Supplementary Information

Age-related shifts in temporal narrative content: A structural topic modeling analysis of a life-span corpus

Tomoe Taniguchi, Izumi Uehara, and Ichiro Kobayashi

This Supplementary Information provides all preprocessing and analysis scripts used in the study (Code availability statement). It contains:

- Supplementary Note 1: overview of the code, software environment and how to run it;
- Supplementary Table S1: correspondence between scripts and results in the article;
- Supplementary Note 2: reproducibility notes;
- Supplementary Note 3: full source code of every script (Listings S1–S21).

The code is released under the MIT License (Listing S21). To run it, save each listing under the relative path shown in its heading; together the listings form the complete code package.

Supplementary Note 1: Overview of the analysis code

Pipeline. The essays were morphologically analyzed with MeCab and UniDic. The preprocessing script `01_preprocess.py` extracts content-word lemmas (nouns, verbs, adjectives and adverbs), removes numerals and a list of 32 high-frequency function-like lemmas, and writes one row per essay with the covariates (age band, representative age, gender and temporal reference). The R scripts then fit the Structural Topic Model with the `stm` package (prevalence $\sim s(\text{age}) \times \text{gender} + \text{temporal reference}$, content $\sim$ gender, $K = 10$, spectral initialization), estimate covariate effects, draw the figures and run the supplementary analyses. All downstream scripts read the saved $K = 10$ model.

Data. The essay corpus is not included, for the reasons given in the Data availability statement. The scripts expect one MeCab/UniDic output file per essay in `data/`. The net-positivity analysis additionally requires the Japanese Sentiment Polarity Dictionary (Inui–Okazaki Laboratory, Tohoku University), which must be obtained from its original distributor.

Software environment.

Software	Version used
R	4.6.0
R package `stm`	1.3.8
R packages `dplyr` / `readr`	1.2.1 / 2.2.0
Python	3.8 (standard library only)
MeCab with UniDic	morphological analysis of the raw essays

How to run. After saving the listings under the paths shown in their headings, the whole analysis is run from the package root, in the order used for the article, by

```
bash run_all.sh           # or: PYTHON=python3 RSCRIPT=Rscript bash run_all.sh
```

Each script can also be run on its own (for example `Rscript src/runners/_run_fit.R`); relative paths are resolved from the package root. Console output of each step is written to `output/logs/`.

Supplementary Table S1: Scripts and the results they produce

Listing	Script	Purpose	Item in the article
S1	`run_all.sh`	Runs all steps in order	—
S2	`src/01_preprocess.py`	Content-word extraction from MeCab output	Methods
S3	`_run_searchK.R`	Diagnostics for choosing K ($K = 5, 10, 15, 20$)	Methods
S4	`_run_fit.R`	Main model ($K = 10$) and topic words	Tables 2 and 3
S5	`_run_table1.R`	Corpus composition	Table 1
S6	`_run_effect.R`	Age × gender effects on topic prevalence	Table 4
S7	`fig1.R`	Age trends by gender	Figure 1
S8	`_run_spread.R`	Concentration of the mean topic mixture	Table 5
S9	`_run_timeref.R`	Past/future topic shares by age band	Results (text)
S10	`fig2.R`	Topic composition of past and future essays	Figure 2
S11	`_run_ee_timeage.R`	Age × temporal-reference effects	Table 6
S12	`_run_findthoughts.R`	Representative documents of the planning topic	Discussion
S13	`_run_content_age.R`	Auxiliary model with age band as content covariate	Results, Discussion
S14	`_run_K20.R`	Re-estimation at $K = 20$	Results, Discussion
S15	`_exp2_features.py`	Abstract and concrete noun counts per essay	Results (supplementary analyses)
S16	`_exp2_clt.R`	Regression of the abstractness index	Results (supplementary analyses)
S17	`_exp2c_sentiment_tohoku.py`	Positive and negative word counts per essay	Results (supplementary analyses)
S18	`_exp2c_sentiment_tohoku_stats.R`	Regression of net positivity	Results (supplementary analyses)
S19	`_exp1_continuity.R`	Within-writer continuity and k-means types	Results (supplementary analyses)
S20	`_exp1_clusters_breakdown.R`	Age band × gender breakdown of the types	Results (supplementary analyses)
S21	`README.md`	Description of the code package	—

Scripts listed without a directory are in `src/runners/`.

Supplementary Note 2: Reproducibility notes

- The random seed is fixed at 20260607 in every script that uses random numbers.
- `stm()` is initialized with the spectral method, which is deterministic, so the estimated topics are reproducible. `estimateEffect()` draws from the approximate posterior of the topic proportions, so p values can differ slightly across package versions.
- Cluster numbers from k-means are arbitrary; clusters are identified by their dominant topics.
- Japanese string literals that remain in the code (UniDic part-of-speech tags, the stop-lemma list, lemma lists and lexicon labels) must match the data and are therefore not translated. English glosses are given in the comments.
- `_run_findthoughts.R` prints preprocessed essay content; its output is for the research team only.

Supplementary Note 3: Source code

Listing S1. `run_all.sh`

```sh
#!/usr/bin/env bash
# ======================================================================
# Runs the complete analysis in the order used for the manuscript.
# Run from the root of this code package. Requires the (non-public)
# essay data in data/*.mecab and the sentiment lexicon in data/lexicon/.
#   PYTHON  : Python 3 interpreter   (default: python3)
#   RSCRIPT : Rscript of R >= 4.6    (default: Rscript)
# Console output of each step is written to output/logs/<script>.log
# ======================================================================
set -euo pipefail
PYTHON="${PYTHON:-python3}"
RSCRIPT="${RSCRIPT:-Rscript}"

mkdir -p output/models output/figures output/logs

run_py() { echo ">> $1"; "$PYTHON" "$@" > "output/logs/$(basename "$1" .py).log" 2>&1; }
run_r()  { echo ">> $1"; "$RSCRIPT" "$1"  > "output/logs/$(basename "$1" .R).log"  2>&1; }

# 1. Preprocessing: data/*.mecab -> output/corpus.csv
run_py src/01_preprocess.py data output/corpus.csv

# 2. Topic model
run_r  src/runners/_run_searchK.R            # K selection (Methods)
run_r  src/runners/_run_fit.R                # K = 10 main model; Tables 2-3
run_r  src/runners/_run_table1.R             # Table 1

# 3. Main results
run_r  src/runners/_run_effect.R             # age x gender interaction; Table 4
run_r  src/runners/fig1.R                    # Figure 1
run_r  src/runners/_run_spread.R             # Table 5
run_r  src/runners/_run_timeref.R            # past/future topic shares by age band
run_r  src/runners/fig2.R                    # Figure 2
run_r  src/runners/_run_ee_timeage.R         # age x temporal reference; Table 6

# 4. Supplementary analyses
run_r  src/runners/_run_findthoughts.R       # representative documents of T6
run_r  src/runners/_run_content_age.R        # content covariate = age band
run_r  src/runners/_run_K20.R                # re-estimation at K = 20
run_py src/runners/_exp2_features.py data output/exp2_features.csv
run_r  src/runners/_exp2_clt.R               # abstractness index
run_py src/runners/_exp2c_sentiment_tohoku.py
run_r  src/runners/_exp2c_sentiment_tohoku_stats.R   # net positivity
run_r  src/runners/_exp1_continuity.R        # within-writer continuity, k-means
run_r  src/runners/_exp1_clusters_breakdown.R

echo "All steps finished. Logs: output/logs/"
```

Listing S2. `src/01_preprocess.py`

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# ======================================================================
# Preprocessing of the child-to-adult essay corpus.
#   Input : a directory of MeCab/UniDic morphological-analysis files
#           (*.mecab), one file per essay
#   Output: corpus.csv (columns: subjid, age_group, age_num, gender,
#           time_ref, n_tokens, text); `text` holds the content-word
#           lemmas of the essay, separated by single half-width spaces
# Usage:
#   python3 01_preprocess.py <input_dir> [output_csv]
#   e.g.  python3 01_preprocess.py data output/corpus.csv
# ======================================================================
import re, glob, os, sys, csv

USAGE = 'Usage: python3 01_preprocess.py <input_dir> [output_csv]'

# -- Parts of speech kept (content words) ---------------------------------
#    UniDic POS labels: noun, verb, adjective, adverb.
```

```
#    These Japanese literals must match the UniDic tag set and are not translated.
CONTENT_POS = {'名詞', '動詞', '形容詞', '副詞'}

