyr_1.0.4          Rcpp_1.0.6
##  [37] rle_0.9.2            carData_3.0-2        cellranger_1.1.0
##  [40] statnet.common_4.4.1 vctrs_0.3.6          nlme_3.1-140
##  [43] iterators_1.0.12     insight_0.10.0       xfun_0.8
##  [46] stringr_1.4.0        network_1.16.0       openxlsx_4.1.0.1
##  [49] trust_0.1-7          lifecycle_0.2.0      DEoptimR_1.0-8
##  [52] zoo_1.8-6            scales_1.1.0         hms_0.5.0
```