Network Construction-Integration

Contents

Introduction

The method described and implemented in this file comprises the steps for implementing a reading model based on the Construction-Integration (CI) framework

@*Kintsch*1998

using the programming language and environment R (R CORE TEAM, 2021). The present method expands the CI framework with the implementation of network analysis of texts in the construction of semantic representations, substituting the use of manually constructed propositions.

Textused

```
# Load the text
text = read_file('NCI_text1.txt')
```

Tokenization, *lemmatization*and*POStagging*

```
# Annotate text with udpipe
# Select lemmas, remove punctuation, remove contractions ('s and 'd), transform to lowercase and remove
text_annotated = text %>%
  udpipe_annotate(m_eng, .) %>%
  as_tibble() %>%
  select(sentence_id, lemma, upos) %>%
  filter(upos != 'PUNCT' & !str_detect(lemma, '\\W')) %>%
  mutate(lemma = lemma %>% str_to_lower()) %>%
  group_by(sentence_id) %>%
  distinct() %>%
  ungroup()
```

Grammaticalwords

```
text_lexical = text_annotated %>%
  filter(!upos %in% c('ADP', 'AUX', 'CCONJ', 'DET', 'PART', 'SCONJ'))
```

Synonymsand Hypernyms

```
# Get synonyms tibble
synonyms_tibble = get_words_synonyms(text_lexical$lemma,
                                     text_lexical$upos)

# Get hypernyms tibble
hypernyms_tibble = get_words_hypernyms(text_lexical$lemma,
                                       text_lexical$upos)
```

##

Search for synonymic relations

```
# Direct synonyms
direct_synonyms = synonyms_tibble %>%
  filter(synonyms %in% text_lexical$lemma)

# Direct hypernyms
direct_hypernyms = hypernyms_tibble %>%
  filter(hypernyms %in% text_lexical$lemma)
```

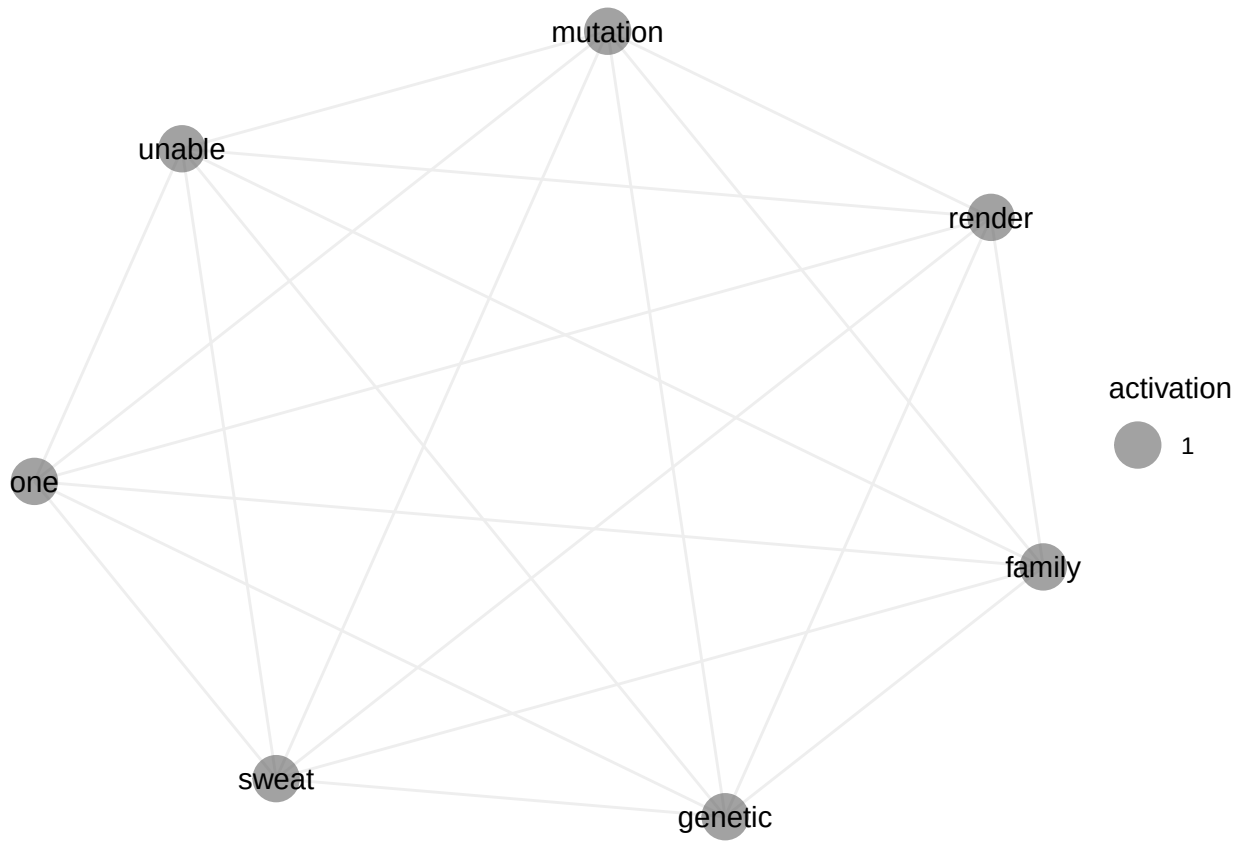
Build the network semantic representation

