Growth Curving

Trevor Day

2/3/2020

Setup

Load packages `tidyverse`, `growthcurver`, `ggplot2`, `scales`, `ggrepel`, `gridExtra`.

Load the first person with five points

```
test <- read_csv("data/test-subjs.csv") %>%
          arrange(data_id, age) %>%
          filter(data_id == 176427)
```

```
## Parsed with column specification:
## cols(
##   data_id = col_double(),
##   age = col_double(),
##   inventory = col_double()
## )
```

Introduction

Throughout, I will be using notation from XXX, where:

- W_0: The starting population. For our purposes, can be estimated at 1 or less.
- A: The upper asymptote, here I use $680 + 1$, the size of MCDI plus one so the participant is actually modeled as learning all words.
- t: Time. Here, age in months.
- k_G: Relative growth rate.
- k_U: Absolute maximum growth rate (at inflection). Given as $k_U = \frac{A \times k_g}{e}$. Has units $\frac{\text{words}}{\text{month}}$.
- W_i, T_i: Value and time at inflection. W_i is fixed at $\frac{A}{e} = 251$, and T_i can be solved for.

```
A <- 681
W_0 <- .Machine$double.eps
```

Basic Gompertz

The Gompertz equation given by `growthcurver` is:

$$y = \frac{K}{1 + \frac{K - N_0}{N_0} e^{-rx}}$$

where N_0 is the initial number, K is the maximum, and r is the *intrinsic growth rate*.

This can be created like so:

```
gomp.fit <- SummarizeGrowth(test$age, test$inventory)
gomp.fit
```

```
## Fit data to K / (1 + ((K - N0) / N0) * exp(-r * t)):
##      K   N0  r
```

```
##   val:   597.031 0.043   0.398
##   Residual standard error: 2.750093 on 2 degrees of freedom
##
## Other useful metrics:
##   DT 1 / DT  auc_l   auc_e
##   1.74   5.7e-01 3745.51 3756.5
```

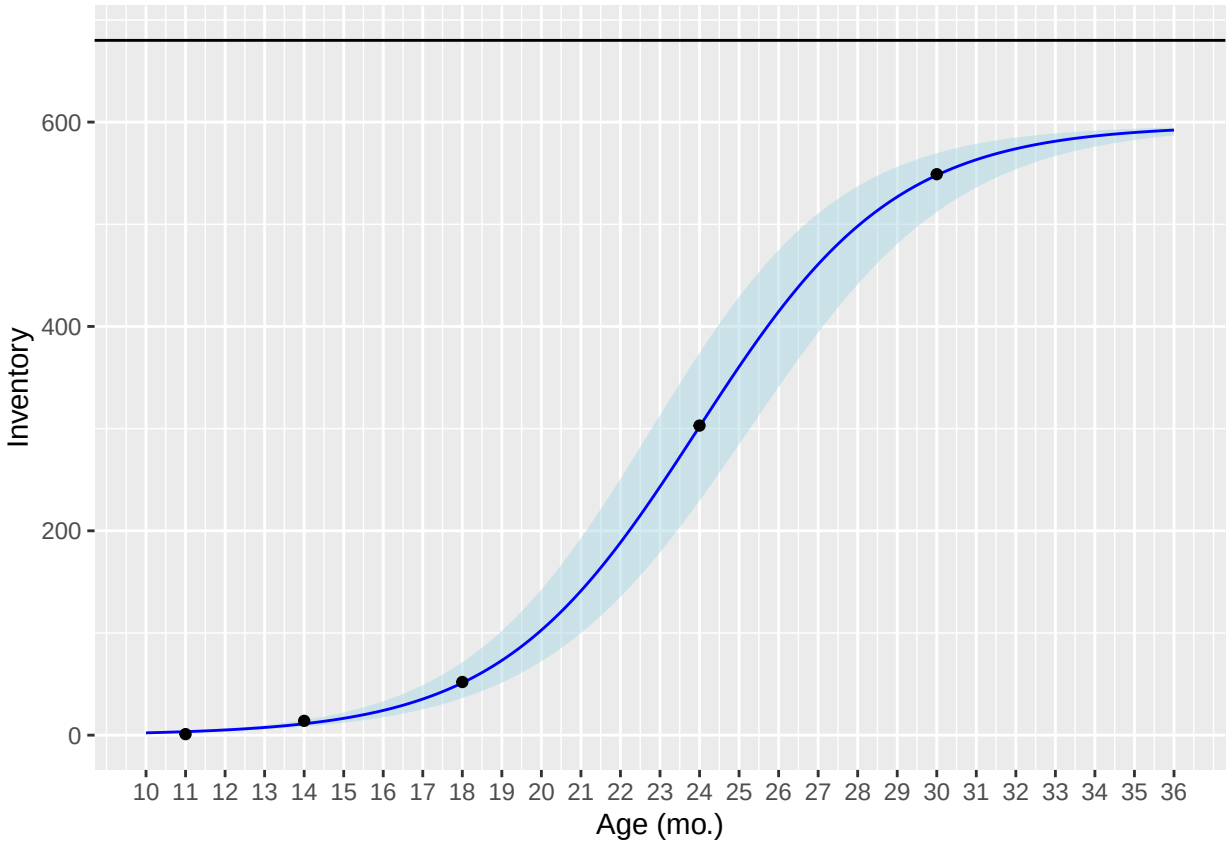
We write a quick function to calculate the fit over a given range. We use `0.01` as a step as a day is `0.03` months.

```
predict.gomp <- function(fit, k_g = NA, k_g.se = NA,
                         range = seq(0, 36, by = 0.01)) {

  # If a fit is passed, use new labels
  if (is.na(k_g)) {

    A <- fit$vals$k
    W_0 <- fit$vals$n0
    K <- fit$vals$r
    K.ci <- 1.96 * fit$vals$r_se

    result    <- A/(1 + ((A - W_0)/W_0) * exp(-K * range))
    result.lo <- A/(1 + ((A - W_0)/W_0) * exp(-(K - K.ci) * range))
    result.hi <- A/(1 + ((A - W_0)/W_0) * exp(-(K + K.ci) * range))

  } else {

    A <- 681
    W_0 <- .Machine$double.eps

    K <- k_g
    K.ci <- k_g.se

    result <- W_0 * (A / W_0) ^ (1 - exp(-K * range))
    result.lo <- W_0 * (A / W_0) ^ (1 - exp(-(K - K.ci) * range))
    result.hi <- W_0 * (A / W_0) ^ (1 - exp(-(K + K.ci) * range))

  }


  out <- cbind(range, result, result.lo, result.hi) %>%
          as.data.frame()

  return(out)

}
```

```
fit1 <- predict.gomp(gomp.fit)

ggplot(NULL) +
  scale_x_continuous(limits = c(10, 36), breaks = 10:36) +
  scale_y_continuous(limits = c(0, 680)) +
  geom_ribbon(data = fit1, aes(x = range, ymin = result.lo, ymax = result.hi),
              fill = "lightblue", alpha = 0.5) +
  geom_line(data = fit1, aes(x = range, y = result), color = "blue") +
```

```
  geom_point(data = test, aes(x = age, y = inventory)) +
  labs(x = "Age (mo.)", y = "Inventory") +
  geom_hline(yintercept = 680)
```

```
## Warning: Removed 1000 rows containing missing values (geom_path).
```



The standard error of `r` is used for two additional lines and plotted here as an error range.

However, this has the limitation that `K` is estimated by the function, which means it may pick an asymptote well below 680.

I also want to point out this is a remarkably good fit; other subjects are not so good.

Gompertz II Electric Pertzaloo

However, there is an alternate formulation for a Gompertz curve that takes both N_0, the starting value, and N_1, the ending value as arguments. As seen below, N_0 cannot be `0`, so we use the smallest number R can represent: 2.22×10^{-16}.

For N_1, we use $680 + 1$, as otherwie, a kid who never learns the 680th word on the MCDI would be modeled. Although, as a reminder, we are most interested in the shape of the first two-thirds of the model, since the greater asymptote is an artifact of the instrument.

This equation is:

$$y = W_0 \left(\frac{A}{W_0} \right)^{1-\exp(-k_G \times t)}$$

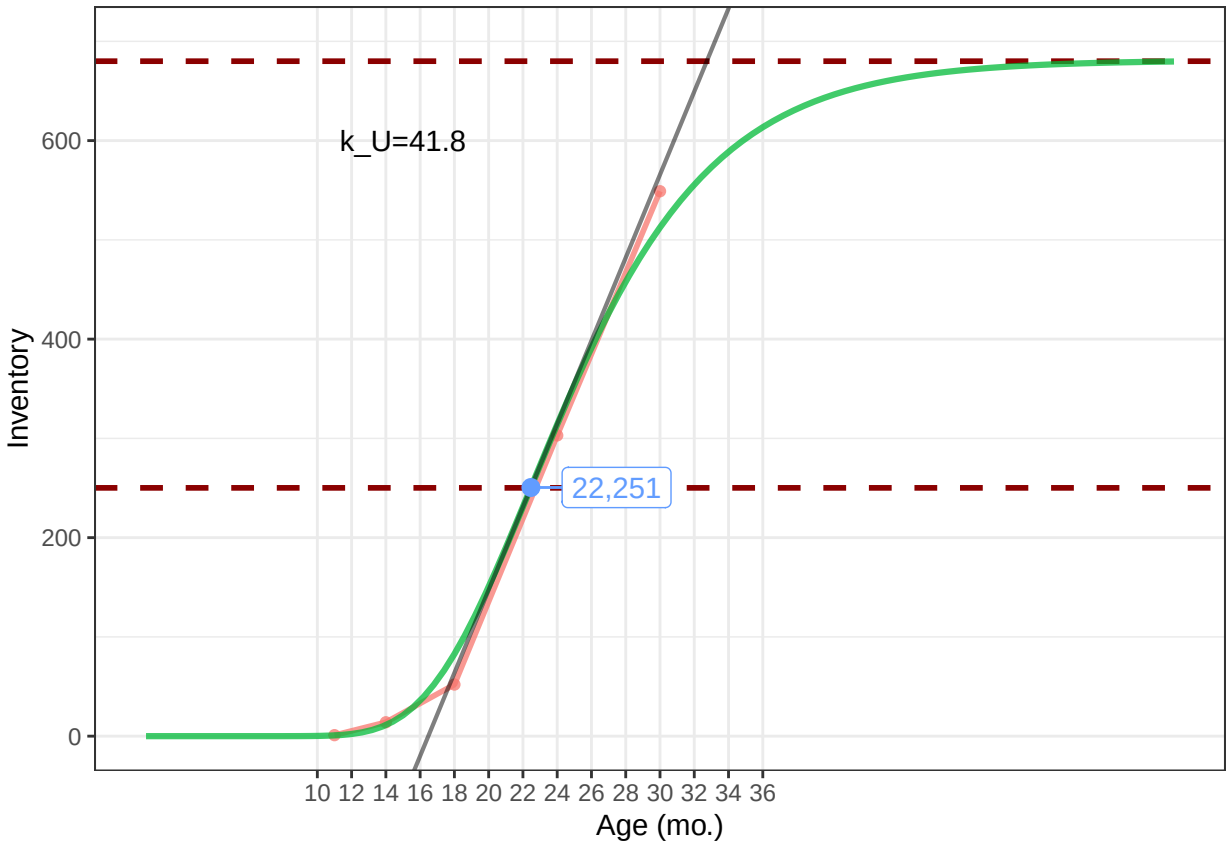
Because we are constraining A and W_0, we can estimate k_G with this form.

```
gomp2.fit <- function(data) {

  A <- 680 + 1
  W_0 <- .Machine$double.eps

  # This is the formula that works best to solve in R
  # 19 refers to its number in the Tjorve and Tjorve paper
  fit19 <- nls(inventory ~ W_0 * (A / W_0) ^ (1 - exp(-k_g * age)),
               data = data,
               start = list(k_g = .1))

  return(fit19)

}

# Estimate relative growth rate
gomp2 <- gomp2.fit(test)
gomp2.kg <- summary(gomp2)$coefficients[1, 1]

# Solve for abs growth rate
gomp2.kU <- 681 * gomp2.kg / exp(1)

solve.gomp2 <- function(x = NA, y = NA, k_g) {

  A <- 680 + 1
  W_0 <- .Machine$double.eps

  if (is.na(y) & !is.na(x))
    result <-  W_0 * (A / W_0) ^ (1 - exp(-k_g * x))
  else if (is.na(x) & !is.na(y))
    result <- -log(1 - log(y / W_0) / log(A / W_0)) / k_g
  else
    message("Exactly one of x/y must be specified")

  return(result)

}
```