# -- Stop lemmas: frequent, low-information function-like lemmas -----------
#    Japanese literals must match UniDic lemma forms; English glosses in comments.
STOP_LEMMA = {
    # light / auxiliary-like verbs: do, be (exist), be (animate), become,
    # can, do (colloq.), come, give (to me)
    '為る', '有る', '居る', '成る', '出来る', '遣る', '来る', '呉れる',
    # general verbs: say, think
    '言う', '思う',
    # formal / generic nouns: thing (abstract), thing (concrete), manner,
    # direction/side, time, place, purpose, around (time)
    '事', '物', '様', '方', '時', '所', '為', '頃',
    # demonstratives: this, that, that (over there), here, there
    '此れ', '其れ', '彼れ', '此処', '其処',
    # generic person words: oneself, everyone
    '自分', '皆',
    # degree adverb and generic evaluation: very, good, not exist / none
    '迚も', '良い', '無い',
    # functional time / position nouns: now, before, after, inside
    '今', '前', '後', '中',
}

def normalize_lemma(lemma: str) -> str:
    # Normalize a UniDic lemma by stripping the foreign-spelling / gloss suffix.
    if '-' in lemma:
        # e.g. 'ハワイ-Hawaii' -> 'ハワイ', '私-代名詞' (pronoun gloss) -> '私'
        lemma = re.split(r'-', lemma, maxsplit=1)[0]
    return lemma

def extract_tokens(path: str):
    # Return the list of content-word lemmas in one .mecab file.
    toks = []
    with open(path, encoding='utf-8') as f:
        for line in f:
            line = line.rstrip('\n')
            if line in ('EOS', '') or '\t' not in line:
                continue
            surface, feat = line.split('\t', 1)
            cols = feat.split(',')
            pos = cols[0]
            if pos not in CONTENT_POS:
                continue
            if len(cols) > 1 and cols[1] == '数詞':   # drop numerals ('数詞') at the POS level
                continue
            # UniDic: field 8 (index 7) is the lemma; fall back to the surface form.
            lemma = cols[7] if (len(cols) > 7 and cols[7] != '*') else surface
            lemma = normalize_lemma(lemma)
            if lemma and lemma != '*' and lemma not in STOP_LEMMA:
                toks.append(lemma)
    return toks

def parse_filename(fn: str):
    # Derive writer attributes from the file name (children and adults differ).
    # Elementary school children: <number>(f|m)_(f|p).mecab, e.g. 1f_f.mecab, 92m_p.mecab
    #   (f|m) = female/male; trailing (f|p) = future/past essay
    m = re.match(r'^(\d+)([fm])_([fp])\.mecab$', fn)
    if m:
        num, sex, tense = m.groups()
        return dict(subjid=num + sex, age_group='elementary', age_num=10,
                    gender=('female' if sex == 'f' else 'male'),
                    time_ref=('future' if tense == 'f' else 'past'))
    # Adults: <number>_(f|p).mecab, e.g. 1_f.mecab, 550_p.mecab
    m = re.match(r'^(\d+)_([fp])\.mecab$', fn)
    if m:
        idn, tense = int(m.group(1)), m.group(2)
        # ID range -> (gender, age band, representative age value).
        # Representative ages are equally spaced: 10 / 30 / 50 / 70.
        table = [(100, 'female', '20_30', 30), (200, 'female', '40_50', 50),
                 (300, 'female', '60_80', 70), (400, 'male',   '20_30', 30),
                 (500, 'male',   '40_50', 50), (600, 'male',   '60_80', 70)]
        for hi, g, a, n in table:
            if idn <= hi:
                return dict(subjid=str(idn), age_group=a, age_num=n, gender=g,
```

```
                            time_ref=('future' if tense == 'f' else 'past'))
    return None  # file name does not follow the naming convention

def main():
    if len(sys.argv) < 2:
        print(USAGE); sys.exit(1)
    in_dir = sys.argv[1]
    out_csv = sys.argv[2] if len(sys.argv) > 2 else 'corpus.csv'

    files = sorted(glob.glob(os.path.join(in_dir, '*.mecab')))
    if not files:
        print(f'[!] no .mecab files found in {in_dir}'); sys.exit(1)

    rows, skipped = [], []
    for p in files:
        fn = os.path.basename(p)
        meta = parse_filename(fn)
        if meta is None:
            skipped.append(fn); continue
        toks = extract_tokens(p)
        meta['n_tokens'] = len(toks)
        meta['text'] = ' '.join(toks)
        rows.append(meta)

    # write CSV
    cols = ['subjid', 'age_group', 'age_num', 'gender', 'time_ref', 'n_tokens', 'text']
    with open(out_csv, 'w', encoding='utf-8', newline='') as f:
        w = csv.DictWriter(f, fieldnames=cols)
        w.writeheader()
        for r in rows:
            w.writerow(r)

    # summary
    print(f'Processed: {len(rows)} documents / output: {out_csv}')
    if skipped:
        print(f'Skipped (name does not match convention): {len(skipped)} -> {skipped[:10]}')
    # documents per cell (age band x gender x temporal reference)
    from collections import Counter
    cells = Counter((r['age_group'], r['gender'], r['time_ref']) for r in rows)
    print('\n--- Documents per cell (age_group, gender, time_ref) ---')
    for k in sorted(cells):
        print(f'  {k}: {cells[k]}')
    # token-count distribution (check for empty documents)
    ntok = [r['n_tokens'] for r in rows]
    if ntok:
        ntok_sorted = sorted(ntok)
        print(f'\nTokens per document: min={min(ntok)}, median={ntok_sorted[len(ntok)//2]}, max={max(ntok)}')
        empties = [r['subjid'] for r in rows if r['n_tokens'] == 0]
        if empties:
            print(f'[!] empty documents (0 tokens): {len(empties)} -> {empties[:10]}')

if __name__ == '__main__':
    main()
```

Listing S3. `src/runners/_run_searchK.R`

```
# ======================================================================
# K selection (searchK): held-out likelihood, residuals, semantic
#   coherence and exclusivity for K = 5, 10, 15, 20.
# The content covariate is omitted here so that exclusivity can be
#   computed; content = ~ gender is used only in the final stm() fit.
# Output: output/models/searchK_result.rds,
#         output/figures/searchK_diagnostics.pdf
# ======================================================================
library(stm)
library(dplyr)
library(readr)

set.seed(20260607)

df <- read_csv("output/corpus.csv", show_col_types = FALSE) %>%
  filter(n_tokens >= 5)                     # drop very short essays (< 5 content words)

```

```r
df <- df %>%
  mutate(
    gender    = factor(gender,   levels = c("female", "male")),
    time_ref  = factor(time_ref, levels = c("past", "future")),
    age_group = factor(age_group,
                       levels = c("elementary", "20_30", "40_50", "60_80"),
                       ordered = TRUE)
    # age_num is the representative age (10/30/50/70), used in the spline s(age_num)
  )

cat("Loaded:", nrow(df), "documents\n")

# The text is already segmented by MeCab and reduced to content-word lemmas,
# so all English-oriented text processing is switched off.
processed <- textProcessor(
  documents          = df$text,
  metadata           = df,
  onlycharacter      = FALSE,
  removestopwords    = FALSE,   # stop lemmas were removed in 01_preprocess.py
  removenumbers      = FALSE,
  removepunctuation  = FALSE,   # text contains content words only
  stem               = FALSE,   # English stemming does not apply to Japanese
  wordLengths        = c(1, Inf),
  language           = "na"
)

# Remove rare words that occur in fewer than 5 documents.
out <- prepDocuments(processed$documents, processed$vocab,
                     processed$meta, lower.thresh = 5)

cat("Vocabulary:", length(out$vocab),
    " / documents:", length(out$documents), "\n")

ksearch <- searchK(
  out$documents, out$vocab,
  K          = c(5, 10, 15, 20),
  prevalence = ~ s(age_num) * gender + time_ref,
  data       = out$meta,
  init.type  = "Spectral",
  proportion = 0.5,                          # held-out proportion
  N          = floor(0.1 * length(out$documents))
)

pdf("output/figures/searchK_diagnostics.pdf", width = 9, height = 7)
plot(ksearch)
dev.off()

saveRDS(ksearch, "output/models/searchK_result.rds")
cat("\n=== searchK results ===\n")
res <- as.data.frame(ksearch$results)
res <- res[order(unlist(res$K)), ]
print(res)
cat("\nDONE_SEARCHK\n")
```

Listing S4. `src/runners/_run_fit.R`

