Figure_4_dplyr_based_cum_sum

Diego Ellis Soto

2026-01-17

Code to calculative cumulative sampling density in a dplyr-plyr free conflicting environment

We show how the cumulative sampling density is calculated in the same way as our originally deposited code and calculated Figure 4, but with dplyr only

Rationale for cumulative sampling density and validation checks

The purpose of this script is to **explicitly reconstruct Figure 4 from Ellis-Soto et al. (2023)** using a *dplyr-only workflow*, while making all aggregation steps transparent and verifiable. This addresses concerns raised in subsequent replication efforts regarding unintended data aggregation or multiplication.

We intentionally use **cumulative sampling density** to visualize how *biodiversity knowledge accumulates through time.*

Cumulative metrics can represent knowledge growth and information accumulation, particularly when raw observations are sparse in early years and increase rapidly following technological or social shifts (e.g. the expansion of eBird and smartphone-based biodiversity data collection after ~2010).

Annual vs cumulative metrics

To avoid ambiguity, this script also computes **annual observation counts (`n_obs`)** alongside our originally published cumulative totals (`cumsum_n_obs`). Annual counts represent the true number of observations recorded in each year, whereas cumulative counts intentionally re-count prior years by design.

As a result: - The *sum of annual counts* equals the total number of bird observations. - The *sum of cumulative counts across years* is necessarily larger (in our case the cumulative counts of 2020 equals the total number of bird observations) - This aggregation is **expected, intentional, and mathematically designed** when using cumulative sums.

Why these sanity checks?

The checks reported below serve three purposes:

1. **Prevent unintended data multiplication**
   We verify that there is exactly one row per `(Year × HOLC grade)` after aggregation.
2. **Demonstrate numerical equivalence across representations**
   We confirm that:
   - The final cumulative value per HOLC grade equals the sum of annual observations for that grade.
   - The sum of final cumulative values across grades equals the total number of observations in the raw data.

3. **Why cumulative sums appear "large"**
   We show that large cumulative totals arise solely from re-counting earlier years, not from duplicated records or join errors.

   By definition, cumulative sampling density is non-decreasing through time: once an observation has occurred, it remains part of the accumulated total in all subsequent years, so cumulative curves can increase or plateau, but they cannot decrease. **Consequently, summing cumulative values across years necessarily yields a number that is larger than the total number of unique observations**, because each observation is counted once for every year following its occurrence.

This behavior is a mathematical property of cumulative sums and does not reflect additional data, replication, or aggregation error.

These checks ensure that the cumulative trends shown in Fig. 4 are a **faithful transformation of the underlying yearly data**, rather than an artefact of coding errors or implicit aggregation.

Scope and limitations

We agree that cumulative, grade-level aggregation: - Does not capture polygon-level heterogeneity, - Does not model spatial or temporal non-independence

For this reason, Fig. 4 is presented strictly as a **descriptive visualization of knowledge accumulation** and we made no claims about temporal patterns of polygon-level heterogeneity

In summary, this code demonstrates that: - The cumulative sampling density shown in Fig. 4 is **intentional computed**, - The reported trends follow directly from the underlying annual data cummulative sum of sampling density as mentioned in our paper,

```r
# "We intentionally used cumulative sampling density to visualize how knowledge accumulates over time; 
# ---- Inputs ----

# holc <- st_read('../../indir/holc_shp/holc_ad_data.shp') %>%
#   sf::st_cast('POLYGON') %>% # IMPORTANT
#   dplyr::filter(!st_is_empty(.)) %>%
#   sf::st_make_valid(.) %>%
#   tibble::rowid_to_column() %>%
#   dplyr::mutate(  id = paste(state, city, holc_id, holc_grade, rowid, sep = '_')
#                   , city_state = paste0(city, ', ', state)
#                   , area_holc_km2 = as.double(st_area(.) / 1e+6)) %>%
#   dplyr::select(id, state, city, holc_id, holc_grade, city_state, area_holc_km2)
#
# # Calculate the area of holc polygons
# holc_area <-  holc %>% dplyr::select(city, holc_grade, area_holc_km2) %>% dplyr::group_by(holc_grade)

    holc_area = read.csv(
  '/Users/diegoellis/Desktop/Projects/Postdoc/Replic_HOLC_NHB/indir/Biodiv_Greeness_Social/main_combined
  group_by(holc_grade) |>
  dplyr::summarise(area_sum = sum(area_holc_km2)) %>%
  dplyr::filter(holc_grade != 'E')  %>%
  as_tibble()


# ---- 1) Load the "trend" data (already Year, holc_grade, Sum) ----
temporal_trend <- read_csv(
  "../../indir/Biodiv_Greeness_Social/R1_biodiv_trend_by_time_holc_id_1933_2022.csv",
```

```r
  col_names = FALSE,
  show_col_types = FALSE
) |>
  setNames(c("Year", "holc_grade", "Sum")) |>
  mutate(
    Year = as.integer(Year),
    holc_grade = as.character(holc_grade),
    Sum = as.numeric(Sum)
  ) |>
  filter(holc_grade %in% c("A","B","C","D"))
```

