



Oblig 3c

Gruppe 98

av

Christopher Sanden

i

MA-223

Statistikk

Fakultet for teknologi og realfag

Universitetet i Agder

April 2026

Innhold

1	Innledning	1
2	Oppgave 1: Kapittel 17, oppgave 1c	2
2.1	Tema	2
2.2	Prior og hyperparametre	2
2.3	Posterior for τ	2
2.4	Posterior for $y(x)$	2
2.5	Prediktiv fordeling for $Y_+(x)$	2
2.6	Intervallestimater	3
2.7	Kommentar	3
2.8	R-kode	3
3	Oppgave 2: Kapittel 17, oppgave 1d	4
3.1	Tema	4
3.2	Prior og hyperparametre	4
3.3	Posterior for τ	4
3.4	Posterior for $y(x)$	4
3.5	Prediktiv fordeling for $Y_+(x)$	4
3.6	Intervallestimater	4
3.7	Kommentar	5
4	Oppgave 3: Terningdropp	6
4.1	Tema	6
4.2	Datagrunnlag	6
4.3	a) Punktsky og regresjonslinje	8
4.4	b) Posterior- og prediktive fordelinger	8
4.5	c) 80% kredibilitetsintervall for stigningstallet b	9
4.6	d) 80% kredibilitetsintervall for standardavviket σ	9
4.7	e) 80% kredibilitetsintervall for $y(x)$	9
4.8	f) Kurver over og under regresjonslinjen	9
4.9	g) Forklaringsgrad R^2	10
4.10	h) Regresjon mellom z og x , og mellom t og x	10
4.11	R-kode	11
5	Oppgave 4: Utvalgsforsøk	12
5.1	Tema	12
5.2	a) 50 runder med $N = 5$	12
5.3	b) 50 runder med $N = 15, 50, 200$	13
5.4	c) Oppgave 3c gjentatt 50 ganger med $N = 5$	16
5.5	d) Samme analyse for $N = 15, 50, 200$	16
5.6	e) Illustrasjoner som i oppgave 3f	19
5.7	Kommentar	23
5.8	R-kode	24

Figurer

1	Punktsky for terningdropp med regresjonslinje.	8
2	80% kredibilitetsbånd for $y(x)$	9
3	50 regresjonslinjer basert på utvalg med $N = 5$	12
4	Regresjonslinjer for $N = 15$	13
5	Regresjonslinjer for $N = 50$	14
6	Regresjonslinjer for $N = 200$	15
7	50 intervalestimater for b når $N = 5$	16
8	Intervallestimater for b når $N = 15$	17
9	Intervallestimater for b når $N = 50$	18
10	Intervallestimater for b når $N = 200$	19
11	Kredibilitetsbånd for $N = 5$	20
12	Kredibilitetsbånd for $N = 15$	21
13	Kredibilitetsbånd for $N = 50$	22
14	Kredibilitetsbånd for $N = 200$	23

List of Listings

1	R-kode for bokoppgavene 17.1c og 17.1d	3
2	R-kode for innlesing og standardisering av terningdropp-data	7
3	R-kode for analyse og figurer i terningdropp-oppgaven	11
4	R-kode for gjentatte delutvalg og intervallillustrasjoner	24

1 Innledning

I denne rapporten ser vi på lineær regresjon med usikkerhet i en Bayesiansk ramme. Målet er å bruke teorien fra kapittel 17 til å beskrive både selve regresjonslinjen og usikkerheten rundt den. Det innebærer at vi ikke bare finner ett enkelt uttrykk for en linje, men også undersøker hvordan usikkerheten i dataene påvirker stigningstall, standardavvik, posteriorfordelinger og intervallestimater.

Rapporten er delt i to hoveddeler. Først løses to bokoppgaver fra kapittel 17, der vi bruker de generelle formlene for posterior- og prediktive fordelinger i lineær regresjon. Deretter brukes de samme ideene på terningdropp-dataene fra oppgavesettet, hvor vi analyserer sammenhengen mellom dropphøyde og sprettlengde. Til slutt undersøker vi hvordan regresjonslinjer og intervallestimater varierer når vi bare bruker tilfeldige delutvalg av observasjonene.

I arbeidet brukes R til å beregne regresjonslinjer, posteriorfordelinger, kredibilitetsintervall og figurer. Figurene brukes videre for å synliggjøre både mønsteret i dataene og hvordan usikkerheten endrer seg når datagrunnlaget blir større eller mindre.

2 Oppgave 1: Kapittel 17, oppgave 1c

I denne oppgaven bruker vi observasjonene

$$\{(2, 10), (3, 8), (4, 8), (6, 7)\},$$

og vi er gitt $\sigma_0 = 0.5$ og $n_0 = 4$.

2.1 Tema

Temaet er Bayesiansk lineær regresjon med informativ prior for usikkerheten. Vi skal finne posteriorfordelingen til τ , posteriorfordelingen til $y(x)$, prediktiv fordeling for $Y_+(x)$, samt tilhørende intervallestimater.

2.2 Prior og hyperparametre

Siden $\sigma_0 = 0.5$ og $n_0 = 4$, får vi

$$\nu_0 = n_0 - 2 = 2, \quad SS_0 = \sigma_0^2 \nu_0 = 0.5^2 \cdot 2 = 0.5.$$

Dette brukes videre sammen med regresjonsstatistikene fra datasettet.

2.3 Posterior for τ

Når regresjonslinjen er estimert, får vi posteriorfordelingen

$$\tau \mid \text{data} \sim \Gamma\left(\frac{\nu_1}{2}, \frac{SS_1}{2}\right).$$

Her settes de konkrete tallene inn fra R-skriptet.

2.4 Posterior for $y(x)$

For en vilkårlig verdi x får vi

$$y(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right).$$

De konkrete parameterverdiene kan hentes fra skriptet og settes inn her.

2.5 Prediktiv fordeling for $Y_+(x)$

Den prediktive fordelingen for en ny observasjon er

$$Y_+(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right).$$

2.6 Intervallestimater

Et 95%-kredibilitetsintervall for regresjonslinjen er

$$I_{0.05}(x) = \alpha_0 + \beta x \pm t_{\nu_1, 0.025} s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}.$$

Et 95%-prediktivt intervall fås tilsvarende ved å legge til 1 inne i roten.

2.7 Kommentar

Denne oppgaven er en direkte anvendelse av formlene i kapittel 17. Hovedpoenget er at vi kombinerer observasjonene med en svak informativ prior for usikkerheten, og deretter leser av posterior- og prediktive fordelinger fra de oppdaterte hyperparametrene.

2.8 R-kode

Listing 1 viser delen av R-skriptet som løser bokoppgavene 1c og 1d.

```
task_1c <- data.frame(
  x = c(2, 3, 4, 6),
  y = c(10, 8, 8, 7)
)

fit_1c <- fit_simple_regression(task_1c$x, task_1c$y, nu0 = 4 - 2, SS0 = 0.5^2 * (4 - 2))
print_regression_summary(
  label = "Task 1: Chapter 17, problem 1c",
  fit = fit_1c,
  x_eval = mean(task_1c$x),
  level_y = 0.95,
  level_pred = 0.95
)

task_1d <- data.frame(
  x = c(0, 1, 2, 3),
  y = c(0, 2, 7, 5)
)

fit_1d <- fit_simple_regression(task_1d$x, task_1d$y, nu0 = -2, SS0 = 0)
print_regression_summary(
  label = "Task 2: Chapter 17, problem 1d",
  fit = fit_1d,
  x_eval = mean(task_1d$x),
  level_y = 0.90,
  level_pred = 0.95
)
```

Listing 1: R-kode for bokoppgavene 17.1c og 17.1d

3 Oppgave 2: Kapittel 17, oppgave 1d

I denne oppgaven bruker vi observasjonene

$$\{(0, 0), (1, 2), (2, 7), (3, 5)\},$$

og vi antar nøytral prior for usikkerheten.

3.1 Tema

Temaet er Bayesiansk lineær regresjon når σ er ukjent. Da bruker vi de nøytrale hyperparametrene fra boka.