```r
# ======================================================================
# Main model: STM with K = 10 (chosen from searchK).
#   prevalence = ~ s(age_num) * gender + time_ref
#   content    = ~ gender
# Prints labelTopics (source of Table 2 and Table 3).
# Output: output/models/stm_fit_K10.rds (used by all downstream scripts)
# ======================================================================
library(stm)
library(dplyr)
library(readr)

set.seed(20260607)

df <- read_csv("output/corpus.csv", show_col_types = FALSE) %>%
  filter(n_tokens >= 5)                      # drop very short essays (< 5 content words)

df <- df %>%
```

```r
  mutate(
    gender    = factor(gender,   levels = c("female", "male")),
    time_ref  = factor(time_ref, levels = c("past", "future")),
    age_group = factor(age_group,
                       levels = c("elementary", "20_30", "40_50", "60_80"),
                       ordered = TRUE)
    # age_num is the representative age (10/30/50/70), used in the spline s(age_num)
  )

# The text is already segmented by MeCab and reduced to content-word lemmas,
# so all English-oriented text processing is switched off.
processed <- textProcessor(
  documents          = df$text,
  metadata           = df,
  onlycharacter      = FALSE,
  removestopwords    = FALSE,   # stop lemmas were removed in 01_preprocess.py
  removenumbers      = FALSE,
  removepunctuation  = FALSE,   # text contains content words only
  stem               = FALSE,   # English stemming does not apply to Japanese
  wordLengths        = c(1, Inf),
  language           = "na"
)

# Remove rare words that occur in fewer than 5 documents.
out <- prepDocuments(processed$documents, processed$vocab,
                     processed$meta, lower.thresh = 5)

cat("Vocabulary:", length(out$vocab),
    " / documents:", length(out$documents), "\n")

K_CHOSEN <- 10

fit <- stm(
  documents  = out$documents,
  vocab      = out$vocab,
  K          = K_CHOSEN,
  prevalence = ~ s(age_num) * gender + time_ref,
  content    = ~ gender,        # within-topic lexical differences by gender
  data       = out$meta,
  init.type  = "Spectral",      # deterministic initialization
  max.em.its = 100,
  verbose    = TRUE
)

saveRDS(list(fit = fit, out = out, K = K_CHOSEN), "output/models/stm_fit_K10.rds")

# Topic words; with a content covariate, gender-specific words are also shown.
cat("\n=== labelTopics (n=12) ===\n")
print(labelTopics(fit, n = 12))
cat("\nDONE_FIT\n")
```

Listing S5. `src/runners/_run_table1.R`

```r
# ======================================================================
# Table 1 (corpus composition).
#   Upper block: writers at collection (subjid units in output/corpus.csv),
#                by age band x gender.
#   Lower block: essays in the final analysis set (after the short-essay
#                filter n_tokens >= 5 and prepDocuments), taken from the
#                metadata of the saved K = 10 model.
# ======================================================================
library(stm)
ags <- c("elementary", "20_30", "40_50", "60_80")

df <- read.csv("output/corpus.csv", fileEncoding = "UTF-8")
writers <- unique(df[, c("subjid", "age_group", "gender")])
cat("=== Writers at collection (age band x gender) ===\n")
tab_w <- table(factor(writers$gender, levels = c("female", "male")),
               factor(writers$age_group, levels = ags))
print(addmargins(tab_w))

meta <- readRDS("output/models/stm_fit_K10.rds")$out$meta
cat("\n=== Final analysis set: essays (gender x time_ref x age band) ===\n")
```

```r
tab_e <- table(factor(meta$gender, levels = c("female", "male")),
               factor(meta$time_ref, levels = c("past", "future")),
               factor(meta$age_group, levels = ags))
print(ftable(tab_e, row.vars = c(1, 2)))
cat("\nTotal essays per age band:\n")
print(table(factor(meta$age_group, levels = ags)))
cat("Total essays:", nrow(meta), "\n")
```

Listing S6. `src/runners/_run_effect.R`

```r
# ======================================================================
# Covariate-effect estimation (estimateEffect) for age x gender.
#   (A) Spline fit, prevalence ~ s(age_num) * gender + time_ref:
#       age trends by gender for every topic (diagnostic figure).
#   (B) Linear fit, ~ age_num * gender + time_ref: estimate and p value
#       of the age x gender (male) interaction per topic (Table 4).
# Output: output/figures/age_trend_by_gender.pdf
# ======================================================================
library(stm)
library(dplyr)
library(readr)

set.seed(20260607)

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out; K <- obj$K

# (A) visualization: spline smoothing (same formula as the prevalence model)
prep <- estimateEffect(1:K ~ s(age_num) * gender + time_ref,
                       fit, metadata = out$meta, uncertainty = "Global")

pdf("output/figures/age_trend_by_gender.pdf", width = 9, height = 6)
for (k in 1:K) {
  plot(prep, "age_num", method = "continuous", topics = k,
       moderator = "gender", moderator.value = "female",
       printlegend = FALSE, linecol = "red",
       main = paste("Topic", k), xlab = "age (continuous)",
       ylab = "expected topic proportion")
  plot(prep, "age_num", method = "continuous", topics = k,
       moderator = "gender", moderator.value = "male",
       add = TRUE, linecol = "blue")
  legend("topright", c("female", "male"), lwd = 2, col = c("red", "blue"))
}
dev.off()

# (B) significance test: age_num entered linearly; p value of age_num:gendermale
prep_lin <- estimateEffect(1:K ~ age_num * gender + time_ref,
                           fit, metadata = out$meta, uncertainty = "Global")
s <- summary(prep_lin)

cat("\n=== age_num:gendermale interaction (by topic) ===\n")
res <- data.frame()
for (k in 1:K) {
  tab <- s$tables[[k]]
  ix <- grep("age_num:gender", rownames(tab))
  if (length(ix) == 1) {
    est <- tab[ix, 1]; se <- tab[ix, 2]; p <- tab[ix, 4]
    sig <- ifelse(p < .001, "***", ifelse(p < .01, "**",
           ifelse(p < .05, "*", ifelse(p < .1, ".", ""))))
    res <- rbind(res, data.frame(topic = k,
            estimate = round(est, 5), std_err = round(se, 5),
            p_value = signif(p, 3), sig = sig))
  }
}
print(res, row.names = FALSE)
cat("\nDONE_EFFECT\n")
```

Listing S7. `src/runners/fig1.R`

```r
# ======================================================================
```

```r
# Figure 1: Age trends by gender for the 5 topics with a
#   significant age x gender interaction.
# Data source: STM model reproduced from output/models/stm_fit_K10.rds
#   (estimateEffect, prevalence ~ s(age_num)*gender + time_ref, same as
#   src/runners/_run_effect.R). Significant topics (T1,T2,T3,T5,T7; Table 4)
#   and labels as in Table 2.
# Output: output/figures/fig1_agetrend.pdf  (vector, fonts embedded)
# Run from project root:  Rscript src/runners/fig1.R
# ======================================================================
suppressMessages(library(stm))
set.seed(20260607)

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out; K <- obj$K

prep <- estimateEffect(1:K ~ s(age_num) * gender + time_ref, fit,
                       metadata = out$meta, uncertainty = "Global")

# Topics with significant age x gender interaction and their Table-2 labels
sig_topics <- c(1, 2, 3, 5, 7)
labels <- c("1" = "T1  Theme parks / excursions",
            "2" = "T2  Work / research",
            "3" = "T3  School events",
            "5" = "T5  Club sports",
            "7" = "T7  Marriage / grandchildren / family")

# Colour-blind-safe pair (Okabe-Ito); distinguished also by line type
col_f <- "#D55E00"; col_m <- "#0072B2"
fill_f <- adjustcolor(col_f, alpha.f = 0.18)
fill_m <- adjustcolor(col_m, alpha.f = 0.18)

# Capture continuous predictions (means + 95% CI) for both genders ------
get_pred <- function(topic, gval) {
  tf <- tempfile(fileext = ".pdf"); pdf(tf)
  r <- plot(prep, "age_num", method = "continuous", topics = topic,
            moderator = "gender", moderator.value = gval)
  dev.off(); unlink(tf)
  list(x = r$x, mean = r$means[[1]],
       lo = r$ci[[1]][1, ], hi = r$ci[[1]][2, ])
}
preds <- lapply(sig_topics, function(k)
  list(f = get_pred(k, "female"), m = get_pred(k, "male")))
names(preds) <- as.character(sig_topics)