Now we plot the Gompertz fit, including the point of inflection, and the maximum absolute growth rate, the rate at T_i, W_i.

```
# Value at inflection: constant
W_i <- 681 / exp(1)

# Time at inflection
# Slope at inflection is gomp2.kU
T_i <- solve.gomp2(y = W_i, k_g = gomp2.kg)

# Solve for intercept of tangent line
b <- W_i - gomp2.kU * T_i

# This block solves for upper and lower estimates
# gomp2.kg.se <- summary(gomp2)$coefficients[1, 2]
```

```r
# gomp2.kgs <- c(gomp2.kg - gomp2.kg.se, gomp2.kg + gomp2.kg.se)

colors <- hue_pal()(3)

ggplot(NULL) +
  geom_point(data = test,
             aes(x = age, y = inventory),
             alpha = 0.75,
             color = colors[1]) +
  geom_line(data = test,
             aes(x = age, y = inventory),
             alpha = 0.75,
             color = colors[1],
             size = 1) +
  scale_x_continuous(limits = c(0, 60), breaks = seq(10, 36, by = 2),
                     minor_breaks = NULL) +
  scale_y_continuous(limits = c(NA, 700)) +
  labs(x = "Age (mo.)", y = "Inventory") +
  geom_hline(yintercept = c(680, 680 / exp(1)), size = 1, linetype = "dashed",
             color = "darkred") +
  stat_function(data = test,
                fun = function(age) { W_0 * (A / W_0) ^ (1 - exp(-gomp2.kg * age)) },
                color = colors[2],
                size = 1.1,
                alpha = 0.75) +
  geom_abline(intercept = b,
              slope = gomp2.kU,
              color = "black",
              size = 0.75,
              alpha = 0.5) +
  geom_point(aes(x = T_i, y = W_i),
             size = 3,
             shape = 16,
             color = colors[3]) +
  geom_label_repel(aes(x = T_i, y = W_i),
                   size = 4,
                   label = paste0(round(T_i), ",", round(W_i)),
                   nudge_x = 5,
                   color = colors[3]) +
  annotate("text",
           x = 15, y = 600,
           label = paste0("k_U=", round(gomp2.kU, 1))) +
  theme_bw()
```

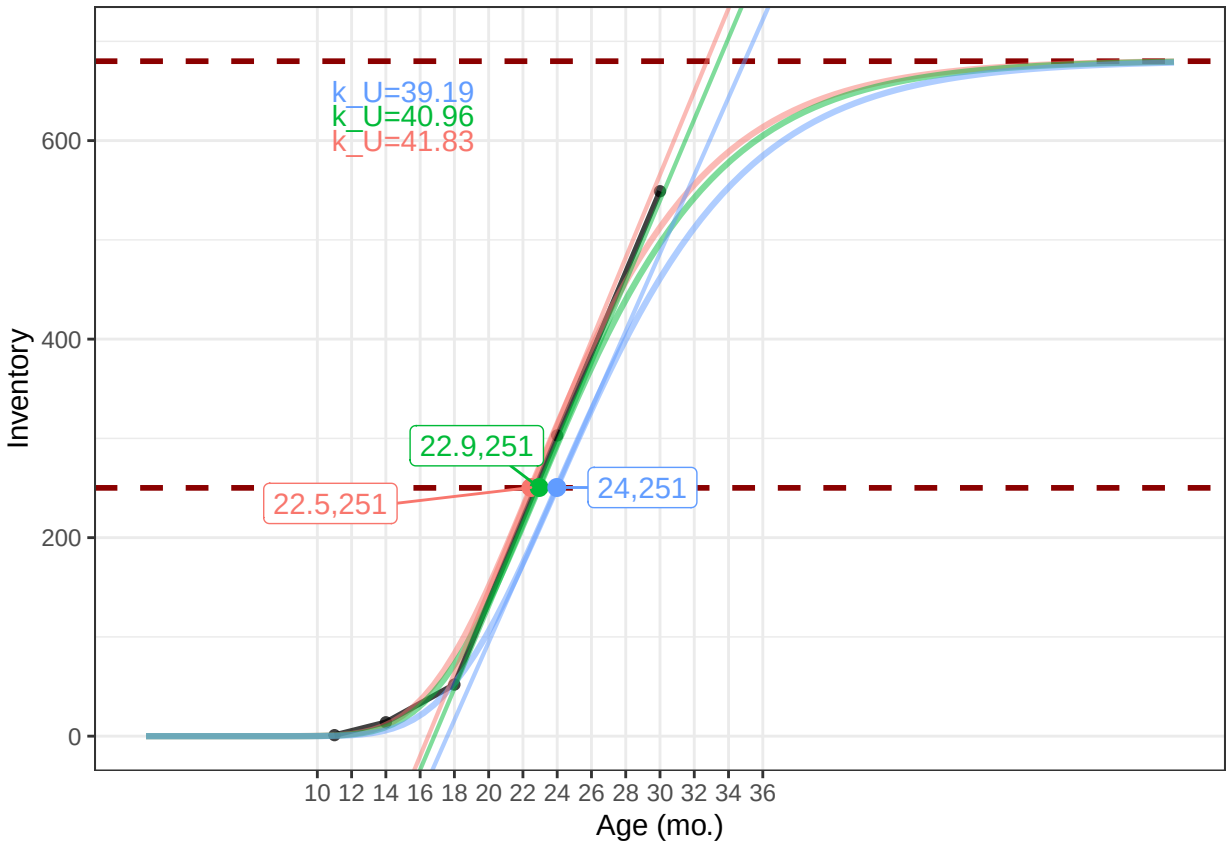


Reducing observations

Here we are modeling five points. However, not everyone has five points, so how few can we use?

```
gomp2s <- lapply(5:3, function(x) gomp2.fit(data = test[1:x, ]))
gomp2.kgs <- sapply(gomp2s, function(x) summary(x)$coefficients[1, 1])
gomp2.kUs <- A * gomp2.kgs / exp(1)

T_is <- sapply(gomp2.kgs, function(x) solve.gomp2(y = W_i, k_g = x))
bs <- W_i - gomp2.kUs * T_is

#colors <- hue_pal()(3)

ggplot(NULL) +
  geom_point(data = test,
             aes(x = age, y = inventory),
             alpha = 0.75,
             color = "black") +
  geom_line(data = test,
             aes(x = age, y = inventory),
             alpha = 0.75,
             color = "black",
             size = 1) +
  scale_x_continuous(limits = c(0, 60), breaks = seq(10, 36, by = 2),
                     minor_breaks = NULL) +
  scale_y_continuous(limits = c(NA, 700)) +
```

```
  labs(x = "Age (mo.)", y = "Inventory") +
  geom_hline(yintercept = c(680, 680 / exp(1)), size = 1, linetype = "dashed",
             color = "darkred") +
  stat_function(data = test,
                fun = function(age) { W_0 * (A / W_0) ^ (1 - exp(-gomp2.kgs[1] * age)) },
                color = colors[1],
                size = 1.1,
                alpha = 0.5) +
  stat_function(data = test,
                fun = function(age) { W_0 * (A / W_0) ^ (1 - exp(-gomp2.kgs[2] * age)) },
                color = colors[2],
                size = 1.1,
                alpha = 0.5) +
  stat_function(data = test,
                fun = function(age) { W_0 * (A / W_0) ^ (1 - exp(-gomp2.kgs[3] * age)) },
                color = colors[3],
                size = 1.1,
                alpha = 0.5) +
  geom_abline(intercept = bs,
              slope = gomp2.kUs,
              color = colors,
              size = 0.75,
              alpha = 0.5) +
  geom_point(aes(x = T_is, y = W_i),
             size = 3,
             shape = 16,
             color = colors) +
  geom_label_repel(aes(x = T_is, y = W_i),
                   size = 4,
                   label = paste0(round(T_is, 1), ",", round(W_i)),
                   nudge_x = 5,
                  force = 25,
                   color = colors) +
  annotate("text",
           x = 15, y = 600 + c(0, 25, 50),
           label = paste0("k_U=", round(gomp2.kUs, 2)),
           color = colors) +
  theme_bw()
```



Modeling across subjects

The `test` file contains data from 51 subjects who have at least four observations. Here, we're going to repeat the same steps as above and plot all 51 to examine how consistent curves are with other data.

```
# Load data as a list
test_all <- read_csv("data/test-subjs.csv") %>%
              arrange(data_id, age) %>%
              split(., f = .$data_id)
```

```
## Parsed with column specification:
## cols(
##   data_id = col_double(),
##   age = col_double(),
##   inventory = col_double()
## )
```

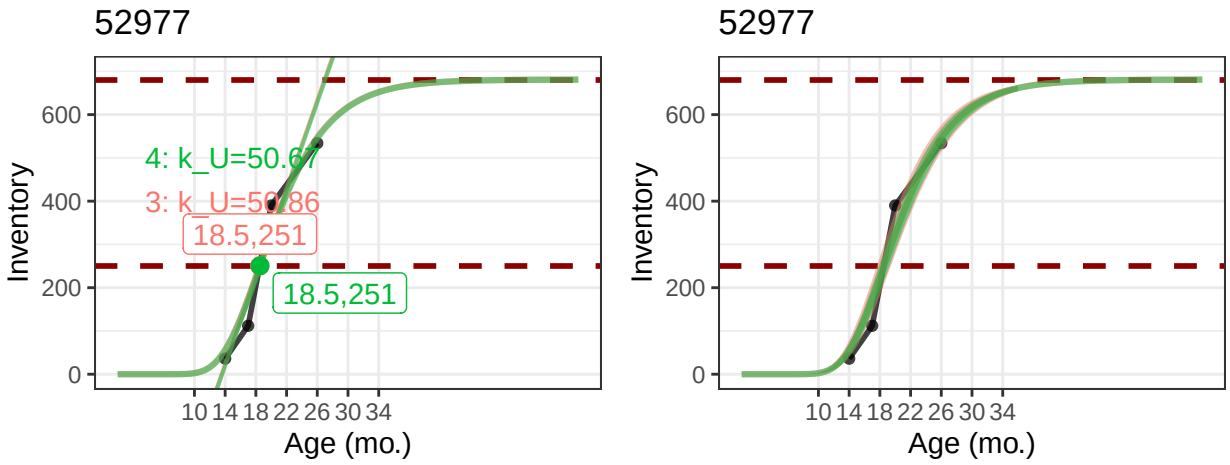
I'm going to invisibly fit the curves here.

```
all.fits5 <- lapply(test_all, function(x) if (nrow(x) == 5) { gomp2.fit(x) } else { NA })
all.fits4 <- lapply(test_all, function(x) gomp2.fit(x[1:4, ]))
all.fits3 <- lapply(test_all, function(x) gomp2.fit(x[1:3, ]))

# Plots three curves/tangents, list fits 3 -> 5
plot.tri <- function(data, fit3, fit4, fit5 = NA) {

  if (is.list(fit5)) {
    fits <- list(fit3, fit4, fit5)
```

```
} else {
  fits <- list(fit3, fit4)
}

kgs <- sapply(fits, function(x) summary(x)$coefficients[1, 1])

# The maximum number of colors is 3
colors <- hue_pal()(3)

p <- ggplot(NULL) +
      geom_point(data = data,
                 aes(x = age, y = inventory),
                 alpha = 0.75,
                 color = "black") +
      geom_line(data = data,
                 aes(x = age, y = inventory),
                 alpha = 0.75,
                 color = "black",
                 size = 1) +
      scale_x_continuous(limits = c(0, 60),
                         breaks = seq(10, 36, by = 4),
                         minor_breaks = NULL) +
      scale_y_continuous(limits = c(NA, 700)) +
      labs(x = "Age (mo.)", y = "Inventory") +
      geom_hline(yintercept = c(680, 680 / exp(1)),
                 size = 1,
                 linetype = "dashed",
                 color = "darkred")

# Here are the basic curves
curves <- lapply(1:length(fits), function(x)
                    stat_function(data = data,
                                  fun = function(age) { W_0 * (A / W_0) ^ (1 - exp(-kgs[x] * age)) },
                                  color = colors[x],
                                  size = 1.1,
                                  alpha = 0.5))

p2 <- p + curves[[1]] + curves[[2]]

if (length(curves) == 3) {
  p2 <- p2 + curves[[3]]
}

# Ranges using predict.gomp
ribbons <- lapply(fits, function(x)
                          predict.gomp(k_g = summary(x)$coefficients[1, 1],
                                       k_g.se = summary(x)$coefficients[1, 2]))

geom_ribbons <- lapply(1:length(ribbons),
                       function(x) {

                          dat <- ribbons[[x]]
```

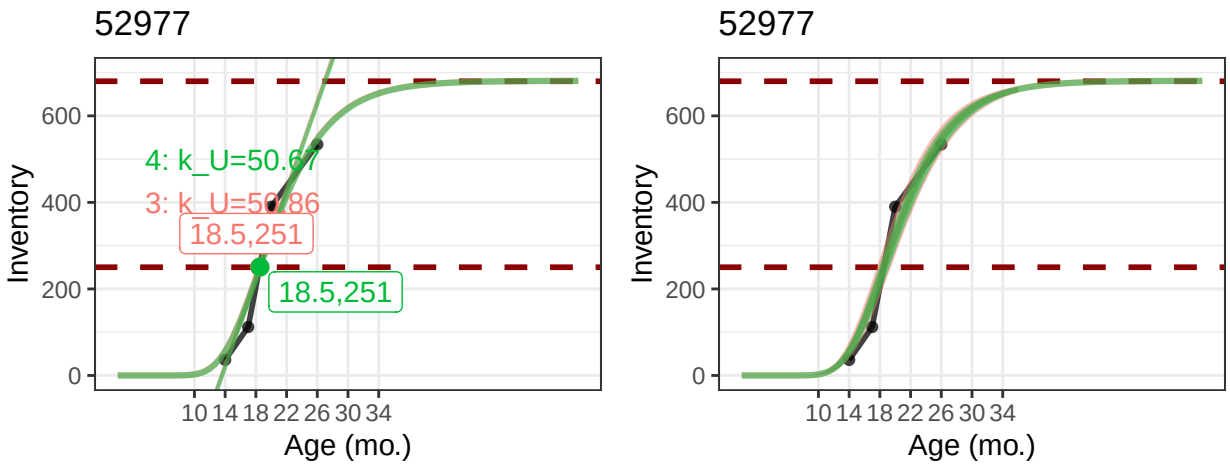
```
                            r <- geom_ribbon(data = dat,
                                             aes(x = range,
                                                 ymin = result.lo,
                                                 ymax = result.hi),
                                             fill = colors[x],
                                             alpha = 0.5)

                            return(r)

                          })


  kUs <- 681 * kgs / exp(1)
  T_is <- sapply(kgs, function(x) solve.gomp2(y = W_i, k_g = x))
  bs <- W_i - kUs * T_is

  tangents <- list(geom_abline(intercept = bs,
                    slope = kUs,
                    color = colors[1:length(bs)],
                    size = 0.75,
                    alpha = 0.5),
                  geom_point(aes(x = T_is, y = W_i),
                             size = 3,
                             shape = 16,
                             color = colors[1:length(T_is)]),
                  geom_label_repel(aes(x = T_is, y = W_i),
                                   size = 4,
                                   label = paste0(round(T_is, 1), ",", round(W_i)),
                                   nudge_x = 5,
                                   force = 25,
                                   color = colors[1:length(T_is)]) )

  p3 <- p2 + tangents

  y_pos <- (400 + c(0, 100, 200))[1:length(kUs)]
  labels <- paste0(2 + 1:length(kUs), ": k_U=", round(kUs, 2))

  annot <- list(annotate("text",
                         x = 15, y = y_pos,
                         label = labels,
                         color = colors[1:length(kUs)]),
                theme_bw(),
                labs(title = unique(data$data_id)))

  p4 <- p3 + annot
  pr1 <- p + geom_ribbons + curves + theme_bw() + labs(title = unique(data$data_id))

  f <- grid.arrange(p4, pr1, ncol = 2)

  return(f)

}
```

