Modeling Data

Howard Nguyen

2022-09-28

COMPUTING FREQUENCIES

INSTALL AND LOAD PACKAGES

pacman must already be installed; then load contributed packages (including pacman) with pacman magrittr: for pipes pacman: for loading/unloading packages rio: for importing data tidyverse: for so many reasons

```
# Install pacman ("package manager") if needed
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(
  caret,          # Predictive analytics
  GGally,         # Scatterplot matrix
  magrittr,       # Pipes
  pacman,         # Load/unload packages
  parallel,       # Parallel processing
  randomForest,   # Random forests (obviously)
  rattle,         # Plot decision trees
  rio,            # Import/export data
  tictoc,         # Time operations
  tidyverse       # So many reasons
  )
```

LOAD AND PREPARE DATA

```
# Load data and transform pyschRegions to factor
df <- import("../Data/StateData.xlsx") %>%
  as_tibble() %>%
  select(state_code, region, psychRegions) %>%
  print()
```

```
## # A tibble: 48 x 3
##    state_code region    psychRegions
##    <chr>      <chr>     <chr>
##  1 AL         South     Friendly and Conventional
##  2 AZ         West      Relaxed and Creative
```

```
##  3 AR         South     Friendly and Conventional
##  4 CA         West      Relaxed and Creative
##  5 CO         West      Friendly and Conventional
##  6 CT         Northeast Temperamental and Uninhibited
##  7 DE         South     Temperamental and Uninhibited
##  8 FL         South     Friendly and Conventional
##  9 GA         South     Friendly and Conventional
## 10 ID         West      Relaxed and Creative
## # ... with 38 more rows
```

SUMMARIZE DATAFRAMES

```
# Summary of all variables.
df %>% summary()
```

```
##   state_code           region          psychRegions
##  Length:48          Length:48          Length:48
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

SUMMARIZE CATEGORICAL VARIABLES

```
# Summary() not very useful for character variables
df %>%
  select(region) %>%
  summary()
```

```
##     region
##  Length:48
##  Class :character
##  Mode  :character
```

```
# table() works better
df %>%
  select(region) %>%
  table()
```

```
## region
##   Midwest Northeast     South      West
##        12         9        16        11
```

SUMMARIZE FACTORS

```
# Convert "region" and "psychRegions" from character
# variables to factors.
df %<>%  # Compound assignment operation = "df <- df %>%"
  mutate(
```

```
    region = as.factor(region),
    psychRegions = as.factor(psychRegions)
  ) %>%
  print()
```

```
## # A tibble: 48 x 3
##    state_code region    psychRegions
##    <chr>      <fct>     <fct>
##  1 AL         South     Friendly and Conventional
##  2 AZ         West      Relaxed and Creative
##  3 AR         South     Friendly and Conventional
##  4 CA         West      Relaxed and Creative
##  5 CO         West      Friendly and Conventional
##  6 CT         Northeast Temperamental and Uninhibited
##  7 DE         South     Temperamental and Uninhibited
##  8 FL         South     Friendly and Conventional
##  9 GA         South     Friendly and Conventional
## 10 ID         West      Relaxed and Creative
## # ... with 38 more rows
```

```
# summary() works well with factors
df %>%
  select(region) %>%
  summary()
```

```
##        region
##  Midwest  :12
##  Northeast: 9
##  South    :16
##  West     :11
```

```
# Summary of all variables
df %>% summary()
```

```
##   state_code              region                          psychRegions
##  Length:48          Midwest  :12   Friendly and Conventional    :24
##  Class :character   Northeast: 9   Relaxed and Creative         :10
##  Mode  :character   South    :16   Temperamental and Uninhibited:14
##                     West     :11
```

COMPUTING DESCRIPTIVE STATISTICS

LOAD AND PREPARE DATA

```
# Use "StateData.xlxs" from exercise files
df <- import("../data/StateData.xlsx") %>%
  as_tibble() %>%
  select(state_code,
    region,
```

```
    psychRegions,
    instagram:entrepreneur) %>%
  mutate(
    region = as.factor(region),
    psychRegions = as.factor(psychRegions)
    ) %>%
  print()
```

```
## # A tibble: 48 x 7
##    state_code region    psychRegions             insta~1 faceb~2 retweet entre~3
##    <chr>      <fct>     <fct>                      <dbl>   <dbl>   <dbl>   <dbl>
##  1 AL         South     Friendly and Convention~   0.64    1.65    0.35    0.257
##  2 AZ         West      Relaxed and Creative       0.183  -0.259  -0.566   0.562
##  3 AR         South     Friendly and Convention~   0.456   1.10   -0.598   0.245
##  4 CA         West      Relaxed and Creative       1.47   -0.422   0.481   0.502
##  5 CO         West      Friendly and Convention~  -1.03   -1.06   -0.902   0.023
##  6 CT         Northeast Temperamental and Uninh~   0.374  -0.982   1.14    0.069
##  7 DE         South     Temperamental and Uninh~   1.48   -1.12    1.19    2.55
##  8 FL         South     Friendly and Convention~   0.85    0.38   -0.23    0.783
##  9 GA         South     Friendly and Convention~   0.807   0.526   0.035   1.95
## 10 ID         West      Relaxed and Creative      -0.736  -0.269  -1.80    0.296
## # ... with 38 more rows, and abbreviated variable names 1: instagram,
## #   2: facebook, 3: entrepreneur
```

SUMMARY

```
# Summary for entire table
df %>% summary()
```

```
##   state_code             region                          psychRegions
##  Length:48          Midwest  :12   Friendly and Conventional    :24
##  Class :character   Northeast: 9   Relaxed and Creative         :10
##  Mode  :character   South    :16   Temperamental and Uninhibited:14
##                     West     :11
##
##
##    instagram           facebook           retweet          entrepreneur
##  Min.   :-1.70000   Min.   :-1.57000   Min.   :-2.5300   Min.   :-1.79900
##  1st Qu.:-0.89475   1st Qu.:-0.42975   1st Qu.:-0.5225   1st Qu.:-0.59675
##  Median :-0.12450   Median :-0.05150   Median : 0.0340   Median : 0.05650
##  Mean   :-0.03363   Mean   : 0.09488   Mean   : 0.0210   Mean   : 0.03448
##  3rd Qu.: 0.65100   3rd Qu.: 0.81575   3rd Qu.: 0.4607   3rd Qu.: 0.51700
##  Max.   : 1.82000   Max.   : 2.24900   Max.   : 1.7930   Max.   : 2.54700
```

```
# Summary for one variable
df %>%
  select(entrepreneur) %>%  # select() returns a data frame
  summary()
```

```
##   entrepreneur
```

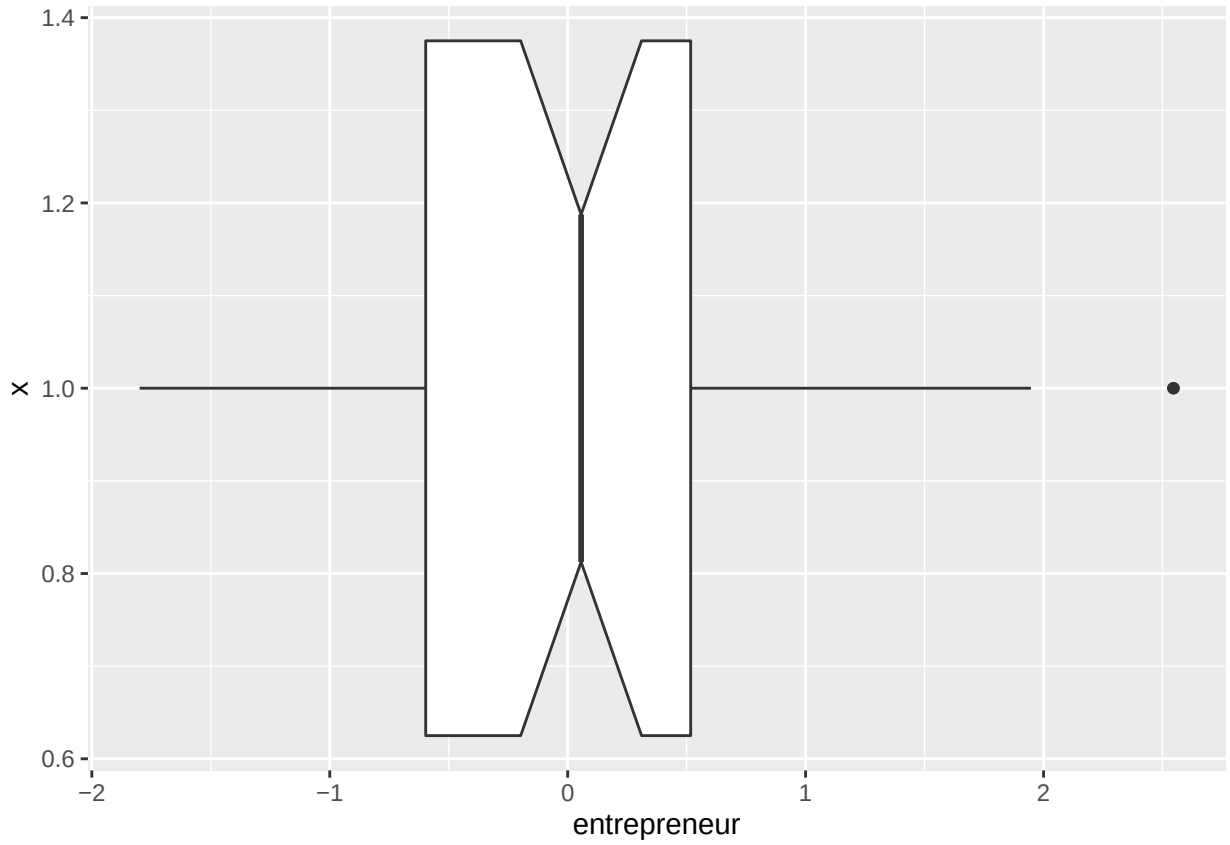
```
##  Min.   :-1.79900
##  1st Qu.:-0.59675
##  Median : 0.05650
##  Mean   : 0.03448
##  3rd Qu.: 0.51700
##  Max.   : 2.54700
```

