Quadratic and Ratio Rescaling of Parameters with Fixed Min and Max

Fan Wang

2022-07-25

Contents

1 Rescale a Parameter with Fixed Min and Max

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.1 Rescale a Fraction Between 0 and 1

We have ratio $0 < \theta < 1$, we want to multiply θ to get closer to 0 or to 1, but to not exceed the bounds of 0 and 1. This is achieved by the following $\widehat{\theta}(\theta, \lambda)$ function, where we adjust λ between negative and positive infinities to move θ closer to 0 (as $\lambda \rightarrow -\infty$) or θ closer to 1 (as $\lambda \rightarrow \infty$).

$$\widehat{\theta}(\theta, \lambda) = \theta \cdot \left(\frac{\exp(\lambda)}{1 + \theta \cdot (\exp(\lambda) - 1)} \right)$$

Given this function form, when $\lambda = 0$, we are back to θ, and $0 \le \widehat{\theta}(\lambda) \le 1$, which allows $\widehat{\theta}$ to be used as the "chance of success" parameter as we freely vary λ.

$$\begin{aligned}
\widehat{\theta}(\theta, \lambda = 0) &= \theta \cdot \left(\frac{1}{1 + \theta \cdot (1 - 1)} \right) = \theta \\
\lim_{\lambda \rightarrow -\infty} \widehat{\theta}(\theta, \lambda) &= \theta \cdot \left(\frac{0}{1 + \theta \cdot (0 - 1)} \right) = 0 \\
\lim_{\lambda \rightarrow \infty} \widehat{\theta}(\theta, \lambda) &= \theta \cdot \left(\frac{\exp(\lambda)}{\theta \cdot \exp(\lambda)} \right) = 1
\end{aligned}$$

To test this, first, we write out the rescaling function.

```
# Construct the formula
ffi_theta_lambda_0t1 <- function(theta, lambda) {
  if (is.finite(exp(lambda))) {
    theta * (exp(lambda) / (1 + theta * (exp(lambda) - 1)))
  } else {
    # If lambda is large, exp(lambda)=inf, ratio above becomes 1
    1
```

```
  }
}
# Test the function
print(ffi_theta_lambda_0t1(0.5, 1e1))
```

```
## [1] 0.9999546
```

```
print(ffi_theta_lambda_0t1(0.5, 1e2))
```

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

```
print(ffi_theta_lambda_0t1(0.5, -1e3))
```

```
## [1] 0
```

Second, given theta, we evaluate the function with differing lambda values.

```
# Create Function
ffi_fixtheta_varylambda_0t1 <-
  function(ar_lambda_pos =
             1e1^(seq(-0.1, 1, length.out = 4)),
           theta = 0.5) {

    # Construct lambda vector
    ar_lambda <- sort(unique((c(-ar_lambda_pos, 0, ar_lambda_pos))))
    # sapply
    ar_theta_hat <- sapply(ar_lambda, ffi_theta_lambda_0t1, theta = theta)
    # Create table
    ar_st_varnames <- c("theta_hat", "lambda")
    tb_theta_hat_lambda <- as_tibble(
      cbind(round(ar_theta_hat, 5), ar_lambda)
    ) %>%
      rename_all(~ c(ar_st_varnames)) %>%
      mutate(theta = theta)
    # return
    return(tb_theta_hat_lambda)
  }

# Test function
tb_theta_hat_lambda <- ffi_fixtheta_varylambda_0t1()
# Print
kable(tb_theta_hat_lambda,
  caption = paste(
    "Theta-hat rescaling",
    ", given different lambda rescalers.",
    separator = " "
  )
) %>% kable_styling_fc()
```

Third, we run the function we just created for two three different θ levels, and we stack the results together.

```
# Evaluate at differing thetas
ar_lambda_pos <- 1e1^(seq(-0.1, 1, length.out = 2))
tb_theta_hat_lambda_low <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.1)
tb_theta_hat_lambda_mid <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.5)
tb_theta_hat_lambda_hgh <- ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.9)
# Combine
```