# --- draw -------------------------------------------------------------
pdf("output/figures/fig1_agetrend.pdf", width = 6.7, height = 4.3, pointsize = 10)
op <- par(mfrow = c(2, 3), mar = c(3.6, 3.8, 2.2, 0.8),
          mgp = c(2.2, 0.7, 0), tcl = -0.3, las = 1)

for (k in sig_topics) {
  p <- preds[[as.character(k)]]
  yr <- range(c(p$f$lo, p$f$hi, p$m$lo, p$m$hi))
  plot(NA, xlim = range(p$f$x), ylim = yr,
       xlab = "Age (years)", ylab = "Expected topic proportion",
       main = labels[as.character(k)], cex.main = 0.92, font.main = 1)
  polygon(c(p$f$x, rev(p$f$x)), c(p$f$lo, rev(p$f$hi)),
          col = fill_f, border = NA)
  polygon(c(p$m$x, rev(p$m$x)), c(p$m$lo, rev(p$m$hi)),
          col = fill_m, border = NA)
  lines(p$f$x, p$f$mean, col = col_f, lwd = 2.2, lty = 1)
  lines(p$m$x, p$m$mean, col = col_m, lwd = 2.2, lty = 2)
}

# legend panel (6th cell)
plot.new()
legend("center", legend = c("Female", "Male"),
       col = c(col_f, col_m), lwd = 2.2, lty = c(1, 2),
       seg.len = 2.6, bty = "n", cex = 1.05,
       title = "Gender (shaded band = 95% CI)")
par(op); dev.off()
cat("WROTE output/figures/fig1_agetrend.pdf\n")
```

Listing S8. `src/runners/_run_spread.R`

```r
# ======================================================================
# Concentration vs dispersion of the mean topic mixture by age cohort and
#   temporal reference (Table 5): share of the top three topics and the
#   effective number of topics, exp(Shannon entropy of the mean theta).
# ======================================================================
library(stm)

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out
meta  <- out$meta
theta <- fit$theta
K <- ncol(theta)

ag <- as.character(meta$age_group)
tr <- as.character(meta$time_ref)
cohorts <- c("elementary", "20_30", "40_50", "60_80")

eff_topics <- function(p) {           # effective number of topics = exp(Shannon entropy)
  p <- p[p > 0]; exp(-sum(p * log(p)))
}

cat(sprintf("%-12s %-7s %5s  %-22s %-10s %s\n",
            "cohort", "time", "n", "top3 topics(share)", "top3sum", "eff.K"))
for (g in cohorts) {
  for (tt in c("past", "future")) {
    idx <- ag == g & tr == tt
    m <- colMeans(theta[idx, , drop = FALSE])
    o <- order(m, decreasing = TRUE)[1:3]
    top3 <- paste(sprintf("T%d(%.0f%%)", o, 100 * m[o]), collapse = " ")
    cat(sprintf("%-12s %-7s %5d  %-22s %6.0f%%   %.2f\n",
                g, tt, sum(idx), top3, 100 * sum(m[o]), eff_topics(m)))
  }
}
cat("\nDONE_SPREAD\n")
```

Listing S9. `src/runners/_run_timeref.R`

```r
# ======================================================================
# Mean topic proportions (theta) of past vs future essays within each
#   age band, and their difference (future - past). Source of the
#   band-wise topic shares quoted in the Results.
# ======================================================================
library(stm)

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out; K <- obj$K
theta <- fit$theta
meta  <- out$meta

ag <- as.character(meta$age_group)
tr <- as.character(meta$time_ref)
ags <- c("elementary", "20_30", "40_50", "60_80")

for (g in ags) {
  pst <- ag == g & tr == "past"
  fut <- ag == g & tr == "future"
  mp <- colMeans(theta[pst, , drop = FALSE])
  mf <- colMeans(theta[fut, , drop = FALSE])
  tab <- data.frame(topic = 1:K,
                    past   = round(mp, 3),
                    future = round(mf, 3),
                    diff_fp = round(mf - mp, 3))
  cat(sprintf("\n==== %s  (past n=%d, future n=%d) ====\n", g, sum(pst), sum(fut)))
  print(tab[order(-abs(tab$diff_fp)), ], row.names = FALSE)
}
cat("\nDONE_TIMEREF\n")
```

Listing S10. `src/runners/fig2.R`

```r
# ======================================================================
# Figure 2: Average topic composition of past vs future
#   essays for each age cohort (100% stacked bars).
# Data source: STM model reproduced from output/models/stm_fit_K10.rds.
#   theta (document x topic) averaged within age_group x time_ref.
#   Topic labels as in Table 2.
# Output: output/figures/fig2_pastfuture.pdf  (vector, fonts embedded)
# Run from project root:  Rscript src/runners/fig2.R
# ======================================================================
suppressMessages(library(stm))

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out; K <- obj$K
theta <- fit$theta
meta  <- out$meta

ag_lv  <- c("elementary", "20_30", "40_50", "60_80")
ag_lab <- c("Elementary", "20-30s", "40-50s", "60-80s")
tr_lv  <- c("past", "future")

# mean theta per (age_group, time_ref): 10 x 8 matrix (rows = topics)
cells <- expand.grid(time = tr_lv, age = ag_lv, stringsAsFactors = FALSE)
M <- sapply(seq_len(nrow(cells)), function(i) {
  sel <- meta$age_group == cells$age[i] & meta$time_ref == cells$time[i]
  colMeans(theta[sel, , drop = FALSE])
})                       # 10 (topics) x 8 (cells)
M <- M / rep(colSums(M), each = K)   # renormalise to 100%

topic_lab <- c("T1 Theme parks/excursions", "T2 Work/research",
               "T3 School events", "T4 Outdoor/family leisure",
               "T5 Club sports", "T6 Child-rearing/future planning",
               "T7 Marriage/grandchildren/family", "T8 Children's play",
               "T9 Sightseeing travel", "T10 Study abroad/language")

# 10-class colour-blind-safe palette (Okabe-Ito + Tol)
topcols <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
             "#D55E00", "#CC79A7", "#999999", "#44AA99", "#882255")
# secondary cue for B/W: distinct hatch angle per topic (dark grey lines)
angles <- c(0, 45, 135, 90, 0, 45, 135, 90, 30, 150)
hatchcol <- "grey20"

# bar positions: pairs (past,future) grouped per cohort
spc <- c(0.4, 0.15, 1.0, 0.15, 1.0, 0.15, 1.0, 0.15)

pdf("output/figures/fig2_pastfuture.pdf", width = 6.7, height = 4.4, pointsize = 10)
op <- par(mar = c(3.0, 3.8, 0.8, 0.6), mgp = c(2.4, 0.7, 0),
          tcl = -0.3, las = 1, xpd = NA)
layout(matrix(c(1, 2), nrow = 1), widths = c(3.05, 1.95))

# --- panel 1: bars ---
bp <- barplot(M, col = topcols, border = "grey25", space = spc,
              names.arg = rep("", ncol(M)), axes = FALSE,
              ylim = c(0, 1))
# overlay sparse hatching (explicit dark line colour, distinct angle/topic)
barplot(M, col = hatchcol, border = "grey25", space = spc, density = 16,
        angle = angles, names.arg = rep("", ncol(M)), axes = FALSE,
        add = TRUE)
axis(2, at = seq(0, 1, 0.2), labels = paste0(seq(0, 100, 20), "%"))
mtext("Topic proportion", side = 2, line = 2.6, las = 0, cex = 1.0)
# P / F labels under each bar
mtext(rep(c("P", "F"), 4), side = 1, line = 0.1, at = bp, cex = 0.85)
# cohort labels centred under each pair
pair_mid <- colMeans(matrix(bp, nrow = 2))
mtext(ag_lab, side = 1, line = 1.2, at = pair_mid, cex = 0.95)

# --- panel 2: legend ---
par(mar = c(0, 0, 0, 0)); plot.new()
legend("left", legend = topic_lab, fill = topcols, border = "grey25",
       bty = "n", cex = 0.82, y.intersp = 1.25,
       title = "Topic (P = past, F = future)", title.adj = 0)
par(op); dev.off()
cat("WROTE output/figures/fig2_pastfuture.pdf\n")
```

Listing S11. src/runners/_run_ee_timeage.R