```
df %>%
  pull(entrepreneur) %>%  # pull() returns a vector
  summary()
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.79900 -0.59675  0.05650  0.03448  0.51700  2.54700
```

BOXPLOT STATISTICS

```
# Boxplot of "entrepreneur" with ggplot2
df %>%
  select(entrepreneur) %>%
  ggplot(., aes(y = entrepreneur, x = 1)) +
  geom_boxplot(notch = TRUE) +  # Boxplot with CI
  coord_flip()  # Display horizontal
```



```r
# Boxplot stats: hinges, n, CI for median, outliers
df %>%
  pull(entrepreneur) %>%
  boxplot.stats()
```

```
## $stats
## [1] -1.7990 -0.6025  0.0565  0.5320  1.9470
##
## $n
## [1] 48
##
## $conf
## [1] -0.2022265  0.3152265
##
## $out
## [1] 2.547 2.545
```

ADDITIONAL DESCRIPTIVES

```r
# The "psych" package has additional descriptive statistics
# with the describe() function
p_load(psych)
```

```r
# Describe all variables
df %>% describe()  # Uses the internal codes for factors
```

```
##               vars  n  mean    sd median trimmed   mad   min   max range  skew
## state_code*      1 48 24.50 14.00  24.50   24.50 17.79  1.00 48.00 47.00  0.00
## region*          2 48  2.54  1.11   3.00    2.55  1.48  1.00  4.00  3.00 -0.15
## psychRegions*    3 48  1.79  0.87   1.50    1.75  0.74  1.00  3.00  2.00  0.40
## instagram        4 48 -0.03  0.93  -0.12   -0.05  1.14 -1.70  1.82  3.52  0.23
## facebook         5 48  0.09  0.93  -0.05    0.07  1.07 -1.57  2.25  3.82  0.23
## retweet          6 48  0.02  0.93   0.03    0.07  0.78 -2.53  1.79  4.32 -0.41
## entrepreneur     7 48  0.03  0.98   0.06    0.01  0.91 -1.80  2.55  4.35  0.35
##               kurtosis   se
## state_code*      -1.28 2.02
## region*          -1.37 0.16
## psychRegions*    -1.60 0.13
## instagram        -0.96 0.13
## facebook         -0.74 0.13
## retweet           0.04 0.13
## entrepreneur      0.35 0.14
```

```r
# Describe one variable
df %>%
  pull(entrepreneur) %>%  # Use pull(), not select()
  describe()
```

```
##    vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
## X1    1 48 0.03 0.98   0.06    0.01 0.91 -1.8 2.55  4.35 0.35     0.35 0.14
```

COMPUTING CORRELATIONS

LOAD AND PREPARE DATA

```
# Save Google Correlate variables
df <- import("../data/StateData.xlsx") %>%
  as_tibble() %>%
  select(instagram:modernDance) %>%
  print()
```

```
## # A tibble: 48 x 12
##    insta~1 faceb~2 retweet entre~3   gdpr privacy unive~4 mortg~5 volun~6 museum
##      <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
##  1   0.64    1.65    0.35    0.257 -0.769   0.583   1.74    1.41   -1.49  -1.10
##  2   0.183  -0.259  -0.566   0.562 -0.306  -0.452  -0.771   1.01    0.96  -0.128
##  3   0.456   1.10   -0.598   0.245 -0.595   0.689   0.024  -0.663  -1.19  -0.954
##  4   1.47   -0.422   0.481   0.502  1.12    0.231  -1.92   -0.881   0.566  0.05
##  5  -1.03   -1.06   -0.902   0.023  0.588  -0.215  -0.444   1.49    1.01   0.723
##  6   0.374  -0.982   1.14    0.069  0.712   0.362   0.37    0.476   1.27   1.18
##  7   1.48   -1.12    1.19    2.55   1.21    0.904   2.19    1.66    1.37   0.701
##  8   0.85    0.38   -0.23    0.783 -0.231  -0.137  -1.12    0.111   0.456 -0.738
##  9   0.807   0.526   0.035   1.95   0.403  -0.398  -0.24    0.188  -0.943 -0.747
## 10  -0.736  -0.269  -1.80    0.296 -0.174   0.075  -0.044   0.638  -0.297 -1.42
## # ... with 38 more rows, 2 more variables: scrapbook <dbl>, modernDance <dbl>,
## #   and abbreviated variable names 1: instagram, 2: facebook, 3: entrepreneur,
## #   4: university, 5: mortgage, 6: volunteering
```

CORRELATION MATRIX

```
# Correlation matrix for data frame
df %>% cor()
```

```
##                instagram    facebook     retweet entrepreneur       gdpr
## instagram     1.00000000 -0.04226378  0.50211252  0.554641518  0.2572551
## facebook     -0.04226378  1.00000000 -0.26215517 -0.370657238 -0.5409009
## retweet       0.50211252 -0.26215517  1.00000000  0.338838992  0.4832409
## entrepreneur  0.55464152 -0.37065724  0.33883899  1.000000000  0.2835559
## gdpr          0.25725507 -0.54090092  0.48324091  0.283555943  1.0000000
## privacy       0.15034972 -0.31415243  0.42268807  0.289310381  0.2129807
## university   -0.11776135  0.01196802  0.19119099  0.198639514 -0.2003535
## mortgage      0.14577857 -0.56075902  0.12709386  0.513706912  0.3094926
## volunteering -0.10454597 -0.65214896  0.37216772  0.089849925  0.6182895
## museum        0.07109595 -0.54528167  0.42468534 -0.005667993  0.5398867
## scrapbook    -0.44238721 -0.06112643 -0.43466043  0.110443217 -0.4225144
## modernDance   0.17774688 -0.49560997  0.09630384  0.251811818  0.2486523
##                  privacy  university    mortgage volunteering       museum
## instagram     0.1503497 -0.11776135  0.14577857  -0.10454597  0.071095949
## facebook     -0.3141524  0.01196802 -0.56075902  -0.65214896 -0.545281673
## retweet       0.4226881  0.19119099  0.12709386   0.37216772  0.424685343
## entrepreneur  0.2893104  0.19863951  0.51370691   0.08984993 -0.005667993
```

```
## gdpr          0.2129807 -0.20035352  0.30949265   0.61828952  0.539886657
## privacy       1.0000000  0.37806640  0.58171006   0.33586375  0.309229855
## university    0.3780664  1.00000000  0.23238214  -0.04674690 -0.117055294
## mortgage      0.5817101  0.23238214  1.00000000   0.38889032  0.290262780
## volunteering  0.3358638 -0.04674690  0.38889032   1.00000000  0.602899916
## museum        0.3092299 -0.11705529  0.29026278   0.60289992  1.000000000
## scrapbook    -0.1426237  0.28946299  0.05948299  -0.29317276 -0.485420240
## modernDance   0.1995072 -0.08870030  0.40850364   0.19537352  0.437701552
##                scrapbook modernDance
## instagram    -0.44238721  0.17774688
## facebook     -0.06112643 -0.49560997
## retweet      -0.43466043  0.09630384
## entrepreneur  0.11044322  0.25181182
## gdpr         -0.42251441  0.24865226
## privacy      -0.14262370  0.19950717
## university    0.28946299 -0.08870030
## mortgage      0.05948299  0.40850364
## volunteering -0.29317276  0.19537352
## museum       -0.48542024  0.43770155
## scrapbook     1.00000000  0.10442248
## modernDance   0.10442248  1.00000000
```

