Income and well-being: not resolved

Ruben Arslan

2023-08-14

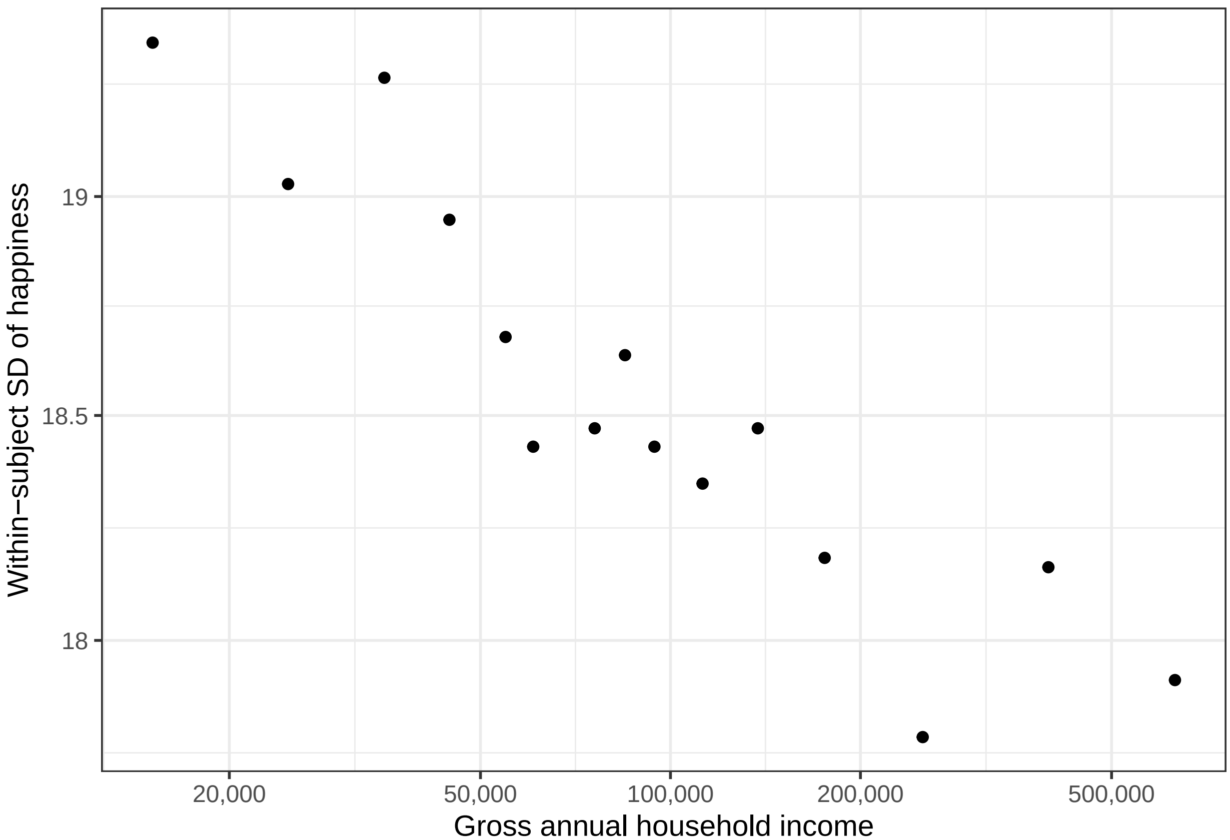
Contents


```
library(tidyverse)
library(brms)
options(mc.cores = parallel::detectCores() - 4,
        brms.backend = "cmdstanr",
        brms.file_refit = "never")

esd <- rio::import("https://osf.io/download/kpnjf/", format = "csv") %>%
  tibble()
```

1 Within-subject SDs

```
killingsworth_within_subject_sds <- readr::read_csv("
log_income, within_subject_sd
9.623577235772357, 19.359504132231404
10.117886178861788, 19.02892561983471
10.46910569105691, 19.27685950413223
10.70650406504065, 18.946280991735534
10.911382113821137, 18.67768595041322
11.012195121951219, 18.4297520661157
11.236585365853658, 18.471074380165287
11.347154471544716, 18.636363636363633
11.454471544715446, 18.4297520661157
11.630081300813007, 18.34710743801653
11.831707317073171, 18.471074380165287
12.075609756097561, 18.18181818181818
12.433333333333334, 17.789256198347104
12.891869918699186, 18.161157024793386
13.353658536585366, 17.913223140495866
```

```r
")

library(ggplot2)
library(tidyverse)
ggplot(killingsworth_within_subject_sds,
       aes(log_income, log(within_subject_sd))) +
  geom_point() +
  scale_y_continuous("Within-subject SD of happiness",
                     breaks = log(c(18, 18.5, 19, 19.5)),
                     labels = c(18, 18.5, 19, 19.5)) +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000")) +
  theme_bw()
```




```r
lm(log(within_subject_sd) ~ log_income, killingsworth_within_subject_sds)
```

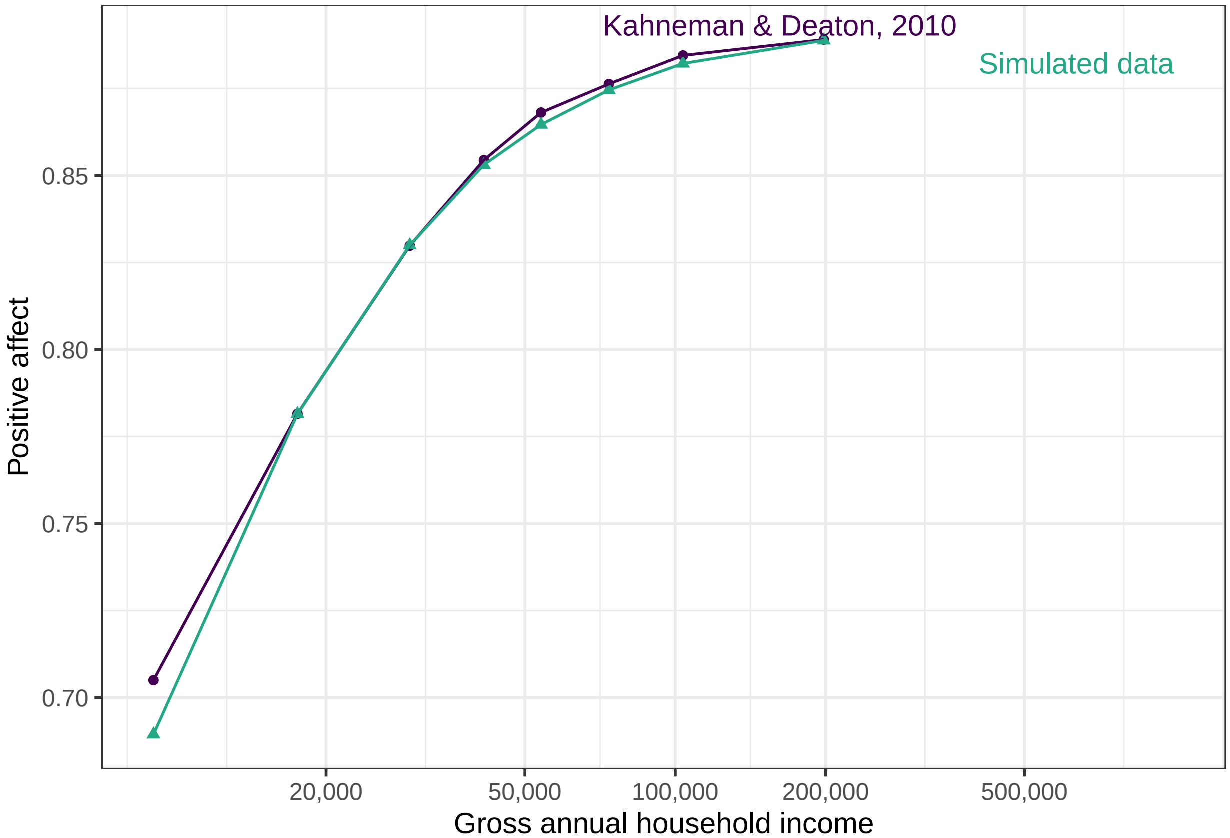
```
##
## Call:
## lm(formula = log(within_subject_sd) ~ log_income, data = killingsworth_within_subject_sds)
##
## Coefficients:
## (Intercept)   log_income
##     3.17420     -0.02231
```

```r
ggsave("Figure2.pdf", width = 9, height = 6, units = "cm", scale = 2)
# ggsave("Figure2.png", width = 7.5, height = 6)
```