First cycle

```
# Create clique network
cycle1 = create_net(text_lexical, 1)

# Spread activation
V(cycle1)$activation = spread_activation(cycle1, NA)

cycle1 %>%
  plot_net()
```

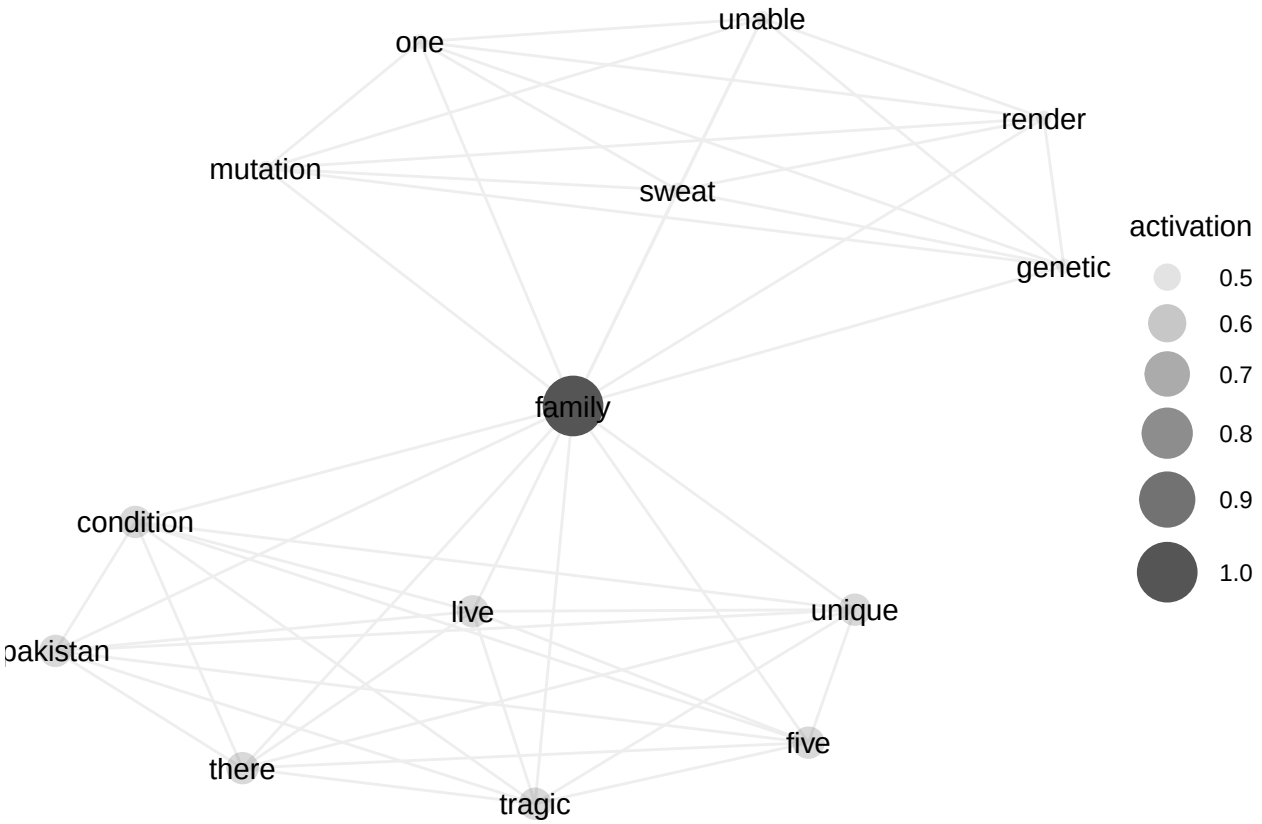


Secondcycle

```
# Create clique network
cycle2 = create_net(text_lexical, 2)

# Spread activation
V(cycle2)$activation = spread_activation(cycle2, cycle1)

cycle2 %>%
  plot_net()
```

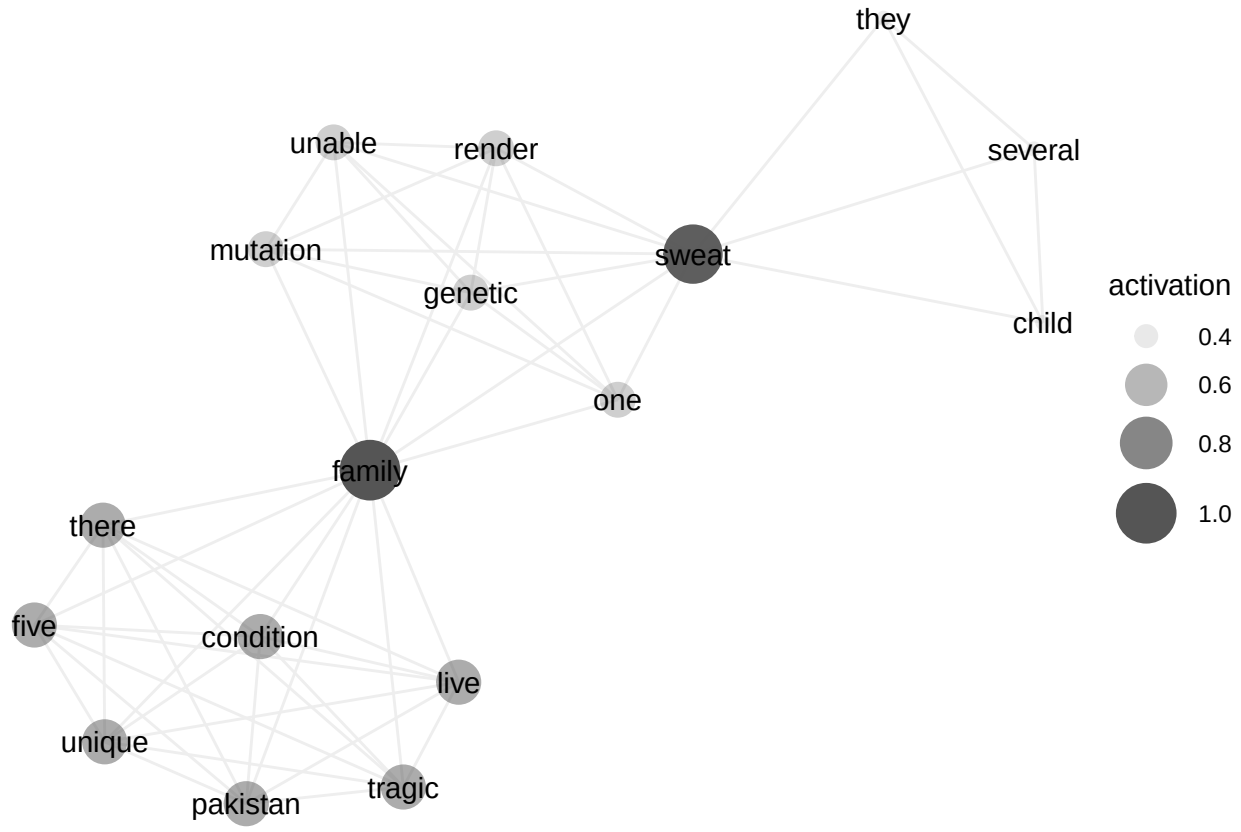


Thirdcycle

```
# Create clique network
cycle3 = create_net(text_lexical, 3)

# Spread activation
V(cycle3)$activation = spread_activation(cycle3, cycle2)

cycle3 %>%
  plot_net()
```