```
# Fewer decimal places
df %>%
  cor() %>%     # Compute correlations
  round(2) %>%  # Round to 2 decimals
  print()
```

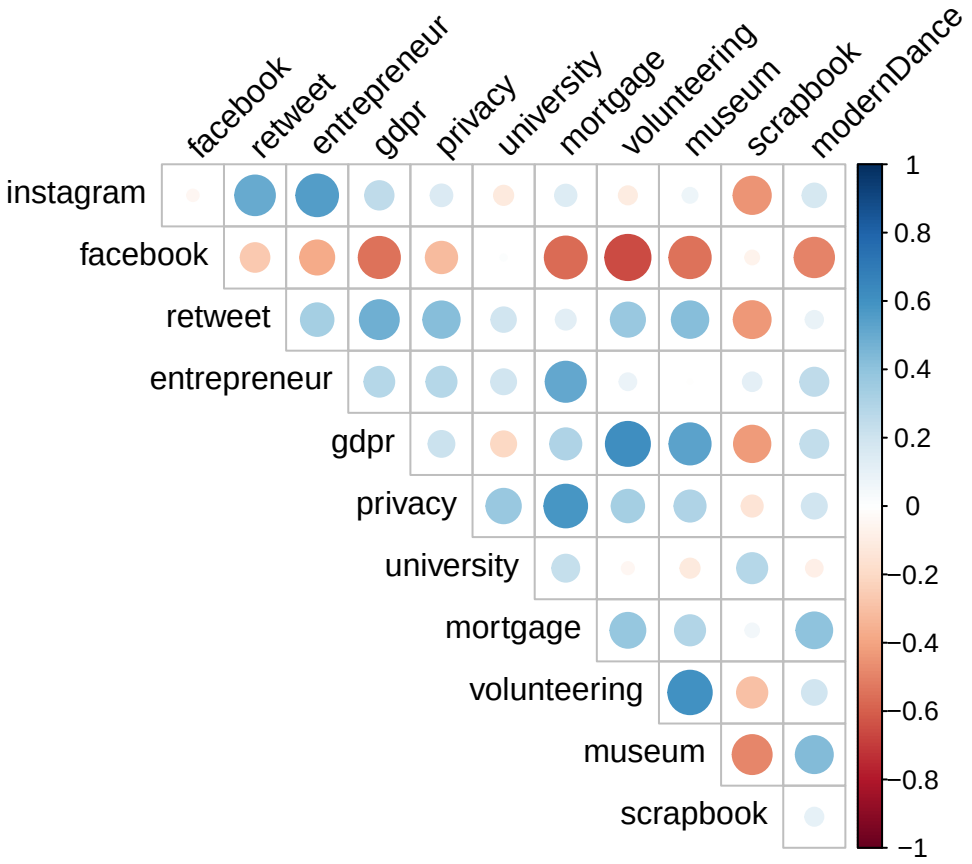
```
##              instagram facebook retweet entrepreneur  gdpr privacy university
## instagram         1.00    -0.04    0.50         0.55  0.26    0.15      -0.12
## facebook         -0.04     1.00   -0.26        -0.37 -0.54   -0.31       0.01
## retweet           0.50    -0.26    1.00         0.34  0.48    0.42       0.19
## entrepreneur      0.55    -0.37    0.34         1.00  0.28    0.29       0.20
## gdpr              0.26    -0.54    0.48         0.28  1.00    0.21      -0.20
## privacy           0.15    -0.31    0.42         0.29  0.21    1.00       0.38
## university       -0.12     0.01    0.19         0.20 -0.20    0.38       1.00
## mortgage          0.15    -0.56    0.13         0.51  0.31    0.58       0.23
## volunteering     -0.10    -0.65    0.37         0.09  0.62    0.34      -0.05
## museum            0.07    -0.55    0.42        -0.01  0.54    0.31      -0.12
## scrapbook        -0.44    -0.06   -0.43         0.11 -0.42   -0.14       0.29
## modernDance       0.18    -0.50    0.10         0.25  0.25    0.20      -0.09
##              mortgage volunteering museum scrapbook modernDance
## instagram        0.15        -0.10   0.07     -0.44        0.18
## facebook        -0.56        -0.65  -0.55     -0.06       -0.50
## retweet          0.13         0.37   0.42     -0.43        0.10
## entrepreneur     0.51         0.09  -0.01      0.11        0.25
## gdpr             0.31         0.62   0.54     -0.42        0.25
## privacy          0.58         0.34   0.31     -0.14        0.20
## university       0.23        -0.05  -0.12      0.29       -0.09
## mortgage         1.00         0.39   0.29      0.06        0.41
## volunteering     0.39         1.00   0.60     -0.29        0.20
## museum           0.29         0.60   1.00     -0.49        0.44
## scrapbook        0.06        -0.29  -0.49      1.00        0.10
## modernDance      0.41         0.20   0.44      0.10        1.00
```

What WE have here are correlation coefficients, these are numbers that range in absolute value from zero to one, they can be positive or negative. So the printed ranges from negative one to positive one, and they indicate the strength of a linear association or a straight line association between two variable. If it's zero or close to it means there's no straight line association. If it's positive one or negative one in indicates there's a perfect linear association, all the dots fall exactly on a line, and they can be used to predict each other. A positive value indicates that higher values on one variable are associated with higher values on the other variable. A negative value indicates that higher values on one variable go with lower values on the other variable.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Visualize correlation matrix with corrplot() from
# corrplot package
df %>%
  cor() %>%
  corrplot(
    type   = "upper",      # Matrix: full, upper, or lower
    diag   = F,            # Remove diagonal
    order  = "original",   # Order for labels
    tl.col = "black",      # Font color
    tl.srt = 45            # Label angle
  )
```



We see here are that large, dark blue circles indicate strong positive associations. So for instance, states

that are more likely to search for GDPR. That's the general data protection regulation, the European Union's online privacy act, are also more likely to search for volunteering, or for museum.

SINGLE CORRELATION

Use cor.test() to test one pair of variables at a time. cor.test() gives r, the hypothesis test, and the confidence interval. This command uses the "exposition pipe," %$%, from magrittr, which passes the columns from the data frame (and not the data frame itself)

```
df %$% cor.test(instagram, privacy)
```

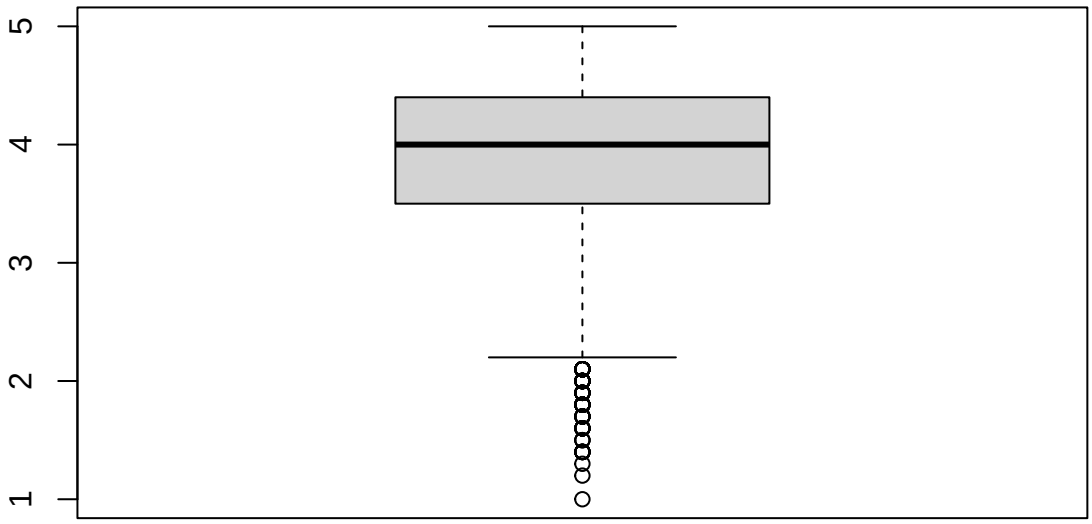
```
##
##  Pearson's product-moment correlation
##
## data:  instagram and privacy
## t = 1.0314, df = 46, p-value = 0.3077
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1397553  0.4166839
## sample estimates:
##       cor
## 0.1503497
```