Theta-hat rescaling , given different lambda rescalers.

theta_hat	lambda	theta
0.00005	-10.0000000	0.5
0.01340	-4.2986623	0.5
0.13613	-1.8478498	0.5
0.31124	-0.7943282	0.5
0.50000	0.0000000	0.5
0.68876	0.7943282	0.5
0.86387	1.8478498	0.5
0.98660	4.2986623	0.5
0.99995	10.0000000	0.5

```
tb_theta_hat_lambda_combo <- bind_rows(
  tb_theta_hat_lambda_low,
  tb_theta_hat_lambda_mid,
  tb_theta_hat_lambda_hgh
)
# Print
kable(tb_theta_hat_lambda_combo,
  caption = paste(
    "Theta-hat rescaling",
    ", with multiple theta values",
    ", given different lambda rescalers.",
    separator = " "
  )
) %>% kable_styling_fc()
```

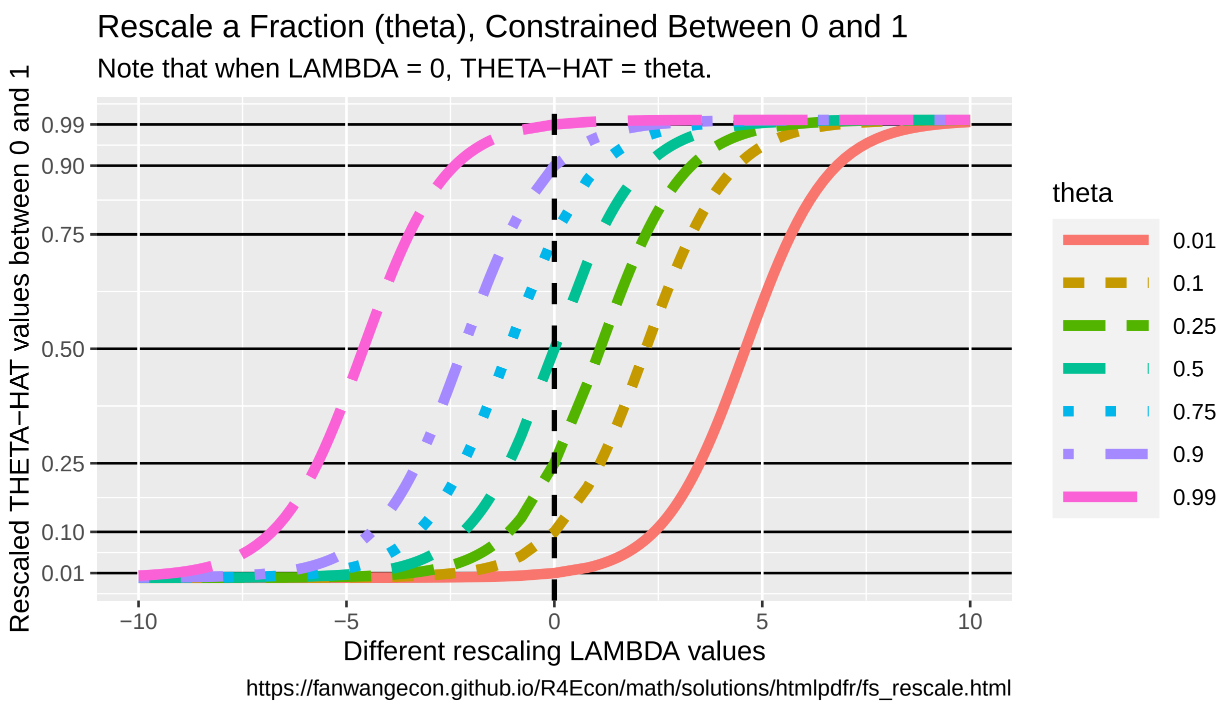
Theta-hat rescaling , with multiple theta values , given different lambda rescalers.