2 Simulation

```r
# load empirical pattern reported in Kahneman & Deaton 2010
kd_graph <- readr::read_csv("income, happy
9025.71216506263, 0.7050113895216401
17530.402913291244, 0.7815489749430525
29410.63692902389, 0.8298405466970389
41390.388056412914, 0.8544419134396356
53873.56232388553, 0.8681093394077449
73629.38698516181, 0.8763097949886106
103620.6358613122, 0.8845102505694762
198333.86843073, 0.8890660592255126
")

set.seed(102019)
n <- 500000
b0 <- -4           # intercept of latent happiness
b <- 0.50           # effect of log income on happiness
b0_sigma <- 0.30 # intercept for the log standard deviation
b_sigma <- -0.05   # decrease of within-subject SD with log income
                   # ca double the observed relationship in Killingsworth's data
                   # reflecting my assumption that that estimate is attenuated
                   # by measurement error
misclassif <- .11  # misclassification rate yes/no in outcome

simulated_data <- tibble(
  income = sample(kd_graph$income, size = n, replace = T), # uniform distribution
  log_income = log(income),
  # latent happiness is normally distributed as a function of log income, varies
  # less within-subject at higher log incomes
  happiness = rnorm(n = n,
                  mean = b0 + b * log_income,
                  sd = exp(b0_sigma + b_sigma * log_income)),
  # to obtain measured happiness, we assume people introspect with error
  happiness_m = happiness + rnorm(n, 0, 1.3),
  # they decide their binary yes/no response based on this introspection
  happiness_b = if_else(happiness < 0, 0, 1),
  # this response is sometimes misreported/misclassified
  happiness_bm = if_else(runif(n) < misclassif, 1 - happiness_b, happiness_b)
)

simulated_data_aggregated <- simulated_data %>% group_by(log_income, income) %>%
  summarise(happiness_bm = mean(happiness_bm))
simulated_data_aggregated <- simulated_data_aggregated %>% left_join(kd_graph)

ggplot(simulated_data_aggregated %>%
         select(`Kahneman & Deaton, 2010` = happy,
                `Simulated data` = happiness_bm, log_income) %>%
         pivot_longer(-log_income),
       aes(log_income, value, color = name, shape = name)) +
```

```r
  geom_point() +
  geom_line() +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000"),
                     limits = log(c(9000, 1000000))) +
  scale_color_viridis_d(end = 0.6, guide = "none") +
  ggrepel::geom_text_repel(aes(
    label = if_else(log_income > 12, name, NA_character_), x = log(400000)),
    size = 4, hjust = 0) +
  ylab("Positive affect") +
  scale_shape(guide = "none") +
  theme_bw()
```



3 Reanalysis

3.1 Model comparison

```r
m_homoskedasticity <- brm(wellbeing ~ log_income, data = esd,
                          file = "m_homoskedasticity") %>%
  add_criterion("loo")
m_continuous_var_increase <- brm(bf(
  wellbeing ~ log_income,
```

```r
  sigma ~ log_income), data = esd,
  file = "m_continuous_var_increase") %>%
  add_criterion("loo")
m_flattening_at_100 <- brm(bf(
  wellbeing ~ log_income + log_income: income_above_100,
  sigma ~ log_income + log_income: income_above_100 ), data = esd,
  file = "m_flattening_at_100") %>%
  add_criterion("loo")
m_spline_sigma <- brm(bf(wellbeing ~ log_income,
             sigma ~ s(log_income)), data = esd, file = "m_spline_sigma") %>%
  add_criterion("loo")
m_spline_mu_sigma <- brm(bf(wellbeing ~ s(log_income),
             sigma ~ s(log_income)), data = esd,
             control = list(adapt_delta = 0.99),
             file = "m_spline_mu_sigma") %>%
  add_criterion("loo")

loo_compare(m_homoskedasticity, m_continuous_var_increase, m_flattening_at_100,
            m_spline_sigma, m_spline_mu_sigma)
```

```
##                           elpd_diff se_diff
## m_spline_sigma              0.0       0.0
## m_spline_mu_sigma          -0.9       0.4
## m_continuous_var_increase -10.9       6.2
## m_flattening_at_100       -12.5       6.3
## m_homoskedasticity        -16.7       8.0
```

```r
loo_compare(m_homoskedasticity, m_flattening_at_100)
```

```
##                     elpd_diff se_diff
## m_flattening_at_100  0.0       0.0
## m_homoskedasticity  -4.2       4.8
```

```r
loo_compare(m_continuous_var_increase, m_flattening_at_100)
```

```
##                           elpd_diff se_diff
## m_continuous_var_increase  0.0       0.0
## m_flattening_at_100       -1.5       1.4
```

```r
loo_compare(m_homoskedasticity, m_continuous_var_increase)
```

```
##                           elpd_diff se_diff
## m_continuous_var_increase  0.0       0.0
## m_homoskedasticity        -5.7       4.7
```