3.2 Prior og hyperparametre

Ved nøytral prior setter vi

$$\nu_0 = -2, \quad SS_0 = 0.$$

Dermed er det dataene alene som bestemmer posterioren.

3.3 Posterior for τ

Posteriorfordelingen blir igjen

$$\tau \mid \text{data} \sim \Gamma\left(\frac{\nu_1}{2}, \frac{SS_1}{2}\right),$$

der $\nu_1 = \nu_0 + n$ og $SS_1 = SS_0 + SSe$.

3.4 Posterior for $y(x)$

For regresjonslinjen får vi

$$y(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right).$$

3.5 Prediktiv fordeling for $Y_+(x)$

For en ny observasjon får vi

$$Y_+(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right).$$

3.6 Intervallestimater

Oppgaven ber om 90%-kredibilitetsintervall og 95%-prediktivt intervall. Disse blir

$$I_{0.10}(x) = \alpha_0 + \beta x \pm t_{\nu_1, 0.05} s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}},$$

og

$$I_{0.05}^+(x) = \alpha_0 + \beta x \pm t_{\nu_1, 0.025} s_1 \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}.$$

3.7 Kommentar

Forskjellen fra oppgave 1c er at vi nå ikke legger inn noen forhåndsinformasjon om usikkerheten. Det gir en mer datadrevet analyse, og intervallene blir derfor bestemt av spredningen i observasjonene alene.

4 Oppgave 3: Terningdropp

4.1 Tema

Her studerer vi sammenhengen mellom dropphøyde x og hvor langt terningen spretter ut fra veggen y . Vi bruker nøytrale priorhyperparametre og analyserer datasettet med Bayesiansk lineær regresjon.

4.2 Datagrunnlag

Dataene er hentet fra alle CSV-filene i mappen `terningDroppFiler`. Siden filene bruker litt ulike kolonnenavn, er de først normalisert i R-skriptet slik at vi får felles variabler for dropphøyde x , sprettlengde y og terningverdi z .

Listing 2 viser delen av skriptet som leser inn filene og standardiserer kolonnenavnene før analysen.

```

read_dice_file <- function(path) {
  raw_df <- read.csv(path, check.names = FALSE, fileEncoding = "UTF-8-BOM")
  raw_df <- raw_df[, colSums(!is.na(raw_df)) > 0, drop = FALSE]

  original_names <- names(raw_df)
  clean_names <- vapply(original_names, standardize_name, character(1))

  rename_map <- c(
    "k" = "k",
    "dropp" = "x",
    "dropphoyde" = "x",
    "x" = "x",
    "lengde" = "y",
    "sprettlengde" = "y",
    "y" = "y",
    "verdi" = "z",
    "terningverdi" = "z",
    "z" = "z",
    "tid" = "t",
    "t" = "t"
  )

  mapped_names <- rename_map[clean_names]
  names(raw_df) <- ifelse(is.na(mapped_names), clean_names, mapped_names)

  keep <- intersect(c("k", "x", "y", "z", "t"), names(raw_df))
  out <- raw_df[, keep, drop = FALSE]
  out$source_file <- basename(path)

  for (name in setdiff(names(out), "source_file")) {
    out[[name]] <- suppressWarnings(as.numeric(out[[name]]))
  }

  out
}

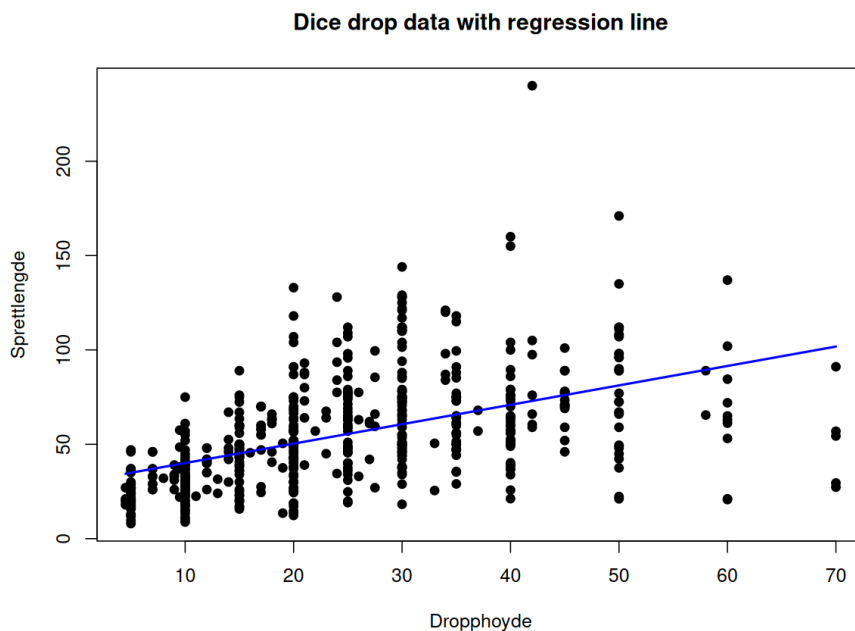
read_all_dice_data <- function(folder) {
  files <- list.files(folder, pattern = "\\\\.csv$", full.names = TRUE)
  df_list <- lapply(files, read_dice_file)
  combined <- do.call(rbind, df_list)
  rownames(combined) <- NULL
  combined
}

```

Listing 2: R-kode for innlesing og standardisering av terningdropp-data

4.3 a) Punktsky og regresjonslinje

Først tegnes alle datapunktene i et spredningsdiagram, og den lineære regresjonslinjen legges oppå.



Figur 1: Punktsky for terningdropp med regresjonslinje.

4.4 b) Posterior- og prediktive fordelinger

Med nøytral prior setter vi

$$\nu_0 = -2, \quad SS_0 = 0.$$

Da får vi posteriorfordelingene

$$\tau \mid \text{data} \sim \Gamma\left(\frac{\nu_1}{2}, \frac{SS_1}{2}\right),$$

$$b \mid \text{data} \sim t\left(\beta, s_1 \sqrt{\frac{1}{SS_x}}, \nu_1\right),$$

$$y(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right),$$

og

$$Y_+(x) \mid \text{data} \sim t\left(\alpha_0 + \beta x, s_1 \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}, \nu_1\right).$$

Siden $\sigma = 1/\sqrt{\tau}$, kan vi også utlede posterior usikkerhet for σ .

4.5 c) 80% kredibilitetsintervall for stigningstallet b

Intervallestimatet finnes fra posteriorfordelingen til b :

$$b \in \left[\beta - t_{\nu_1, 0.1} s_1 \sqrt{\frac{1}{SS_x}}, \beta + t_{\nu_1, 0.1} s_1 \sqrt{\frac{1}{SS_x}} \right].$$

De numeriske verdiene leses ut fra R-skriptet.

4.6 d) 80% kredibilitetsintervall for standardavviket σ

Her bruker vi sammenhengen mellom σ^2 og χ^2 -fordelingen. Intervallet kan beregnes direkte i R ut fra SS_1 og ν_1 .

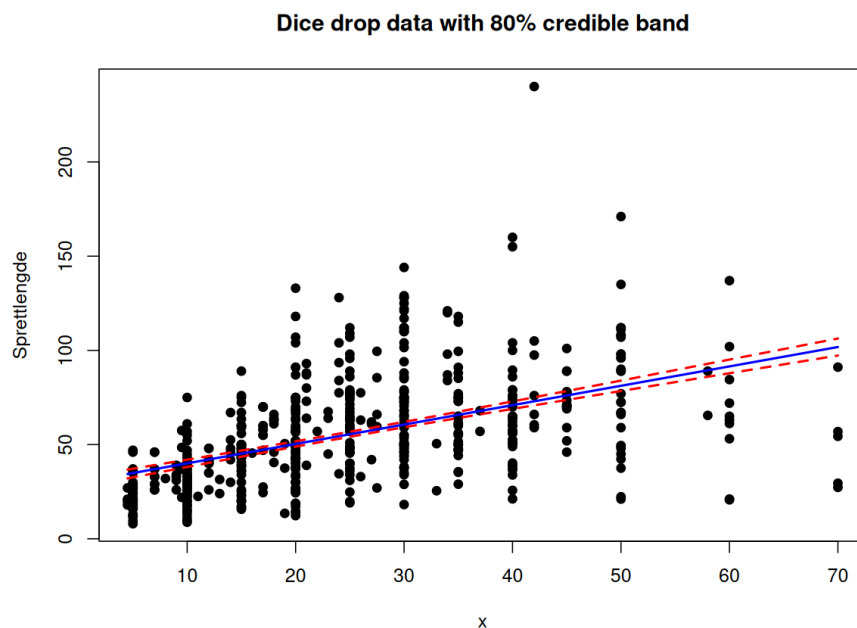
4.7 e) 80% kredibilitetsintervall for $y(x)$

For hver verdi av x får vi

$$I_{0.20}(x) = \alpha_0 + \beta x \pm t_{\nu_1, 0.1} s_1 \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{SS_x}}.$$

4.8 f) Kurver over og under regresjonslinjen

Intervallet i punkt e) gir to kurver: en øvre og en nedre. Disse plottes sammen med regresjonslinjen.