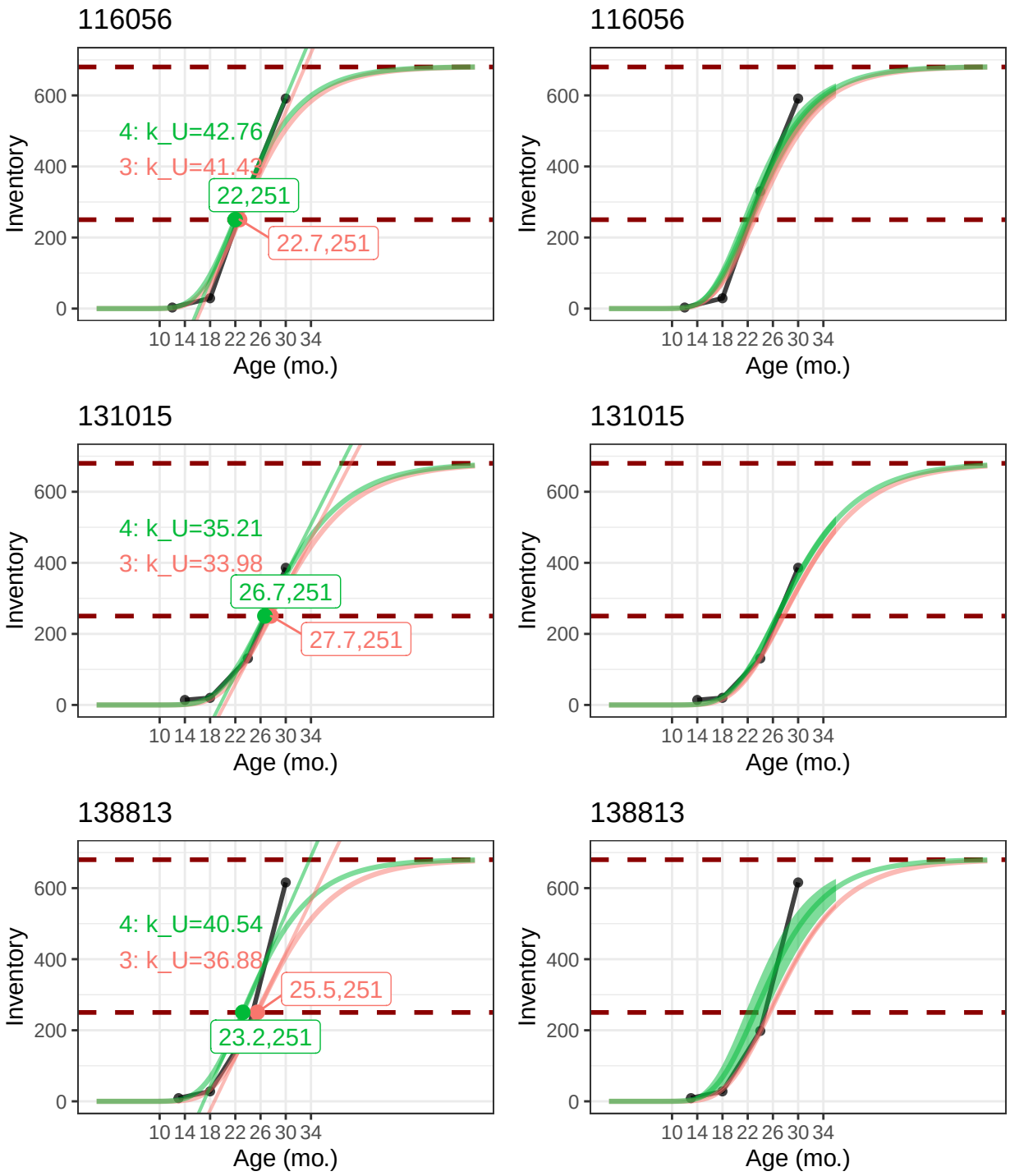
```
plot.tri(test_all[[1]],
         fit3 = all.fits3[[1]],
         fit4 = all.fits4[[1]],
         fit5 = all.fits5[[1]])
```