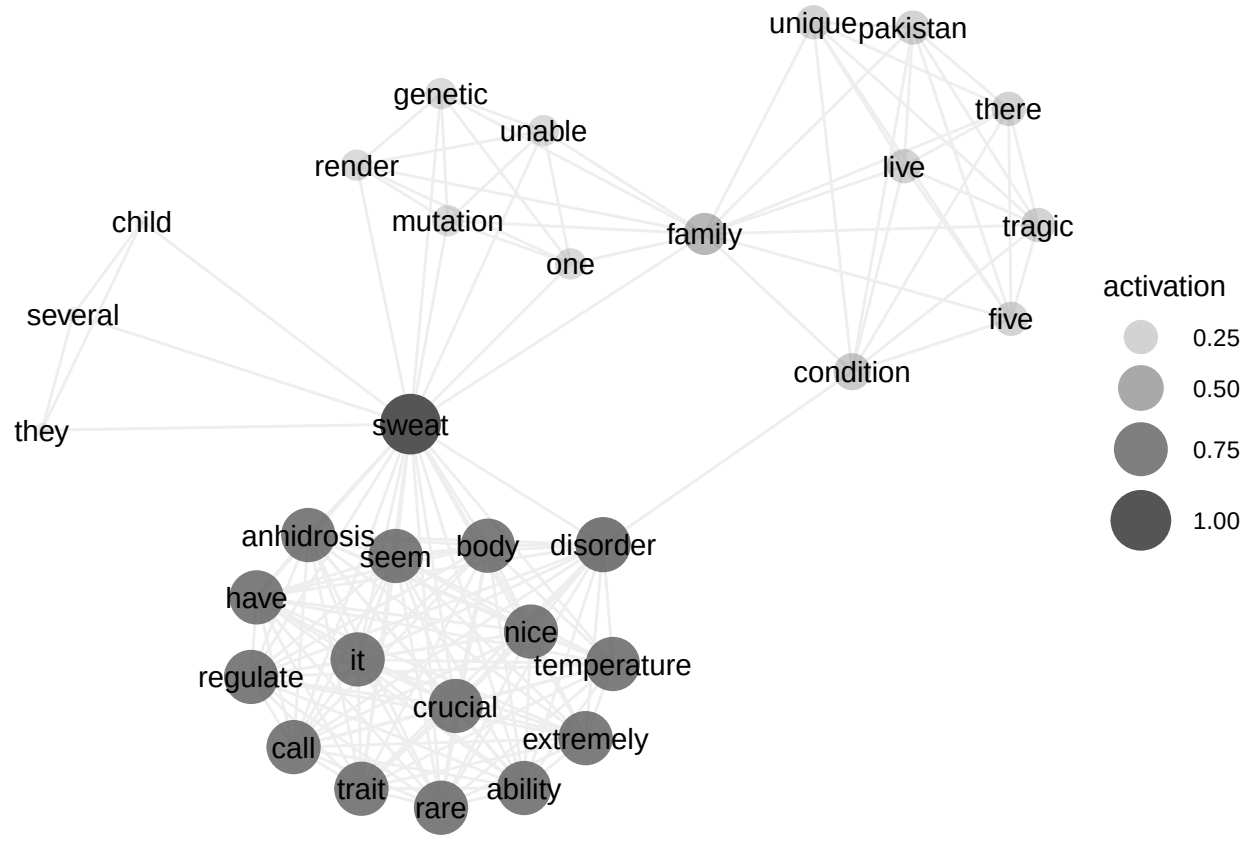


FourthCycle

```
# Create clique network
cycle4 = create_net(text_lexical, 4)

# Spread activation
V(cycle4)$activation = spread_activation(cycle4, cycle3)

cycle4 %>%
  plot_net()
```

