Introduction to Machine Learning in R

Lab 4: Text Mining

Table of contents

Load Required Packages **1**

1. Structural Topic Models **2**

2. Text Classification **6**

Random Forest (with tuning) for text classification 7

Exercise 15

References **16**

Load Required Packages

```
library(here)
library(tidyverse)
library(tidymodels)
library(ranger)
library(vip)
library(stm)
library(textrecipes)
library(kableExtra)
```

1. Structural Topic Models

This code is based on parts of the analyses presented in Kraft and Dolan (2023). The full replication files can be found on the Harvard Dataverse: https://doi.org/10.7910/DVN/OPW1XY. See also Kraft (2024) for more details.

```
load(here("data/data_anes.Rdata"))
```

First, we fit the structural topic model for all three ANES waves.

```
## ANES 2012 ----

## Merge open-ended responses with main survey
df2012 <- anes2012 %>%
  inner_join(tibble(caseid = oe2012$caseid,
                    resp = apply(oe2012[,-1], 1, paste, collapse = " "))) %>%
  mutate(resp = str_trim(resp)) %>%
  filter(resp != "") %>%
  na.omit()

## Text preprocessing
tmp2012 <- textProcessor(
  documents = df2012$resp,
  metadata = dplyr::select(df2012, age, female, educ_cont, pid_cont, educ_pid),
  customstopwords = readLines(here("data/stopwords.txt")),
  verbose = FALSE
)

## Prepare documents, remove infrequent terms
out2012 <- prepDocuments(
  tmp2012$documents,
  tmp2012$vocab,
  tmp2012$meta,
  lower.thresh = 10,
  verbose = FALSE
)

## Estimate structural topic model
fit2012 <- stm(
  out2012$documents,
  out2012$vocab,
  prevalence = as.matrix(out2012$meta),
```

```
  K = 50,
  seed = 12345,
  verbose = FALSE
)
```

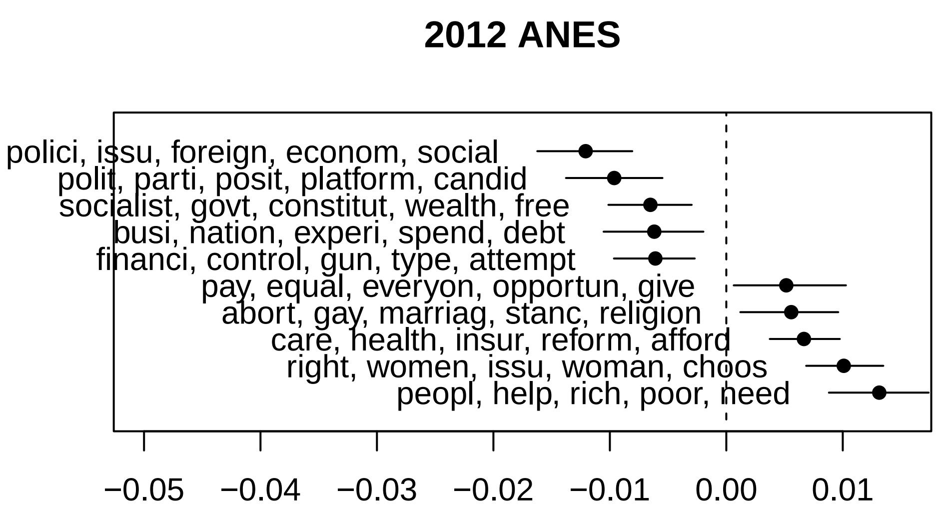
```
## ANES 2016 ----

## Merge open-ended responses with main survey
df2016 <- anes2016 %>%
  inner_join(tibble(caseid = oe2016$caseid,
                    resp = apply(oe2016[,-1], 1, paste, collapse = " "))) %>%
  mutate(resp = str_trim(resp)) %>%
  filter(resp != "") %>%
  na.omit()

## Text preprocessing
tmp2016 <- textProcessor(
  documents = df2016$resp,
  metadata = dplyr::select(df2016, age, female, educ_cont, pid_cont, educ_pid),
  customstopwords = readLines(here("data/stopwords.txt")),
  verbose = FALSE
)

## Prepare documents, remove infrequent terms
out2016 <- prepDocuments(
  tmp2016$documents,
  tmp2016$vocab,
  tmp2016$meta,
  lower.thresh = 10,
  verbose = FALSE
)

## Estimate structural topic model
fit2016 <- stm(
  out2016$documents,
  out2016$vocab,
  prevalence = as.matrix(out2016$meta),
  K = 50,
  seed = 12345,
  verbose = FALSE
)
```

```r
## ANES 2020 ----

## Merge open-ended responses with main survey
df2020 <- anes2020 %>%
  inner_join(tibble(caseid = oe2020$caseid,
                    resp = apply(oe2020[,-1], 1, paste, collapse = " "))) %>%
  mutate(resp = str_trim(resp)) %>%
  filter(resp != "") %>%
  na.omit()

## Text preprocessing
tmp2020 <- textProcessor(
  documents = df2020$resp,
  metadata = dplyr::select(df2020, age, female, educ_cont, pid_cont, educ_pid),
  customstopwords = readLines(here("data/stopwords.txt")),
  verbose = FALSE
)

## Prepare documents, remove infrequent terms
out2020 <- prepDocuments(
  tmp2020$documents,
  tmp2020$vocab,
  tmp2020$meta,
  lower.thresh = 10,
  verbose = FALSE
)

## Estimate structural topic model
fit2020 <- stm(
  out2020$documents,
  out2020$vocab,
  prevalence = as.matrix(out2020$meta),
  K = 50,
  seed = 12345,
  verbose = FALSE
)
```