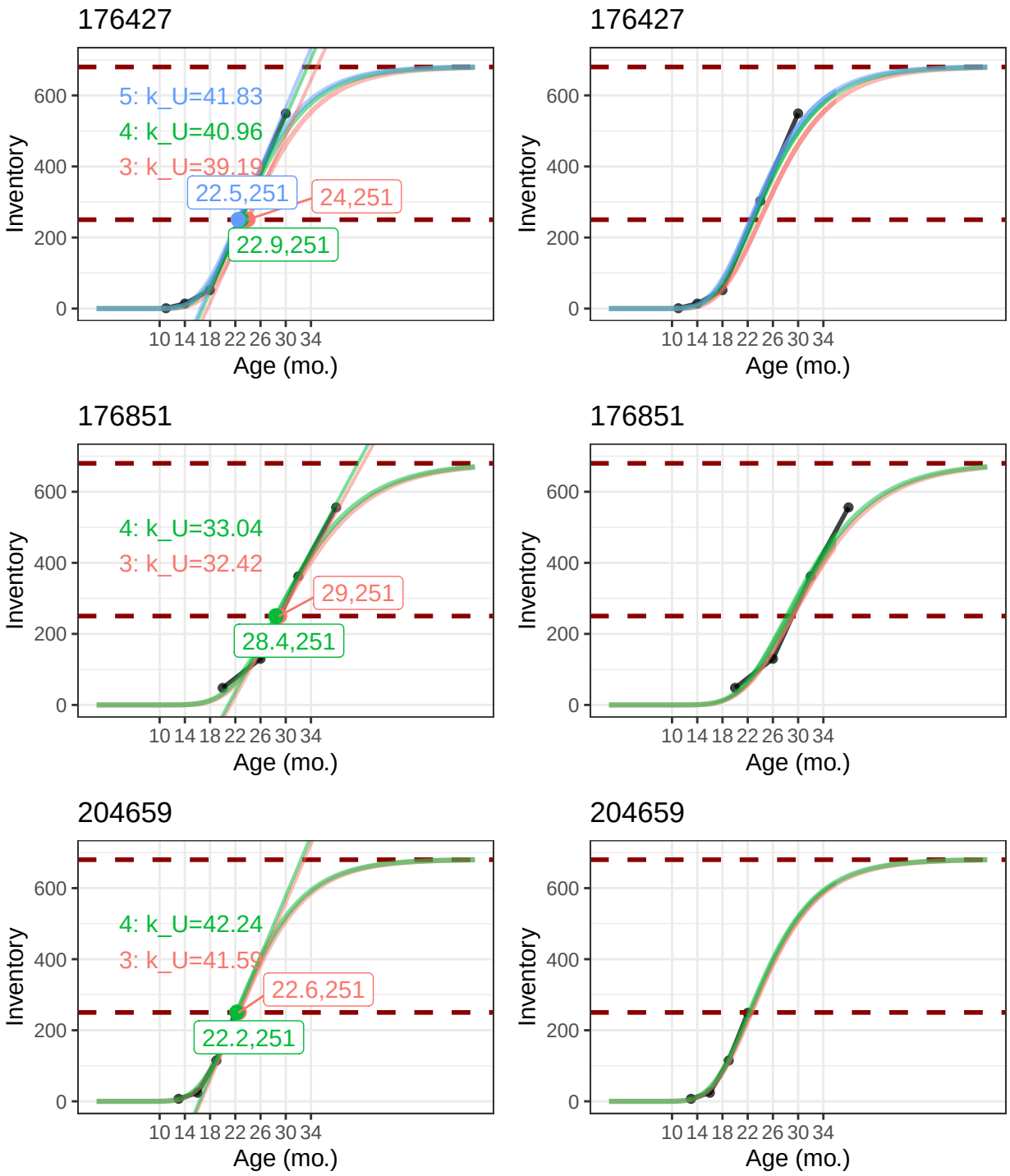


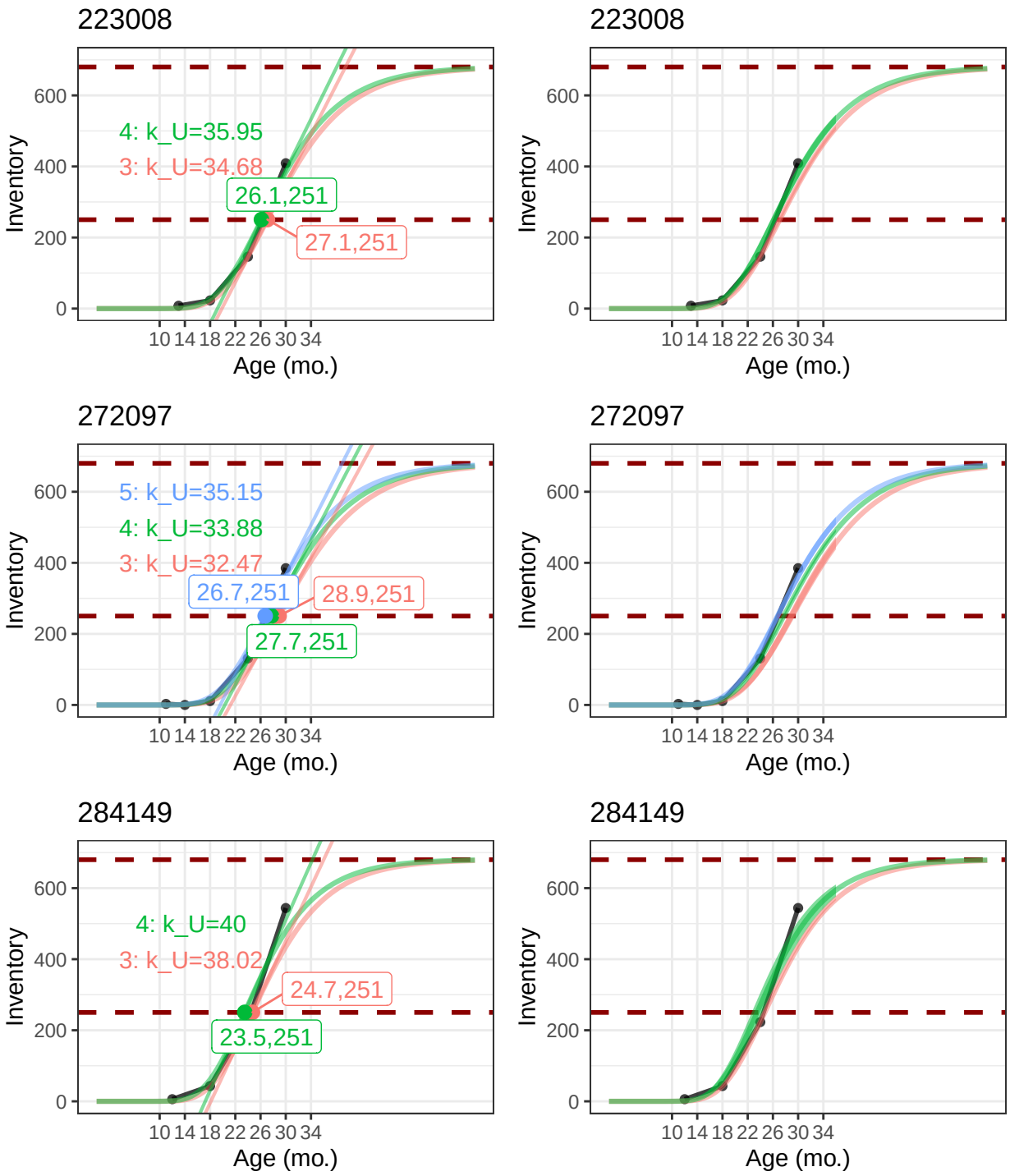

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

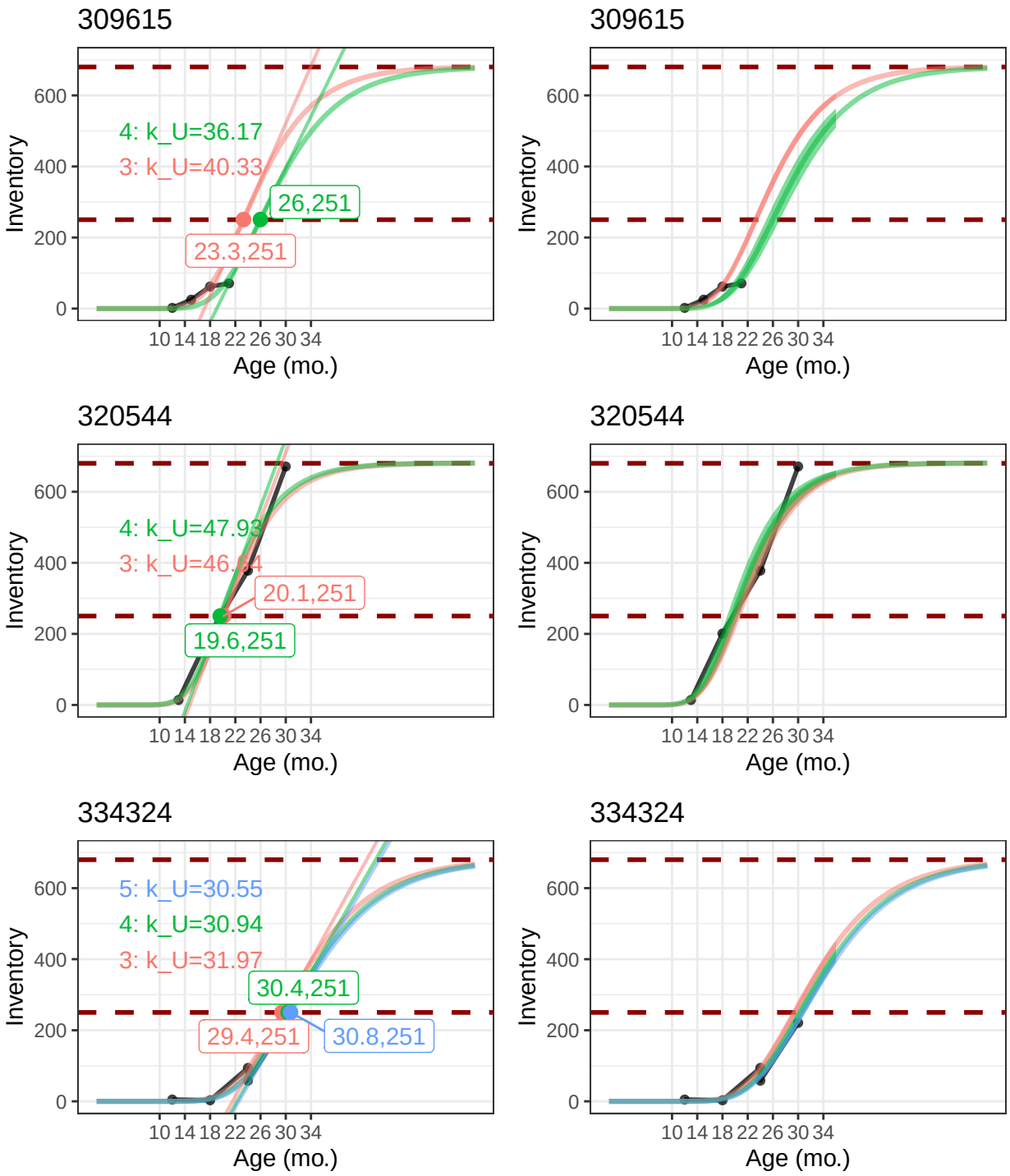
```
all.plots <- lapply(1:51, function(x) plot.tri(test_all[[x]],
                                               all.fits3[[x]],
                                               all.fits4[[x]],
                                               all.fits5[[x]]))
```

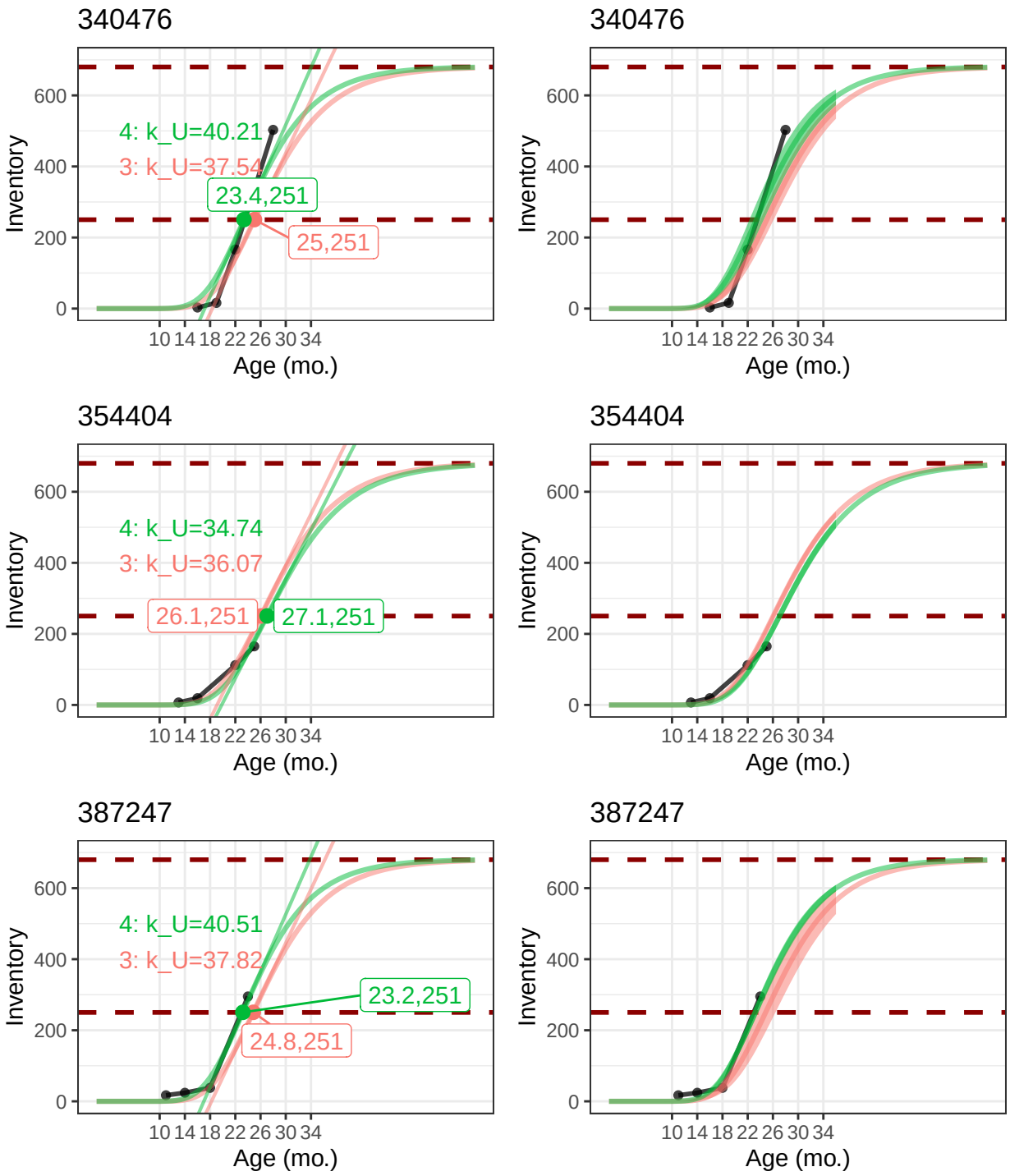


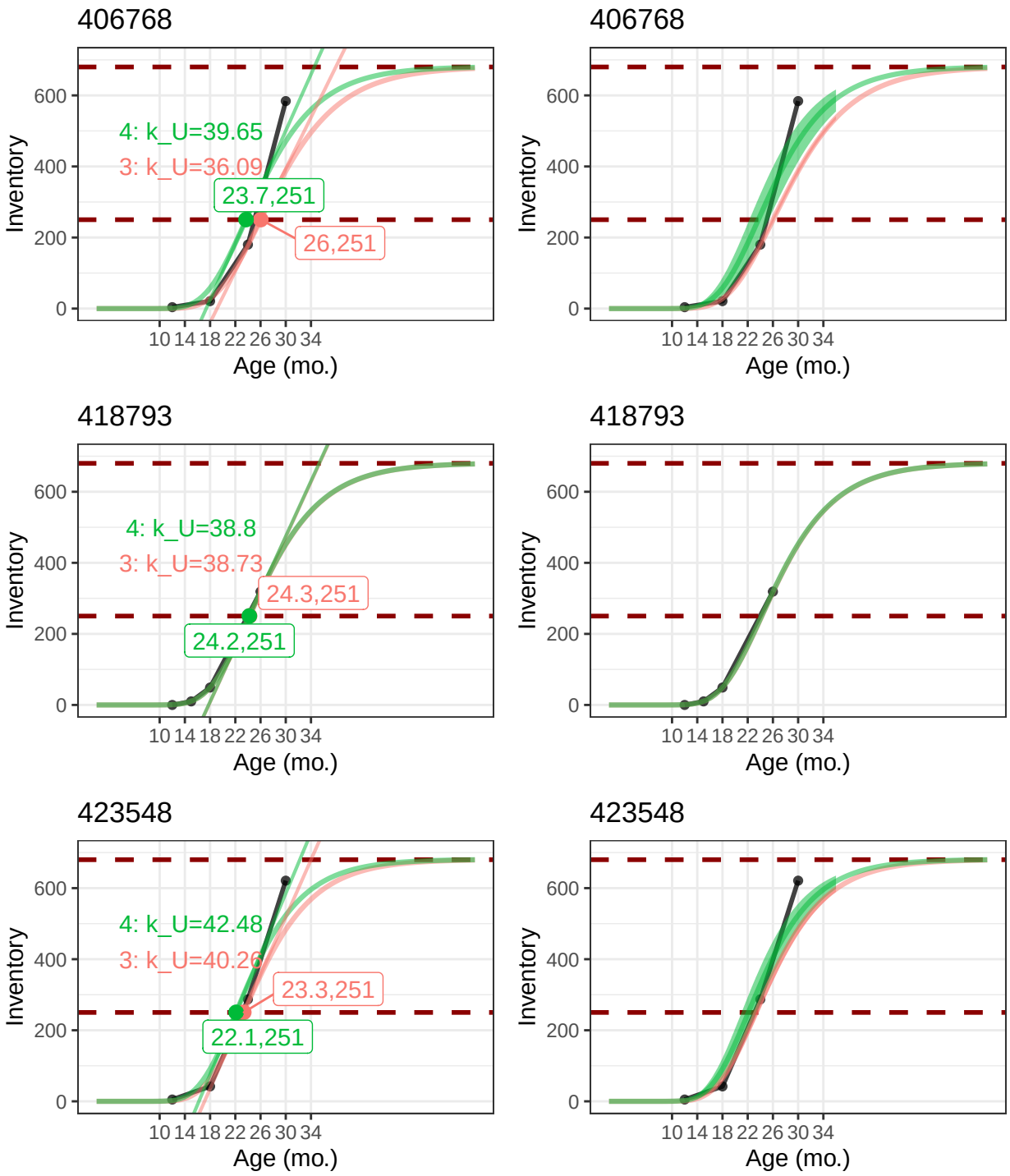

116056
4: k_U=42.76
3: k_U=41.43
22,251
22.7,251
Inventory
Age (mo.)
116056
131015
4: k_U=35.21
3: k_U=33.98
26.7,251
27.7,251
131015
138813
4: k_U=40.54
3: k_U=36.88
25.5,251
23.2,251
138813