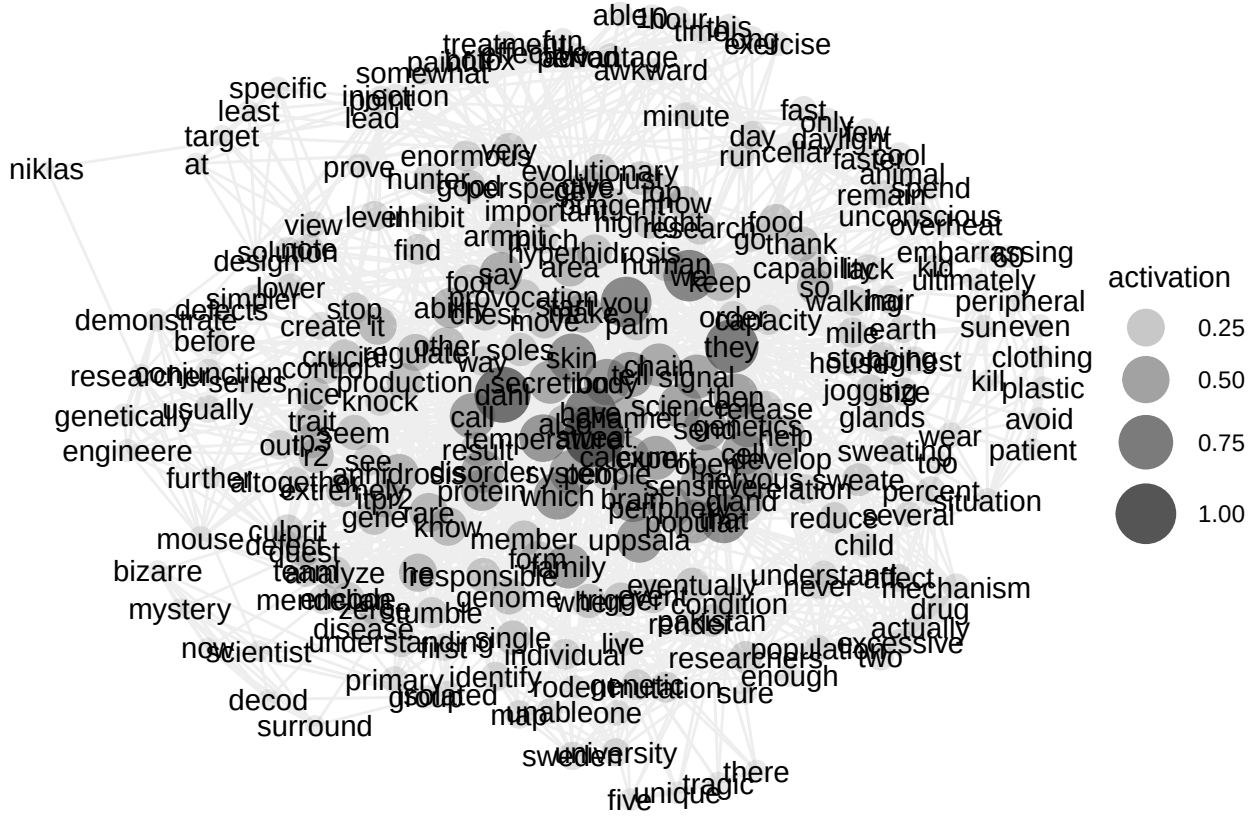


Cycles5to34

```
remaining_cycles = lapply(5:34, function(c) {
  create_net(text_lexical, c)
})

for (c in 1:length(remaining_cycles)) {
  if (c == 1) {
    V(remaining_cycles[[c]])$activation = spread_activation(
      remaining_cycles[[c]], cycle4)
  } else {
    V(remaining_cycles[[c]])$activation = spread_activation(
      remaining_cycles[[c]], remaining_cycles[[c - 1]])
  }
}

remaining_cycles[[30]] %>%
  plot_net()
```