Next, we estimate topic differences betwwn men and women.

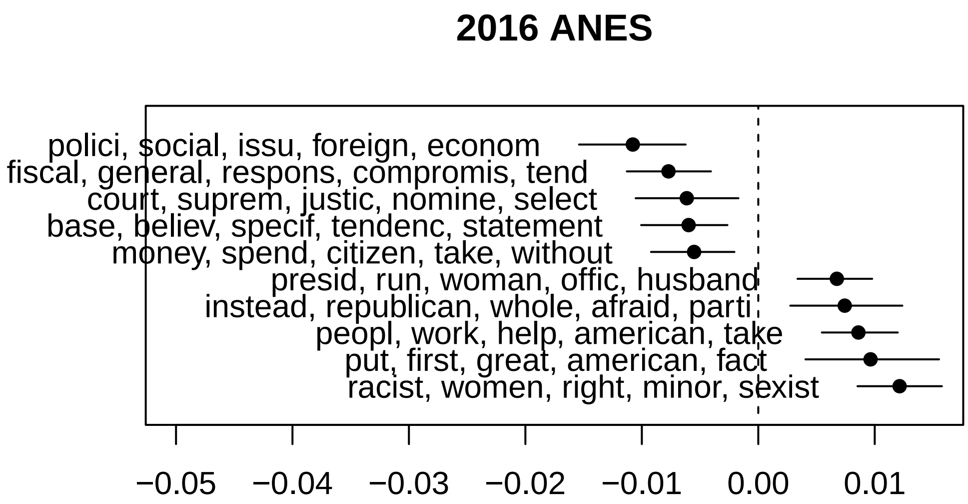
```r
## estimate topic prevalence effects
prep2012 <- estimateEffect(~ age + female + educ_cont + pid_cont + educ_pid,
                           fit2012, meta = out2012$meta, uncertainty = "Global")
```

```
prep2016 <- estimateEffect(~ age + female + educ_cont + pid_cont + educ_pid,
                           fit2016, meta = out2016$meta, uncertainty = "Global")
prep2020 <- estimateEffect(~ age + female + educ_cont + pid_cont + educ_pid,
                           fit2020, meta = out2020$meta, uncertainty = "Global")

## select topics with largest gender effects
tmp2012 <- tibble(estimate = sapply(summary(prep2012)$tables,
                                    function(x) x["female","Estimate"]),
                  topics = prep2012$topics) %>% arrange(estimate)
topics2012 <- c(head(tmp2012$topics, 5), tail(tmp2012$topics, 5))
tmp2016 <- tibble(estimate = sapply(summary(prep2016)$tables,
                                    function(x) x["female","Estimate"]),
                  topics = prep2016$topics) %>% arrange(estimate)
topics2016 <- c(head(tmp2016$topics, 5), tail(tmp2016$topics, 5))
tmp2020 <- tibble(estimate = sapply(summary(prep2020)$tables,
                                    function(x) x["female","Estimate"]),
                  topics = prep2020$topics) %>% arrange(estimate)
topics2020 <- c(head(tmp2020$topics, 5), tail(tmp2020$topics, 5))

## Visualize results: gender differences in topic proportions
plot.estimateEffect(prep2012, covariate = "female", topics = topics2012,
                    model = fit2012, xlim = c(-.05,.015), method = "difference",
                    cov.value1 = 1, cov.value2 = 0, labeltype = "prob", n=5,
                    verbose.labels = F, width=50, main = "2012 ANES")
```