Figur 2: 80% kredibilitetsbånd for $y(x)$.

4.9 g) Forklaringsgrad R^2

Forklaringsgraden

$$R^2 = 1 - \frac{SS_e}{SS_y}$$

forteller hvor stor del av variasjonen i y som forklares av regresjonslinjen. Denne kan enten beregnes direkte fra sums of squares eller hentes fra `lm()` i R.

4.10 h) Regresjon mellom z og x , og mellom t og x

Oppgaven ber om en sammenligning av R^2 for

$$z \text{ mot } x \quad \text{og} \quad t \text{ mot } x.$$

I de tilgjengelige CSV-filene finnes det tydelige kolonner for x , y og z , men ingen entydig kolonne for t . Derfor kan analysen for z gjennomføres direkte, mens delen om t må enten utelates eller suppleres dersom tidsmålingene finnes i en annen fil.

4.11 R-kode

Listing 3 viser delen av skriptet som utfører regresjonen, skriver ut intervallene og lager figurene til oppgave 3.

```
dice_df <- read_all_dice_data(file.path(script_dir, "terningDroppFiler"))
dice_df <- dice_df[complete.cases(dice_df[, intersect(c("x", "y", "z"), names(dice_df))),
drop = FALSE], ]

dice_fit <- fit_simple_regression(dice_df$x, dice_df$y, nu0 = -2, SS0 = 0)
x_grid <- seq(min(dice_df$x), max(dice_df$x), length.out = 300)
cred_band_80 <- credible_band(dice_fit, x_grid, level = 0.80)
pred_band_80 <- predictive_band(dice_fit, x_grid, level = 0.80)

cat("\nTask 3: Dice drop data\n")
cat("-----\n")
cat("Number of observations =", nrow(dice_df), "\n")
cat("Regression line: y =", round(dice_fit$alpha0, 4), "+", round(dice_fit$beta, 4), "* x\n")
cat("80% interval for b:", paste(round(b_interval(dice_fit, 0.80), 4), collapse = " to "), "\n")
cat("80% interval for sigma:", paste(round(sigma_interval(dice_fit, 0.80), 4),
collapse = " to "), "\n")
cat("R^2 for y on x =", round(r_squared(dice_df$x, dice_df$y), 4), "\n")
cat("R^2 for z on x =", round(r_squared(dice_df$x, dice_df$z), 4), "\n")

plot_regression_with_band(
  df = dice_df,
  fit = dice_fit,
  band_df = cred_band_80,
  file_name = "task3_credible_band.png",
  ylab = "Sprettlengde",
  band_label = "Dice drop data with 80% credible band"
)
```

Listing 3: R-kode for analyse og figurer i terningdropp-oppgaven

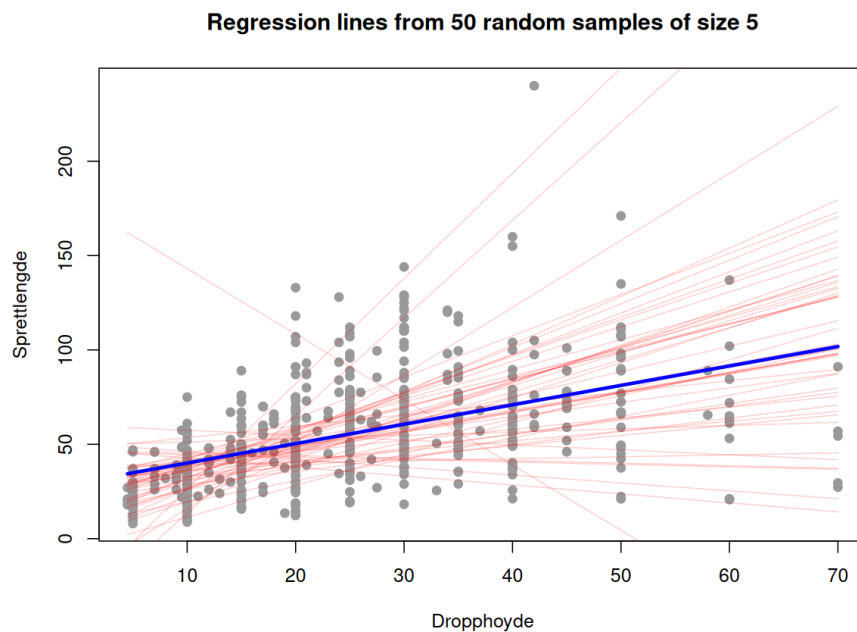
5 Oppgave 4: Utvalgsforsøk

5.1 Tema

I denne delen undersøker vi hvordan regresjonslinjen og intervallestimatene endrer seg når vi bare bruker tilfeldige delutvalg av observasjonene.

5.2 a) 50 runder med $N = 5$

Vi trekker 50 tilfeldige utvalg med $N = 5$, finner regresjonslinjen for hvert utvalg, og tegner alle linjene i samme figur.

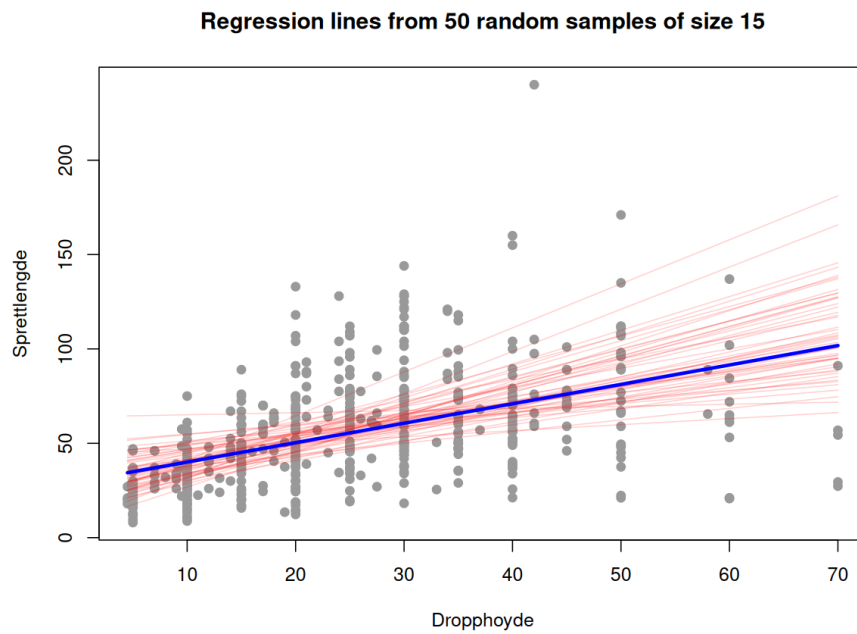


Figur 3: 50 regresjonslinjer basert på utvalg med $N = 5$.