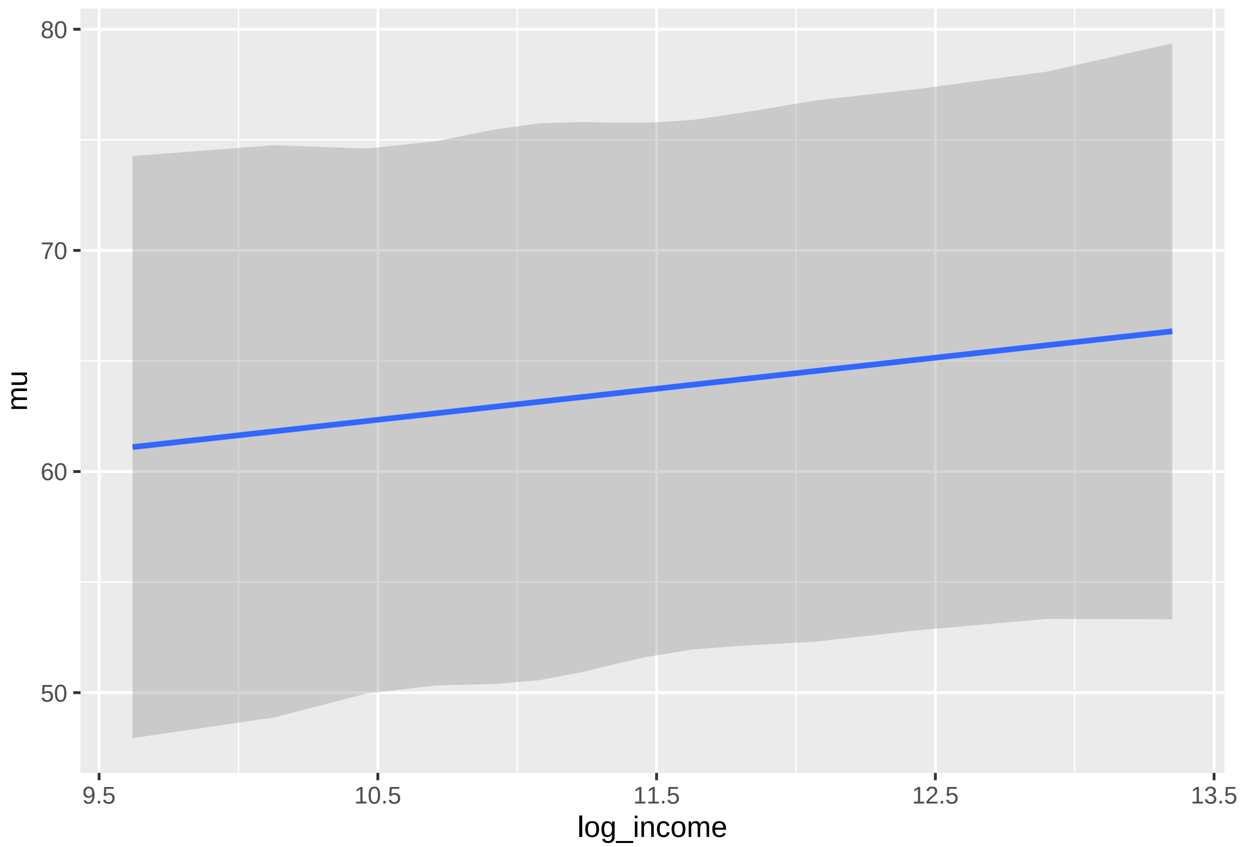
```r
loo_compare(m_homoskedasticity, m_spline_sigma)
```

```
##                    elpd_diff se_diff
## m_spline_sigma       0.0       0.0
## m_homoskedasticity -16.7       8.0
```

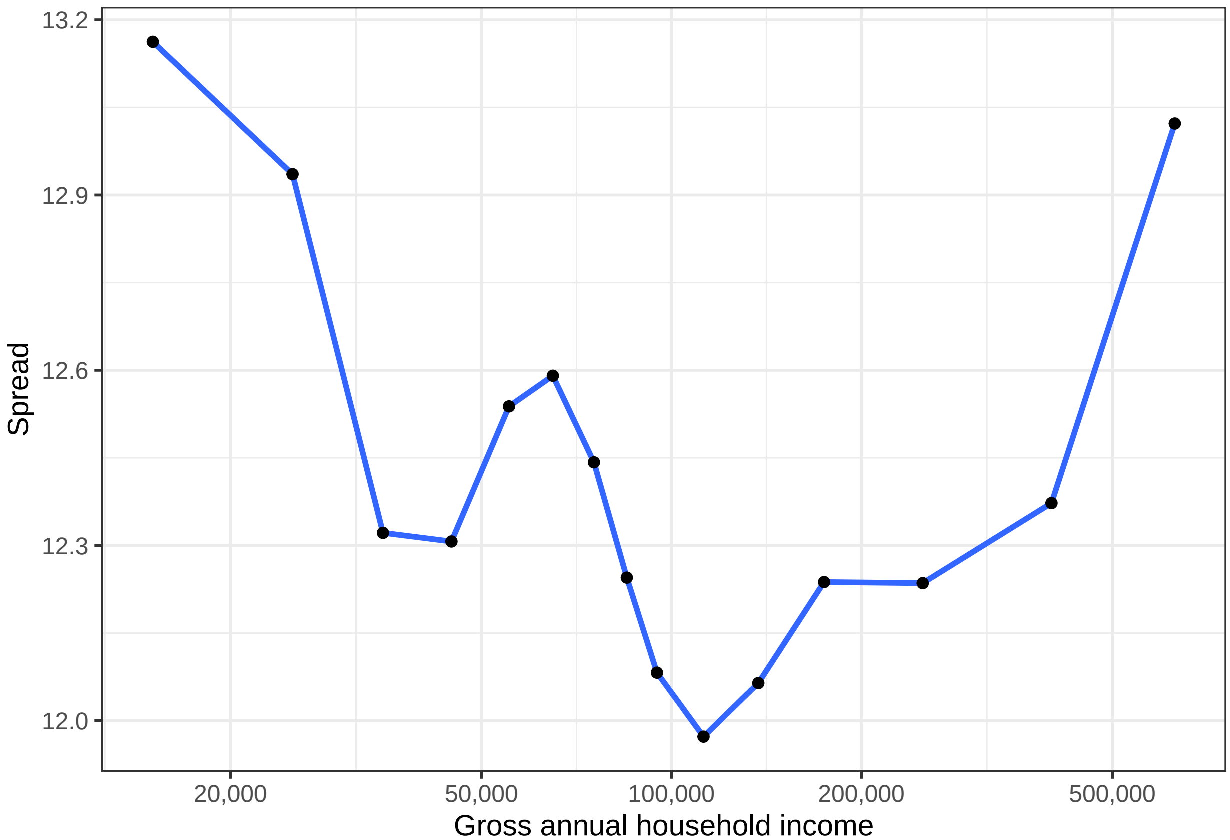
The threshold model barely outperforms the simple model assuming homoskedasticity in LOO. The spline model does better but actually shows an increase in variability at low incomes.

```r
mus <- fitted(m_spline_sigma,
              newdata = esd %>% distinct(log_income),
              summary = T, dpar = "mu")
```

```r
sigmas <- fitted(m_spline_sigma,
                 newdata = esd %>% distinct(log_income),
                 summary = T, dpar = "sigma")
plotdata <- tibble(
  log_income = esd %>% distinct(log_income) %>% pull(log_income),
  mu = mus[,"Estimate"],
  sigma = sigmas[,"Estimate"])

ggplot(plotdata,
       aes(log_income, y = mu, ymin = mu - sigma, ymax = mu + sigma)) +
  geom_smooth(stat = "identity")
```