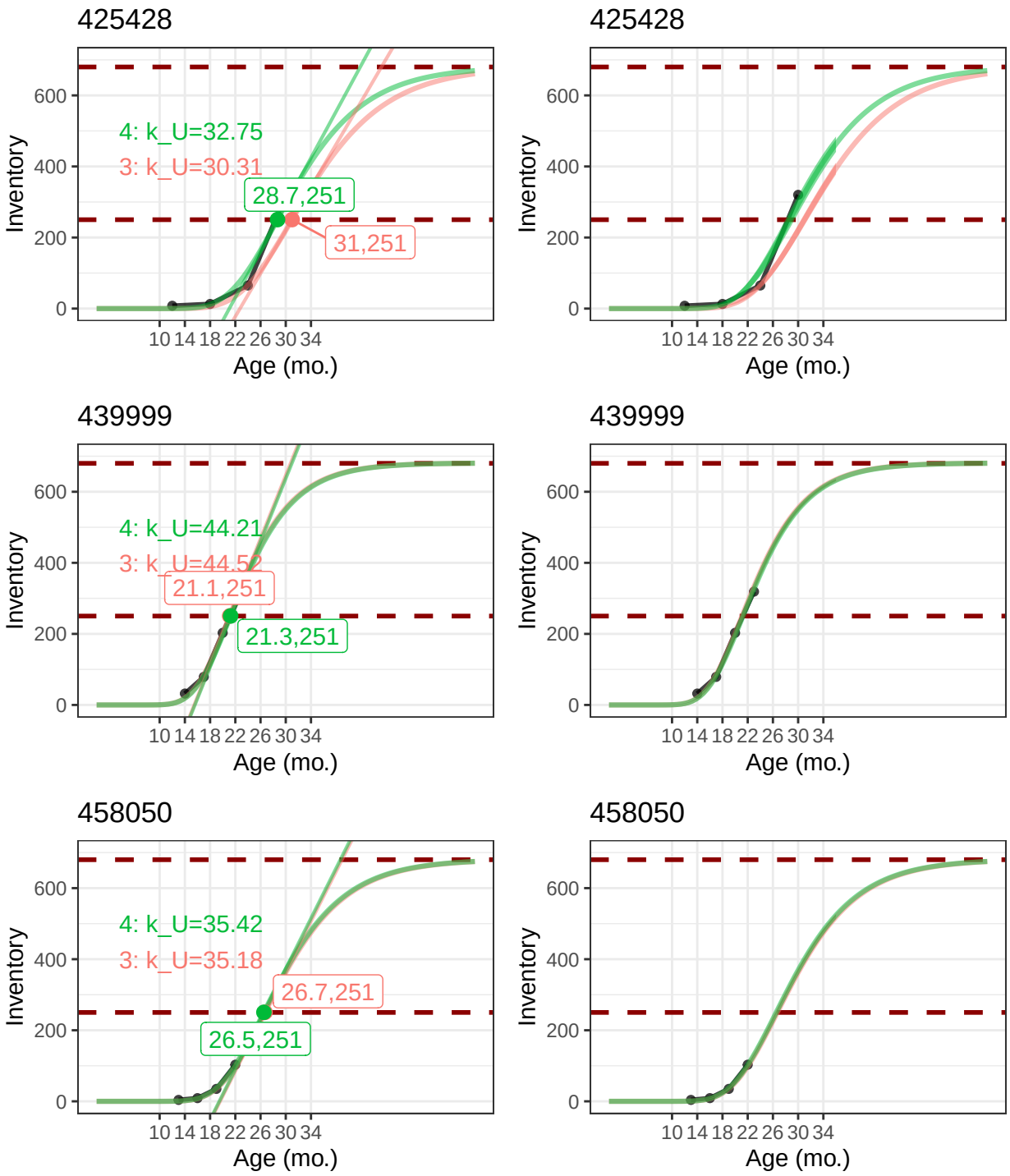

176427
5: k_U=41.83
4: k_U=40.96
3: k_U=39.19
22.5,251
24,251
22.9,251
Inventory
Age (mo.)
176427
176851
4: k_U=33.04
3: k_U=32.42
29,251
28.4,251
176851
204659
4: k_U=42.24
3: k_U=41.59
22.6,251
22.2,251
204659

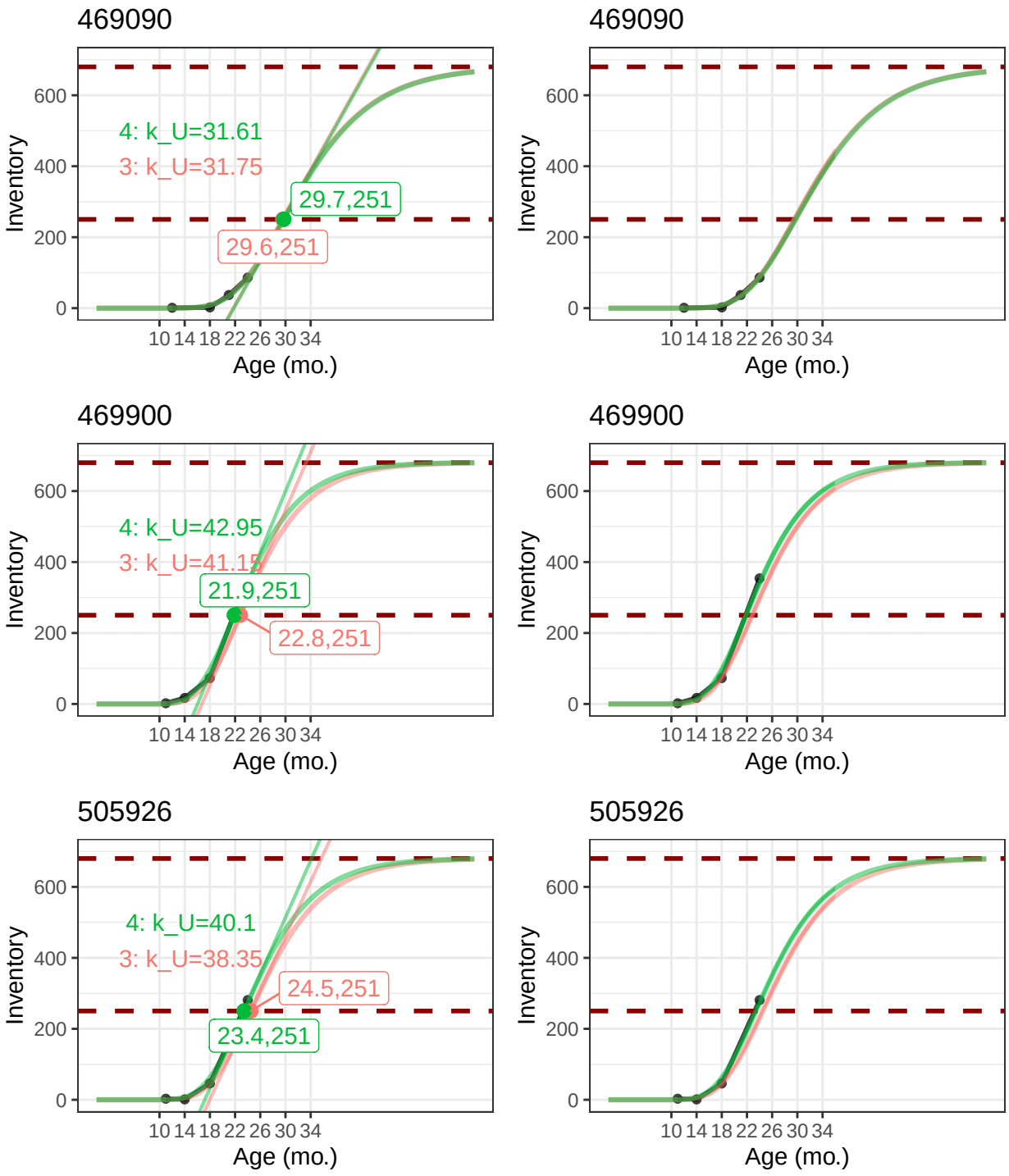

223008
4: k_U=35.95
3: k_U=34.68
26.1,251
27.1,251
Inventory
Age (mo.)
223008
272097
5: k_U=35.15
4: k_U=33.88
3: k_U=32.47
26.7,251
28.9,251
27.7,251
272097
284149
4: k_U=40
3: k_U=38.02
24.7,251
23.5,251
284149

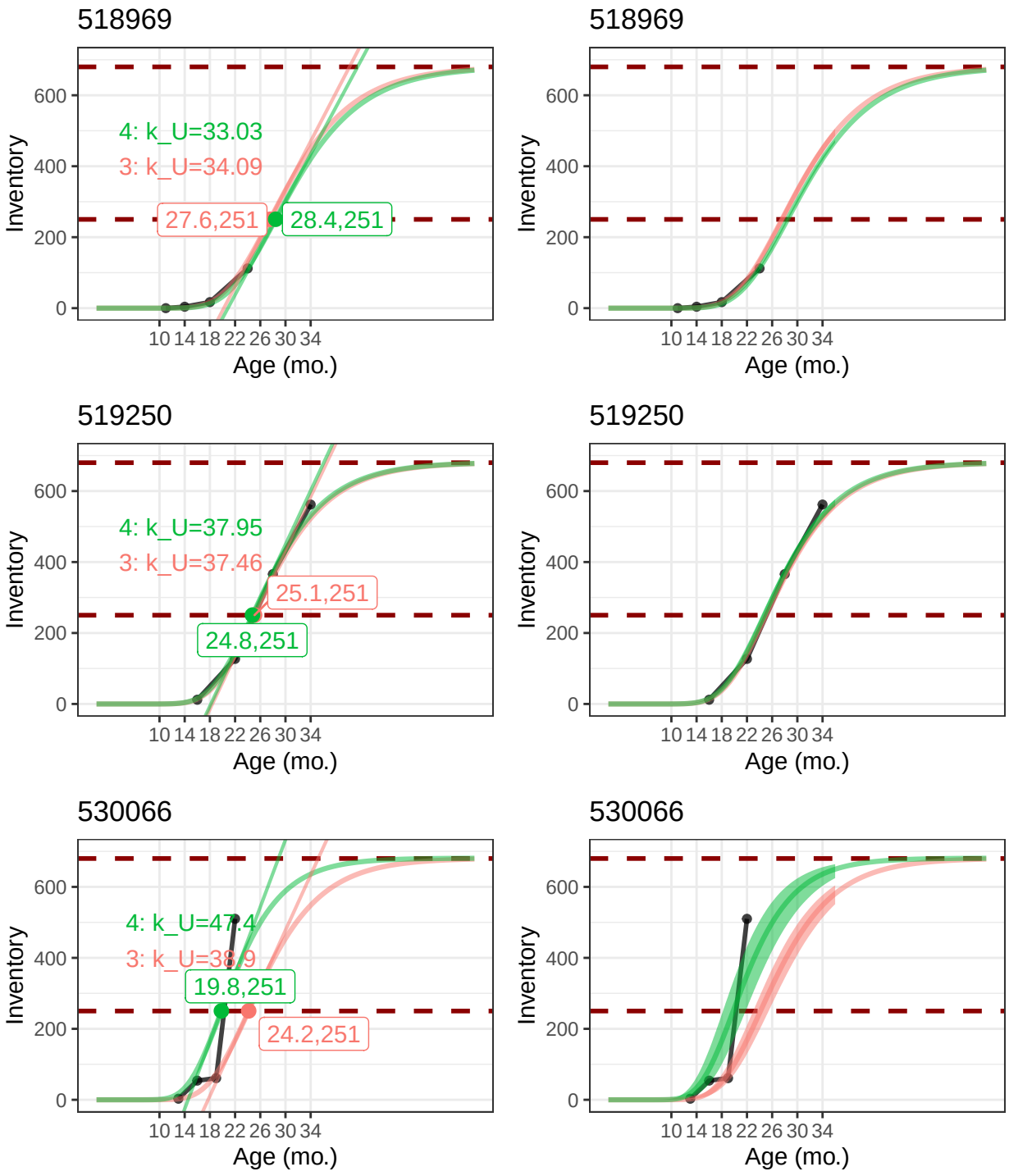

309615
4: k_U=36.17
3: k_U=40.33
26,251
23.3,251
Inventory
Age (mo.)
309615
320544
4: k_U=47.93
3: k_U=46.84
20.1,251
19.6,251
320544
334324
5: k_U=30.55
4: k_U=30.94
3: k_U=31.97
30.4,251
29.4,251
30.8,251
334324

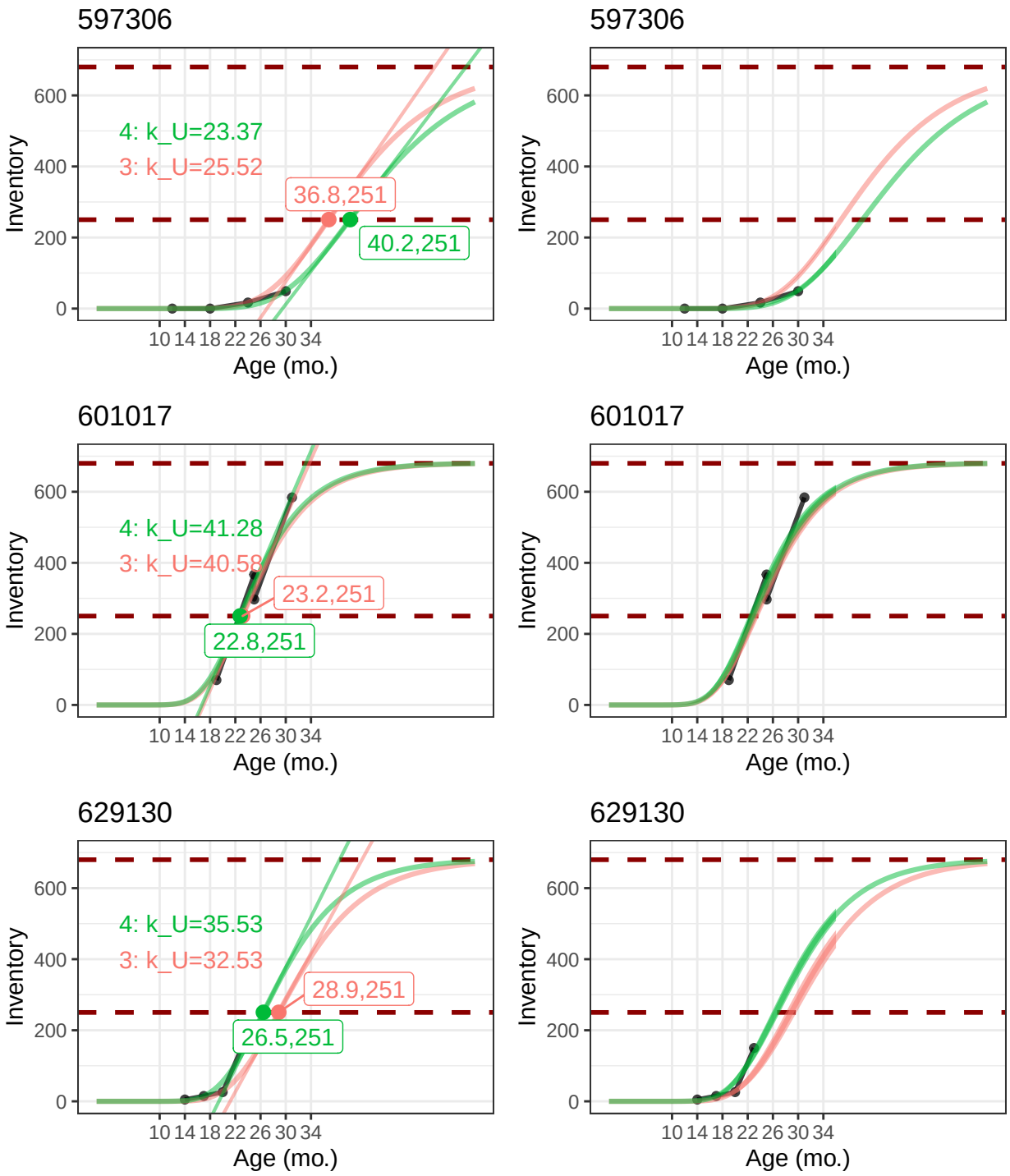

340476
4: k_U=40.21
3: k_U=37.54
23.4,251
25,251
340476
354404
4: k_U=34.74
3: k_U=36.07
26.1,251
27.1,251
354404
387247
4: k_U=40.51
3: k_U=37.82
23.2,251
24.8,251
387247
Inventory
Age (mo.)
0
200
400
600
10 14 18 22 26 30 34