Mtricasdarede

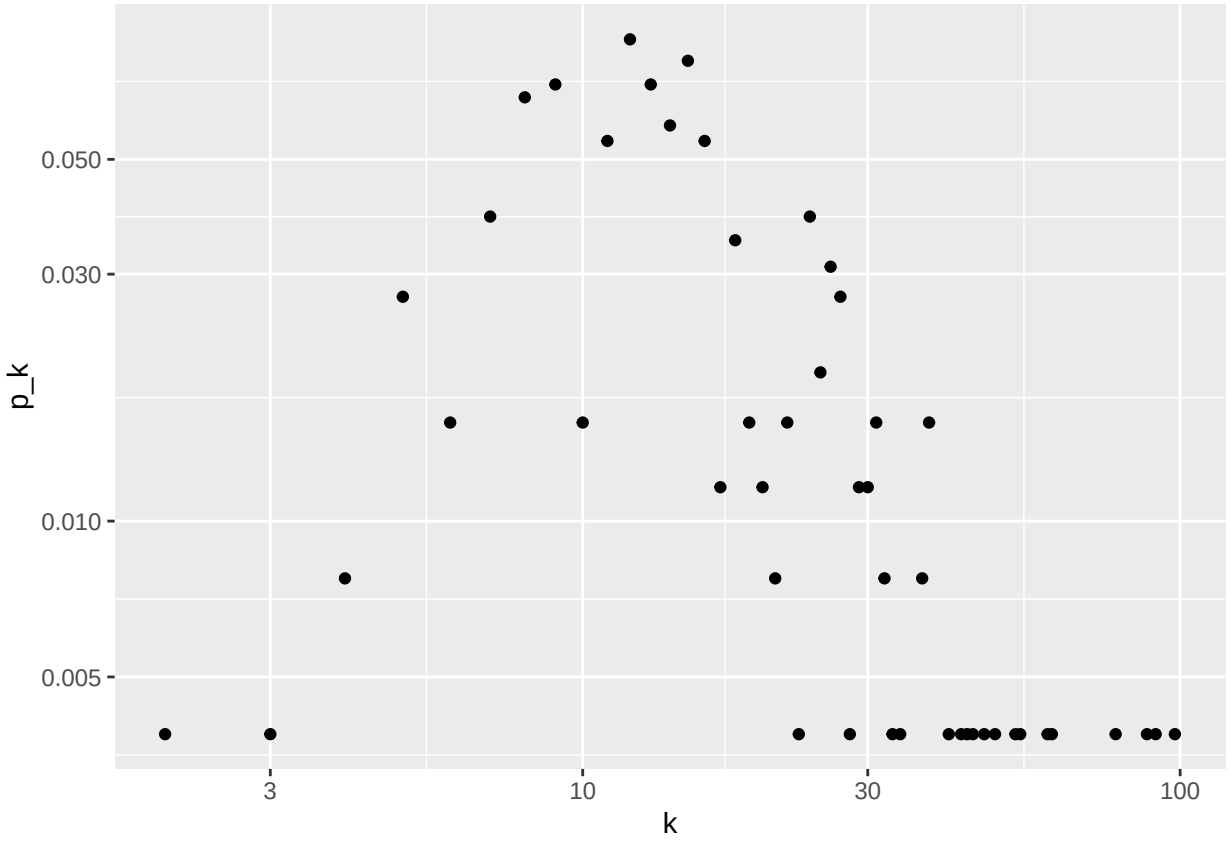
```
# Transitivity of final network
remaining_cycles[[30]] %>% transitivity()
```

```
## [1] 0.4784454
```

```
# Transitivity of random counterpart
length(E(remaining_cycles[[30]])) / (length(V(remaining_cycles[[30]])) * (length(V(remaining_cycles[[30]
```

```
## [1] 0.03520043
```

```
# Degree distribution
remaining_cycles[[30]] %>%
  degree() %>%
  tibble(k = .) %>%
  group_by(k) %>%
  count() %>%
  ungroup() %>%