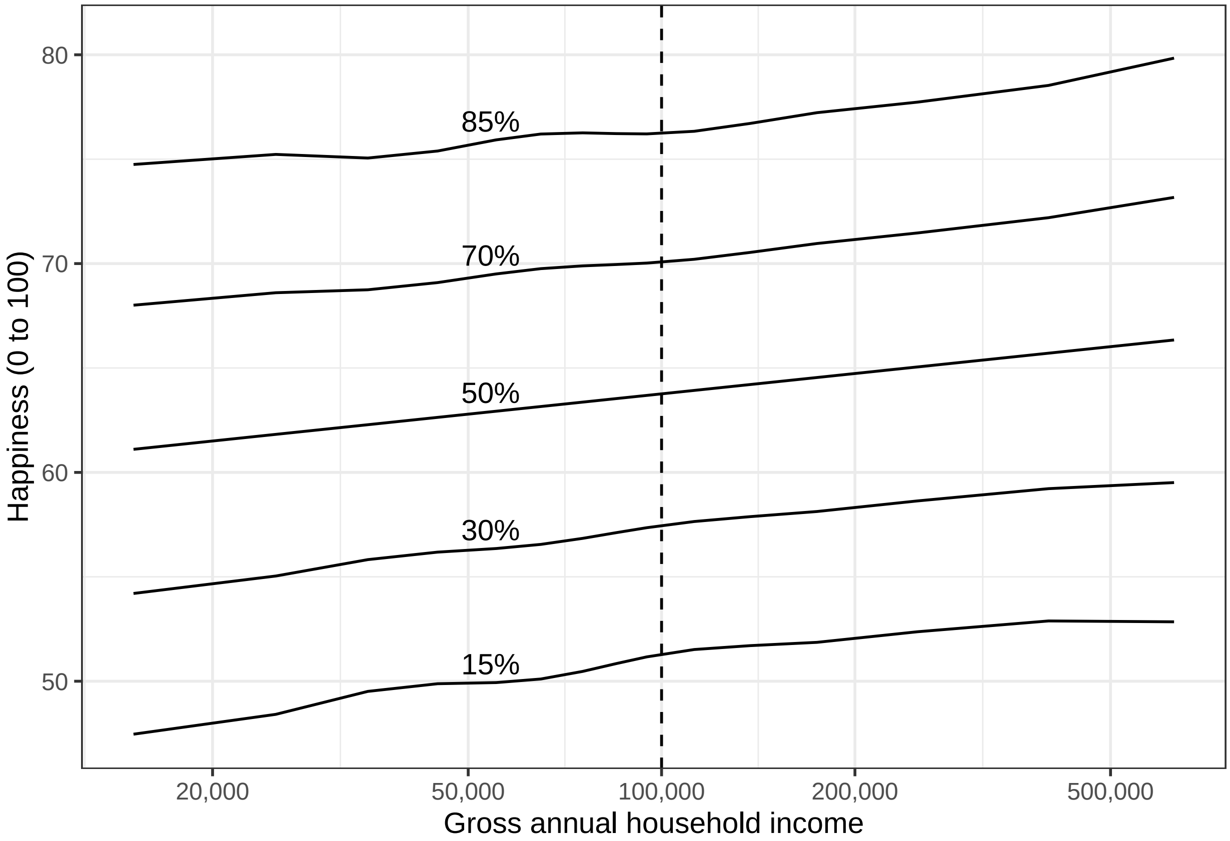



```r
ggplot(plotdata, aes(log_income, y = sigma)) +
  geom_smooth(stat = "identity") +
  geom_point() +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000")) +
  ylab("Spread") +
  theme_bw()
```




```
ggplot(plotdata,
       aes(log_income, y = mu)) + geom_line() +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000")) +
  geom_line(aes(y = mu + qnorm(0.15)*sigma)) +
  geom_line(aes(y = mu + qnorm(0.30)*sigma)) +
  geom_line(aes(y = mu + qnorm(0.70)*sigma)) +
  geom_line(aes(y = mu + qnorm(0.85)*sigma)) +
  ylab("Happiness (0 to 100)") +
  geom_text(aes(
    x = if_else(round(log_income,1)==10.9, 10.9, NA_real_),
    y = 0.9 + mu + qnorm(0.15)*sigma), label = "15%") +
  geom_text(aes(
    x = if_else(round(log_income,1)==10.9, 10.9, NA_real_),
    y = 0.9 + mu + qnorm(0.30)*sigma), label = "30%") +
  geom_text(aes(
    x = if_else(round(log_income,1)==10.9, 10.9, NA_real_),
    y = 0.9 + mu), label = "50%") +
  geom_text(aes(
    x = if_else(round(log_income,1)==10.9, 10.9, NA_real_),
    y = 0.9 + mu + qnorm(0.70)*sigma), label = "70%") +
  geom_text(aes(
    x = if_else(round(log_income,1)==10.9, 10.9, NA_real_),
    y = 0.9 + mu + qnorm(0.85)*sigma), label = "85%") +
  geom_vline(xintercept = log(100000), linetype = "dashed") +
```

```
  theme_bw()
```