```r
# ======================================================================
# Covariate-effect estimation for age x temporal reference (Table 6),
#   using the saved K = 10 model (no refitting).
# Question: within future essays, do T6 (future planning) and T7
#   (marriage/family) increase and T2 (work/research) decrease with age,
#   and do these slopes differ from those in past essays?
# Method: fit ~ age_num * time_ref twice, with past and with future as
#   the reference level; the age_num coefficient under each reference is
#   the within-condition age slope, and the interaction term is the
#   past/future difference in slope.
# ======================================================================
library(stm)

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out
meta <- out$meta
topics_test <- c(2, 6, 7)            # T2 work / T6 future planning / T7 marriage & family

star <- function(p) ifelse(p < .001, "***", ifelse(p < .01, "**",
                    ifelse(p < .05, "*", ifelse(p < .1, ".", ""))))

# Run estimateEffect with the given reference level; return coefficient tables
run_ee <- function(base) {
  m <- meta
  m$time_ref <- relevel(factor(as.character(m$time_ref),
                               levels = c("past", "future")), ref = base)
  set.seed(20260607)
  prep <- estimateEffect(topics_test ~ age_num * time_ref, fit,
                         metadata = m, uncertainty = "Global")
  summary(prep)$tables
}

tab_past <- run_ee("past")     # age_num row = slope within past; includes interaction row
tab_fut  <- run_ee("future")   # age_num row = slope within future

getrow <- function(tab, pat) {
  ix <- grep(pat, rownames(tab))
  if (length(ix) != 1) return(c(NA, NA, NA))
  c(est = tab[ix, 1], se = tab[ix, 2], p = tab[ix, 4])
}

cat(sprintf("%-26s %12s %12s %10s\n",
            "topic / term", "est(x10yr)", "SE(x10yr)", "p"))
cat(strrep("-", 64), "\n")
for (i in seq_along(topics_test)) {
  k <- topics_test[i]
  ps <- getrow(tab_past[[i]], "^age_num$")
  fs <- getrow(tab_fut[[i]],  "^age_num$")
  ia <- getrow(tab_past[[i]], "age_num:time_reffuture")
  cat(sprintf("== T%d ==\n", k))
  cat(sprintf("%-26s %12.4f %12.4f %8.3g %s\n",
              "  age slope within past", ps[1]*10, ps[2]*10, ps[3], star(ps[3])))
  cat(sprintf("%-26s %12.4f %12.4f %8.3g %s\n",
              "  age slope within future", fs[1]*10, fs[2]*10, fs[3], star(fs[3])))
  cat(sprintf("%-26s %12.4f %12.4f %8.3g %s\n",
              "  interaction (fut - past)", ia[1]*10, ia[2]*10, ia[3], star(ia[3])))
}
cat("\n(Estimates: change in theta per +10 years of age. ***p<.001 **p<.01 *p<.05 .p<.1)\n")
cat("DONE_EE_TIMEAGE\n")
```

Listing S12. src/runners/_run_findthoughts.R

```r
# ======================================================================
# Representative documents of T6 (future planning) among future essays,
#   by age cohort, to inspect the age-related shift in its referents.
# out$meta is aligned with out$documents; meta$text holds the
#   preprocessed content-word string of each essay.
# NOTE: the output contains (preprocessed) essay content and must not be
#   shared outside the research team.
# ======================================================================
library(stm)
```

```r

obj <- readRDS("output/models/stm_fit_K10.rds")
fit <- obj$fit; out <- obj$out
meta <- out$meta
texts <- as.character(meta$text)

K_T6   <- 6
theta6 <- fit$theta[, K_T6]
ag <- as.character(meta$age_group)
tr <- as.character(meta$time_ref)

cohorts <- c("20_30", "40_50", "60_80")
N_TOP   <- 5

for (g in cohorts) {
  idx <- which(ag == g & tr == "future")
  ord <- idx[order(theta6[idx], decreasing = TRUE)]
  top <- head(ord, N_TOP)
  cat(sprintf("\n========== %s / future / top %d T6 documents (n=%d) ==========\n",
              g, length(top), length(idx)))
  for (i in top) {
    cat(sprintf("[theta=%.3f] %s\n", theta6[i], texts[i]))
  }
}
cat("\nDONE_FINDTHOUGHTS\n")
```

Listing S13. `src/runners/_run_content_age.R`

```r
# ======================================================================
# Auxiliary model: content covariate = age band instead of gender
#   (K = 10, prevalence unchanged). Shows how the word distribution
#   (beta) of the same topic differs across age bands (sageLabels).
# Output: output/models/stm_fit_K10_contentAge.rds
# ======================================================================
library(stm)
library(dplyr)
library(readr)
set.seed(20260607)

df <- read_csv("output/corpus.csv", show_col_types = FALSE) %>% filter(n_tokens >= 5)
df <- df %>% mutate(
  gender    = factor(gender,   levels = c("female", "male")),
  time_ref  = factor(time_ref, levels = c("past", "future")),
  # unordered factor for content, so that a label is obtained for each level
  age_group = factor(age_group, levels = c("elementary","20_30","40_50","60_80"))
)

processed <- textProcessor(df$text, metadata = df, onlycharacter = FALSE,
                           removestopwords = FALSE, removenumbers = FALSE,
                           removepunctuation = FALSE, stem = FALSE,
                           wordLengths = c(1, Inf), language = "na")
out <- prepDocuments(processed$documents, processed$vocab, processed$meta, lower.thresh = 5)
cat("Vocabulary:", length(out$vocab), " / documents:", length(out$documents), "\n")

fit <- stm(documents = out$documents, vocab = out$vocab, K = 10,
           prevalence = ~ s(age_num) * gender + time_ref,
           content    = ~ age_group,
           data = out$meta, init.type = "Spectral", max.em.its = 100, verbose = TRUE)

saveRDS(list(fit = fit, out = out, K = 10), "output/models/stm_fit_K10_contentAge.rds")

cat("\n=== labelTopics (content = age band) ===\n")
print(labelTopics(fit, n = 10))

cat("\n=== sageLabels: characteristic words per age band (covariate-topic) ===\n")
sl <- sageLabels(fit, n = 10)
print(sl)

cat("\nDONE_CONTENT_AGE\n")
```

Listing S14. `src/runners/_run_K20.R`

```r
# ======================================================================
# Re-estimation at K = 20: does the future-planning topic (T6 at K = 10)
#   separate into distinct themes at higher resolution?
# Data preparation and stm() arguments are identical to _run_fit.R;
#   only K is changed to 20.
# Criterion: correlation between theta[, T6] of the K = 10 model and each
#   theta column of the K = 20 model (which K = 20 topics inherit T6).
# Output: output/models/stm_fit_K20.rds
# ======================================================================
library(stm)
library(dplyr)
library(readr)

set.seed(20260607)

df <- read_csv("output/corpus.csv", show_col_types = FALSE) %>%
  filter(n_tokens >= 5)                      # drop very short essays (< 5 content words)

df <- df %>%
  mutate(
    gender    = factor(gender,   levels = c("female", "male")),
    time_ref  = factor(time_ref, levels = c("past", "future")),
    age_group = factor(age_group,
                       levels = c("elementary", "20_30", "40_50", "60_80"),
                       ordered = TRUE)
    # age_num is the representative age (10/30/50/70), used in the spline s(age_num)
  )

# The text is already segmented by MeCab and reduced to content-word lemmas,
# so all English-oriented text processing is switched off.
processed <- textProcessor(
  documents          = df$text,
  metadata           = df,
  onlycharacter      = FALSE,
  removestopwords    = FALSE,   # stop lemmas were removed in 01_preprocess.py
  removenumbers      = FALSE,
  removepunctuation  = FALSE,   # text contains content words only
  stem               = FALSE,   # English stemming does not apply to Japanese
  wordLengths        = c(1, Inf),
  language           = "na"
)

# Remove rare words that occur in fewer than 5 documents.
out <- prepDocuments(processed$documents, processed$vocab,
                     processed$meta, lower.thresh = 5)

cat("Vocabulary:", length(out$vocab),
    " / documents:", length(out$documents), "\n")