The value is right here its 0.15. That's a pretty weak correlation. We can get a hypothesis test it gets a T value of 1.0314 with 46 degrees of freedom. The P value is 0.3077. Normally, it needs to be less than .05 to be considered significant. So this is nowhere close. We also get a confidence interval for the individual correlation coefficient, and it ranges from negative 0.139 to positive 0.4167. it's not statistically significant, but this does give us a much more detailed analysis of that one Association.

```
# Import Big 5 data
df <- import("../data/b5_df.rds") %>%
  print()
```

```
## # A tibble: 18,837 x 8
##      age gender engnat Extrav Neurot Agree Consc  Open
##    <int> <fct>  <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1    53 Male   Yes       4.4    1.1   4.6   4.7   4.3
##  2    46 Female Yes       2.2    3.1   3.5   4.2   2.6
##  3    14 Female No        3.5    4.6   3.8   4.9   4.5
##  4    19 Female No        2.2    4.3   3.7   2.6   4.1
##  5    25 Female No        3.4    3     4.4   3.4   3.4
##  6    31 Female Yes       1.6    2.4   3.6   3.1   3.3
##  7    20 Female Yes       4.6    2.1   4.5   2.8   4.1
##  8    23 Male   No        3.9    1.5   4.1   4.4   4.2
##  9    39 Female Yes       4.5    3.5   4.9   4.1   4.1
## 10    18 Female Yes       1.5    3.7   3.5   4     4.1
## # ... with 18,827 more rows
```

```
# Boxplot and median for "Open"
df %>%
  pull(Open) %T>%  # use pull() for vector; T-pipe
  boxplot() %>%
  median()
```




```
## [1] 4
```

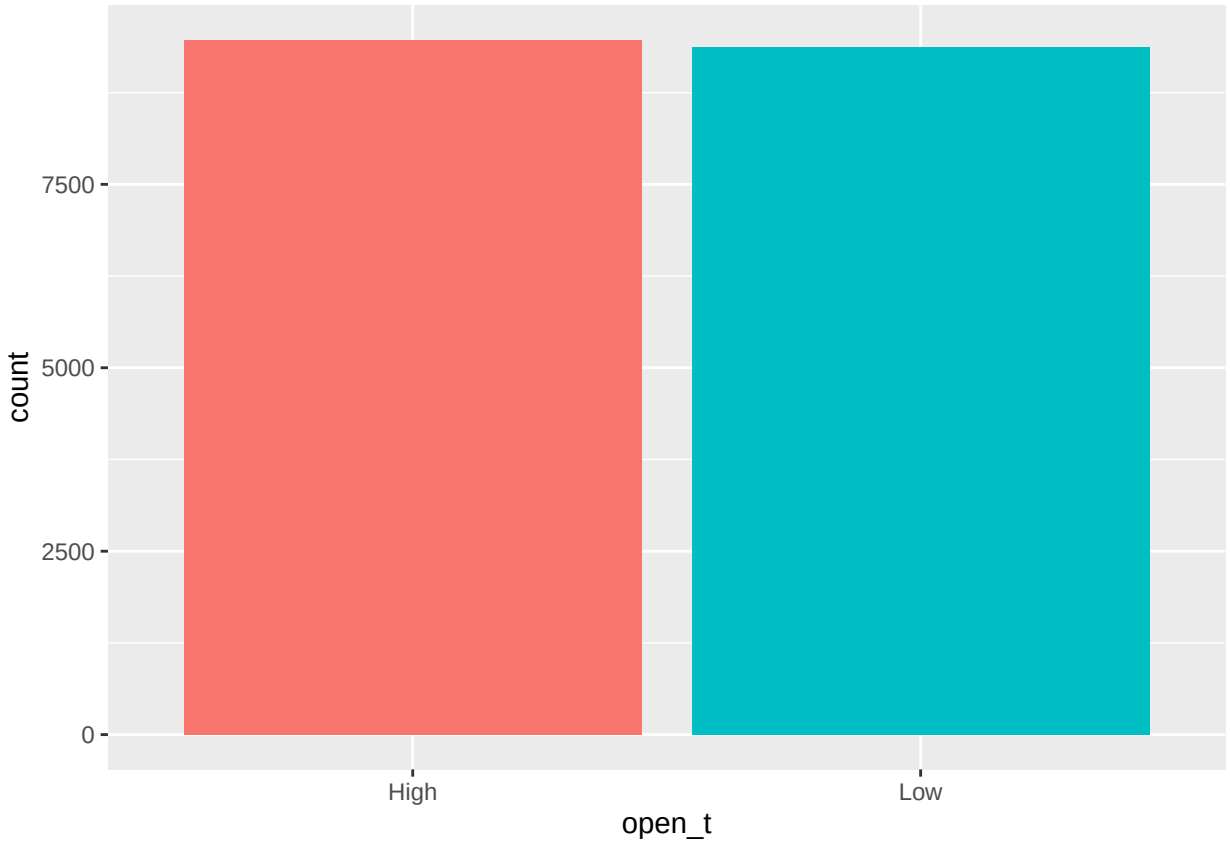
```
# Dichotomize "Open" for outcome
df %<>%                          # Overwrite data
  mutate(
    open_t = ifelse(             # open_t ("text")
      Open >= 4,                 # Test
      "High",                    # Value if true
      "Low"                      # Value if false
    ),
    open_t = as_factor(open_t)   # Convert to factor
  ) %>%
  print()
```

```
## # A tibble: 18,837 x 9
##      age gender engnat Extrav Neurot Agree Consc  Open open_t
##    <int> <fct>  <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl> <fct>
##  1    53 Male   Yes       4.4    1.1   4.6   4.7   4.3 High
##  2    46 Female Yes       2.2    3.1   3.5   4.2   2.6 Low
##  3    14 Female No        3.5    4.6   3.8   4.9   4.5 High
##  4    19 Female No        2.2    4.3   3.7   2.6   4.1 High
##  5    25 Female No        3.4    3     4.4   3.4   3.4 Low
##  6    31 Female Yes       1.6    2.4   3.6   3.1   3.3 Low
##  7    20 Female Yes       4.6    2.1   4.5   2.8   4.1 High
##  8    23 Male   No        3.9    1.5   4.1   4.4   4.2 High
```

```
##  9    39 Female Yes       4.5    3.5   4.9   4.1   4.1 High
## 10    18 Female Yes       1.5    3.7   3.5   4     4.1 High
## # ... with 18,827 more rows
```

EXPLORE DATA

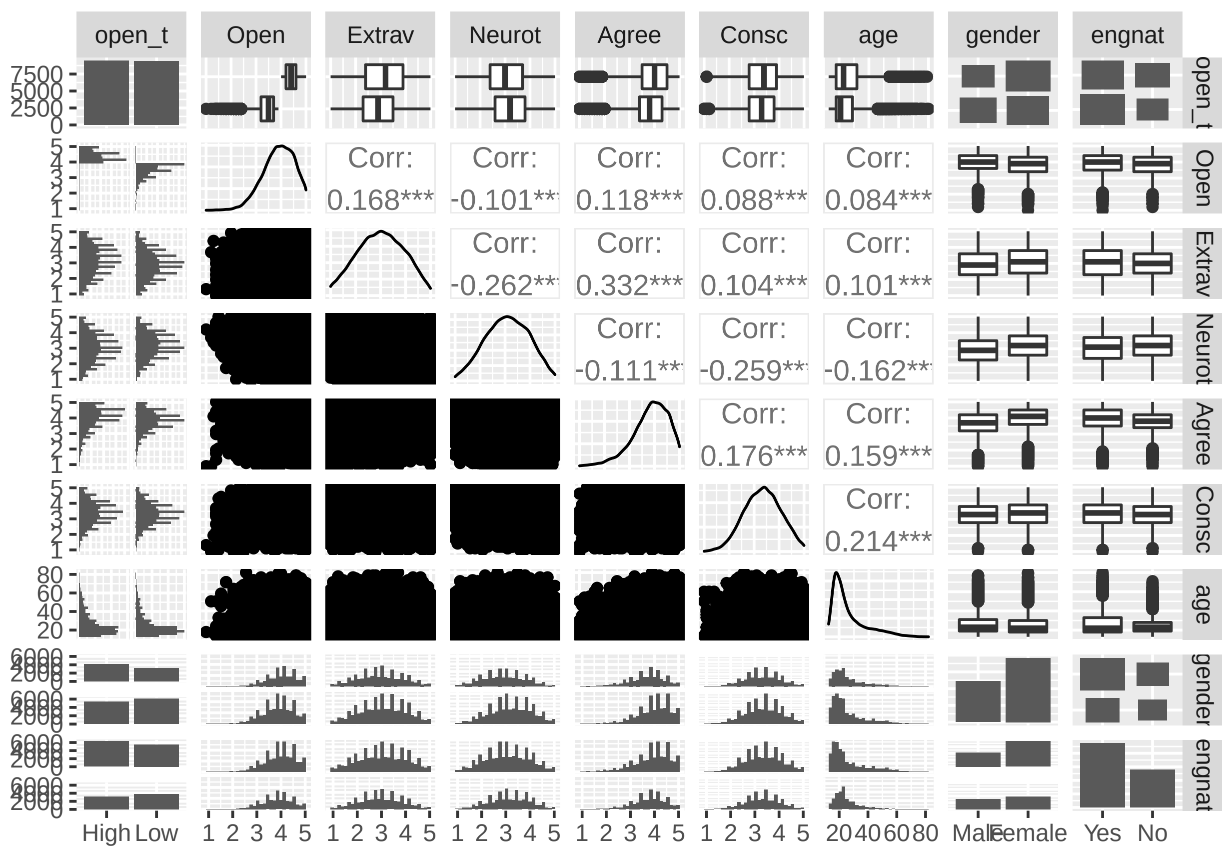
```
# Bar chart of "open_t"
df %>%
  ggplot() +
  geom_bar(
    aes(
      x    = open_t,  # Variable to chart
      fill = open_t   # Color bars by variable
    )
  ) +
  theme(legend.position = "none")
```




```
# Scatterplot matrix of all variables (takes a moment)
tic("Scatterplot matrix")
df %>%
  select(          # Reorder variables
    open_t,        # Put dichotomous "open" first
    Open,          # Then quantitative "open"
```

```
    Extrav:Consc,   # Then other Big 5
    age:engnat      # Then demographics
  ) %>%
  ggpairs()
```