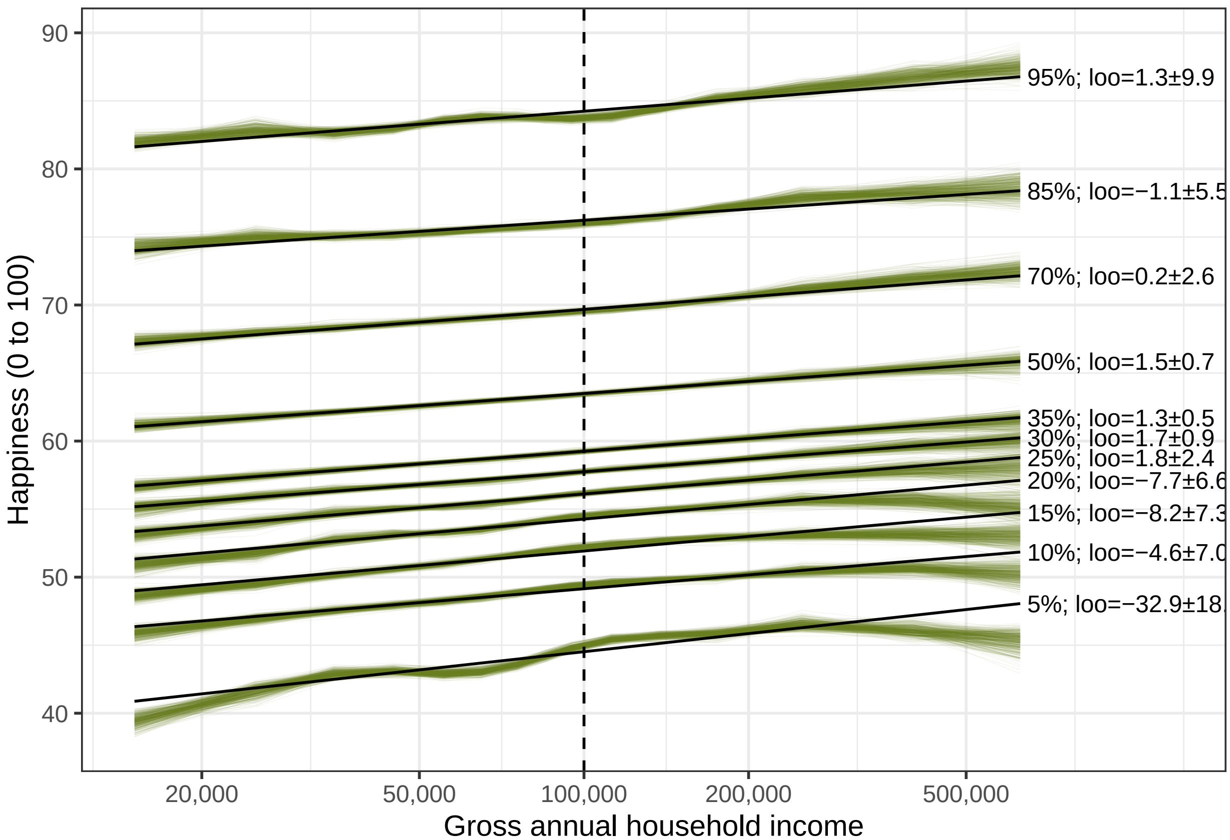
3.2 Spline quantile plot

```
quantiles <- c(0.05, 0.10, 0.15, 0.20, 0.25, 0.3, 0.35, 0.5, 0.7, 0.85, 0.95)

fit_model <- function(qu) {
  brm(
    bf(wellbeing ~ s(log_income), quantile = qu),
    data = esd, family = asym_laplace(),
    control = list(adapt_delta = 0.99),
    file = str_c("m_spline_q", qu)
  ) %>% add_criterion("loo")
}

fits <- tibble(
  quantile = quantiles
) %>%
  mutate(
    m = map(quantile, fit_model)
  )


fit_linear_model <- function(qu) {
  brm(
```

```r
    bf(wellbeing ~ log_income, quantile = qu),
    data = esd, family = asym_laplace(),
    file = str_c("m_linear_q", qu)
  ) %>% add_criterion("loo")
}


fit_cut_model <- function(qu) {
  brm(
    bf(wellbeing ~ log_income + log_income:income_above_100, quantile = qu),
    data = esd, family = asym_laplace(),
    file = str_c("m_cut_q", qu)
  ) %>% add_criterion("loo")
}

fits2 <- tibble(
  quantile = quantiles
) %>%
  mutate(
    m = map(quantile, fit_linear_model)
  )

fits_cut <- tibble(
  quantile = quantiles
) %>%
  mutate(
    m = map(quantile, fit_cut_model)
  )




predict_mu_model <- function(model) {
  fits <- fitted(model, newdata = esd %>%
                  distinct(log_income) %>%
                  arrange(log_income),
                summary = F, ndraws = 500, dpar = "mu") %>% as_tibble()
  colnames(fits) <- esd %>% distinct(log_income) %>%
    arrange(log_income) %>% pull(log_income)
  fits$sample <- 1:nrow(fits)

  fits %>% pivot_longer(-sample, names_to = "log_income", values_to = "mu") %>%
    mutate(log_income = as.numeric(log_income))
}
fits3 <- fits %>% mutate(
  spaghetti = m %>% map(predict_mu_model)
)

predict_mu_model_summarize <- function(model) {
  fits <- fitted(model, newdata = esd %>%
                  distinct(log_income, income_above_100) %>%
                  arrange(log_income),
                summary = T, dpar = "mu") %>% as_tibble()
```

```
  fits$log_income <- esd %>% distinct(log_income) %>%
    arrange(log_income) %>% pull(log_income)

  fits %>% rename(mu = Estimate)
}

fits_nonlinear_cis <- fits %>% mutate(
  spaghetti = m %>% map(predict_mu_model_summarize)
) %>% select(-m) %>%
  unnest(spaghetti)

fits_linear <- fits2 %>% mutate(
  spaghetti = m %>% map(predict_mu_model_summarize)
)

fits_cut <- fits_cut %>% mutate(
  spaghetti = m %>% map(predict_mu_model_summarize)
)

fits_linear <- fits_linear %>% select(-m) %>%
  unnest(spaghetti)


fits_cut <- fits_cut %>% select(-m) %>%
  unnest(spaghetti)

summarise_loo_comp <- function(m_linear, m_spline) {
  comp <- loo_compare(m_linear, m_spline)
  best_mod <- rownames(comp)[1]
  elpd_diff <- comp[2,1]
  se_diff <- comp[2,2]
  elpd_diff <- if_else(best_mod == "m_linear", -1, 1) * elpd_diff
  sprintf("%.1f±%.1f", elpd_diff, se_diff)
}
fits_both <- fits %>% left_join(fits2, by = "quantile",
                                suffix = c("_spline", "_linear"))
fits_both <- fits_both %>% rowwise() %>%
  mutate(elpd_loo = summarise_loo_comp(m_linear, m_spline))

fits3 %>%
  select(-m) %>%
  unnest(spaghetti, names_repair = "universal") %>%
  ggplot(aes(log_income, mu, group = interaction(quantile,sample))) +
  geom_line(alpha = 0.1, color = "#697f1f", linewidth = 0.05) +
  geom_line(aes(log_income, mu, group = quantile),
            alpha = 1, color = 'black', data = fits_linear) +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000"),
                     limits = log(c(15000, 1200000))) +
  geom_text(aes(
    label = str_c(quantile*100, "%; loo=", elpd_loo), group = 1),
```