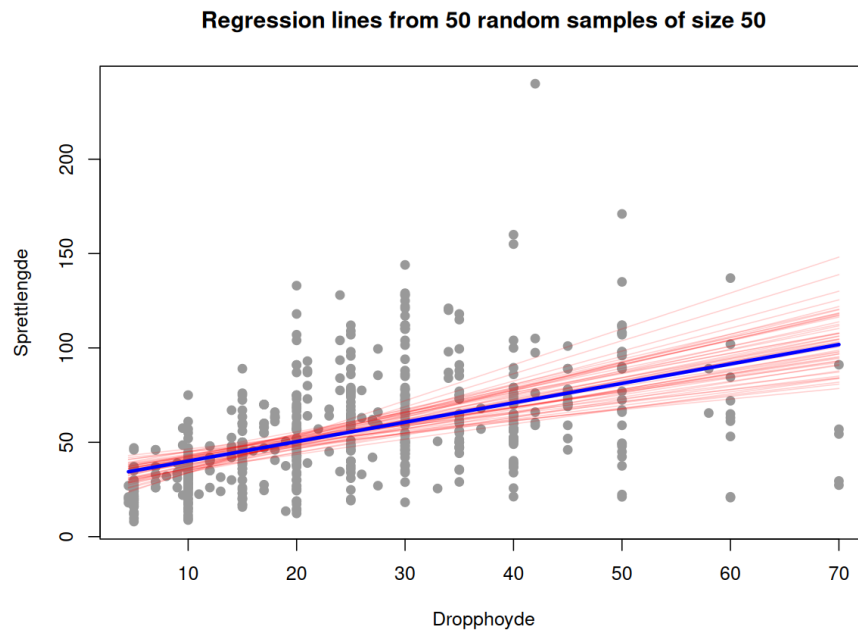
Det vi forventer å se, er stor variasjon fra linje til linje fordi utvalgene er små.

5.3 b) 50 runder med $N = 15, 50, 200$

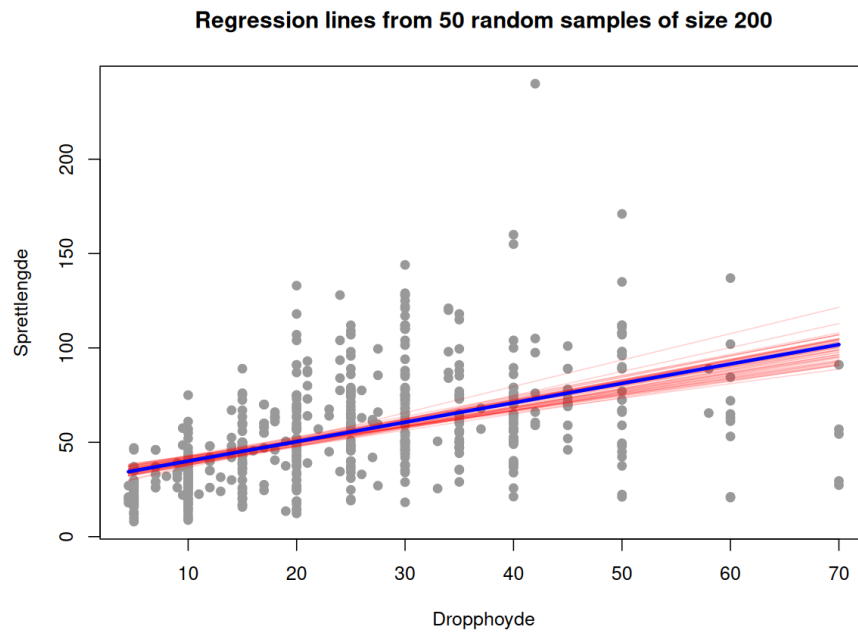
Vi gjentar samme prosedyre for større utvalg. Når N øker, bør linjene samle seg mer rundt regresjonslinjen for hele datasettet.



Figur 4: Regresjonslinjer for $N = 15$.



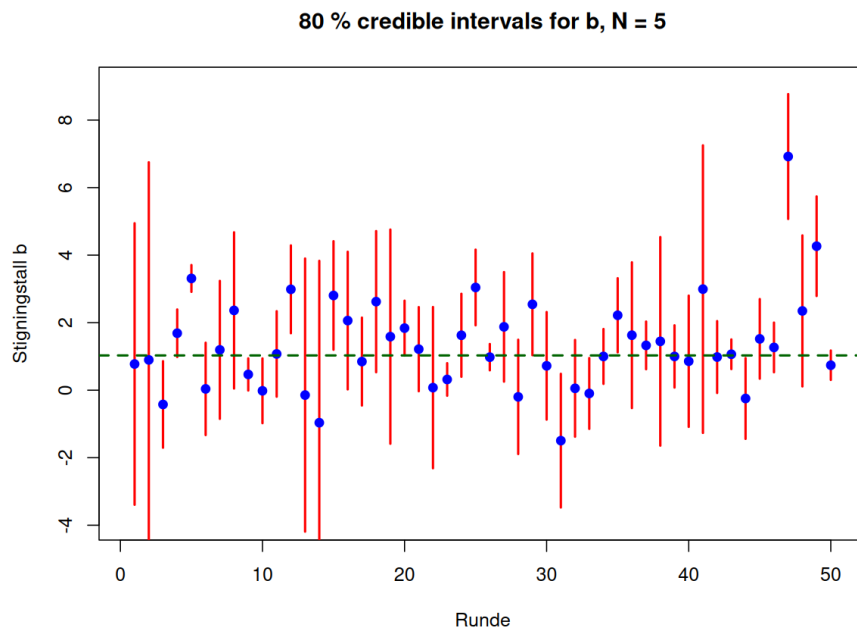
Figur 5: Regresjonslinjer for $N = 50$.



Figur 6: Regresjonslinjer for $N = 200$.

5.4 c) Oppgave 3c gjentatt 50 ganger med $N = 5$

Her beregner vi 80%-intervallestimatet for stigningstallet b i 50 runder med $N = 5$, og tegner intervallene samlet i én figur.

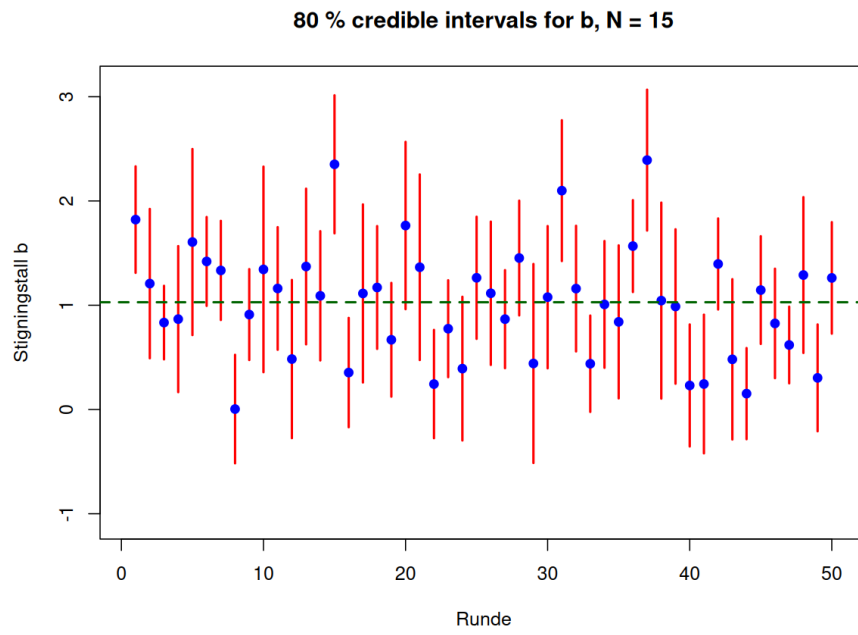


Figur 7: 50 intervallestimater for b når $N = 5$.