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `Year = as.integer(Year)`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```r
sum(temporal_trend$Sum)
```

```
## [1] 12290294
```

```r
# ---- 2) Annual aggregation (1 row per Year x grade) ----
annual_by_grade_year <- temporal_trend %>%
  dplyr::group_by(holc_grade, Year) %>%
  dplyr::summarise(
    n_obs = sum(Sum, na.rm = TRUE),
    .groups = "drop"
  )

annual_by_grade_year %>% head()
```

```
## # A tibble: 6 x 3
##   holc_grade  Year n_obs
##   <chr>      <int> <dbl>
## 1 A           1932   113
## 2 A           1933    56
## 3 A           1934    47
## 4 A           1935    38
## 5 A           1936    13
## 6 A           1937    30
```

```r
names(annual_by_grade_year)
```

```
## [1] "holc_grade" "Year"       "n_obs"
```

```r
nrow(annual_by_grade_year)
```

```
## [1] 364
```

```r
sum(annual_by_grade_year$n_obs)
```

```
## [1] 12290294
```

```r
# ---- 3) Join grade areas + compute cumulative totals + density ----
temporal_all_data <- annual_by_grade_year |>
  dplyr::arrange(holc_grade, Year) |>
  dplyr::group_by(holc_grade) |>
  dplyr::mutate(cumsum_n_obs = cumsum(n_obs)) |>
  ungroup() |>
  dplyr::left_join(holc_area |>
                     dplyr::select(holc_grade, area_sum), by = "holc_grade") |>
  dplyr::mutate(sampling_density = cumsum_n_obs / area_sum)

sum(temporal_all_data$cumsum_n_obs)
```

```
## [1] 92008824
```

```r
sum(temporal_all_data$n_obs)
```

```
## [1] 12290294
```

```r
# ---- 4) Sanity checks (these are the receipts) ----

# A) Ensure exactly 1 row per Year x grade (no multiplication)
dup_check <- temporal_all_data %>%
  dplyr::count(holc_grade, Year) %>%
  dplyr::summarise(max_n = max(n), .groups = "drop")
print(dup_check)   # max_n should be 1
```

```
## # A tibble: 1 x 1
##   max_n
##   <int>
## 1     1
```

```r
# B) Annual totals should equal the raw totals
raw_total <- temporal_trend %>% dplyr::summarise(raw_sum = sum(Sum, na.rm = TRUE)) %>% pull(raw_sum)
annual_total <- temporal_all_data %>% dplyr::summarise(annual_sum = sum(n_obs)) %>% pull(annual_sum)
cat("\nRaw total Sum =", raw_total, "\nAnnual aggregated total =", annual_total, "\n")
```

```
##
## Raw total Sum = 12290294
## Annual aggregated total = 12290294
```

```r
# C) Final cumulative (end of series) should equal annual totals (within each grade)
end_vs_sum <- temporal_all_data %>%
  dplyr::group_by(holc_grade) %>%
  dplyr::summarise(
    sum_annual = sum(n_obs),
```

```r
    end_cum    = cumsum_n_obs[Year == max(Year)],
    .groups = "drop"
  ) %>%
  dplyr::mutate(ok = (sum_annual == end_cum))
print(end_vs_sum)
```