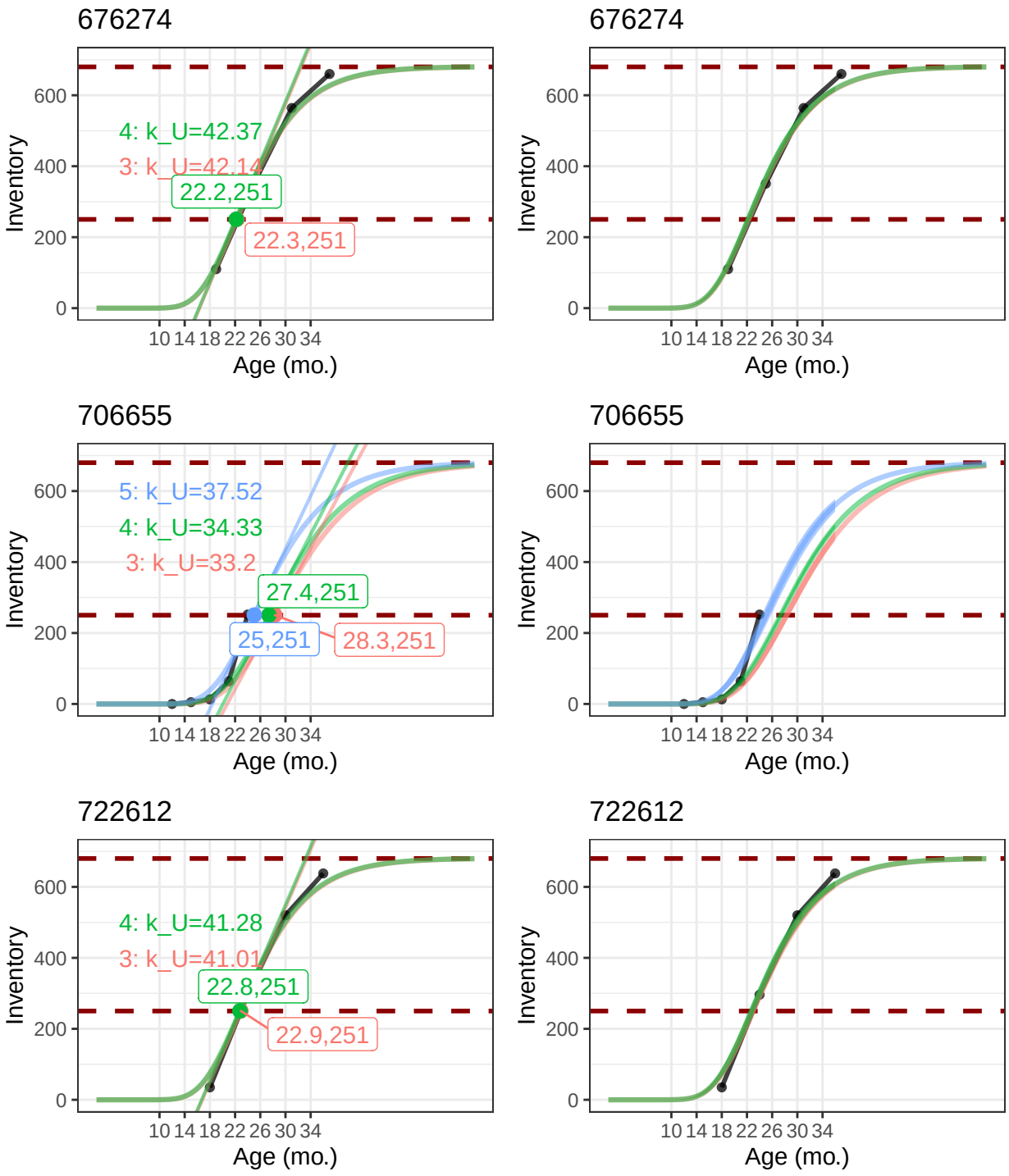

406768
4: k_U=39.65
3: k_U=36.09
23.7,251
26,251
406768
418793
4: k_U=38.8
3: k_U=38.73
24.3,251
24.2,251
418793
423548
4: k_U=42.48
3: k_U=40.26
23.3,251
22.1,251
423548
Inventory
Age (mo.)
0
200
400
600
10 14 18 22 26 30 34

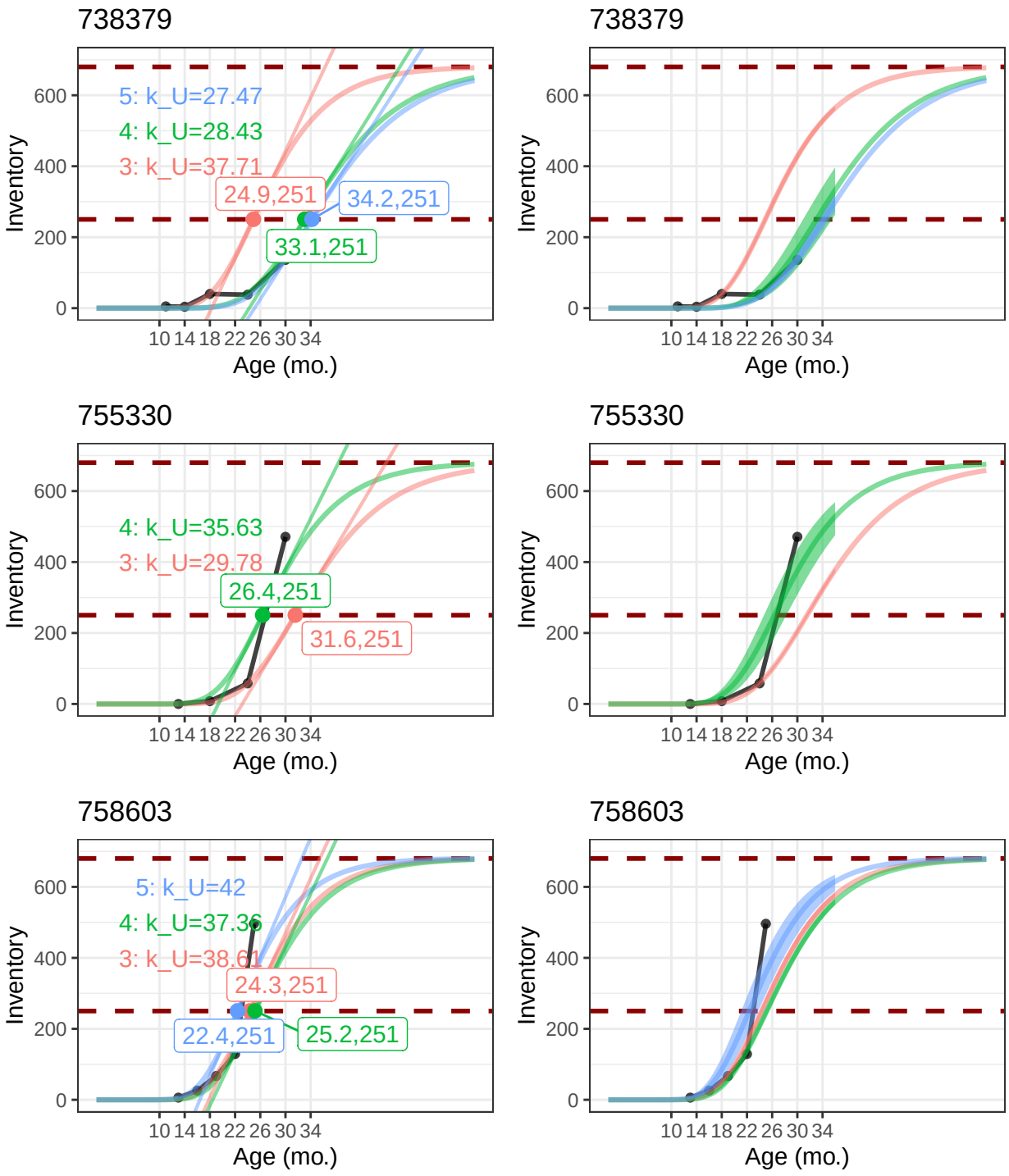

425428
4: k_U=32.75
3: k_U=30.31
28.7,251
31,251
425428
439999
4: k_U=44.21
3: k_U=44.52
21.1,251
21.3,251
439999
458050
4: k_U=35.42
3: k_U=35.18
26.7,251
26.5,251
458050
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

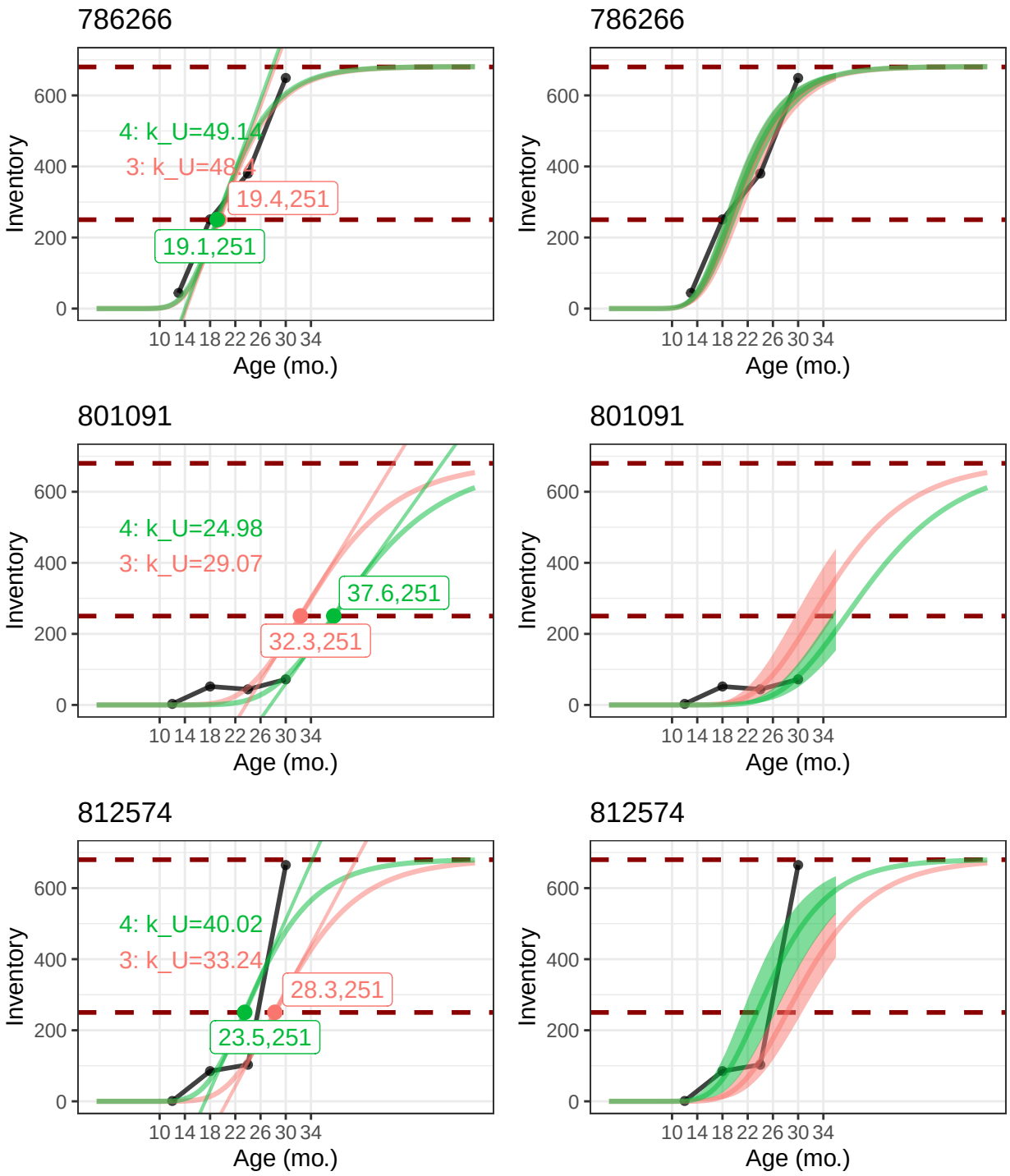

469090
4: k_U=31.61
3: k_U=31.75
29.7,251
29.6,251
469090
469900
4: k_U=42.95
3: k_U=41.15
21.9,251
22.8,251
469900
505926
4: k_U=40.1
3: k_U=38.35
24.5,251
23.4,251
505926
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