K_HIGH <- 20

fit20 <- stm(
  documents  = out$documents,
  vocab      = out$vocab,
  K          = K_HIGH,
  prevalence = ~ s(age_num) * gender + time_ref,
  content    = ~ gender,
  data       = out$meta,
  init.type  = "Spectral",
  max.em.its = 100,
  verbose    = FALSE
)

saveRDS(list(fit = fit20, out = out, K = K_HIGH), "output/models/stm_fit_K20.rds")

# Align with the K = 10 model (same 'out', hence identical document order)
obj10 <- readRDS("output/models/stm_fit_K10.rds")
stopifnot(nrow(obj10$fit$theta) == nrow(fit20$theta))
T6 <- 6
r <- cor(obj10$fit$theta[, T6], fit20$theta)[1, ]
ord <- order(r, decreasing = TRUE)
cat("\n=== Correlation of K=10 T6 with each K=20 topic (descending) ===\n")
print(data.frame(topic = paste0("K20-T", ord), r = round(r[ord], 3)))
```

```

lt <- labelTopics(fit20, topics = ord[1:2], n = 10)
cat("\n=== Words of the two most correlated K=20 topics ===\n")
print(lt)
cat("\nDONE_K20\n")
```

Listing S15. src/runners/_exp2_features.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# ======================================================================
# Supplementary analysis: per-essay features of temporal narration,
#   computed from the .mecab files.
# Output: output/exp2_features.csv
#   columns: subjid, age_group, age_num, gender, time_ref, n_content,
#            n_noun, n_abstract_noun, n_concrete_noun, abstract_noun_ratio,
#            n_near, n_far, far_ratio
# Uses the same content-word filter as 01_preprocess.py (stop lemmas and
#   numerals removed) and classifies nouns as abstract or concrete from
#   the UniDic noun subcategories.
# Usage:
#   python3 _exp2_features.py [input_dir=data] [output_csv=output/exp2_features.csv]
# ======================================================================
import re, glob, os, sys, csv

# Japanese literals below must match UniDic tags / lemma forms and are not translated.
CONTENT_POS = {'名詞', '動詞', '形容詞', '副詞'}   # noun, verb, adjective, adverb
STOP_LEMMA = {                                    # identical to 01_preprocess.py
    '為る','有る','居る','成る','出来る','遣る','来る','呉れる','言う','思う',
    '事','物','様','方','時','所','為','頃','此れ','其れ','彼れ','此処','其処',
    '自分','皆','迚も','良い','無い','今','前','後','中',
}
# UniDic noun subcategory 2 (cols[2]) used to classify nouns:
#   abstract-leaning: verbal-noun capable, verbal/adjectival capable,
#                     adjectival capable, adverbial capable
ABSTRACT_NOUN_SUB = {'サ変可能','サ変形状詞可能','形状詞可能','副詞可能'}
#   concrete-leaning: general, counter-capable
CONCRETE_NOUN_SUB = {'一般','助数詞可能'}
#   proper nouns ('固有名詞' in cols[1]) are also counted as concrete

# Lemmas of temporal distance (matched on the lemma, cols[7])
#   near: tomorrow, day after tomorrow, next year, year after next, next month,
#         next week, next time, soon, in a few days, this week, a few days, imminent
NEAR_LEMMA = {'明日','明後日','来年','再来年','来月','来週','今度','近々','近日','今週','数日','間近'}
#   far: old age, rest of life, later years, retirement age, retirement, pension,
#        lifetime, whole life, end-of-life planning, death, die, grave, someday,
#        eternity, decades, future, future (distant), grow old, elderly,
#        life span, longevity
FAR_LEMMA  = {'老後','余生','晩年','定年','退職','年金','生涯','一生','終活','死','死ぬ','墓',
              'いつか','永遠','数十年','将来','未来','老いる','高齢','寿命','長生き'}

def normalize_lemma(lemma: str) -> str:
    if '-' in lemma:
        lemma = re.split(r'-', lemma, maxsplit=1)[0]
    return lemma

def features(path: str):
    n_content = n_noun = n_abs = n_con = n_near = n_far = 0
    with open(path, encoding='utf-8') as f:
        for line in f:
            line = line.rstrip('\n')
            if line in ('EOS','') or '\t' not in line:
                continue
            surface, feat = line.split('\t', 1)
            cols = feat.split(',')
            pos = cols[0]
            if pos not in CONTENT_POS:
                continue
            sub1 = cols[1] if len(cols) > 1 else '*'
            sub2 = cols[2] if len(cols) > 2 else '*'
            if sub1 == '数詞':                      # numeral
                continue
            lemma = cols[7] if (len(cols) > 7 and cols[7] != '*') else surface
```

```
            lemma = normalize_lemma(lemma)
            if not lemma or lemma == '*' or lemma in STOP_LEMMA:
                continue
            n_content += 1
            if pos == '名詞':                        # noun
                n_noun += 1
                if sub2 in ABSTRACT_NOUN_SUB:
                    n_abs += 1
                elif sub1 == '固有名詞' or sub2 in CONCRETE_NOUN_SUB:   # proper noun or concrete
                    n_con += 1
            if lemma in NEAR_LEMMA:
                n_near += 1
            if lemma in FAR_LEMMA:
                n_far += 1
    abs_ratio = (n_abs / (n_abs + n_con)) if (n_abs + n_con) > 0 else ''
    far_ratio = (n_far / n_content) if n_content > 0 else ''
    return dict(n_content=n_content, n_noun=n_noun, n_abstract_noun=n_abs,
                n_concrete_noun=n_con, abstract_noun_ratio=abs_ratio,
                n_near=n_near, n_far=n_far, far_ratio=far_ratio)

def parse_filename(fn: str):
    # identical to 01_preprocess.py
    m = re.match(r'^(\d+)([fm])_([fp])\.mecab$', fn)
    if m:
        num, sex, tense = m.groups()
        return dict(subjid=num+sex, age_group='elementary', age_num=10,
                    gender=('female' if sex=='f' else 'male'),
                    time_ref=('future' if tense=='f' else 'past'))
    m = re.match(r'^(\d+)_([fp])\.mecab$', fn)
    if m:
        idn, tense = int(m.group(1)), m.group(2)
        table = [(100,'female','20_30',30),(200,'female','40_50',50),
                 (300,'female','60_80',70),(400,'male','20_30',30),
                 (500,'male','40_50',50),(600,'male','60_80',70)]
        for hi,g,a,n in table:
            if idn <= hi:
                return dict(subjid=str(idn), age_group=a, age_num=n, gender=g,
                            time_ref=('future' if tense=='f' else 'past'))
    return None

def main():
    in_dir = sys.argv[1] if len(sys.argv) > 1 else 'data'
    out_csv = sys.argv[2] if len(sys.argv) > 2 else 'output/exp2_features.csv'
    files = sorted(glob.glob(os.path.join(in_dir, '*.mecab')))
    rows = []
    for p in files:
        meta = parse_filename(os.path.basename(p))
        if meta is None:
            continue
        meta.update(features(p))
        rows.append(meta)
    cols = ['subjid','age_group','age_num','gender','time_ref','n_content','n_noun',
            'n_abstract_noun','n_concrete_noun','abstract_noun_ratio','n_near','n_far','far_ratio']
    with open(out_csv,'w',encoding='utf-8',newline='') as f:
        w = csv.DictWriter(f, fieldnames=cols); w.writeheader()
        for r in rows: w.writerow(r)
    print(f'Wrote: {len(rows)} documents -> {out_csv}')

if __name__ == '__main__':
    main()
```