```
## # A tibble: 4 x 4
##   holc_grade sum_annual end_cum ok
##   <chr>           <dbl>   <dbl> <lgl>
## 1 A             2036676 2036676 TRUE
## 2 B             3594743 3594743 TRUE
## 3 C             4320873 4320873 TRUE
## 4 D             2338002 2338002 TRUE
```

```r
# D) Demonstrate why "sum of cumulative" is huge BY DESIGN (not a bug)
cum_sum_all_years <- temporal_all_data %>%
  dplyr::summarise(sum_of_cumsums = sum(cumsum_n_obs)) %>%
  pull(sum_of_cumsums)

cat("\nNOTE:\n",
    "Sum of annual counts (true records) =", annual_total, "\n",
    "Sum of cumsum across years (intentionally large) =", cum_sum_all_years, "\n",
    "This is large because cumulative sums re-count prior years on purpose.\n\n")
```

```
##
## NOTE:
##  Sum of annual counts (true records) = 12290294
##  Sum of cumsum across years (intentionally large) = 92008824
##  This is large because cumulative sums re-count prior years on purpose.
```

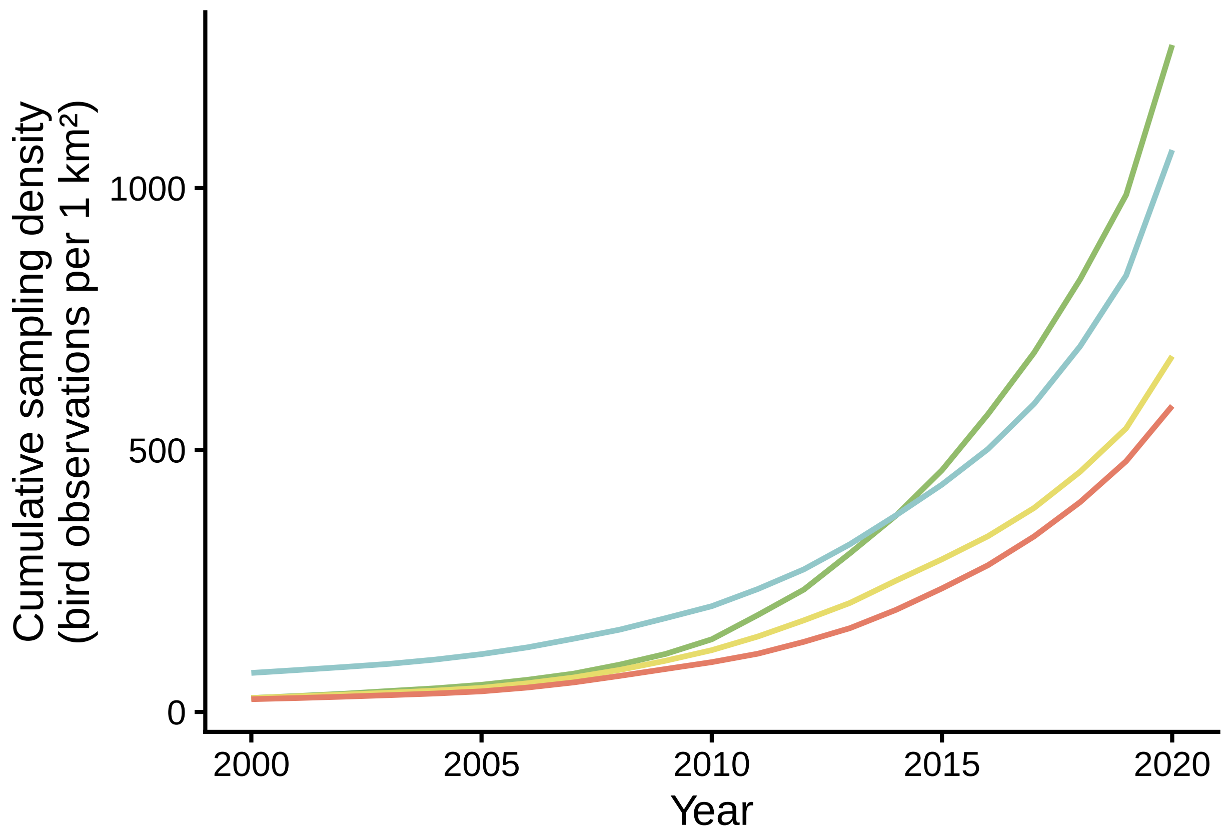
```r
# ---- 5) Plot your Fig 4 equivalent (2000-2020) ----

holc_pal <- c('#92BC6B' # green
              , '#92C7C9' # blue
              , '#E7DC6B' # yellow
              , '#E47D67' # red