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



```
toc()  # 15.404 sec to calculate, about 10 sec to appear
```

```
## Scatterplot matrix: 3.407 sec elapsed
```

```
# Summary statistics
df %>% summary()
```

```
##       age          gender      engnat          Extrav          Neurot
##  Min.   :13.0   Male  : 7326   Yes:11865   Min.   :1.000   Min.   :1.000
##  1st Qu.:18.0   Female:11511   No : 6972   1st Qu.:2.300   1st Qu.:2.500
##  Median :22.0                              Median :3.000   Median :3.100
##  Mean   :26.2                              Mean   :3.015   Mean   :3.094
##  3rd Qu.:31.0                              3rd Qu.:3.700   3rd Qu.:3.700
##  Max.   :80.0                              Max.   :5.000   Max.   :5.000
##      Agree           Consc            Open        open_t
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   High:9470
##  1st Qu.:3.400   1st Qu.:2.800   1st Qu.:3.500   Low :9367
##  Median :3.900   Median :3.400   Median :4.000
##  Mean   :3.847   Mean   :3.346   Mean   :3.909
##  3rd Qu.:4.400   3rd Qu.:3.900   3rd Qu.:4.400
##  Max.   :5.000   Max.   :5.000   Max.   :5.000
```

T-TEST ON FULL DATA

```
# Just an example
df %>% t.test(Extrav ~ open_t, data = .)
```

```
##
##  Welch Two Sample t-test
##
## data:  Extrav by open_t
## t = 18.371, df = 18724, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group High and group Low is not equal to 0
## 95 percent confidence interval:
##  0.2182320 0.2703623
## sample estimates:
## mean in group High  mean in group Low
##           3.136589           2.892292
```

```
# Not a good idea to do several t-tests
```

LINEAR REGRESSION ON FULL DATA

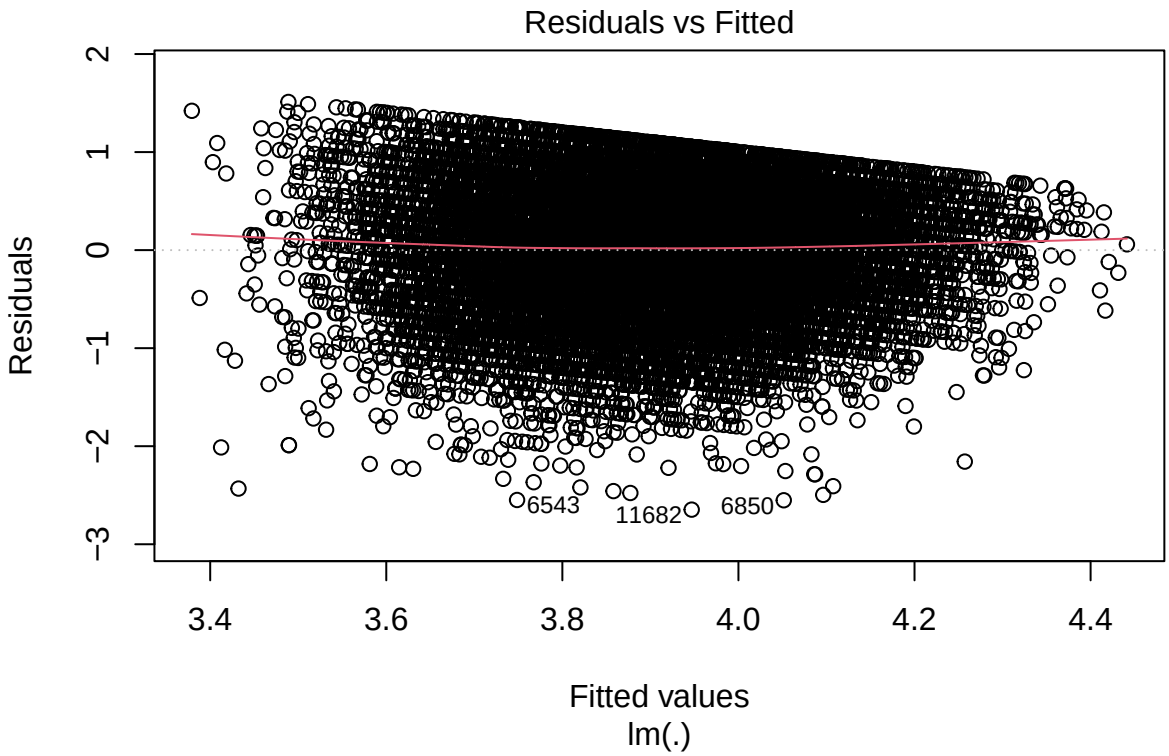
```
# Simultaneous entry linear regression
fit_lm <- df %>%   # Use full data, save as "fit_lm"
  select(          # Put outcome first
    Open,          # Then quantitative "open"
```

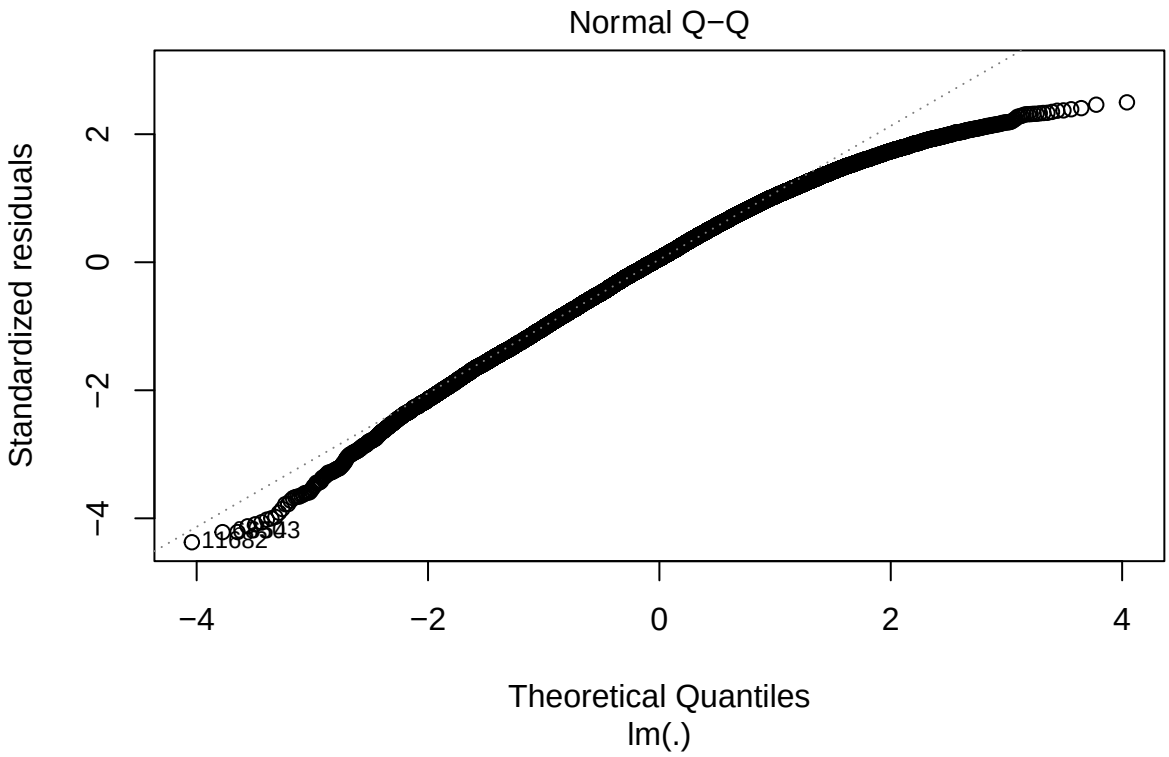
```
    Extrav:Consc,  # Then other Big 5
    age:engnat     # Then demographics
  ) %>%
  lm()             # Default linear regression
```