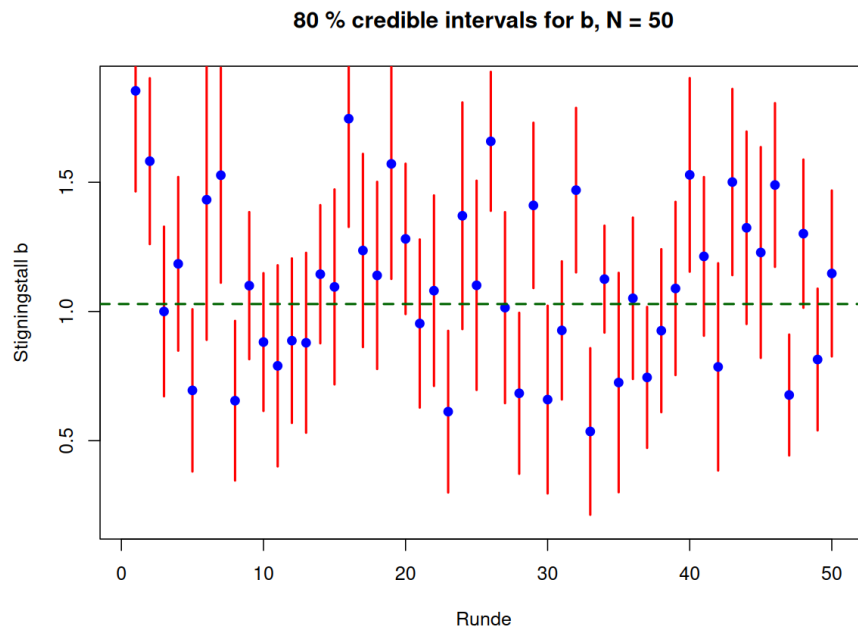
Små utvalg vil typisk gi brede intervaller og stor variasjon mellom rundene.

5.5 d) Samme analyse for $N = 15, 50, 200$

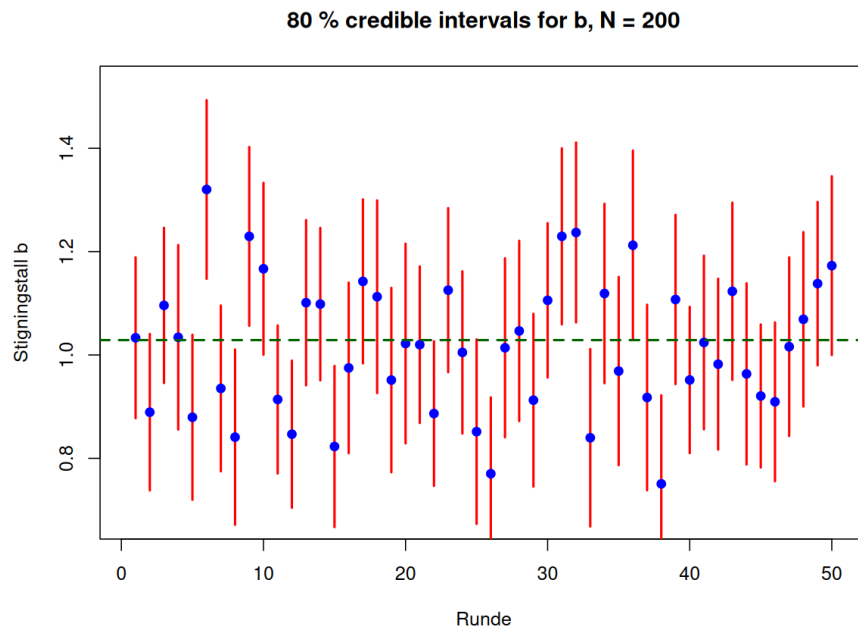
Når vi øker N , blir intervallene vanligvis smalere, og estimatene for b blir mer stabile.



Figur 8: Intervallestimator for b når $N = 15$.



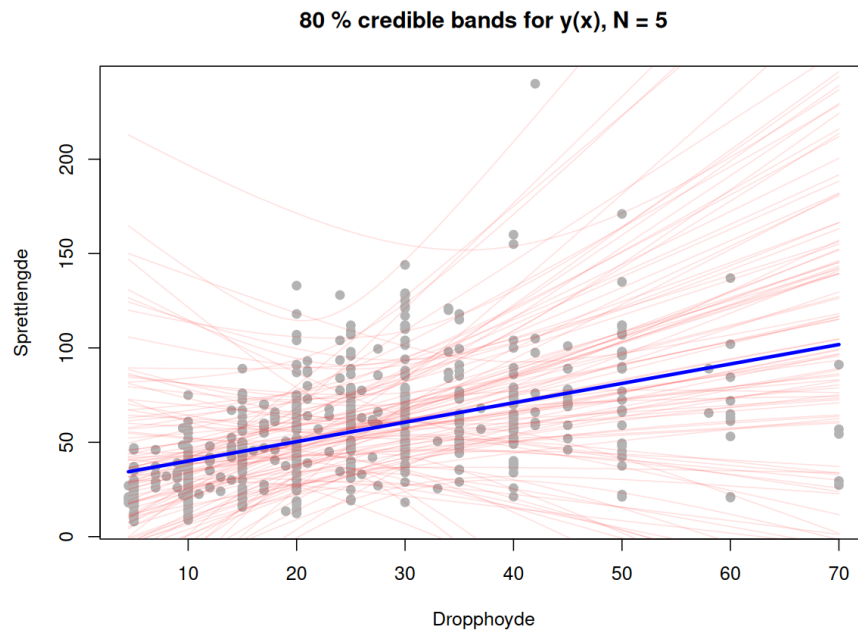
Figur 9: Intervallestimator for b når $N = 50$.



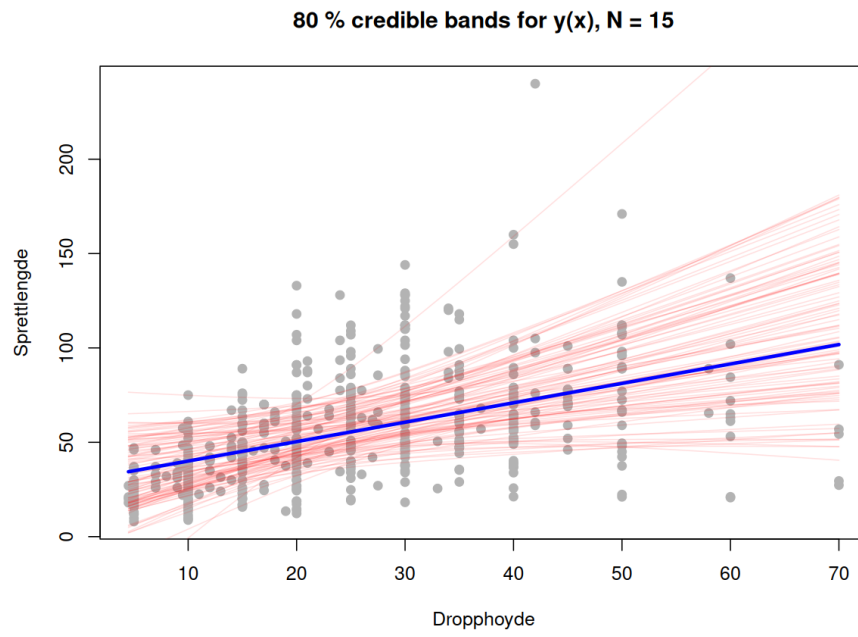
Figur 10: Intervallestimater for b når $N = 200$.

5.6 e) Illustrasjoner som i oppgave 3f

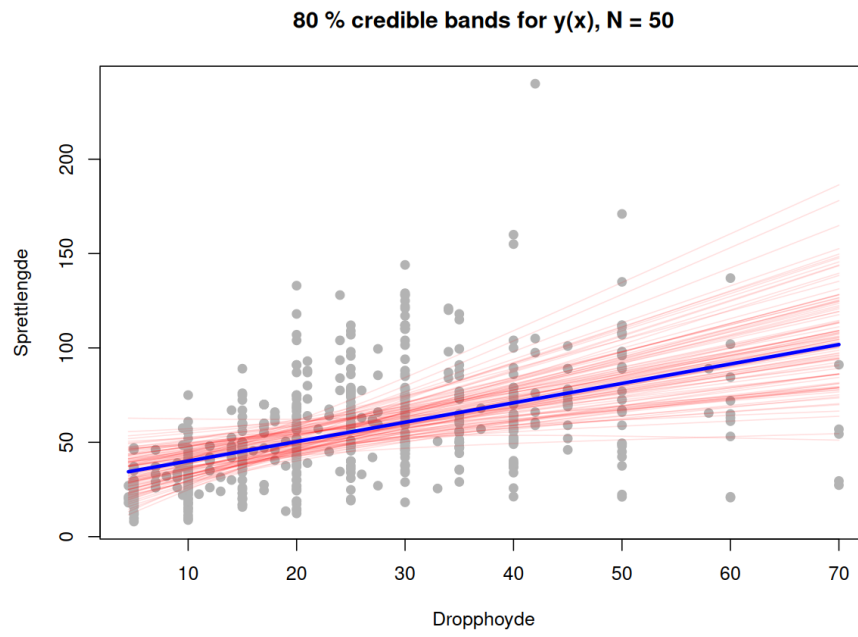
Til slutt lager vi figurer med regresjonslinje og tilhørende 80%-kredibilitetsbånd for utvalg med $N = 5, 15, 50$ og 200 .