)

p <- temporal_all_data %>%
  filter(Year >= 2000, Year <= 2020) %>%
  ggplot(aes(x = Year, y = sampling_density, color = holc_grade)) +
  geom_line(linewidth = 1) +
  scale_color_manual(values = holc_pal) +
  theme_classic(base_size = 16) +
  theme(legend.position = "none") +
  ylab("Cumulative sampling density\n(bird observations per 1 km²)")

print(p)
```




```
ggplot2::ggsave('../../outdir/Fig_4_dplyr_based_cum_sum.png', p, width = 7, height = 7, dpi = 300)
```

```
# Should be 1 row per (Year, holc_grade) if already aggregated
temporal_trend %>%
  dplyr::count(Year, holc_grade) %>%
  dplyr::summarise(max_n = max(n), .groups="drop")
```

```
## # A tibble: 1 x 1
##   max_n
##   <int>
## 1  2650
```

```
ncol(temporal_trend)
```

```
## [1] 3
```

```
names(temporal_trend)
```

```
## [1] "Year"       "holc_grade" "Sum"
```

```
temporal_trend %>% dplyr::summarise(
  n_rows = n(),
```

```r
  n_years = n_distinct(Year),
  n_grades = n_distinct(holc_grade)
)
```

```
## # A tibble: 1 x 3
##   n_rows n_years n_grades
##    <int>   <int>    <int>
## 1  81738      91        4
```

```r
temporal_trend %>% dplyr::summarise(
  n_na_sum = sum(is.na(Sum)),
  min_sum = min(Sum, na.rm=TRUE),
  max_sum = max(Sum, na.rm=TRUE)
)
```

```
## # A tibble: 1 x 3
##   n_na_sum min_sum max_sum
##      <int>   <dbl>   <dbl>
## 1        0       1   98151
```

```r
annual_by_grade_year %>% dplyr::count(holc_grade)      # should be ~#years per grade
```

```
## # A tibble: 4 x 2
##   holc_grade     n
##   <chr>      <int>
## 1 A             91
## 2 B             91
## 3 C             91
## 4 D             91
```

```r
annual_by_grade_year %>% dplyr::count(Year)            # should be 4 per year (A-D) for most years
```

```
## # A tibble: 91 x 2
##     Year     n
##    <int> <int>
##  1  1932     4
##  2  1933     4
##  3  1934     4
##  4  1935     4
##  5  1936     4
##  6  1937     4
##  7  1938     4
##  8  1939     4
##  9  1940     4
## 10  1941     4
## # i 81 more rows
```

```r
# This is "sum of the final cumulative totals per grade" (i.e., total observations across all years, by

check_final_cum_sum <- temporal_all_data %>%
```

```r
  group_by(holc_grade) %>%
  summarise(final_cum = max(cumsum_n_obs, na.rm = TRUE), .groups = "drop") %>%
  summarise(sum_final_cum = sum(final_cum)) %>%
  pull(sum_final_cum)

check_final_cum_sum
```

```
## [1] 12290294
```

```r
# Extra sanity (this should equal total annual counts across all grades/years):

check_total_annual <- temporal_all_data %>% summarise(total = sum(n_obs, na.rm = TRUE)) %>% pull(total)

c(final_cum_sum = check_final_cum_sum, total_annual = check_total_annual)
```

```
## final_cum_sum  total_annual
##      12290294      12290294
```

```r
# This is "sum of cumulative totals over the window 2000-2020, across grades".
check_sum_cumsums_2000_2020 <- temporal_all_data %>%
  filter(Year >= 2000, Year <= 2020) %>%
  summarise(sum_cumsums = sum(cumsum_n_obs, na.rm = TRUE)) %>%
  pull(sum_cumsums)