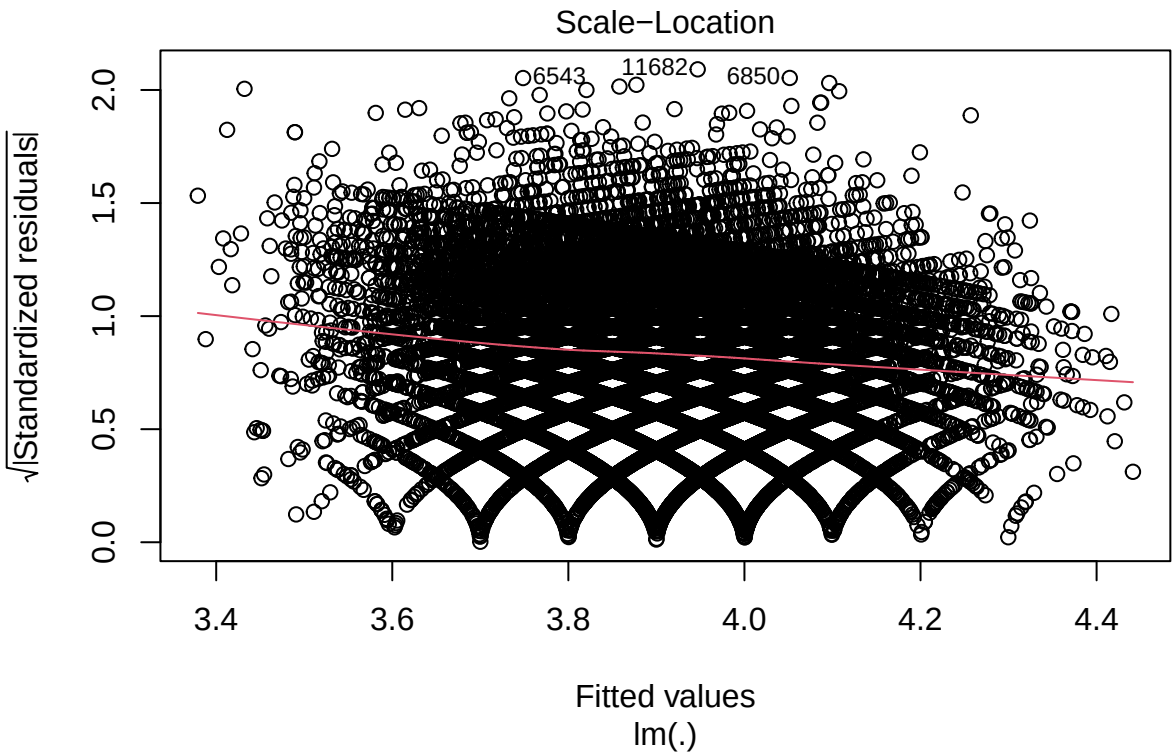
```
# Show model summary
fit_lm %>% summary()
```

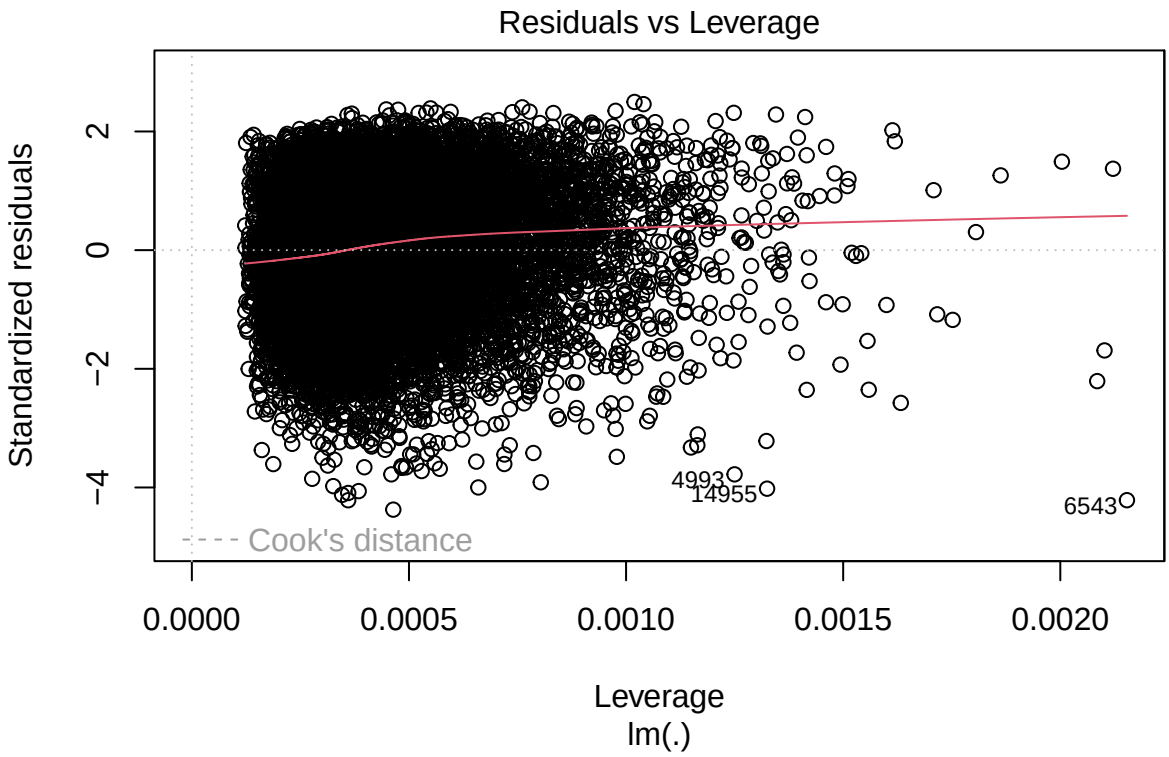
```
##
## Call:
## lm(formula = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.64690 -0.40035  0.03498  0.45182  1.51116
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.3476679  0.0400808  83.523  < 2e-16 ***
## Extrav        0.0898844  0.0052319  17.180  < 2e-16 ***
## Neurot       -0.0092399  0.0056025  -1.649   0.0991 .
## Agree         0.0689122  0.0068283  10.092  < 2e-16 ***
## Consc         0.0433337  0.0064187   6.751 1.51e-11 ***
## age           0.0019151  0.0004006   4.780 1.76e-06 ***
## genderFemale -0.1666066  0.0094945 -17.548  < 2e-16 ***
## engnatNo     -0.1055003  0.0092208 -11.442  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6054 on 18829 degrees of freedom
## Multiple R-squared:  0.06057,    Adjusted R-squared:  0.06022
## F-statistic: 173.4 on 7 and 18829 DF,  p-value: < 2.2e-16
```

```
# Diagnostic plots; run command then hit return in Console
# 1: Residuals vs. Fitted
# 2: Normal Q-Q
# 3: Scale-Location
# 4: Residuals vs. Leverage
fit_lm %>% plot()
```


Residuals vs Fitted
Residuals
2
1
0
−1
−2
−3
6543
11682
6850
3.4
3.6
3.8
4.0
4.2
4.4
Fitted values
lm(.)


Normal Q−Q
Standardized residuals
2
0
−2
−4
11082
−4
−2
0
2
4
Theoretical Quantiles
lm(.)


Scale−Location
6543
11682
6850
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
3.4
3.6
3.8
4.0
4.2
4.4
Fitted values
lm(.)