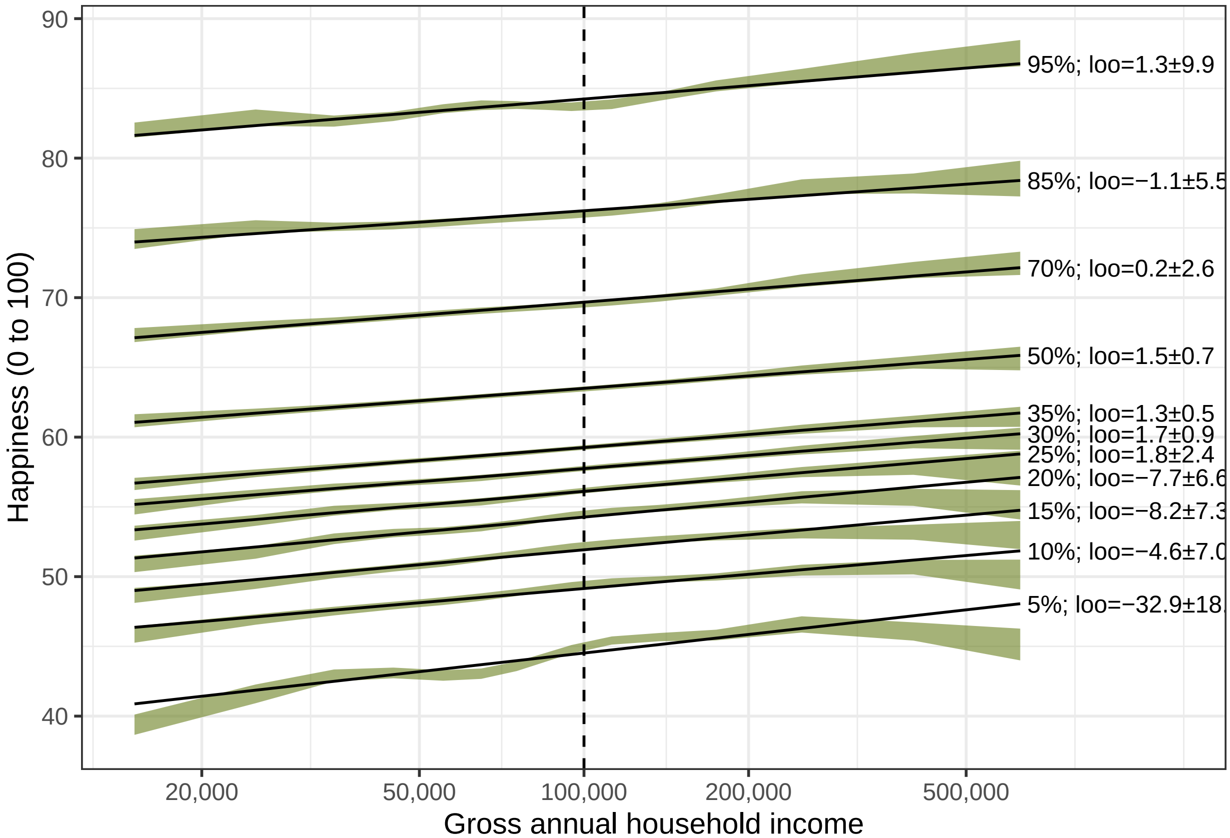
```
    data = fits_linear %>%
      group_by(quantile) %>%
      filter(log_income == max(log_income)) %>%
      ungroup() %>%
      left_join(fits_both %>% select(quantile, elpd_loo)),
    hjust = 0, nudge_x = 0.03, size = 3.3) +
  ylab("Happiness (0 to 100)") +
  geom_vline(xintercept = log(100000), linetype = "dashed") +
  theme_bw()
```




```
ggsave("Figure1.pdf", width = 9, height = 6, units = "cm", scale = 2)
# ggsave("Figure1.png", width = 7.5, height = 6)
```

```
fits_nonlinear_cis %>%
  ggplot(aes(log_income, y = mu, group = quantile)) +
  geom_ribbon(aes(ymin = `Q2.5`, ymax = `Q97.5`), stat = 'identity',
              alpha = 0.6, fill = "#697f1f") +
  geom_line(aes(log_income, mu, group = quantile),
            alpha = 1, color = 'black', data = fits_linear) +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000"),
                     limits = log(c(15000, 1200000))) +
  geom_text(aes(
    label = str_c(quantile*100, "%; loo=", elpd_loo), group = 1),
    data = fits_linear %>%
```

```r
      group_by(quantile) %>%
      filter(log_income == max(log_income)) %>%
      ungroup() %>%
      left_join(fits_both %>% select(quantile, elpd_loo)),
    hjust = 0, nudge_x = 0.03, size = 3.3) +
  ylab("Happiness (0 to 100)") +
  geom_vline(xintercept = log(100000), linetype = "dashed") +
  theme_bw()
```



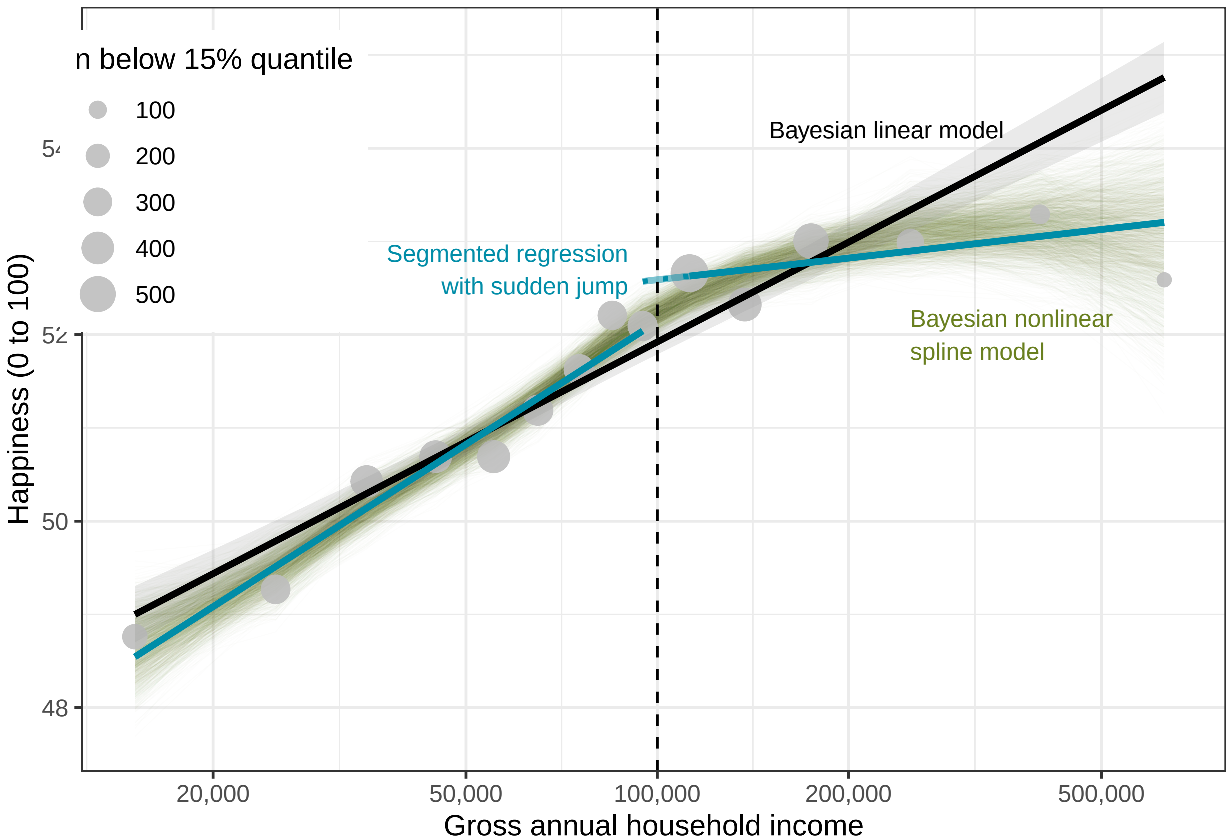

```r
ggsave("Figure1_bands.pdf", width = 9, height = 6, units = "cm", scale = 2)
```

3.3 Comparison to (dis)continuous segmented regression

```r
library(quantreg)
segmented_rq <- (rq(wellbeing ~ log_income + income_above_100:log_income, esd, tau = quantiles))

jump_rq <- (rq(wellbeing ~ log_income + income_above_100:log_income + income_above_100, esd, tau = quan

newd <- esd %>% distinct(log_income, income_above_100) %>%
  mutate(row = row_number())
fits_segmented <- predict(segmented_rq, newdata = esd %>%
    distinct(log_income, income_above_100)) %>%
  as_tibble(rownames = "row") %>%
  pivot_longer(-row) %>%
  mutate(row = as.integer(row)) %>%
```