check_sum_cumsums_2000_2020
```

```
## [1] 59109884
```

```r
check_sum_cumsums_2000_2020_by_grade <- temporal_all_data %>%
  filter(Year >= 2000, Year <= 2020) %>%
  group_by(holc_grade) %>%
  summarise(sum_cumsums = sum(cumsum_n_obs, na.rm = TRUE), .groups = "drop")

check_sum_cumsums_2000_2020_by_grade
```

```
## # A tibble: 4 x 2
##   holc_grade sum_cumsums
##   <chr>            <dbl>
## 1 A              8442006
## 2 B             18100565
## 3 C             21250429
## 4 D             11316884
```

```r
sum(check_sum_cumsums_2000_2020_by_grade$sum_cumsums)
```

```
## [1] 59109884
```

```r
# That's just "sum of cumsums across all years and grades" (the intentionally large number).

check_sum_cumsums_all <- temporal_all_data %>%
  summarise(sum_cumsums_all = sum(cumsum_n_obs, na.rm = TRUE)) %>%
  pull(sum_cumsums_all)

check_sum_cumsums_all
```

```
## [1] 92008824
```

```r
temporal_all_data %>%
  summarise(
    sum_annual_counts = sum(n_obs, na.rm = TRUE),
    sum_of_cumsums    = sum(cumsum_n_obs, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 2
##   sum_annual_counts sum_of_cumsums
##               <dbl>          <dbl>
## 1          12290294       92008824
```

```r
summary(gam(sampling_density ~ Year * holc_grade, data = temporal_all_data[temporal_all_data$Year %in% c
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sampling_density ~ Year * holc_grade
##
## Parametric coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.026e+05  8.691e+03 -11.802  < 2e-16 ***
## Year              5.119e+01  4.324e+00  11.838  < 2e-16 ***
## holc_gradeB       2.198e+04  1.229e+04   1.788 0.077732 .
## holc_gradeC       4.679e+04  1.229e+04   3.807 0.000283 ***
## holc_gradeD       5.504e+04  1.229e+04   4.478 2.61e-05 ***
## Year:holc_gradeB -1.093e+01  6.115e+00  -1.788 0.077777 .
## Year:holc_gradeC -2.334e+01  6.115e+00  -3.817 0.000274 ***
## Year:holc_gradeD -2.746e+01  6.115e+00  -4.490 2.50e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.794   Deviance explained = 81.1%
## GCV =  15910  Scale est. = 14395     n = 84
```

```r
# model_sampling = glm((sampling_density) ~ Year * holc_grade, data = temporal_all_data_tmp[temporal_al
model_sampling = gam(sampling_density ~ Year * holc_grade, data = temporal_all_data[temporal_all_data$Ye
# model_sampling |> tab_model( show.aic = TRUE,   dv.labels = "Cumulative sampling density"
# )
```

```
model_sampling_log = gam((log(sampling_density)) ~ Year * holc_grade, data = temporal_all_data[temporal_

# model_sampling_log |> tab_model( show.aic = TRUE,   dv.labels = "Cumulative log sampling density")


tab_model(
  model_sampling_log,
  model_sampling,
  show.aic = TRUE,
  dv.labels = c(
    "Cumulative log sampling density<br>(original Suppl. Table 4)",
    "Cumulative sampling density"
  )
)
```

Cumulative log sampling density(original Suppl. Table 4)

Cumulative sampling density

Predictors

Estimates

CI

p

Estimates

CI

p

(Intercept)

-400.14

-414.27 – -386.01

<0.001

-102569.30

-119878.24 – -85260.36

<0.001

Year

0.20

0.19 – 0.21

<0.001

51.19

42.57 – 59.80

<0.001

holc grade [B]

138.36

118.38 – 158.34

<0.001

21977.53

-2501.00 – 46456.06

0.078

holc grade [C]

67.07

47.09 – 87.05

<0.001

46789.84

22311.31 – 71268.38

<0.001

holc grade [D]

73.20

53.22 – 93.18

<0.001

55036.44

30557.91 – 79514.97

<0.001

Year × holc grade [B]

-0.07

-0.08 – -0.06

<0.001

-10.93

-23.11 – 1.25

0.078

Year × holc grade [C]

-0.03

-0.04 – -0.02

<0.001

-23.34

-35.52 – -11.16

<0.001

Year × holc grade [D]

-0.04

-0.05 – -0.03

<0.001

-27.46

-39.63 – -15.28

<0.001

Observations

84

84

R2

0.992

0.794

AIC

693.528

1052.242