  mutate(p_k = n/sum(n)) %>%
  ggplot(aes(k, p_k)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



Correlationwithmostusedwordsonrecall

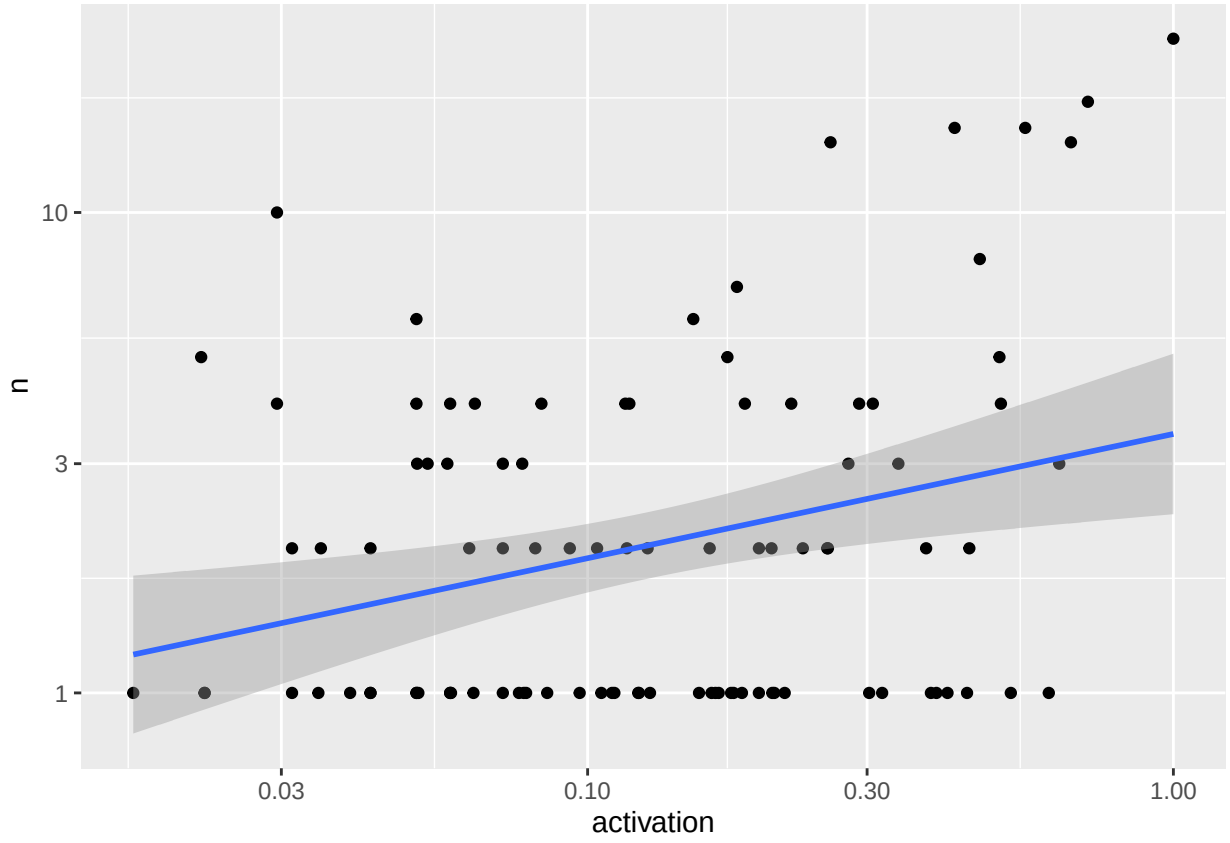
```
recall = read_file('NCI_recall.txt') %>%
  udpipe_annotate(m_eng, .) %>%
  as_tibble() %>%
  select(sentence_id, lemma, upos) %>%
  filter(upos != 'PUNCT' & !str_detect(lemma, '\\W')) %>%
  mutate(lemma = lemma %>% str_to_lower()) %>%
  group_by(sentence_id) %>%
  distinct() %>%
  ungroup()