SPLIT DATA FOR MACHINE LEARNING

```
# Set random seed
set.seed(333)
```

```
# Take random subsample to save time (total n = 18,837)
# df %<>% sample_n(1000)

# Split data into train and test sets
train <- df %>% sample_frac(.70)  # Sample 70% of data
test  <- anti_join(df, train)     # Use remaining cases
```

```
## Joining, by = c("age", "gender", "engnat", "Extrav", "Neurot", "Agree",
## "Consc", "Open", "open_t")
```

KNN ON TRAINING DATA

```
# Define parameters
statctrl <- trainControl(
  method = "repeatedcv",  # Repeated cross-validation
  number = 10,            # Number of folds
```

```
  repeats = 3              # Number of complete sets of folds
)
```

```
# Define and save model
fit_knn <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,           # Use training data
  method = "knn",
  trControl = statctrl,
  tuneLength = 20,        # 20 dif values for k
  na.action = "na.omit"
)
```

```
# Apply model to training data (takes a moment)
tic("k-NN")
fit_knn
```

```
## k-Nearest Neighbors
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 11867, 11867, 11868, 11867, 11868, 11867, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.5464878  0.09303443
##    7  0.5516698  0.10341346
##    9  0.5592035  0.11849353
##   11  0.5635015  0.12710126
##   13  0.5686085  0.13732431
##   15  0.5703782  0.14087382
##   17  0.5717690  0.14366493
##   19  0.5737396  0.14760974
##   21  0.5739415  0.14801979
##   23  0.5729309  0.14600902
##   25  0.5753328  0.15081889
##   27  0.5757371  0.15163171
##   29  0.5771280  0.15442140
##   31  0.5792010  0.15857497
##   33  0.5770520  0.15428226
##   35  0.5768744  0.15393276
##   37  0.5765203  0.15322771
##   39  0.5771526  0.15449302
##   41  0.5764955  0.15318687
##   43  0.5754086  0.15101927
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 31.
```

```
toc() # 0.04 sec for n = 700; 0.01 for full data (13k)??
```

```
## k-NN: 0.003 sec elapsed
```

```
# Predict training set
open_p <- fit_knn %>%  # "predicted"
  predict(newdata = train)
```

```
# Accuracy of model on training data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # .6180 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 3528 3072
##        Low  1877 4709
##
##                Accuracy : 0.6247
##                  95% CI : (0.6163, 0.633)
##     No Information Rate : 0.5901
##     P-Value [Acc > NIR] : 2.675e-16
##
##                   Kappa : 0.2495
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6527
##             Specificity : 0.6052
##          Pos Pred Value : 0.5345
##          Neg Pred Value : 0.7150
##              Prevalence : 0.4099
##          Detection Rate : 0.2676
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.6290
##
##        'Positive' Class : High
##
```

KNN ON TEST DATA

```
# Predict test set
open_p <- predict(  # Create new variable ("predicted")
  fit_knn,          # Apply saved model
  newdata = test    # Use test data
)
```

```r
# Accuracy of model on test data
table(
  actualclass = test$open_t,  # True outcome
  predictedclass = open_p     # Predicted outcome
) %>%
confusionMatrix() %>%         # Accuracy statistics
print()                       # .5772 accuracy on full data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1412 1458
##        Low   938 1843
##
##                Accuracy : 0.576
##                  95% CI : (0.563, 0.5889)
##     No Information Rate : 0.5841
##     P-Value [Acc > NIR] : 0.8952
##
##                   Kappa : 0.1543
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6009
##             Specificity : 0.5583
##          Pos Pred Value : 0.4920
##          Neg Pred Value : 0.6627
##              Prevalence : 0.4159
##          Detection Rate : 0.2499
##    Detection Prevalence : 0.5079
##       Balanced Accuracy : 0.5796
##
##        'Positive' Class : High
##
```

DECISION TREE ON TRAINING DATA

```r
# Train decision tree on training data (takes a moment)
tic("Decision tree")
fit_dt <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,       # Use training data
  method = "rpart",   # Recursive partitioning
  trControl = trainControl(method = "cv")  # Cross-validate
)
toc()  # 1.1 sec for n = 700; 1.8 sec for full data
```

```
## Decision tree: 0.668 sec elapsed
```

```r
# Show processing summary
fit_dt
```

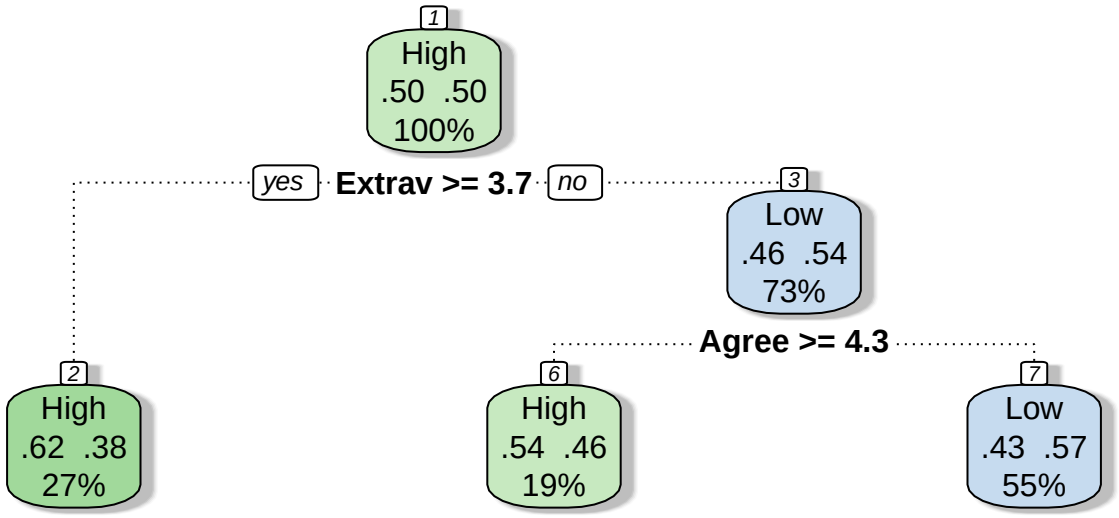
```
## CART
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 11867, 11867, 11868, 11868, 11867, 11867, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.004555117  0.5764478  0.15292936
##   0.032645005  0.5643858  0.12898642
##   0.126935925  0.5232800  0.04603416
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.004555117.
```

```r
# Description of final training model
fit_dt$finalModel
```

```
## n= 13186
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 13186 6586 High (0.5005309 0.4994691)
##   2) Extrav>=3.65 3512 1331 High (0.6210137 0.3789863) *
##   3) Extrav< 3.65 9674 4419 Low (0.4567914 0.5432086)
##     6) Agree>=4.25 2471 1128 High (0.5435047 0.4564953) *
##     7) Agree< 4.25 7203 3076 Low (0.4270443 0.5729557) *
```

```r
# Plot final training model
fit_dt$finalModel %>%
  fancyRpartPlot(
    main = "Predicting Open",
    sub = "Training Data"
  )
```

Predicting Open



Training Data

```
# Predict training set
open_p <- fit_dt %>%  # "predicted"
  predict(newdata = train)
```

```
# Accuracy of model on training data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # .5802 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 3524 3076
##        Low  2459 4127
##
##                Accuracy : 0.5802
##                  95% CI : (0.5718, 0.5887)
##     No Information Rate : 0.5463
##     P-Value [Acc > NIR] : 2.13e-15
##
##                   Kappa : 0.1606
```

```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5890
##             Specificity : 0.5730
##          Pos Pred Value : 0.5339
##          Neg Pred Value : 0.6266
##              Prevalence : 0.4537
##          Detection Rate : 0.2673
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.5810
##
##        'Positive' Class : High
##
```

DECISION TREE ON TEST DATA

```
# Predict test set
open_p <- fit_dt %>%
  predict(newdata = test)

# Accuracy of model on test data
table(
  actualclass = test$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.5776 on full test data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1544 1326
##        Low  1061 1720
##
##                Accuracy : 0.5776
##                  95% CI : (0.5646, 0.5905)
##     No Information Rate : 0.539
##     P-Value [Acc > NIR] : 2.967e-09
##
##                   Kappa : 0.1562
##
##  Mcnemar's Test P-Value : 6.534e-08
##
##             Sensitivity : 0.5927
##             Specificity : 0.5647
##          Pos Pred Value : 0.5380
##          Neg Pred Value : 0.6185
##              Prevalence : 0.4610
##          Detection Rate : 0.2732
##    Detection Prevalence : 0.5079
```

```
##       Balanced Accuracy : 0.5787
##
##        'Positive' Class : High
##
```

RANDOM FOREST ON TRAINING DATA

```
# Define parameters for the random forest
control <- trainControl(
  method  = "repeatedcv",  # Repeated cross-validation
  number  = 10,            # Number of folds
  repeats = 3,             # Number of sets of folds
  search  = "random",      # Max number of tuning parameters
  allowParallel = TRUE     # Allow parallel processing
)
```

```
# Train random forest on training data (can take a while)
tic("Random forest")
fit_rf <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,          # Use training data
  method = "rf",         # Use random forests
  metric = "Accuracy",   # Use accuracy as criterion
  tuneLength = 15,       # Number of levels for parameters
  ntree = 300,           # Number of trees
  trControl = control    # Link to parameters
)
toc()  # 32 sec when n = 700; 635 sec for full data
```