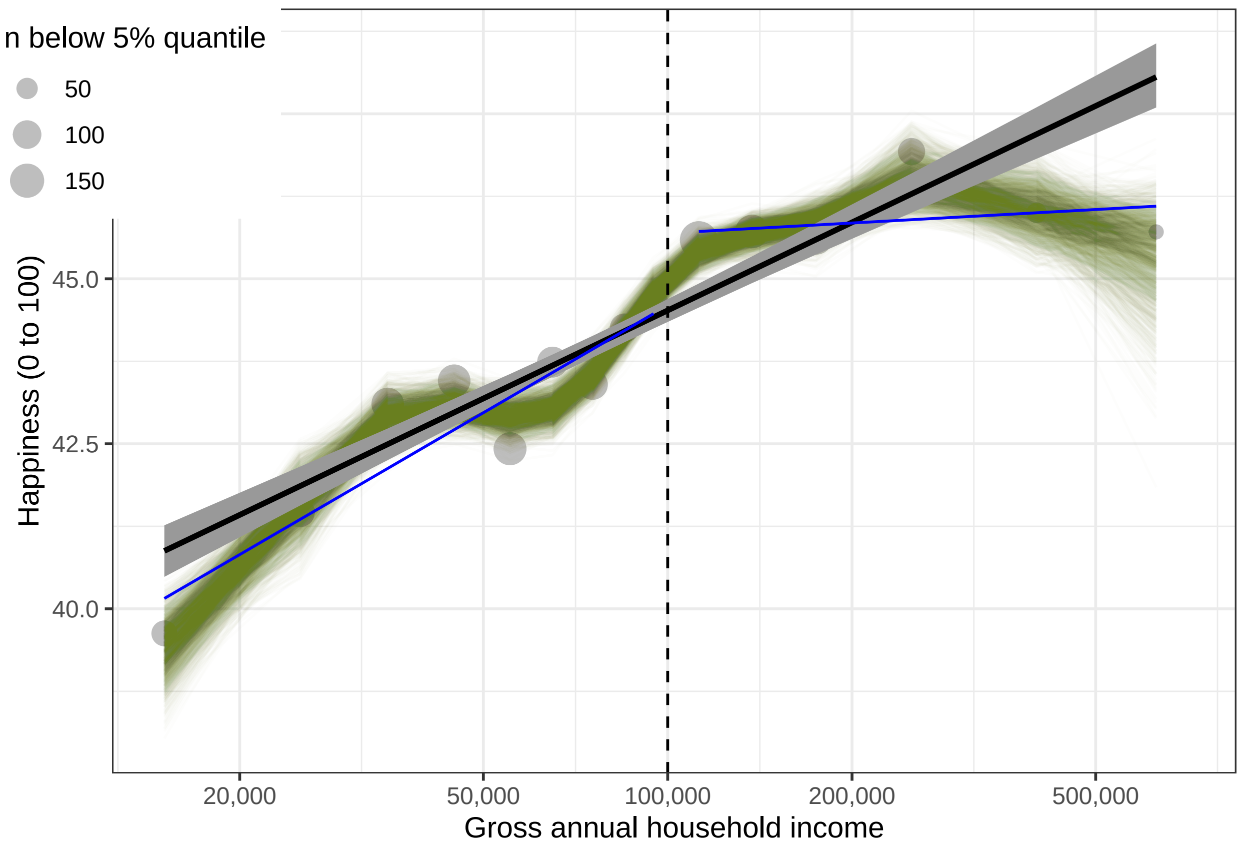
```
  left_join(newd) %>%
  mutate(quantile = as.numeric(str_sub(name, 5)))

newd <- esd %>% distinct(log_income, income_above_100) %>%
  mutate(row = row_number())
fits_jump <- predict(jump_rq, newdata = newd) %>%
  as_tibble(rownames = "row") %>%
  pivot_longer(-row) %>%
  mutate(row = as.integer(row)) %>%
  left_join(newd) %>%
  mutate(quantile = as.numeric(str_sub(name, 5)))

newd <- esd %>% distinct(log_income) %>% filter(log_income > 11.4) %>%
  mutate(income_above_100 = 1) %>% mutate(row = row_number())

fits_jump_extrapolated <- predict(jump_rq, newdata = newd) %>%
  as_tibble(rownames = "row") %>%
  pivot_longer(-row) %>%
  mutate(row = as.integer(row)) %>%
  left_join(newd) %>%
  mutate(quantile = as.numeric(str_sub(name, 5)))

predict_mu_model <- function(model) {
  fits <- fitted(model, newdata = esd %>%
                   distinct(log_income) %>%
                   arrange(log_income),
                 summary = F, ndraws = 2000, dpar = "mu") %>% as_tibble()
  colnames(fits) <- esd %>% distinct(log_income) %>%
    arrange(log_income) %>% pull(log_income)
  fits$sample <- 1:nrow(fits)

  fits %>% pivot_longer(-sample, names_to = "log_income", values_to = "mu") %>%
    mutate(log_income = as.numeric(log_income))
}

fits3 <- fits %>% mutate(
  spaghetti = m %>% map(predict_mu_model)
)

ns_by_inc <-  esd %>%
  mutate(income = exp(log_income)) %>%
  group_by(income, log_income) %>%
  summarise(n = n(),q = quantile(wellbeing, probs = .15) %>% as.vector(),
            n15 = sum(wellbeing <= q))

fits3 %>%
  filter(quantile == .15) %>%
  select(-m) %>%
  unnest(spaghetti, names_repair = "universal") %>%
  ggplot(aes(log_income, mu, group = interaction(quantile,sample))) +
  geom_line(alpha = 0.01, color = "#697f1f", linewidth = 0.2) +
  geom_point(aes(log_income, q, size = (n15), group = q), data =
               ns_by_inc, color = "gray", alpha = 0.9) +
```

```r
geom_smooth(aes(log_income, mu, group = quantile,
                ymin = `Q2.5`, ymax = `Q97.5`),
            stat = "identity",
          linewidth = 1.2,
          alpha = 0.2, color = 'black', data = fits_linear %>%
filter(quantile == .15)) +
# geom_line(aes(log_income, value, group = quantile),
#           alpha = 1, color = 'brown', data = fits_segmented %>%
# filter(quantile == .15)) +
geom_line(aes(log_income, value, group = quantile),
          linewidth = 0.9,
          alpha = 1, color = '#008da8', linetype = 'dotted',
          data = fits_jump_extrapolated %>%
            filter(quantile == .15)) +
geom_line(aes(log_income, value, group = quantile),
          linewidth = 1.2,
          alpha = 0.5, color = '#008da8',
          data = fits_jump_extrapolated %>%
            filter(quantile == .15)) +
geom_line(aes(log_income, value, group = quantile),
          linewidth = 1.2,
          alpha = 1, color = '#008da8',
          data = fits_jump %>%
            filter(income_above_100 == 0) %>%
            filter(quantile == .15)) +
geom_line(aes(log_income, value, group = quantile),
          linewidth = 1.2,
          alpha = 1, color = '#008da8',
          data = fits_jump %>%
            filter(income_above_100 == 1) %>%
            filter(quantile == .15)) +
scale_x_continuous("Gross annual household income",
                   breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                   labels = c("20,000", "50,000", "100,000",
                              "200,000", "500,000"),
                   limits = log(c(15000, 650000))) +
annotate("text", label = "Bayesian linear model", color = "black",
         x = log(150000), y = 54.2, size = 3, hjust = 0) +
annotate("text", label = "Segmented regression\nwith sudden jump",
         color = "#008da8",
         x = log(90000), y = 52.7, size = 3, hjust = 1) +
annotate("text", label = "Bayesian nonlinear\nspline model",
         color = "#697f1f",
         x = log(250000), y = 52, size = 3, hjust = 0) +
scale_size_area(max_size = 6) +
ylab("Happiness (0 to 100)") +
scale_size_area("n below 15% quantile", max_size = 6) +
geom_vline(xintercept = log(100000), linetype = "dashed") +
theme_bw() +
theme(legend.position = c(0.25,0.97), legend.justification = c(1,1))
```




```
ggsave("Figure2b.pdf", width = 9, height = 6, units = "cm", scale = 2)

ns_by_inc <-  esd %>%
  mutate(income = exp(log_income)) %>%
  group_by(income, log_income) %>%
  summarise(n = n(),q = quantile(wellbeing, probs = .05) %>% as.vector(),
            n15 = sum(wellbeing <= q))

fits3 %>%
  filter(quantile == .05) %>%
  select(-m) %>%
  unnest(spaghetti, names_repair = "universal") %>%
  ggplot(aes(log_income, mu, group = interaction(quantile,sample))) +
  geom_point(aes(log_income, q, size = (n15), group = q),
             data = ns_by_inc, color = "gray",
             fill = "white") +
  geom_line(alpha = 0.01, color = "#697f1f") +
  geom_smooth(aes(log_income, mu, group = quantile,
                  ymin = `Q2.5`, ymax = `Q97.5`),
             stat = "identity",
            alpha = 1, color = 'black', data = fits_linear %>%
  filter(quantile == .05)) +
  # geom_line(aes(log_income, value, group = quantile),
  #             alpha = 1, color = 'brown', data = fits_segmented %>%
  # filter(quantile == .15)) +
  geom_line(aes(log_income, value, group = quantile),
            alpha = 1, color = 'blue', linetype = 'dotted',
```

```
            data = fits_jump_extrapolated %>%
              filter(income_above_100 == 0) %>%
              filter(quantile == .05)) +
  geom_line(aes(log_income, value, group = quantile),
            alpha = 1, color = 'blue', data = fits_jump %>%
              filter(income_above_100 == 0) %>%
              filter(quantile == .05)) +
  geom_line(aes(log_income, value, group = quantile),
            alpha = 1, color = 'blue',
            data = fits_jump %>% filter(income_above_100 == 1) %>%
              filter(quantile == .05)) +
  scale_x_continuous("Gross annual household income",
                     breaks = log(c(20000, 50000, 100000, 200000, 500000)),
                     labels = c("20,000", "50,000", "100,000",
                                "200,000", "500,000"),
                     limits = log(c(15000, 700000))) +
  scale_size_area("n below 5% quantile", max_size = 6) +
  ylab("Happiness (0 to 100)") +
  geom_vline(xintercept = log(100000), linetype = "dashed") +
  theme_bw() +
  theme(legend.position = c(0.15,1), legend.justification = c(1,1))
```