Listing S16. `src/runners/_exp2_clt.R`

```
# ======================================================================
# Supplementary analysis: construal level (abstractness) and the temporal
#   reach of future essays. Input: output/exp2_features.csv
#   (2a) abstractness index = abstract / (abstract + concrete) nouns;
#        lm(abstract_noun_ratio ~ age_num * gender + time_ref)
#   (2b) share of far-future lemmas within future essays (exploratory;
#        not reported in the manuscript)
# ======================================================================
options(width = 110)
df <- read.csv("output/exp2_features.csv", fileEncoding = "UTF-8")
```

```r
df <- subset(df, n_content >= 5)
df$gender   <- factor(df$gender,   levels = c("female","male"))
df$time_ref <- factor(df$time_ref, levels = c("past","future"))
df$age_group<- factor(df$age_group, levels = c("elementary","20_30","40_50","60_80"))
cat("N =", nrow(df), "\n")

## ===== 2a Abstractness (abstract-noun ratio) =====
cat("\n========== 2a Abstract-noun ratio ==========\n")
a <- df[!is.na(df$abstract_noun_ratio), ]
cat("\n--- Mean abstract_noun_ratio: age band x temporal reference ---\n")
tab <- tapply(a$abstract_noun_ratio, list(a$age_group, a$time_ref), mean)
print(round(tab, 3))
cat("\nOverall: past =", round(mean(a$abstract_noun_ratio[a$time_ref=="past"]),3),
    " future =", round(mean(a$abstract_noun_ratio[a$time_ref=="future"]),3), "\n")

m_a <- lm(abstract_noun_ratio ~ age_num * gender + time_ref, data = a)
cat("\n--- lm(abstract_noun_ratio ~ age_num*gender + time_ref) ---\n")
print(round(summary(m_a)$coefficients, 5))

## ===== 2b Temporal reach (far-lemma ratio, future essays only) =====
cat("\n========== 2b Temporal reach (far_ratio, future only) ==========\n")
fu <- df[df$time_ref == "future" & !is.na(df$far_ratio), ]
cat("\n--- Mean far_ratio by age band (future) ---\n")
print(round(tapply(fu$far_ratio, fu$age_group, mean), 4))
cat("\n--- Mean near / far counts by age band (future) ---\n")
print(round(tapply(fu$n_near, fu$age_group, mean), 3))
print(round(tapply(fu$n_far,  fu$age_group, mean), 3))

m_b <- lm(far_ratio ~ age_num * gender, data = fu)
cat("\n--- lm(far_ratio ~ age_num*gender), future only ---\n")
print(round(summary(m_b)$coefficients, 6))

## far_ratio in past essays (reference: do far lemmas also appear in recollection?)
pa <- df[df$time_ref == "past" & !is.na(df$far_ratio), ]
cat("\nReference: mean far_ratio  past =", round(mean(pa$far_ratio),4),
    " future =", round(mean(fu$far_ratio),4), "\n")

cat("\nDONE_EXP2_CLT\n")
```

Listing S17. `src/runners/_exp2c_sentiment_tohoku.py`

```python
# -*- coding: utf-8 -*-
# ======================================================================
# Supplementary analysis: net positivity of each essay, scored with the
#   Japanese Sentiment Polarity Dictionary (Inui-Okazaki Laboratory,
#   Tohoku University).
# Lexicon (NOT included; obtain it from the original distributor and
#   place it in data/lexicon/):
#   nouns           : pn.csv.m3.120408.trim (column 2: p = positive / n = negative)
#   verbs/adjectives: wago.121808.pn        (column 1: label starting with
#                                            'ポジ' = positive / 'ネガ' = negative)
# Output: output/exp2c_sentiment_tohoku.csv
#   (pos, neg and net = (pos - neg) / n_tokens for each essay)
# ======================================================================
import csv

def load_lexicon():
    pos, neg = set(), set()
    with open("data/lexicon/pn.csv.m3.120408.trim", encoding="utf-8") as f:
        for line in f:
            p = line.rstrip("\n").split("\t")
            if len(p) < 2:
                continue
            w, lab = p[0].strip().strip('"'), p[1].strip()
            if lab == "p":
                pos.add(w)
            elif lab == "n":
                neg.add(w)
    with open("data/lexicon/wago.121808.pn", encoding="utf-8") as f:
        for line in f:
            p = line.rstrip("\n").split("\t")
            if len(p) < 2:
                continue
```

```
            lab, phrase = p[0], p[1].split(" ")[0]   # for multi-word entries, use the head word
            # The labels in the lexicon file are Japanese: 'ポジ' (positive), 'ネガ' (negative).
            if lab.startswith("ポジ"):
                pos.add(phrase)
            elif lab.startswith("ネガ"):
                neg.add(phrase)
    return pos, neg

POS, NEG = load_lexicon()

rows_out = []
n_in = 0
with open("output/corpus.csv", encoding="utf-8") as f:
    for row in csv.DictReader(f):
        if int(row["n_tokens"]) < 5:        # same short-essay filter as the main analysis
            continue
        n_in += 1
        words = [w for w in row["text"].split(" ") if w]
        pos = sum(1 for w in words if w in POS)
        neg = sum(1 for w in words if w in NEG)
        nt = int(row["n_tokens"])
        rows_out.append({
            "subjid": row["subjid"], "age_group": row["age_group"],
            "age_num": row["age_num"], "gender": row["gender"],
            "time_ref": row["time_ref"], "n_tokens": nt,
            "pos": pos, "neg": neg, "net": (pos - neg) / nt,
        })

with open("output/exp2c_sentiment_tohoku.csv", "w", encoding="utf-8", newline="") as f:
    w = csv.DictWriter(f, fieldnames=list(rows_out[0].keys()))
    w.writeheader()
    w.writerows(rows_out)

tp = sum(r["pos"] for r in rows_out)
tn = sum(r["neg"] for r in rows_out)
print("Lexicon POS=%d NEG=%d" % (len(POS), len(NEG)))
print("N=%d  total pos=%d  total neg=%d" % (len(rows_out), tp, tn))
print("DONE_SCORE")
```

Listing S18. `src/runners/_exp2c_sentiment_tohoku_stats.R`

```
# ======================================================================
# Supplementary analysis: summaries and regression of net positivity
#   (scored by _exp2c_sentiment_tohoku.py).
#   lm(net ~ age_num * time_ref + gender)
# ======================================================================
options(width = 110)
d <- read.csv("output/exp2c_sentiment_tohoku.csv", fileEncoding = "UTF-8")
d$gender    <- factor(d$gender,    levels = c("female", "male"))
d$time_ref  <- factor(d$time_ref,  levels = c("past", "future"))
d$age_group <- factor(d$age_group, levels = c("elementary", "20_30", "40_50", "60_80"))

cat("N =", nrow(d), " total pos =", sum(d$pos), " total neg =", sum(d$neg), "\n")

cat("\n--- Mean net positivity: age band x temporal reference ---\n")
print(round(tapply(d$net, list(d$age_group, d$time_ref), mean), 4))

cat("\n--- Mean count of negative words: age band x temporal reference ---\n")
print(round(tapply(d$neg, list(d$age_group, d$time_ref), mean), 3))

cat("\n--- lm(net ~ age_num * time_ref + gender) ---\n")
m <- lm(net ~ age_num * time_ref + gender, data = d)
print(round(summary(m)$coefficients, 6))

cat("\n--- Overall future net =", round(mean(d$net[d$time_ref=="future"]),4),
    " / overall past net =", round(mean(d$net[d$time_ref=="past"]),4), "\n")
cat("DONE_EXP2C_TOHOKU\n")
```