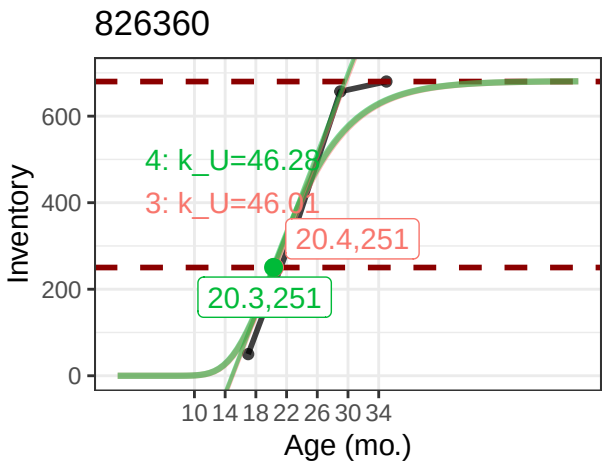

518969
4: k_U=33.03
3: k_U=34.09
27.6,251
28.4,251
518969
519250
4: k_U=37.95
3: k_U=37.46
25.1,251
24.8,251
519250
530066
4: k_U=47.4
3: k_U=38.9
19.8,251
24.2,251
530066
Inventory
Age (mo.)
0
200
400
600
10 14 18 22 26 30 34

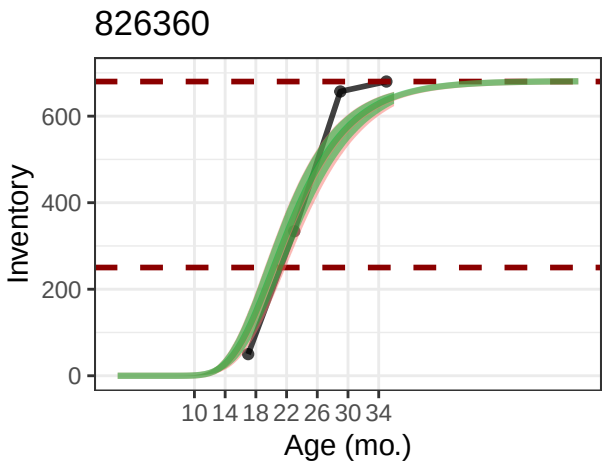

597306
4: k_U=23.37
3: k_U=25.52
36.8,251
40.2,251
Inventory
Age (mo.)
597306
601017
4: k_U=41.28
3: k_U=40.58
23.2,251
22.8,251
601017
629130
4: k_U=35.53
3: k_U=32.53
28.9,251
26.5,251
629130

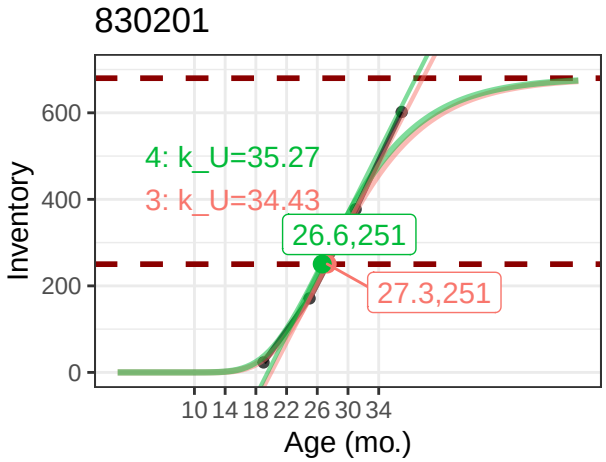

676274
4: k_U=42.37
3: k_U=42.14
22.2,251
22.3,251
Inventory
Age (mo.)
676274
706655
5: k_U=37.52
4: k_U=34.33
3: k_U=33.2
27.4,251
25,251
28.3,251
706655
722612
4: k_U=41.28
3: k_U=41.01
22.8,251
22.9,251
722612

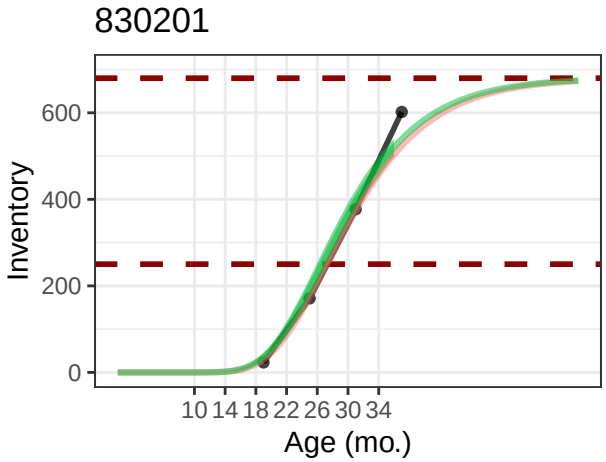

738379
5: k_U=27.47
4: k_U=28.43
3: k_U=37.71
24.9,251
34.2,251
33.1,251
738379
755330
4: k_U=35.63
3: k_U=29.78
26.4,251
31.6,251
755330
758603
5: k_U=42
4: k_U=37.36
3: k_U=38.61
24.3,251
22.4,251
25.2,251
758603
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

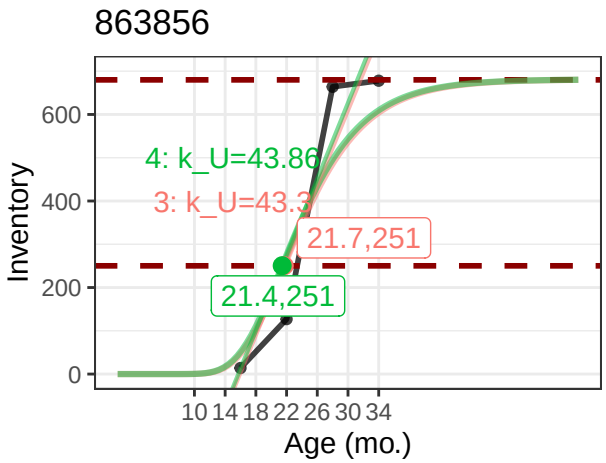

786266
4: k_U=49.14
3: k_U=48.4
19.4,251
19.1,251
Inventory
Age (mo.)
786266
801091
4: k_U=24.98
3: k_U=29.07
37.6,251
32.3,251
801091
812574
4: k_U=40.02
3: k_U=33.24
28.3,251
23.5,251
812574

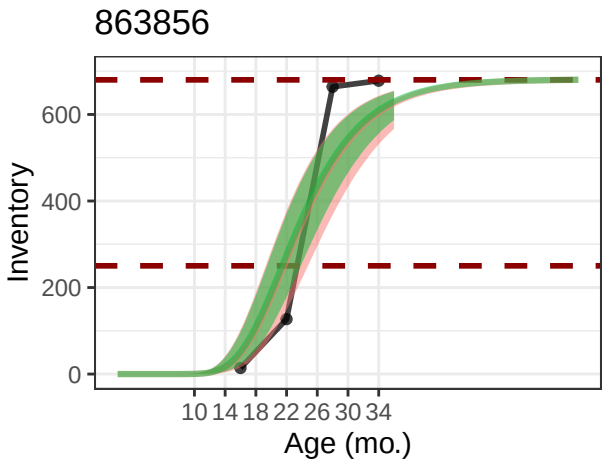

826360
4: k_U=46.28
3: k_U=46.01
20.4,251
20.3,251
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

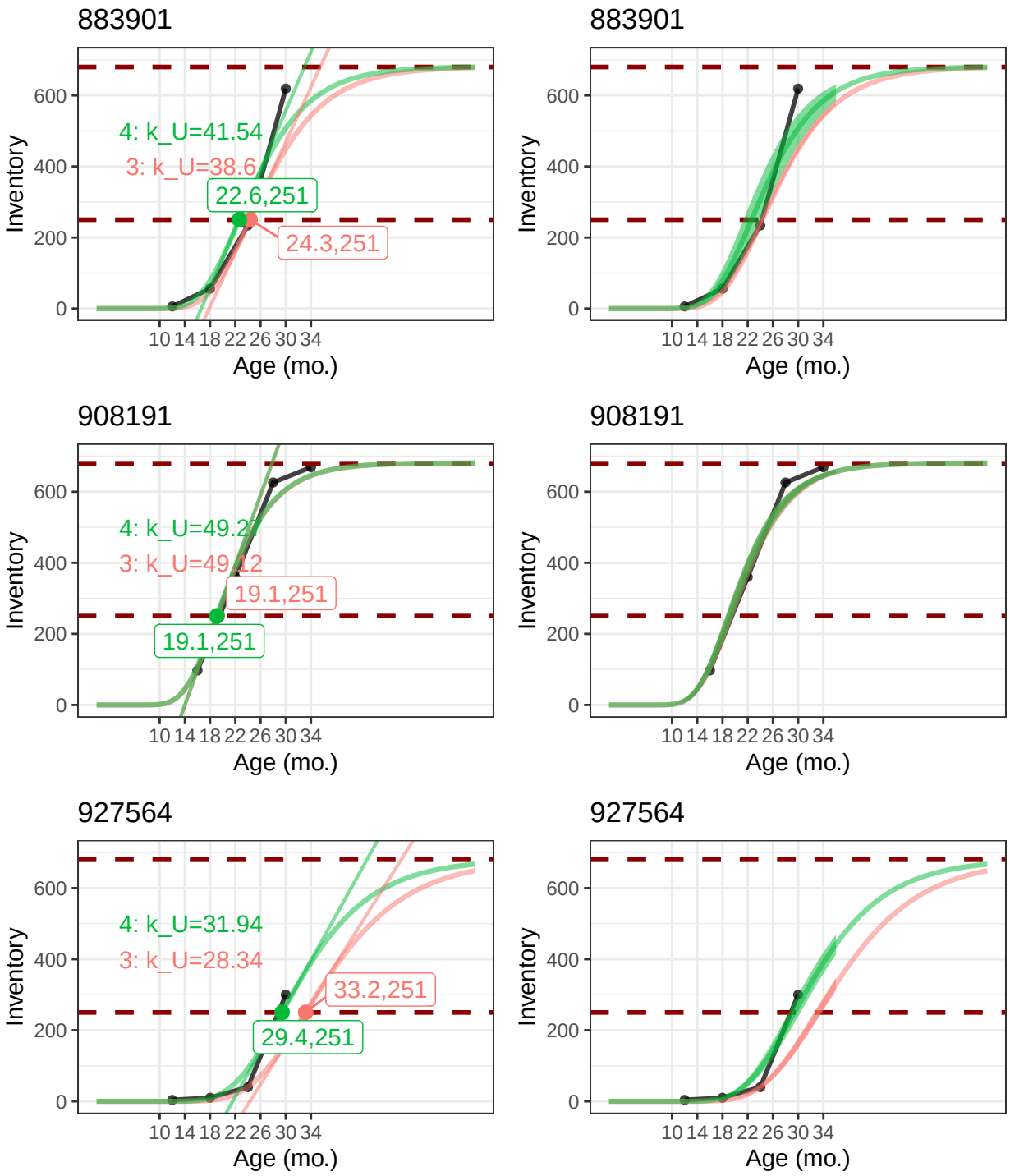

826360
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

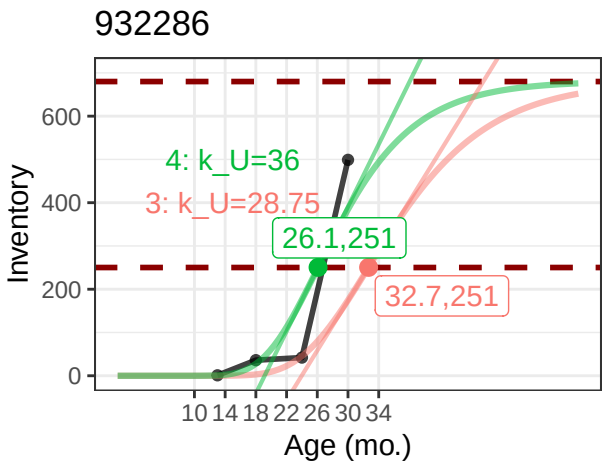

830201
4: k_U=35.27
3: k_U=34.43
26.6,251
27.3,251
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