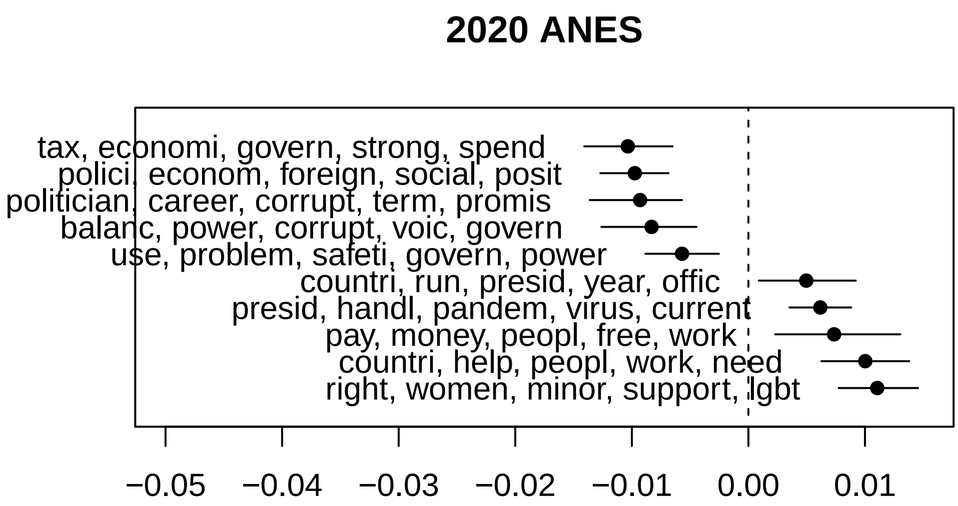



```
plot.estimateEffect(prep2016, covariate = "female", topics = topics2016,
                    model = fit2016, xlim = c(-.05,.015), method = "difference",
```

```
                          cov.value1 = 1, cov.value2 = 0, labeltype = "prob", n=5,
                          verbose.labels = F, width=50, main = "2016 ANES")
```




```
plot.estimateEffect(prep2020, covariate = "female", topics = topics2020,
                          model = fit2020, xlim = c(-.05,.015), method = "difference",
                          cov.value1 = 1, cov.value2 = 0, labeltype = "prob", n=5,
                          verbose.labels = F, width=50, main = "2020 ANES")
```



2. Text Classification

This code is based on material provided by Paul C. Bauer. The data comes from his project on measuring trust (see Landesvatter & Bauer (forthcoming) in *Sociological Methods & Research*).

The data for the lab was pre-processed. 56 open-ended answers that revealed the respondent's profession, age, area of living/rown or others' specific names/categories, particular activities (e.g., town elections) or city were deleted for reasons of anonymity.

- **Research questions**: Do individuals interpret trust questions similar? Do they have a higher level if they **think of someone personally known** to them?
  - **Objective**: Predict whether they think of personally known person (yes/no).

We start by loading our data that contains the following variables:

- `respondent_id`: Individual's identification number (there is only one response per individual - so it's also the id for the response)
- `social_trust_score`: Individual's value on the trust scale
  - Question: *Generally speaking, would you say that most people can betrusted, or that you can't be too careful in dealing with people? Please tell me on a score of 0 to 6, where 0 means you can't be too careful and 6 means that most people can be trusted.*
    - **Original scale**: 0 - You can't be too careful; 1; 2; 3; 4; 5; 6 - Most people can be trusted; Don't know;
    - **Recoded scale**: `Don't know = NA` and values `0-6` standardized to `0-1`.
- `text`: Individual's response to the probing question
  - Question: *In answering the previous question, who came to your mind when you were thinking about 'most people?' Please describe.*
- `human_classified`: Variable that contains the manual human classification of whether person was thinking about **someone personally known to them or not** (this is based on the open-ended response to `text`)
  - `N = 295` were classified as `1 = yes`
  - `N = 666` were classified as `0 = no`
  - `N = 482` were not classified (we want to make predictions on those those!)