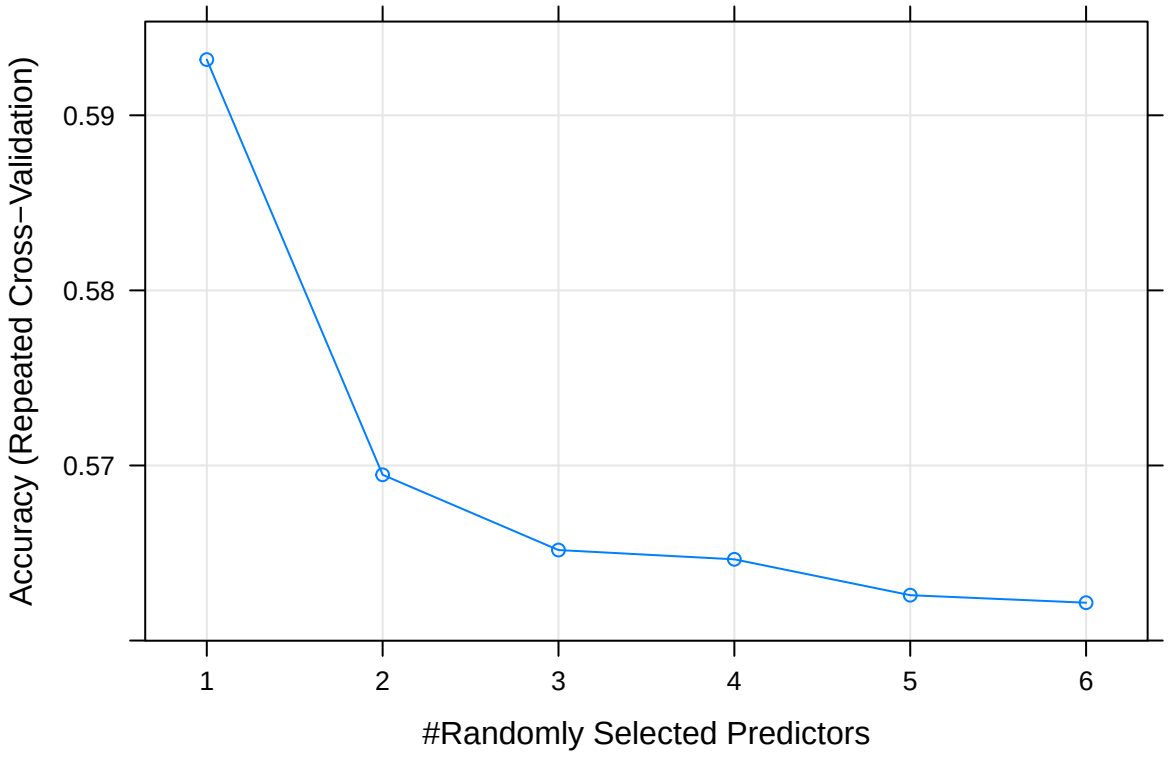
```
## Random forest: 533.291 sec elapsed
```

```
# Show processing summary
fit_rf
```

```
## Random Forest
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 11867, 11867, 11868, 11868, 11867, 11867, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.5931825  0.1864808
##   2     0.5694682  0.1389488
##   3     0.5651699  0.1303523
##   4     0.5646404  0.1292891
##   5     0.5625916  0.1251975
```

```
##   6     0.5621620  0.1243339
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

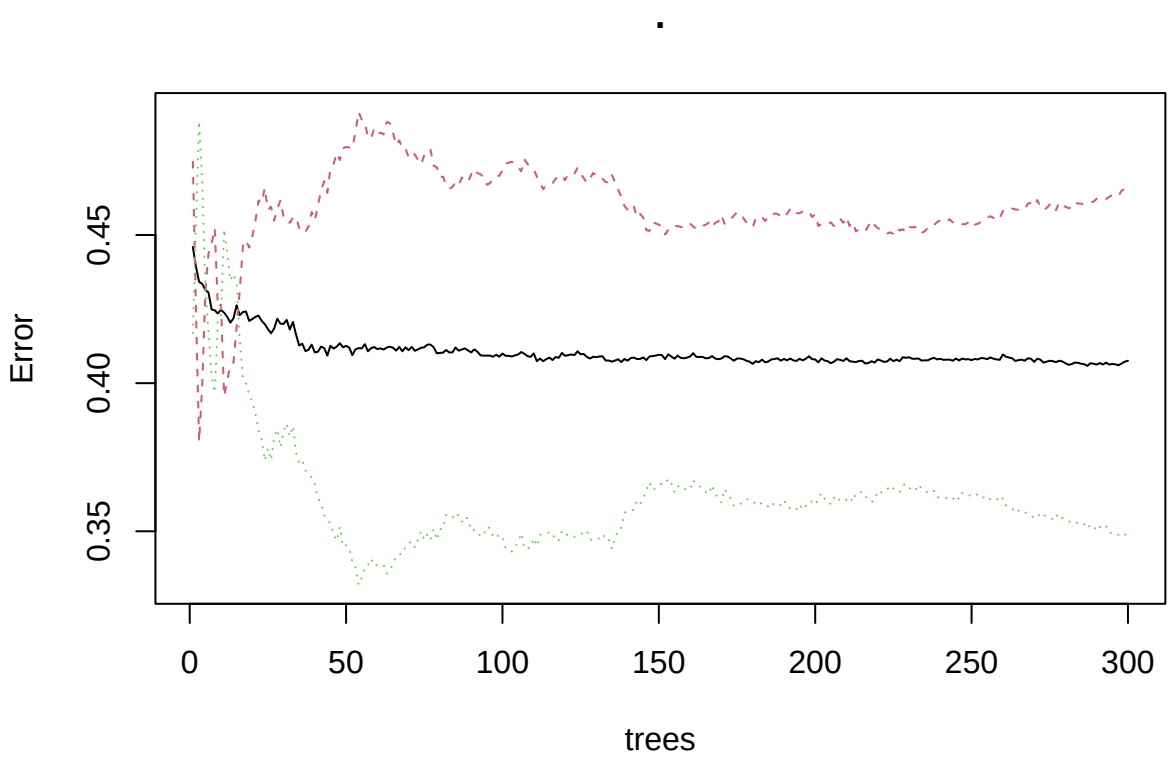
```
# Plot accuracy by number of predictors
fit_rf %>% plot()
```




```
# Accuracy of model with training data
fit_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 40.76%
## Confusion matrix:
##      High  Low class.error
## High 3524 3076   0.4660606
## Low  2298 4288   0.3489220
```

```
# Plot error by number of trees; Red is error for "High,"
# green is error for "Low," and black is error or "OOB,"
# or "out of bag" (i.e., the probability that any given
# prediction is not correct within the test data, or the
# overall accuracy)
fit_rf$finalModel %>% plot()
```




```
# Predict training data
open_p <- fit_rf %>%        # "predicted"
  predict(newdata = train)  # Use train data
```

```
# Accuracy of model on test data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.7145 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 4277 2323
```

```
##        Low  1471 5115
##
##                Accuracy : 0.7123
##                  95% CI : (0.7045, 0.72)
##     No Information Rate : 0.5641
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4246
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7441
##             Specificity : 0.6877
##          Pos Pred Value : 0.6480
##          Neg Pred Value : 0.7766
##              Prevalence : 0.4359
##          Detection Rate : 0.3244
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.7159
##
##        'Positive' Class : High
##
```

RANDOM FOREST ON TEST DATA

```
# Predict test set
open_p <- fit_rf %>%       # "predicted"
  predict(newdata = test)  # Use test data
```

```
# Accuracy of model on test data
table(
  actualclass = test$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.5912 accuracy on full test data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1525 1345
##        Low  1000 1781
##
##                Accuracy : 0.585
##                  95% CI : (0.5721, 0.5979)
##     No Information Rate : 0.5532
##     P-Value [Acc > NIR] : 7.360e-07
##
##                   Kappa : 0.1714
##
```

```
##  Mcnemar's Test P-Value : 1.214e-12
##
##             Sensitivity : 0.6040
##             Specificity : 0.5697
##          Pos Pred Value : 0.5314
##          Neg Pred Value : 0.6404
##              Prevalence : 0.4468
##          Detection Rate : 0.2699
##    Detection Prevalence : 0.5079
##       Balanced Accuracy : 0.5868
##
##        'Positive' Class : High
##
```

SUMMARIZING PREDICTIONS

```
# TOTAL n = 18,837
# TRAIN n = 13,186
# TEST  n =  5,651

# How many cases in "high"?
test %>% pull(open_t) %>% summary()
```

```
## High  Low
## 2870 2781
```

```
# 2870 / 5651 = .5078 (about 51%)

#      TRAIN  TEST
# KNN    62%   58%
# DT     58%   58%
# RF     71%   59%
```

```
# CLEAN UP ############################################################

# Clear data
# rm(list = ls())  # Removes all objects from environment

# Clear packages
# p_unload(all)  # Remove all contributed packages

# Clear plots
#graphics.off()  # Clears plots, closes all graphics devices

# Clear console
#cat("\014")  # Mimics ctrl+L

# Clear R
#   You may want to use Session > Restart R, as well, which
#   resets changed options, relative paths, dependencies,
#   and so on to let you start with a clean slate
```

```
# Clear mind :)
```