activation_recall = tibble(lemma = V(remaining_cycles[[30]])$name,
       activation = V(remaining_cycles[[30]])$activation) %>%
  left_join(recall %>%
              filter(!upos %in%
                       c('ADP', 'AUX', 'CCONJ', 'DET', 'PART', 'SCONJ')) %>%
              group_by(lemma) %>%
              count(),
            by = 'lemma')

activation_recall %>%
  filter(!is.na(n)) %>%
  ggplot(aes(activation, n)) +
  geom_point() +
```

```
  scale_x_log10() +
  scale_y_log10() +
  geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Linearmodel

```
# TODO Refit model considering activation as a predictor of the probability of word recall and not of f
activation_recall %>%
  lm(n ~ activation, data = .) %>%
  summary()
```

```
##
## Call:
## lm(formula = n ~ activation, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1075 -1.7597 -0.4323  1.2095 10.2726
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7948     0.4674    1.70   0.0923 .
## activation   11.9325     1.7471    6.83 7.73e-10 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.198 on 96 degrees of freedom
##   (160 observations deleted due to missingness)
## Multiple R-squared:  0.327,  Adjusted R-squared:   0.32
## F-statistic: 46.65 on 1 and 96 DF,  p-value: 7.728e-10
```

References

R CORE TEAM. **R: A language and environment for statistical computing**. Vienna, Austria: R Foundation for Statistical Computing, 2021. Available at: https://www.R-project.org/.