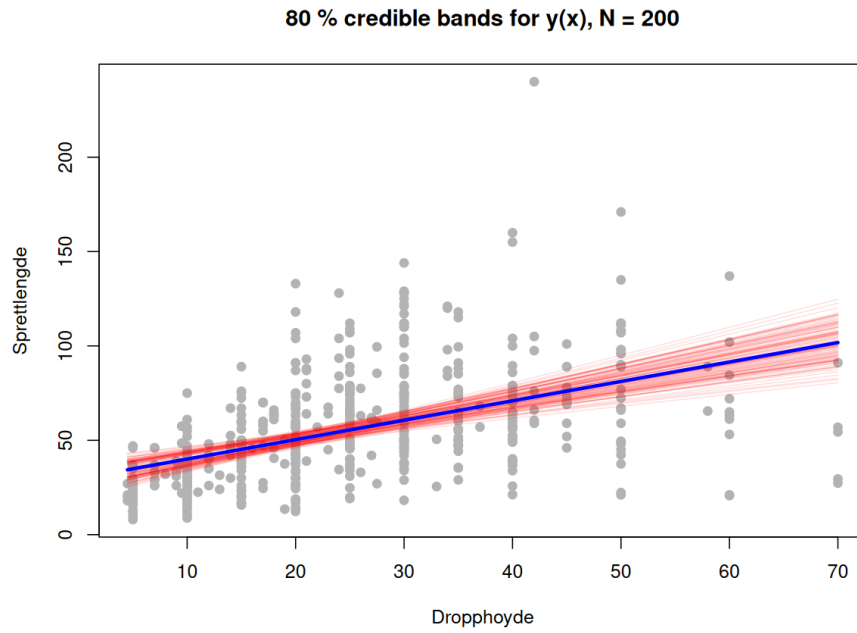
Figur 11: Kredibilitetsbånd for $N = 5$.



Figur 12: Kredibilitetsbånd for $N = 15$.



Figur 13: Kredibilitetsbånd for $N = 50$.



Figur 14: Kredibilitetsbånd for $N = 200$.

5.7 Kommentar

Hovedpoenget i denne oppgaven er å se hvordan usikkerheten minker når utvalgsstørrelsen øker. Små utvalg gir mer ustabile linjer og bredere intervaller, mens store utvalg gir mer presise estimater og tydeligere mønstre.

5.8 R-kode

Listing 4 viser delen av skriptet som trekker delutvalg og lager figurene for oppgave 4.

```
plot_many_sample_lines(dice_df, sample_size = 5, rounds = 50,
file_name = "task4a_lines_N5.png")
plot_many_sample_lines(dice_df, sample_size = 15, rounds = 50,
file_name = "task4b_lines_N15.png")
plot_many_sample_lines(dice_df, sample_size = 50, rounds = 50,
file_name = "task4b_lines_N50.png")
plot_many_sample_lines(dice_df, sample_size = 200, rounds = 50,
file_name = "task4b_lines_N200.png")

plot_many_b_intervals(dice_df, sample_size = 5, rounds = 50, level = 0.80,
file_name = "task4c_b_intervals_N5.png")
plot_many_b_intervals(dice_df, sample_size = 15, rounds = 50, level = 0.80,
file_name = "task4d_b_intervals_N15.png")
plot_many_b_intervals(dice_df, sample_size = 50, rounds = 50, level = 0.80,
file_name = "task4d_b_intervals_N50.png")
plot_many_b_intervals(dice_df, sample_size = 200, rounds = 50, level = 0.80,
file_name = "task4d_b_intervals_N200.png")

plot_many_credible_bands(dice_df, sample_size = 5, rounds = 50, level = 0.80,
file_name = "task4e_bands_N5.png")
plot_many_credible_bands(dice_df, sample_size = 15, rounds = 50, level = 0.80,
file_name = "task4e_bands_N15.png")
plot_many_credible_bands(dice_df, sample_size = 50, rounds = 50, level = 0.80,
file_name = "task4e_bands_N50.png")
plot_many_credible_bands(dice_df, sample_size = 200, rounds = 50, level = 0.80,
file_name = "task4e_bands_N200.png")
```

Listing 4: R-kode for gjentatte delutvalg og intervallillustrasjoner

Vedlegg

R-kode

```
1  # Oblig 3c: Linear regression with uncertainty
2
3  options(stringsAsFactors = FALSE)
4
5  script_path_arg <- grep("^--file=", commandArgs(trailingOnly = FALSE), value = TRUE)
6  script_dir <- if (length(script_path_arg) > 0) {
7    dirname(normalizePath(sub("^--file=", "", script_path_arg[1])))
8  } else {
9    getwd()
10 }
11
12 output_dir <- file.path(script_dir, "output")
13 if (!dir.exists(output_dir)) {
14   dir.create(output_dir, recursive = TRUE)
15 }
16
17 # -----
18 # Generic helper functions
19 # -----
20
21 strip_bom <- function(x) {
22   sub("^\\ufeff", "", x)
23 }
24
25 standardize_name <- function(x) {
26   cleaned <- tolower(strip_bom(iconv(x, to = "ASCII//TRANSLIT")))
27   cleaned <- gsub("[^a-z0-9]+", "", cleaned)
28   cleaned
29 }
30
31 read_dice_file <- function(path) {
32   raw_df <- read.csv(path, check.names = FALSE, fileEncoding = "UTF-8-BOM")
33   raw_df <- raw_df[, colSums(!is.na(raw_df)) > 0, drop = FALSE]
34
35   original_names <- names(raw_df)
36   clean_names <- vapply(original_names, standardize_name, character(1))
37
38   rename_map <- c(
39     "k" = "k",
40     "dropp" = "x",
41     "dropphoyde" = "x",
42     "x" = "x",
43     "lengde" = "y",
44     "sprettlengde" = "y",
45     "y" = "y",
46     "verdi" = "z",
47     "terningverdi" = "z",
48     "z" = "z",
```

```

49     "tid" = "t",
50     "t" = "t"
51 )
52
53 mapped_names <- rename_map[clean_names]
54 names(raw_df) <- ifelse(is.na(mapped_names), clean_names, mapped_names)
55
56 keep <- intersect(c("k", "x", "y", "z", "t"), names(raw_df))
57 out <- raw_df[, keep, drop = FALSE]
58 out$source_file <- basename(path)
59
60 for (name in setdiff(names(out), "source_file")) {
61   out[[name]] <- suppressWarnings(as.numeric(out[[name]]))
62 }
63
64 out
65 }
66
67 read_all_dice_data <- function(folder) {
68   files <- list.files(folder, pattern = "\\*.csv$", full.names = TRUE)
69   df_list <- lapply(files, read_dice_file)
70   combined <- do.call(rbind, df_list)
71   rownames(combined) <- NULL
72   combined
73 }
74
75 fit_simple_regression <- function(x, y, nu0 = -2, SS0 = 0) {
76   stopifnot(length(x) == length(y))
77
78   n <- length(x)
79   x_bar <- mean(x)
80   x_centered <- x - x_bar
81   SSx <- sum(x_centered^2)
82
83   beta_hat <- sum(x_centered * y) / SSx
84   alpha_star <- mean(y)
85   alpha0 <- alpha_star - beta_hat * x_bar
86
87   fitted <- alpha0 + beta_hat * x
88   residuals <- y - fitted
89   SSe <- sum(residuals^2)
90
91   nu1 <- nu0 + n
92   SS1 <- SS0 + SSe
93   s1 <- sqrt(SS1 / nu1)
94
95   list(
96     n = n,
97     x = x,
98     y = y,
99     x_bar = x_bar,
100    SSx = SSx,
101    alpha0 = alpha0,