Listing S19. `src/runners/_exp1_continuity.R`

```r
# ======================================================================
# Supplementary analysis: within-writer continuity of self-narratives,
#   using writers who contributed both a past and a future essay.
#   (1) Transition matrix: dominant past topic -> dominant future topic
#   (2) Continuity score = cosine(theta_past, theta_future);
#       lm(cosine ~ age_num * gender)
#   (3) Narrative types: k-means (k = 4) on [theta_past, theta_future]
# ======================================================================
options(width = 110)
library(stm)
set.seed(20260607)

obj  <- readRDS("output/models/stm_fit_K10.rds")
fit  <- obj$fit; meta <- obj$out$meta; K <- obj$K
theta <- fit$theta
sid <- as.character(meta$subjid); tr <- as.character(meta$time_ref)

# Pair the past and future essays of the same writer
common <- intersect(sid[tr=="past"], sid[tr=="future"])
idxP <- match(common, ifelse(tr=="past", sid, NA))
idxF <- match(common, ifelse(tr=="future", sid, NA))
TP <- theta[idxP, , drop=FALSE]   # theta_past
TF <- theta[idxF, , drop=FALSE]   # theta_future
ag <- meta$age_group[idxP]; an <- meta$age_num[idxP]; gd <- meta$gender[idxP]
ag <- factor(as.character(ag), levels=c("elementary","20_30","40_50","60_80"))
gd <- factor(as.character(gd), levels=c("female","male"))
cat("Number of pairs =", length(common), "\n")

## (1) Transition matrix: dominant past topic -> dominant future topic
domP <- max.col(TP); domF <- max.col(TF)
cat("\n--- Transition matrix (rows = dominant past topic, cols = dominant future topic, counts) ---\n")
tabm <- table(past=paste0("T",domP), future=paste0("T",domF))
print(tabm)
cat("\n--- Most frequent future destination for each dominant past topic ---\n")
for (k in sort(unique(domP))) {
  row <- tabm[paste0("T",k), ]
  cat(sprintf("past T%-2d (n=%3d) -> most frequent: %s (%d)\n",
              k, sum(domP==k), names(which.max(row)), max(row)))
}

## (2) Continuity score = cosine(theta_past, theta_future)
cosine <- rowSums(TP*TF) / (sqrt(rowSums(TP^2))*sqrt(rowSums(TF^2)))
cat("\n--- Mean continuity score (cosine) by age band ---\n")
print(round(tapply(cosine, ag, mean), 3))
cat("Overall mean =", round(mean(cosine),3), " median =", round(median(cosine),3), "\n")
m <- lm(cosine ~ an * gd)
cat("\n--- lm(cosine ~ age_num * gender) ---\n")
print(round(summary(m)$coefficients, 6))

## (3) Narrative types (k-means on the concatenated [theta_past, theta_future])
X <- cbind(TP, TF)
km <- kmeans(X, centers=4, nstart=25)
cat("\n--- Clusters (narrative types, k=4) ---\n")
for (c in 1:4) {
  ii <- which(km$cluster==c)
  cat(sprintf("\n[Cluster %d] n=%d, mean continuity=%.3f\n", c, length(ii), mean(cosine[ii])))
  cat("  Top dominant past topics  :", paste0("T", as.integer(names(sort(table(domP[ii]),decreasing=TRUE
    ↪ )[1:2]))), "\n")
  cat("  Top dominant future topics:", paste0("T", as.integer(names(sort(table(domF[ii]),decreasing=TRUE
    ↪ )[1:2]))), "\n")
  cat("  Age-band composition:", paste(names(table(ag[ii])), as.integer(table(ag[ii])), sep="=", collapse=" "), "
    ↪ \n")
}
cat("\nDONE_EXP1\n")
```

Listing S20. `src/runners/_exp1_clusters_breakdown.R`

```r
# ======================================================================
# Supplementary analysis (continued): age band x gender breakdown of the
#   four narrative types (k-means, k = 4). Reproduces the procedure and
#   random-number sequence of _exp1_continuity.R exactly.
# Cluster numbers are arbitrary; match clusters by their dominant topics.
```

```r
# ======================================================================
options(width = 120)
library(stm)
set.seed(20260607)

obj  <- readRDS("output/models/stm_fit_K10.rds")
fit  <- obj$fit; meta <- obj$out$meta
theta <- fit$theta
sid <- as.character(meta$subjid); tr <- as.character(meta$time_ref)

common <- intersect(sid[tr=="past"], sid[tr=="future"])
idxP <- match(common, ifelse(tr=="past", sid, NA))
idxF <- match(common, ifelse(tr=="future", sid, NA))
TP <- theta[idxP, , drop=FALSE]; TF <- theta[idxF, , drop=FALSE]
ag <- factor(as.character(meta$age_group[idxP]), levels=c("elementary","20_30","40_50","60_80"))
gd <- factor(as.character(meta$gender[idxP]),    levels=c("female","male"))
domP <- max.col(TP); domF <- max.col(TF)
cosine <- rowSums(TP*TF) / (sqrt(rowSums(TP^2))*sqrt(rowSums(TF^2)))

X <- cbind(TP, TF)
km <- kmeans(X, centers=4, nstart=25)

for (c in 1:4) {
  ii <- which(km$cluster==c)
  cat(sprintf("\n[Cluster %d] n=%d continuity=%.3f  top past T=%s  top future T=%s\n",
      c, length(ii), mean(cosine[ii]),
      paste0("T", as.integer(names(sort(table(domP[ii]),decreasing=TRUE)[1:2]))[1:2], collapse=","),
      paste0("T", as.integer(names(sort(table(domF[ii]),decreasing=TRUE)[1:2]))[1:2], collapse=",")))
  cat("  Counts by age band x gender:\n")
  print(addmargins(table(age_band=ag[ii], gender=gd[ii])))
}
cat("\nDONE_CLUST\n")
```

Listing S21. `README.md`

````
# Analysis code: Structural Topic Model of past and future essays across the life span

This package contains all preprocessing and analysis scripts used in the study.
It contains **code only**. The essay corpus is not included because it consists of
participant-generated free text that could permit re-identification; it is available
from the corresponding author on reasonable request, subject to approval by the
institutional research ethics committee (see the Data availability statement of the
article).

## Requirements

| Software | Version used |
|---|---|
| R | 4.6.0 |
| R package `stm` | 1.3.8 |
| R packages `dplyr` / `readr` | 1.2.1 / 2.2.0 |
| Python | 3.8 (standard library only) |
| MeCab with UniDic | to produce the `.mecab` input files |

## Input data (not included)

- `data/*.mecab`: one MeCab/UniDic output file per essay (tab-separated surface form and
  comma-separated features; the lemma is feature field 8). File names encode the writer:
  - elementary school children: `<id>(f|m)_(f|p).mecab` (gender f/m; future f / past p)
  - adults: `<id>_(f|p).mecab`; the ID range encodes gender and age band
    (see `parse_filename()` in `src/01_preprocess.py`).
- `data/lexicon/`: Japanese Sentiment Polarity Dictionary (Inui-Okazaki Laboratory,
  Tohoku University), files `pn.csv.m3.120408.trim` and `wago.121808.pn`. It is used only
  for the net-positivity analysis and must be obtained from its original distributor.

## How to run

Run everything from the root of this package. Relative paths are resolved from there.

```bash
bash run_all.sh                       # whole pipeline, in manuscript order
# or with explicit interpreters:
PYTHON=python3 RSCRIPT=Rscript bash run_all.sh
````
````

```

Each step can also be run on its own, for example `Rscript src/runners/_run_fit.R`.
All downstream scripts read the saved model `output/models/stm_fit_K10.rds`.

## Scripts and the results they produce

| Script | Purpose | Manuscript item |
|---|---|---|
| `src/01_preprocess.py` | Content-word extraction from `.mecab` files to `output/corpus.csv` | Methods (
    ↪ Preprocessing) |
| `src/runners/_run_searchK.R` | K selection diagnostics, K = 5, 10, 15, 20 | Methods (Model) |
| `src/runners/_run_fit.R` | Main STM, K = 10; topic words | Tables 2 and 3 |
| `src/runners/_run_table1.R` | Corpus composition | Table 1 |
| `src/runners/_run_effect.R` | Age x gender effects on topic prevalence | Table 4 |
| `src/runners/fig1.R` | Age trends by gender | Figure 1 |
| `src/runners/_run_spread.R` | Concentration of the mean topic mixture | Table 5 |
| `src/runners/_run_timeref.R` | Past/future topic shares by age band | Results (text) |
| `src/runners/fig2.R` | Topic composition of past vs future essays | Figure 2 |
| `src/runners/_run_ee_timeage.R` | Age x temporal-reference effects | Table 6 |
| `src/runners/_run_findthoughts.R` | Representative documents of the planning topic | Discussion |
| `src/runners/_run_content_age.R` | Auxiliary model, content covariate = age band | Results, Discussion |
| `src/runners/_run_K20.R` | Re-estimation at K = 20 | Results, Discussion |
| `src/runners/_exp2_features.py`, `_exp2_clt.R` | Abstractness index | Results (supplementary analyses) |
| `src/runners/_exp2c_sentiment_tohoku.py`, `_exp2c_sentiment_tohoku_stats.R` | Net positivity | Results (
    ↪ supplementary analyses) |
| `src/runners/_exp1_continuity.R`, `_exp1_clusters_breakdown.R` | Within-writer continuity and k-means types |
    ↪ Results (supplementary analyses) |

## Reproducibility notes

- The random seed is fixed at `20260607` in every script that uses random numbers.
- `stm()` uses spectral initialization, which is deterministic, so the topics are
  reproducible. `estimateEffect()` draws from the approximate posterior, so p values
  can vary slightly with the package version.
- k-means cluster numbers are arbitrary. Identify clusters by their dominant topics.
- `_run_findthoughts.R` prints preprocessed essay content. Do not share its output.
- Japanese string literals that remain in the code (UniDic part-of-speech tags, stop
  lemmas, lexicon labels) must match the data and are therefore not translated;
  English glosses are given in the comments.

## License

```
MIT License

Copyright (c) 2026 Tomoe Taniguchi

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```
```