```r
library(mgcv)
library(dplyr)
library(sjPlot)

s <- summary(model_sampling_log)

param_df <- as.data.frame(s$p.table) |>
  tibble::rownames_to_column("Term") |>
  rename(
    Estimate  = Estimate,
    `Std Error` = `Std. Error`,
    `t-value`  = `t value`,
    `p-value`  = `Pr(>|t|)`
  ) |>
  mutate(
    Component = "A. parametric coefficients",
    `p-value`  = format.pval(`p-value`, digits = 4, eps = 1e-4)
  ) |>
  select(Component, Term, Estimate, `Std Error`, `t-value`, `p-value`)

tab_df(param_df, show.rownames = FALSE)
```

Component

Term

Estimate

Std.Error

t.value

p.value

A. parametric coefficients

(Intercept)

-400.14

7.09

-56.41

< 1e-04

A. parametric coefficients

Year

0.20

0.00

57.12

< 1e-04

A. parametric coefficients

holc_gradeB

138.36

10.03

13.79

< 1e-04

A. parametric coefficients

holc_gradeC

67.07

10.03

6.69

< 1e-04

A. parametric coefficients

holc_gradeD

73.20

10.03

7.30

< 1e-04

A. parametric coefficients

Year:holc_gradeB

-0.07

0.00

-13.76

< 1e-04

A. parametric coefficients

Year:holc_gradeC

-0.03

0.00

-6.71

< 1e-04

A. parametric coefficients

Year:holc_gradeD

-0.04

0.00

-7.34

< 1e-04

```r
# Parametric table
param_df <- as.data.frame(s$p.table) |>
  tibble::rownames_to_column("Term") |>
  rename(
    Estimate   = Estimate,
    `Std Error` = `Std. Error`,
    `t-value`   = `t value`,
    `p-value`   = `Pr(>|t|)`
  ) |>
  mutate(
    Component = "A. parametric coefficients",
    `p-value` = format.pval(`p-value`, digits = 4, eps = 1e-4)
  ) |>
  select(Component, Term, Estimate, `Std Error`, `t-value`, `p-value`)

# Create model summary rows
fit_rows <- tibble::tibble(
  Component = "",
  Term = c("Adjusted R-squared", "Deviance explained", "GCV", "Scale est", "N"),
  Estimate = c(s$r.sq, s$dev.expl, s$sp.criterion, s$scale, s$n),
  `Std Error` = NA,
  `t-value` = NA,
  `p-value` = NA
)

# Combine
final_df <- dplyr::bind_rows(param_df, fit_rows)

# Print
sjPlot::tab_df(final_df, show.rownames = FALSE)
```

Component

Term

Estimate

Std.Error

t.value

p.value

A. parametric coefficients

(Intercept)

-400.14

7.09

-56.41

< 1e-04

A. parametric coefficients

Year

0.20

0.00

57.12

< 1e-04

A. parametric coefficients

holc_gradeB

138.36

10.03

13.79

< 1e-04

A. parametric coefficients

holc_gradeC

67.07

10.03

6.69

< 1e-04

A. parametric coefficients

holc_gradeD

73.20

10.03

7.30

< 1e-04

A. parametric coefficients

Year:holc_gradeB

-0.07

0.00

-13.76

< 1e-04

A. parametric coefficients

Year:holc_gradeC

-0.03

0.00

-6.71

< 1e-04

A. parametric coefficients

Year:holc_gradeD

-0.04

0.00

-7.34

< 1e-04

Adjusted R-squared

0.99

NA

NA

NA

Deviance explained

0.99

NA

NA

NA

GCV

0.01

NA

NA

NA

Scale est

0.01

NA

NA

NA

N

84.00

NA

Component	Term	Estimate	Std.Error	t.value	p.value
A. parametric coefficients	(Intercept)	-400.13977669	7.093669949	-56.408006	< 1e-04
	Year	0.20159364	0.003529173	57.122063	< 1e-04
	holc_gradeB	138.36420791	10.031964248	13.792335	< 1e-04
	holc_gradeC	67.07353907	10.031964248	6.685983	< 1e-04
	holc_gradeD	73.20392432	10.031964248	7.297068	< 1e-04
	Year:holc_gradeB	-0.06865717	0.004991004	-13.756184	< 1e-04
	Year:holc_gradeC	-0.03350489	0.004991004	-6.713056	< 1e-04
	Year:holc_gradeD	-0.03664046	0.004991004	-7.341299	< 1e-04

Adjusted R-squared: 0.992, Deviance explained: 0.992GCV: 0.0106, Scale est: 0.00959, N: 84

NA

NA