```

```

102     alpha_star = alpha_star,
103     beta = beta_hat,
104     fitted = fitted,
105     residuals = residuals,
106     SSe = SSe,
107     nu0 = nu0,
108     SS0 = SS0,
109     nu1 = nu1,
110     SS1 = SS1,
111     s1 = s1
112   )
113 }
114
115 tau_posterior_parameters <- function(fit) {
116   list(shape = fit$nu1 / 2, rate = fit$SS1 / 2)
117 }
118
119 sigma_interval <- function(fit, level = 0.80) {
120   alpha <- 1 - level
121   lower_var <- fit$SS1 / qchisq(1 - alpha / 2, df = fit$nu1)
122   upper_var <- fit$SS1 / qchisq(alpha / 2, df = fit$nu1)
123   c(lower = sqrt(lower_var), upper = sqrt(upper_var))
124 }
125
126 b_interval <- function(fit, level = 0.80) {
127   alpha <- 1 - level
128   t_crit <- qt(1 - alpha / 2, df = fit$nu1)
129   margin <- t_crit * fit$s1 / sqrt(fit$SSx)
130   c(lower = fit$beta - margin, upper = fit$beta + margin)
131 }
132
133 y_posterior_parameters <- function(fit, x_new) {
134   scale <- fit$s1 * sqrt(1 / fit$n + (x_new - fit$x_bar)^2 / fit$SSx)
135   list(mean = fit$alpha0 + fit$beta * x_new, scale = scale, df = fit$nu1)
136 }
137
138 y_predictive_parameters <- function(fit, x_new) {
139   scale <- fit$s1 * sqrt(1 + 1 / fit$n + (x_new - fit$x_bar)^2 / fit$SSx)
140   list(mean = fit$alpha0 + fit$beta * x_new, scale = scale, df = fit$nu1)
141 }
142
143 t_interval <- function(mean, scale, df, level) {
144   alpha <- 1 - level
145   t_crit <- qt(1 - alpha / 2, df = df)
146   c(lower = mean - t_crit * scale, upper = mean + t_crit * scale)
147 }
148
149 credible_band <- function(fit, x_grid, level = 0.80) {
150   alpha <- 1 - level
151   t_crit <- qt(1 - alpha / 2, df = fit$nu1)
152   mean_curve <- fit$alpha0 + fit$beta * x_grid
153   margin <- t_crit * fit$s1 * sqrt(1 / fit$n + (x_grid - fit$x_bar)^2 / fit$SSx)

```



```

154   data.frame(x = x_grid, mean = mean_curve, lower = mean_curve - margin, upper =
      ↪ mean_curve + margin)
155 }
156
157 predictive_band <- function(fit, x_grid, level = 0.80) {
158   alpha <- 1 - level
159   t_crit <- qt(1 - alpha / 2, df = fit$nul)
160   mean_curve <- fit$alpha0 + fit$beta * x_grid
161   margin <- t_crit * fit$s1 * sqrt(1 + 1 / fit$n + (x_grid - fit$x_bar)^2 / fit$SSx)
162   data.frame(x = x_grid, mean = mean_curve, lower = mean_curve - margin, upper =
      ↪ mean_curve + margin)
163 }
164
165 r_squared <- function(x, y) {
166   fit <- lm(y ~ x)
167   summary(fit)$r.squared
168 }
169
170 print_regression_summary <- function(label, fit, x_eval = NULL, level_y = 0.80, level_pred
      ↪ = 0.80) {
171   tau_post <- tau_posterior_parameters(fit)
172
173   cat("\n", label, "\n", sep = "")
174   cat(strep("-", nchar(label)), "\n", sep = "")
175   cat("n =", fit$n, "\n")
176   cat("x_bar =", fit$x_bar, "\n")
177   cat("alpha0 =", fit$alpha0, "\n")
178   cat("alpha* =", fit$alpha_star, "\n")
179   cat("beta =", fit$beta, "\n")
180   cat("SSe =", fit$SSe, "\n")
181   cat("Posterior tau ~ Gamma(shape =", tau_post$shape, ", rate =", tau_post$rate, ")\n")
182
183   if (!is.null(x_eval)) {
184     y_post <- y_posterior_parameters(fit, x_eval)
185     y_pred <- y_predictive_parameters(fit, x_eval)
186
187     cat("Posterior y(", x_eval, ") ~ t(mean =", y_post$mean, ", scale =", y_post$scale, ",
      ↪ df =", y_post$df, ")\n", sep = "")
188     cat("Predictive Y+(", x_eval, ") ~ t(mean =", y_pred$mean, ", scale =", y_pred$scale,
      ↪ ", df =", y_pred$df, ")\n", sep = "")
189     cat(level_y * 100, "% credible interval for y(", x_eval, "): ",
      ↪ paste(round(t_interval(y_post$mean, y_post$scale, y_post$df, level_y), 4),
      ↪ collapse = " to "), "\n", sep = "")
190     cat(level_pred * 100, "% predictive interval for Y+(", x_eval, "): ",
      ↪ paste(round(t_interval(y_pred$mean, y_pred$scale, y_pred$df, level_pred), 4),
      ↪ collapse = " to "), "\n", sep = "")
191   }
192 }
193
194 plot_regression_with_band <- function(df, fit, band_df, file_name, ylab, band_label) {
195   png(file.path(output_dir, file_name), width = 1200, height = 900, res = 150)
196   plot(df$x, df$y,

```

```

197     pch = 19, col = "black",
198     xlab = "x", ylab = ylab,
199     main = band_label)
200   lines(band_df$x, band_df$mean, col = "blue", lwd = 2)
201   lines(band_df$x, band_df$lower, col = "red", lwd = 2, lty = 2)
202   lines(band_df$x, band_df$upper, col = "red", lwd = 2, lty = 2)
203   dev.off()
204 }
205
206 plot_regression_only <- function(df, fit, file_name, ylab, plot_title) {
207   x_grid <- seq(min(df$x), max(df$x), length.out = 300)
208   png(file.path(output_dir, file_name), width = 1200, height = 900, res = 150)
209   plot(df$x, df$y,
210        pch = 19, col = "black",
211        xlab = "Dropphoyde", ylab = ylab,
212        main = plot_title)
213   lines(x_grid, fit$alpha0 + fit$beta * x_grid, col = "blue", lwd = 2)
214   dev.off()
215 }
216
217 plot_many_sample_lines <- function(df, sample_size, rounds, file_name) {
218   x_grid <- seq(min(df$x), max(df$x), length.out = 200)
219   full_fit <- fit_simple_regression(df$x, df$y)
220
221   png(file.path(output_dir, file_name), width = 1200, height = 900, res = 150)
222   plot(df$x, df$y,
223        pch = 19, col = "grey60",
224        xlab = "Dropphoyde", ylab = "Sprettlengde",
225        main = paste("Regression lines from", rounds, "random samples of size",
226                     ↪ sample_size))
227
228   for (i in seq_len(rounds)) {
229     sample_index <- sort(sample(seq_len(nrow(df)), sample_size))
230     sample_fit <- fit_simple_regression(df$x[sample_index], df$y[sample_index])
231     y_grid <- sample_fit$alpha0 + sample_fit$beta * x_grid
232     lines(x_grid, y_grid, col = rgb(1, 0, 0, alpha = 0.18), lwd = 1)
233   }
234
235   lines(x_grid, full_fit$alpha0 + full_fit$beta * x_grid, col = "blue", lwd = 3)
236   dev.off()
237 }
238
239 plot_many_b_intervals <- function(df, sample_size, rounds, level = 0.80, file_name) {
240   png(file.path(output_dir, file_name), width = 1200, height = 900, res = 150)
241   plot(NA,
242        xlim = c(0.5, rounds + 0.5),
243        ylim = range(vapply(seq_len(rounds), function(i) {
244          sample_index <- sort(sample(seq_len(nrow(df)), sample_size))
245          fit <- fit_simple_regression(df$x[sample_index], df$y[sample_index])
246          b_interval(fit, level)
247        }, numeric(2)))),
248        xlab = "Runde", ylab = "Stigningstall b",