The variable `human_classified` contains the values `NA` (was not classified), `1` (respondents were thinking about people known to them) and `0` (respondents were not thinking about people known to them).

Random Forest (with tuning) for text classification

- Steps
  1. Load and initial split of the data
  2. Create folds for cross-validation

3. Define recipe (text preprocessing) & model (random forest + parameters to tune) & workflow
4. **1st fitting & tuning session**: Fit model to resampled training data (folds) + tuning in parallel and inspect accuracy & tuning parameters afterwards
5. If happy, `select_best` hyperparameters (identified in tuning), `finalize_model` the model with those parameters and create a final `workflow_final`. Train/fit `workflow_final` to the full training dataset and obtain `fit_final`.
6. Use `fit_final` to predict outcome both in `data_train` and `data_test` and evaluate accuracy.
7. To explore which predictors are important calcuculate and visualize variable importance.

We first import the data into R:

```
load(file = here("data/data_text_trust.Rdata"))
```

```
# Extract data with missing outcome
  data_missing_outcome <- data %>%
                  filter(is.na(human_classified))
  dim(data_missing_outcome)
```

```
[1] 482   4
```

```
# Omit individuals with missing outcome from data
  data <- data %>% drop_na(human_classified) # ?drop_na
  dim(data)
```

```
[1] 961   4
```

```
# 1.

# Split the data into training and test data
  set.seed(345)
  data_split <- initial_split(data, prop = 0.8)
  data_split # Inspect
```

```
<Training/Testing/Total>
<768/193/961>
```

```r
# Extract the two datasets
  data_train <- training(data_split)
  data_test <- testing(data_split) # Do not touch until the end!


# 2.

# Create resampled partitions of training data
  data_folds <- vfold_cv(data_train, v = 2) # V-fold/k-fold cross-validation
  data_folds # data_folds now contains several resamples of our training data
```

```
#  2-fold cross-validation
# A tibble: 2 x 2
  splits            id
  <list>            <chr>
1 <split [384/384]> Fold1
2 <split [384/384]> Fold2
```

```r
# 3.

# Define the recipe & model
  recipe1 <-
    recipe(human_classified ~ respondent_id + text, data = data_train) %>%
    update_role(respondent_id, new_role = "id") %>% # update role
    step_tokenize(text)  %>% # Tokenize text (split into words)
    step_stopwords(text) %>% # Remove stopwords
    step_stem(text) %>% # Text stemming
    step_tokenfilter(text, max_tokens = 100) %>% # Filter max tokens
    step_tf(text) # convert to term-feature matrix

# Extract and preview data + recipe (direclty with $)
  data_preprocessed <- prep(recipe1, data_train)$template
  dim(data_preprocessed)
```

```
[1] 768 102
```

```r
  # View(data_preprocessed)
  table(data_preprocessed[,3]) # first token frequency table
```

```
tf_text_acquaint
```

```
  0   1   2
746  21   1
```