```r
library(dplyr)
library(tibble)
library(gt)

s <- summary(model_sampling_log)

param_df <- as.data.frame(s$p.table) |>
  rownames_to_column("Term") |>
  rename(
    Estimate    = Estimate,
    `Std.Error` = `Std. Error`,
    `t.value`   = `t value`,
    `p.value`   = `Pr(>|t|)`
  ) |>
  mutate(
    Component = "A. parametric coefficients",
    `p.value` = format.pval(`p.value`, digits = 4, eps = 1e-4),
    Component = ifelse(duplicated(Component), "", Component)
  ) |>
  select(Component, Term, Estimate, `Std.Error`, `t.value`, `p.value`)

stats_note <- sprintf(
  "Adjusted R-squared: %.3f, Deviance explained: %.3f<br>GCV: %.4f, Scale est: %.5f, N: %d",
  s$r.sq, s$dev.expl, s$sp.criterion, s$scale, s$n
)

param_df |>
  gt() |>
  tab_source_note(md(stats_note))
```

```
## Warning: HTML tags found, and they will be removed.
## * Set 'options(gt.html_tag_check = FALSE)' to disable this check.
```

```r
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] gt_1.0.0      tibble_3.3.0  sjPlot_2.8.17 mgcv_1.9-1    nlme_3.1-167
## [6] sf_1.0-21     ggplot2_4.0.1 readr_2.1.5   dplyr_1.1.4
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6       bayestestR_0.15.2  xfun_0.52          insight_1.0.2
##  [5] lattice_0.22-6     tzdb_0.5.0         vctrs_0.6.5        tools_4.4.1
##  [9] sjstats_0.19.0     generics_0.1.4     datawizard_1.0.0   parallel_4.4.1
## [13] proxy_0.4-27       pkgconfig_2.0.3    Matrix_1.7-2       KernSmooth_2.23-26
## [17] RColorBrewer_1.1-3 S7_0.2.1           ggeffects_2.2.0    lifecycle_1.0.4
## [21] compiler_4.4.1     farver_2.1.2       textshaping_1.0.1  sjmisc_2.8.10
## [25] htmltools_0.5.8.1  class_7.3-23       yaml_2.3.10        pillar_1.11.1
## [29] crayon_1.5.3       tidyr_1.3.1        classInt_0.4-11    commonmark_1.9.5
## [33] tidyselect_1.2.1   sjlabelled_1.2.0   digest_0.6.37      performance_0.13.0
## [37] purrr_1.2.0        labeling_0.4.3     splines_4.4.1      fastmap_1.2.0
## [41] grid_4.4.1         cli_3.6.5          magrittr_2.0.4     dichromat_2.0-0.1
## [45] utf8_1.2.6         e1071_1.7-16       withr_3.0.2        scales_1.4.0
## [49] bit64_4.6.0-1      rmarkdown_2.29     bit_4.6.0          ragg_1.4.0
## [53] hms_1.1.3          evaluate_1.0.3     knitr_1.50         parameters_0.24.1
## [57] markdown_1.13      rlang_1.1.6        Rcpp_1.1.0         glue_1.8.0
## [61] DBI_1.2.3          xml2_1.3.8         effectsize_1.0.0   rstudioapi_0.17.1
## [65] vroom_1.6.5        R6_2.6.1           systemfonts_1.2.3  units_0.8-7
```