4 Version info

```r
sessionInfo()
```

```
## R Under development (unstable) (2022-12-06 r83409)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib
##
## 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: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices datasets  utils     methods   base
##
## other attached packages:
##  [1] quantreg_5.96   SparseM_1.81    brms_2.20.1     Rcpp_1.0.11
##  [5] lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0   dplyr_1.1.2
##  [9] purrr_1.0.2     readr_2.1.4     tidyr_1.3.0     tibble_3.2.1
## [13] ggplot2_3.4.3   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##   [1] gridExtra_2.3        inline_0.3.19        readxl_1.4.3
##   [4] rlang_1.1.1          magrittr_2.0.3       rio_0.5.29
##   [7] matrixStats_1.0.0    compiler_4.3.0       mgcv_1.9-0
##  [10] loo_2.6.0            systemfonts_1.0.4    callr_3.7.3
##  [13] vctrs_0.6.3          reshape2_1.4.4       pkgconfig_2.0.3
##  [16] crayon_1.5.2         fastmap_1.1.1        backports_1.4.1
##  [19] ellipsis_0.3.2       labeling_0.4.2       utf8_1.2.3
##  [22] cmdstanr_0.6.1       threejs_0.3.3        promises_1.2.1
##  [25] rmarkdown_2.24       markdown_1.7         tzdb_0.4.0
##  [28] haven_2.5.3          ps_1.7.5             ragg_1.2.5
##  [31] MatrixModels_0.5-2   bit_4.0.5            xfun_0.40
##  [34] highr_0.10           later_1.3.1          parallel_4.3.0
##  [37] prettyunits_1.1.1    R6_2.5.1             dygraphs_1.1.1.6
##  [40] stringi_1.7.12       StanHeaders_2.26.27  cellranger_1.1.0
##  [43] rstan_2.21.8         knitr_1.43           zoo_1.8-12
##  [46] base64enc_0.1-3      bayesplot_1.10.0     splines_4.3.0
##  [49] httpuv_1.6.11        Matrix_1.6-1         igraph_1.5.1
##  [52] timechange_0.2.0     tidyselect_1.2.0     rstudioapi_0.15.0
##  [55] abind_1.4-5          yaml_2.3.7           codetools_0.2-19
##  [58] miniUI_0.1.1.1       curl_5.0.2           processx_3.8.2
##  [61] pkgbuild_1.4.2       lattice_0.22-5       plyr_1.8.8
##  [64] shiny_1.7.5          withr_2.5.0          bridgesampling_1.1-2
##  [67] posterior_1.4.1      coda_0.19-4          evaluate_0.21
##  [70] foreign_0.8-85       survival_3.5-7       RcppParallel_5.1.7
##  [73] zip_2.3.0            xts_0.13.1           pillar_1.9.0
```

```
##  [76] tensorA_0.36.2       checkmate_2.2.0      renv_1.0.2
##  [79] DT_0.28              stats4_4.3.0         shinyjs_2.1.0
##  [82] distributional_0.3.2 generics_0.1.3       vroom_1.6.3
##  [85] hms_1.1.3            rstantools_2.3.1.1   munsell_0.5.0
##  [88] scales_1.2.1         gtools_3.9.4         xtable_1.8-4
##  [91] glue_1.6.2           tools_4.3.0          shinystan_2.6.0
##  [94] data.table_1.14.8    openxlsx_4.2.5.2     colourpicker_1.2.0
##  [97] mvtnorm_1.2-2        grid_4.3.0           crosstalk_1.2.0
## [100] colorspace_2.1-0     nlme_3.1-163         cli_3.6.1
## [103] textshaping_0.3.6    fansi_1.0.4          viridisLite_0.4.2
## [106] Brobdingnag_1.2-9    gtable_0.3.3         digest_0.6.33
## [109] ggrepel_0.9.3        htmlwidgets_1.6.2    farver_2.1.1
## [112] htmltools_0.5.6      lifecycle_1.0.3      mime_0.12
## [115] MASS_7.3-60          bit64_4.0.5          shinythemes_1.2.0
```