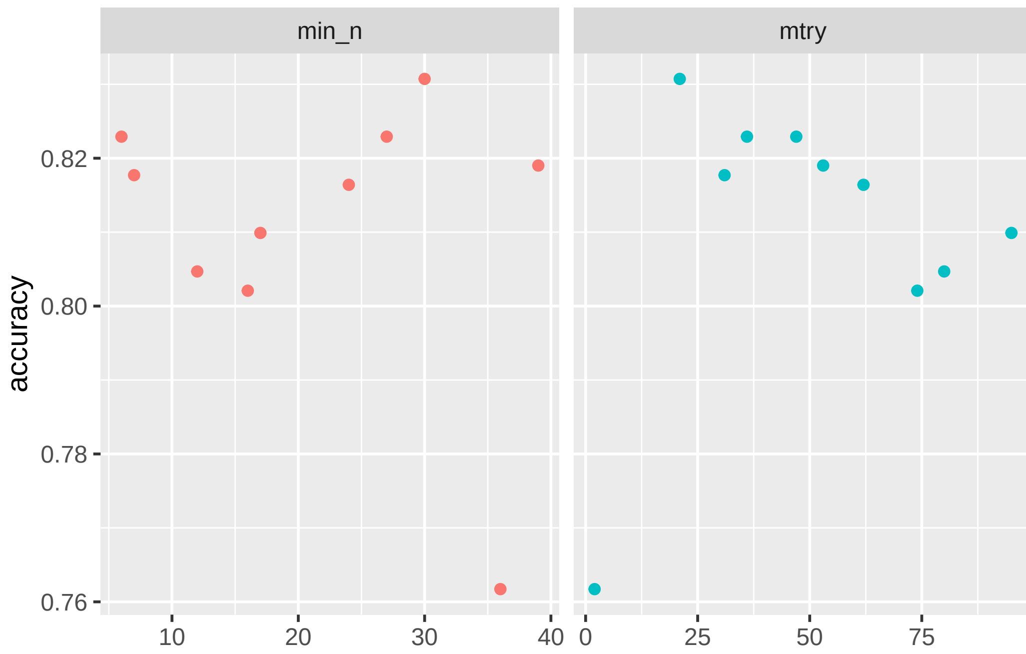
```r
# Specify model with tuning
model1 <- rand_forest(
  mtry = tune(), # tune mtry parameter
  trees = 1000, # grow 1000 trees
  min_n = tune() # tune min_n parameter
) %>%
  set_mode("classification") %>%
  set_engine("ranger",
             importance = "permutation") # potentially computational intensive

# Specify workflow (with tuning)
workflow1 <- workflow() %>%
  add_recipe(recipe1) %>%
  add_model(model1)


# 4. 1st fitting & tuning & evaluation of accuracy

# Specify to use parallel processing
doParallel::registerDoParallel()

set.seed(345)
tune_result <- tune_grid(
  workflow1,
  resamples = data_folds,
  grid = 10 # choose 10 grid points automatically
)
```

```
i Creating pre-processing data to finalize unknown parameter: mtry
```

```r
tune_result
```

```
# Tuning results
# 2-fold cross-validation
# A tibble: 2 x 4
  splits            id    .metrics          .notes
  <list>            <chr> <list>            <list>
1 <split [384/384]> Fold1 <tibble [30 x 6]> <tibble [0 x 3]>
2 <split [384/384]> Fold2 <tibble [30 x 6]> <tibble [0 x 3]>
```

```
tune_result %>%
  collect_metrics() %>% # extract metrics
  filter(.metric == "accuracy") %>% # keep accuracy only
  select(mean, min_n, mtry) %>% # subset variables
  pivot_longer(min_n:mtry, # convert to longer
    values_to = "value",
    names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) + # plot!
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "accuracy")
```




```
# 5. Choose best model after tuning & fit/train

  # Find tuning parameter combination with best performance values
  best_accuracy <- select_best(tune_result, metric = "accuracy")
  best_accuracy
```

```
# A tibble: 1 x 3
   mtry min_n .config
  <int> <int> <chr>
1    21    30 Preprocessor1_Model01
```

```
  # Take list/tibble of tuning parameter values
  # and update model1 with those values.
  model_final <- finalize_model(model1, best_accuracy)
  model_final
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = 21
  trees = 1000
  min_n = 30

Engine-Specific Arguments:
  importance = permutation

Computational engine: ranger
```

```
# Define final workflow
  workflow_final <- workflow() %>%
    add_recipe(recipe1) %>% #  use standard recipe
    add_model(model_final) # use final model

  # Fit final model
  fit_final <- parsnip::fit(workflow_final, data = data_train)
  fit_final
```

```
== Workflow [trained] ==========================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
5 Recipe Steps

* step_tokenize()
* step_stopwords()
* step_stem()
* step_tokenfilter()
* step_tf()

-- Model -----------------------------------------------------------------------
Ranger result
```

```
Call:
 ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~21L,      x), num.trees = ~1

Type:                             Probability estimation
Number of trees:                  1000
Sample size:                      768
Number of independent variables:  100
Mtry:                             21
Target node size:                 30
Variable importance mode:         permutation
Splitrule:                        gini
OOB prediction error (Brier s.):  0.1205772
```

```
# Q: What do the values for `mtry` and `min_n` in the final model mean?

# A:
# mtry = An integer for the number of predictors that will be randomly sampled at each split
# trees = An integer for the number of trees contained in the ensemble.
# min_n = An integer for the minimum number of data points in a node that are required for th


# 6. Predict & evaluate accuracy (both in full training and test data)
  metrics_combined <-
    metric_set(accuracy, precision, recall, f_meas) # Set accuracy metrics

# Accuracy: Full training data
  augment(fit_final, new_data = data_train) %>%
  metrics_combined(truth = human_classified, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 accuracy  binary         0.895
2 precision binary         0.904
3 recall    binary         0.947
4 f_meas    binary         0.925
```

```
# Cross-classification table
  augment(fit_final, new_data = data_train) %>%
      conf_mat(data = .,
               truth = human_classified, estimate = .pred_class)
```

```
          Truth
Prediction   0   1
         0 499  53
         1  28 188
```

```
# Accuracy: Test data
  augment(fit_final, new_data = data_test) %>%
  metrics_combined(truth = human_classified, estimate = .pred_class)
```

```
# A tibble: 4 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 accuracy  binary         0.860
2 precision binary         0.894
3 recall    binary         0.914
4 f_meas    binary         0.904
```

```
# Cross-classification table
  augment(fit_final, new_data = data_test) %>%
      conf_mat(data = .,
               truth = human_classified, estimate = .pred_class)
```