```

```

248     main = paste(round(level * 100), "% credible intervals for b,", "N =",
249                 ↪ sample_size))
250
251 for (i in seq_len(rounds)) {
252     sample_index <- sort(sample(seq_len(nrow(df)), sample_size))
253     fit <- fit_simple_regression(df$x[sample_index], df$y[sample_index])
254     interval <- b_interval(fit, level)
255     segments(i, interval["lower"], i, interval["upper"], col = "red", lwd = 2)
256     points(i, fit$beta, pch = 19, col = "blue")
257 }
258
259 abline(h = fit_simple_regression(df$x, df$y)$beta, col = "darkgreen", lwd = 2, lty = 2)
260 dev.off()
261
262 plot_many_credible_bands <- function(df, sample_size, rounds, level = 0.80, file_name) {
263     x_grid <- seq(min(df$x), max(df$x), length.out = 200)
264
265     png(file.path(output_dir, file_name), width = 1200, height = 900, res = 150)
266     plot(df$x, df$y,
267         pch = 19, col = "grey70",
268         xlab = "Dropphoyde", ylab = "Sprettlengde",
269         main = paste(round(level * 100), "% credible bands for y(x), N =", sample_size))
270
271     for (i in seq_len(rounds)) {
272         sample_index <- sort(sample(seq_len(nrow(df)), sample_size))
273         fit <- fit_simple_regression(df$x[sample_index], df$y[sample_index])
274         band <- credible_band(fit, x_grid, level)
275         lines(band$x, band$lower, col = rgb(1, 0, 0, alpha = 0.12), lwd = 1)
276         lines(band$x, band$upper, col = rgb(1, 0, 0, alpha = 0.12), lwd = 1)
277     }
278
279     full_fit <- fit_simple_regression(df$x, df$y)
280     lines(x_grid, full_fit$alpha0 + full_fit$beta * x_grid, col = "blue", lwd = 3)
281     dev.off()
282 }
283
284 # -----
285 # Book task 17.1c
286 # -----
287
288 task_1c <- data.frame(
289     x = c(2, 3, 4, 6),
290     y = c(10, 8, 8, 7)
291 )
292
293 fit_1c <- fit_simple_regression(task_1c$x, task_1c$y, nu0 = 4 - 2, SS0 = 0.5^2 * (4 - 2))
294 print_regression_summary(
295     label = "Task 1: Chapter 17, problem 1c",
296     fit = fit_1c,
297     x_eval = mean(task_1c$x),
298     level_y = 0.95,

```

```

299     level_pred = 0.95
300 )
301
302 # -----
303 # Book task 17.1d
304 # -----
305
306 task_1d <- data.frame(
307   x = c(0, 1, 2, 3),
308   y = c(0, 2, 7, 5)
309 )
310
311 fit_1d <- fit_simple_regression(task_1d$x, task_1d$y, nu0 = -2, SS0 = 0)
312 print_regression_summary(
313   label = "Task 2: Chapter 17, problem 1d",
314   fit = fit_1d,
315   x_eval = mean(task_1d$x),
316   level_y = 0.90,
317   level_pred = 0.95
318 )
319
320 # -----
321 # Dice-drop assignment
322 # -----
323
324 dice_df <- read_all_dice_data(file.path(script_dir, "terningDroppFile"))
325 dice_df <- dice_df[complete.cases(dice_df[, intersect(c("x", "y", "z"), names(dice_df))),
↪   drop = FALSE), ]
326
327 dice_fit <- fit_simple_regression(dice_df$x, dice_df$y, nu0 = -2, SS0 = 0)
328 x_grid <- seq(min(dice_df$x), max(dice_df$x), length.out = 300)
329 cred_band_80 <- credible_band(dice_fit, x_grid, level = 0.80)
330 pred_band_80 <- predictive_band(dice_fit, x_grid, level = 0.80)
331
332 cat("\nTask 3: Dice drop data\n")
333 cat("-----\n")
334 cat("Number of observations =", nrow(dice_df), "\n")
335 cat("Regression line: y =", round(dice_fit$alpha0, 4), "+", round(dice_fit$beta, 4), "*"
↪   x\n")
336 cat("80% interval for b:", paste(round(b_interval(dice_fit, 0.80), 4), collapse = " to "),
↪   "\n")
337 cat("80% interval for sigma:", paste(round(sigma_interval(dice_fit, 0.80), 4), collapse =
↪   " to "), "\n")
338 cat("R^2 for y on x =", round(r_squared(dice_df$x, dice_df$y), 4), "\n")
339 cat("R^2 for z on x =", round(r_squared(dice_df$x, dice_df$z), 4), "\n")
340
341 if ("t" %in% names(dice_df)) {
342   cat("R^2 for t on x =", round(r_squared(dice_df$x, dice_df$t), 4), "\n")
343 } else {
344   cat("No time variable t was found in the CSV files, so task 3h can only be completed for
↪   z versus x with the current dataset.\n")
345 }

```

```

346
347 plot_regression_with_band(
348     df = dice_df,
349     fit = dice_fit,
350     band_df = cred_band_80,
351     file_name = "task3_credible_band.png",
352     ylab = "Sprettlengde",
353     band_label = "Dice drop data with 80% credible band"
354 )
355
356 plot_regression_only(
357     df = dice_df,
358     fit = dice_fit,
359     file_name = "task3_scatter_regression.png",
360     ylab = "Sprettlengde",
361     plot_title = "Dice drop data with regression line"
362 )
363
364 plot_regression_with_band(
365     df = dice_df,
366     fit = dice_fit,
367     band_df = pred_band_80,
368     file_name = "task3_predictive_band.png",
369     ylab = "Sprettlengde",
370     band_label = "Dice drop data with 80% predictive band"
371 )
372
373 # -----
374 # Repeated subsample experiments
375 # -----
376
377 set.seed(1234)
378
379 plot_many_sample_lines(dice_df, sample_size = 5, rounds = 50, file_name =
↪ "task4a_lines_N5.png")
380 plot_many_sample_lines(dice_df, sample_size = 15, rounds = 50, file_name =
↪ "task4b_lines_N15.png")
381 plot_many_sample_lines(dice_df, sample_size = 50, rounds = 50, file_name =
↪ "task4b_lines_N50.png")
382 plot_many_sample_lines(dice_df, sample_size = 200, rounds = 50, file_name =
↪ "task4b_lines_N200.png")
383
384 plot_many_b_intervals(dice_df, sample_size = 5, rounds = 50, level = 0.80, file_name =
↪ "task4c_b_intervals_N5.png")
385 plot_many_b_intervals(dice_df, sample_size = 15, rounds = 50, level = 0.80, file_name =
↪ "task4d_b_intervals_N15.png")
386 plot_many_b_intervals(dice_df, sample_size = 50, rounds = 50, level = 0.80, file_name =
↪ "task4d_b_intervals_N50.png")
387 plot_many_b_intervals(dice_df, sample_size = 200, rounds = 50, level = 0.80, file_name =
↪ "task4d_b_intervals_N200.png")
388

```

```

389 plot_many_credible_bands(dice_df, sample_size = 5, rounds = 50, level = 0.80, file_name =
↪ "task4e_bands_N5.png")
390 plot_many_credible_bands(dice_df, sample_size = 15, rounds = 50, level = 0.80, file_name =
↪ "task4e_bands_N15.png")
391 plot_many_credible_bands(dice_df, sample_size = 50, rounds = 50, level = 0.80, file_name =
↪ "task4e_bands_N50.png")
392 plot_many_credible_bands(dice_df, sample_size = 200, rounds = 50, level = 0.80, file_name
↪ = "task4e_bands_N200.png")
393
394 cat("\nTask 4 plots written to:", normalizePath(output_dir), "\n")

```

R-kode for hele rapporten