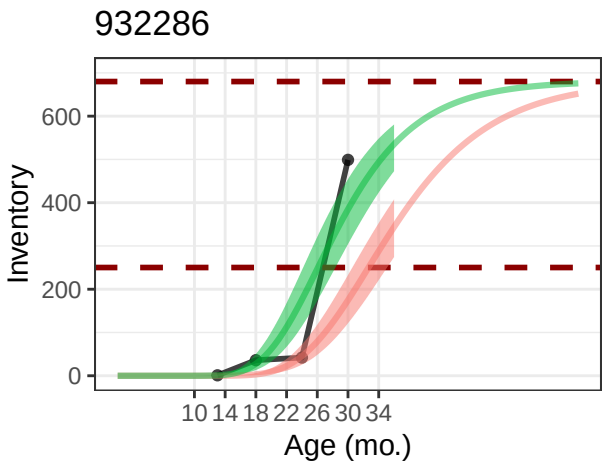

830201
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

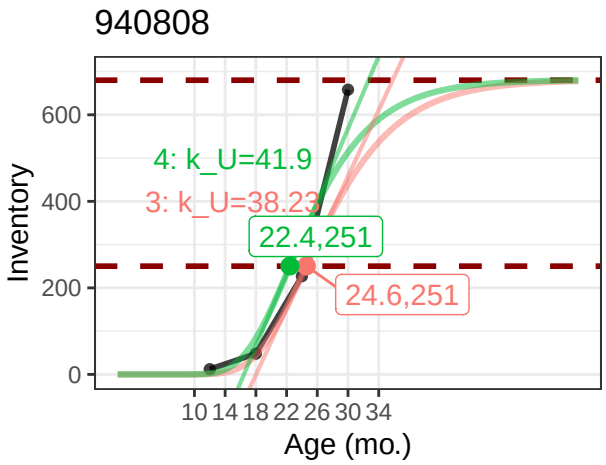

863856
4: k_U=43.86
3: k_U=43.3
21.7,251
21.4,251
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

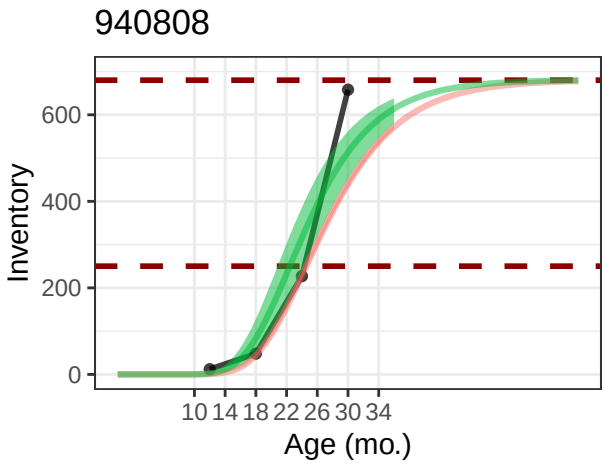

863856
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

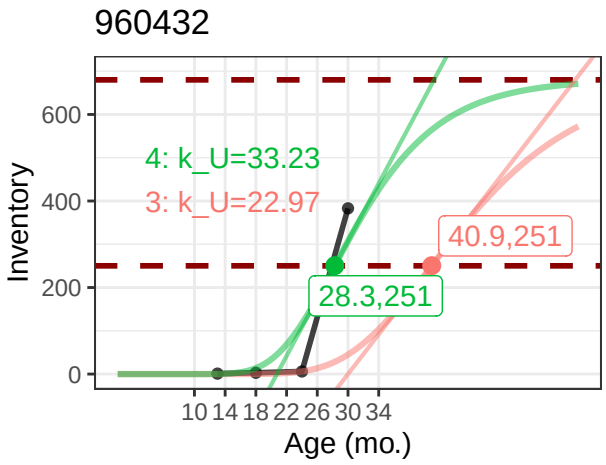

883901
4: k_U=41.54
3: k_U=38.6
22.6,251
24.3,251
883901
908191
4: k_U=49.27
3: k_U=49.12
19.1,251
19.1,251
908191
927564
4: k_U=31.94
3: k_U=28.34
33.2,251
29.4,251
927564
Inventory
0
200
400
600
10 14 18 22 26 30 34
Age (mo.)

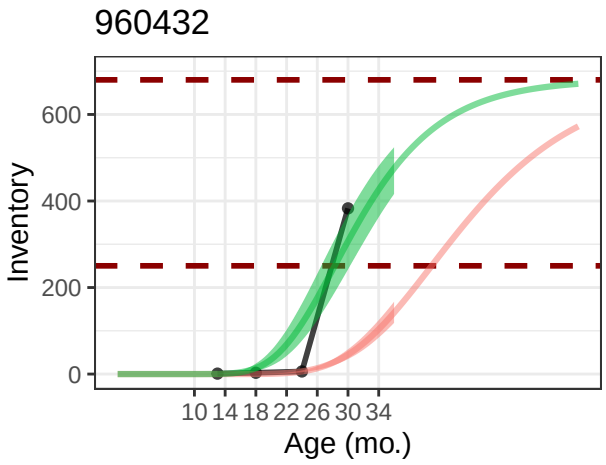

932286
4: k_U=36
3: k_U=28.75
26.1,251
32.7,251
Inventory
Age (mo.)

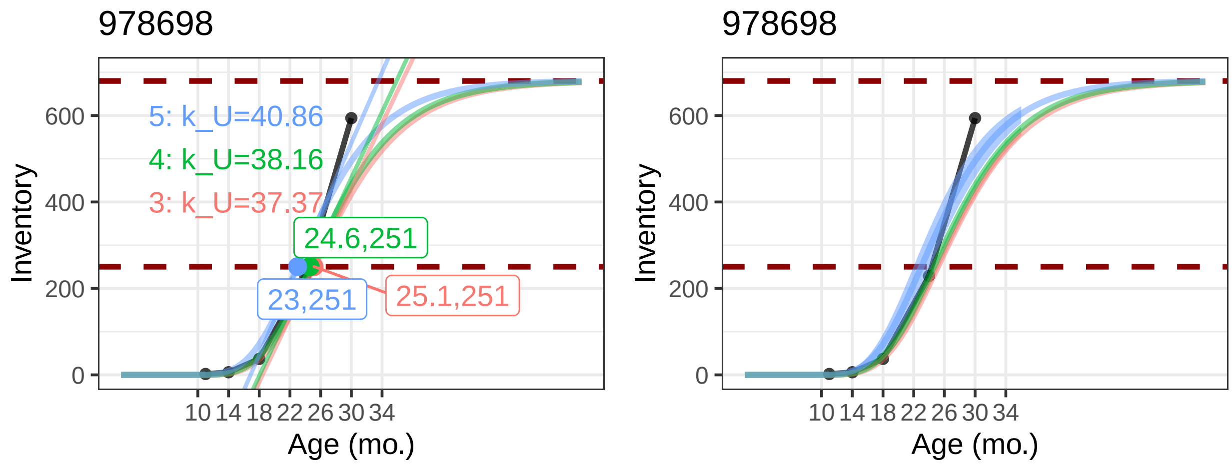

932286
Inventory
Age (mo.)

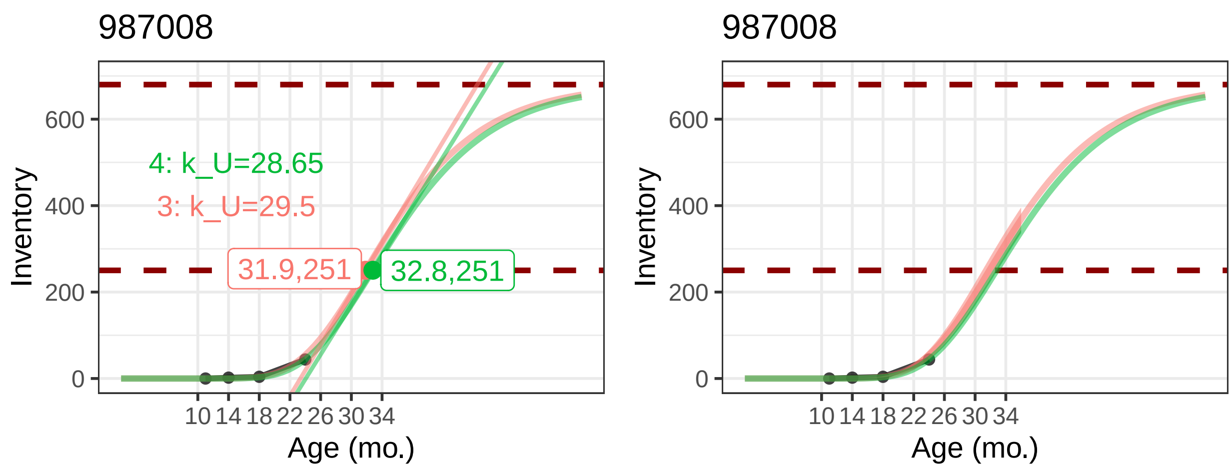

940808
4: k_U=41.9
3: k_U=38.23
22.4,251
24.6,251
Inventory
Age (mo.)


940808
Inventory
Age (mo.)


960432
4: k_U=33.23
3: k_U=22.97
40.9,251
28.3,251
Inventory
Age (mo.)


960432
Inventory
Age (mo.)






```r
for (p in all.plots){
  print(p) %>% suppressMessages()
}
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
```

```
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
```

```
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
```

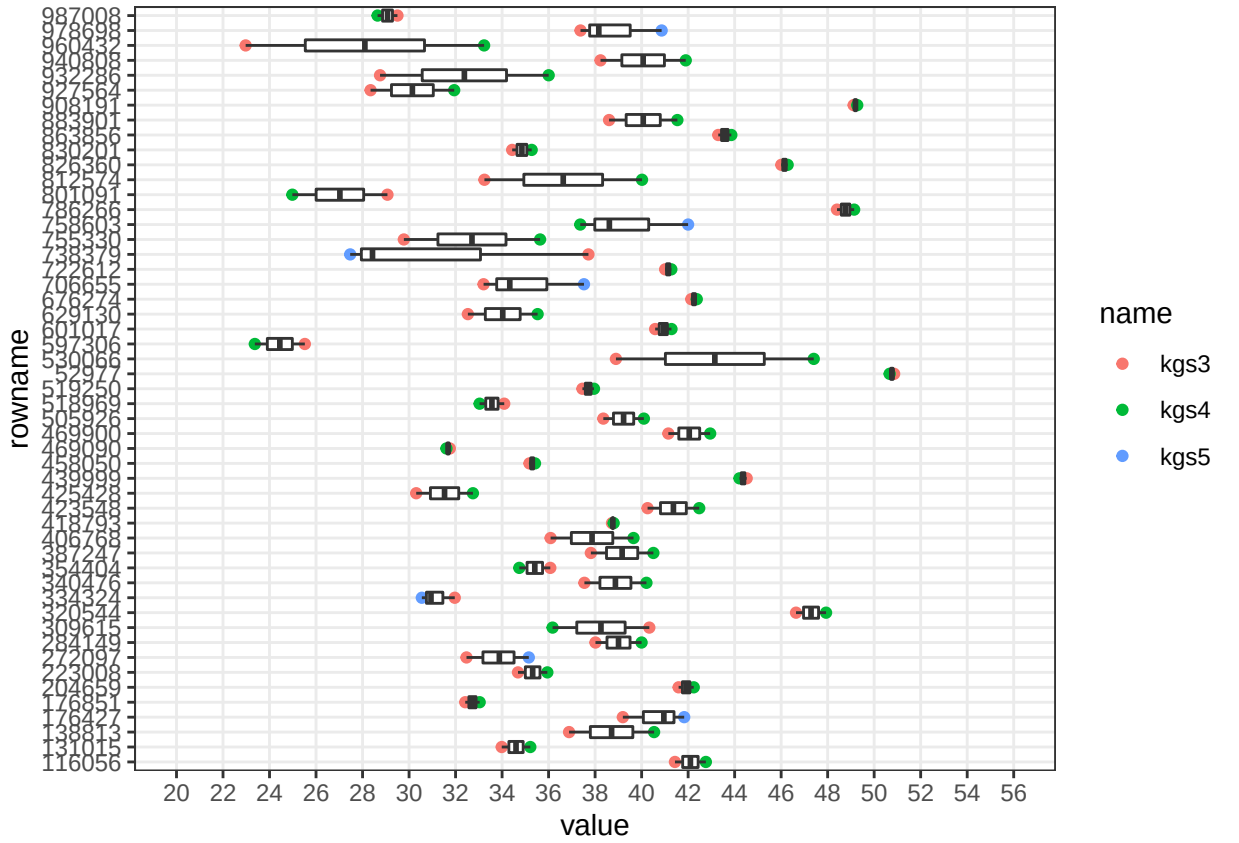
```
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
```

```
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

Distribution of k_U

How widely distributed is k_U?

```
kgs5 <- sapply(all.fits5, function(x) if(is.list(x)) {summary(x)$coefficients[1, 1]} else {NA})
kgs4 <- sapply(all.fits4, function(x) summary(x)$coefficients[1, 1])
kgs3 <- sapply(all.fits3, function(x) summary(x)$coefficients[1, 1])

all.kgs <- cbind(kgs3, kgs4, kgs5)
all.kUs <- (A * all.kgs / exp(1)) %>%
            as.data.frame() %>%
            rownames_to_column() %>%
            pivot_longer(-rowname) %>%
            na.omit()

max_vals <- bind_rows(test_all) %>%
              group_by(data_id) %>%
              summarize(age = max(age), W = max(inventory)) %>%
              mutate(data_id = as.character(data_id))

ggplot(all.kUs, aes(x = rowname, y = value)) +
  geom_point(aes(color = name)) +
  geom_boxplot(outlier.shape = NULL) +
  coord_flip() +
  theme_bw() +
  scale_y_continuous(limits = c(20, 56), breaks = seq(20, 56, by = 2),
                     minor_breaks = NULL)
```



This shows that for the majority of individuals, using 3, 4, or 5 points provide reasonable estimates, however, there are some individuals whose spread is large.

These individuals don't seem to have a systematically low oldest visit or low largest inventory size.