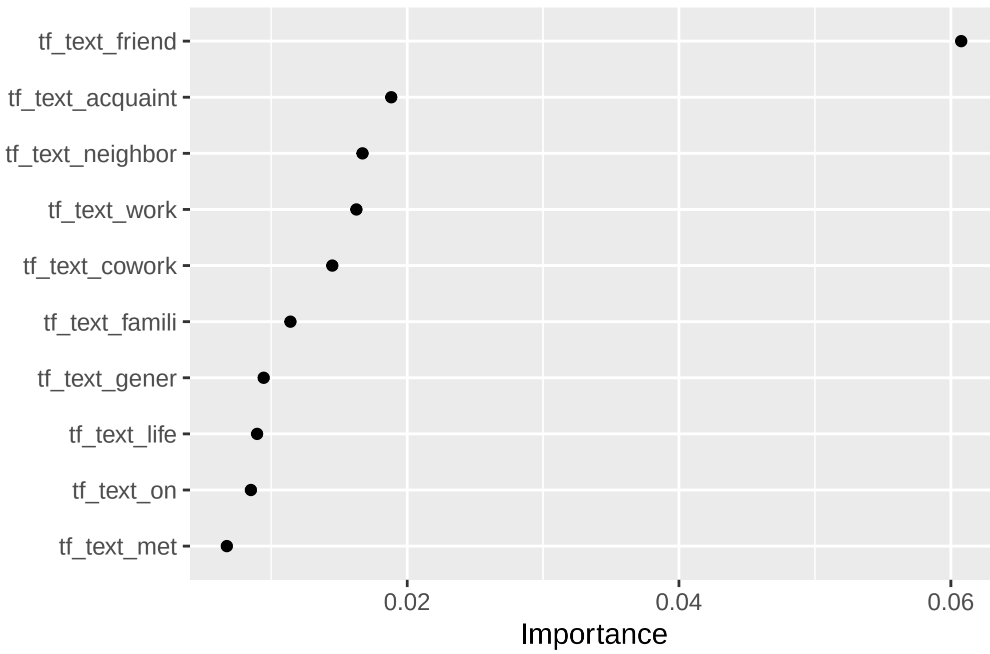
```
          Truth
Prediction   0   1
         0 127  15
         1  12  39
```

```
# 7.   Visualize variable importance

  fit_final$fit$fit %>%
      vip::vi() %>%
    dplyr::slice(1:10) %>%
    kable()
```

Variable	Importance
tf_text_friend	0.0607590
tf_text_acquaint	0.0188360
tf_text_neighbor	0.0167197
tf_text_work	0.0162719

Variable	Importance
tf_text_cowork	0.0144992
tf_text_famili	0.0114177
tf_text_gener	0.0094499
tf_text_life	0.0089688
tf_text_on	0.0085111
tf_text_met	0.0067403

```r
# Visualize variable importance
  fit_final$fit$fit %>%
    vip(geom = "point")
```



Exercise

- In the lab above we used a random forest to built a classifier for our labelled text. Thereby we made different choice in preprocessing the texts. Please modify those choices (e.g., don't remove stopwords, change `max_tokens`). How does this affect the accuracy of your model (and the training process)?

References

Kraft, Patrick W. 2024. "Women Also Know Stuff: Challenging the Gender Gap in Political Sophistication." *American Political Science Review* 118 (2): 903–21.

Kraft, Patrick W, and Kathleen Dolan. 2023. "Asking the Right Questions: A Framework for Developing Gender-Balanced Political Knowledge Batteries." *Political Research Quarterly* 76 (1): 393–406. https://doi.org/10.1177/10659129221092473.