theta_hat	lambda	theta
0.00001	-10.0000000	0.1
0.04781	-0.7943282	0.1
0.10000	0.0000000	0.1
0.19736	0.7943282	0.1
0.99959	10.0000000	0.1
0.00005	-10.0000000	0.5
0.31124	-0.7943282	0.5
0.50000	0.0000000	0.5
0.68876	0.7943282	0.5
0.99995	10.0000000	0.5
0.00041	-10.0000000	0.9
0.80264	-0.7943282	0.9
0.90000	0.0000000	0.9
0.95219	0.7943282	0.9
0.99999	10.0000000	0.9

Fourth, we visualize the results from above. We generate a denser x-grid for this purpose, and we evaluate at 9 different theta values from $\theta = 0.01$ to $\theta = 0.99$. We can see in the graph that all $0 < \hat{\theta} < 1$.

```
# Generate a denser result
ar_lambda_pos <- 1e1^(seq(-0.1, 1, length.out = 100))
tb_theta_hat_lambda_combo <- bind_rows(
```

```r
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.01),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.10),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.25),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.50),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.75),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.90),
  ffi_fixtheta_varylambda_0t1(ar_lambda_pos, 0.99)
)
# Labeling
st_title <- paste0("Rescale a Fraction (theta), Constrained Between 0 and 1")
st_subtitle <- paste0(
  "Note that when LAMBDA = 0, THETA-HAT = theta."
)
st_caption <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_rescale.html"
)
st_x_label <- "Different rescaling LAMBDA values"
st_y_label <- "Rescaled THETA-HAT values between 0 and 1"
ar_y_breaks <- c(0.01, 0.10, 0.25, 0.50, 0.75, 0.90, 0.99)
# Graph
tb_theta_hat_lambda_combo %>%
  mutate(theta = as.factor(theta)) %>%
  ggplot(aes(
    x = lambda, y = theta_hat,
    color = theta, linetype = theta
  )) +
  geom_line(size = 2) +
  geom_vline(
    xintercept = 0,
    linetype = "dashed", color = "black", size = 1
  ) +
  labs(
    title = st_title,
    subtitle = st_subtitle,
    x = st_x_label,
    y = st_y_label,
    caption = st_caption
  ) +
  scale_y_continuous(
    breaks = ar_y_breaks,
    limits = c(0, 1)
  ) +
  theme(
    panel.grid.major.y = element_line(
      color = "black",
      size = 0.5,
      linetype = 1
    ),
    legend.key.width = unit(3, "line")
  )
```



1.2 Fit Three Points with Ends Along 45 Degree Line

Given e < x < f, use f(x) to rescale x, such that f(e)=e, f(f)=f, but f(z)=$\alpha \cdot z$ for one particular z between e and f, with $\alpha > 1$. And in this case, assume that $\alpha \cdot z < f$. Note that this is case where we have three points, and the starting and the ending points are along the 45 degree line.

We can fit these three points using the Quadratic function exactly. In another word, there is a unique quadratic function that crosses these three points. Note the quadratic function is either concave or convex through the entire domain.

First, as an example, suppose that $e = 0$, $f = 10$, $z = 2$, and $\alpha = 1.5$. Using a quadratic to fit:

$$y(x) = a \cdot x^2 + b \cdot x + c$$

We have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c2 \cdot 1.5 = a \cdot 2^2 + b \cdot 2 + c10 = a \cdot 10^2 + b \cdot 10 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$3 = a \cdot 4 + b \cdot 210 = a \cdot 100 + b \cdot 10$$

Hence:

$$a = \frac{3 - 2b}{4}10 = \frac{3 - 2b}{4} \cdot 100 + b \cdot 1010 = 75 - 50b + 10b$$

And finally:

$$a = \frac{3 - 2 * 1.625}{4} = -0.0625b = \frac{65}{40} = 1.625c = 0$$

Generate the a, b and c points above for the quadratic function:

```r
# set values
e <- 0
f <- 10
z <- 2
alpha <- 1.5
# apply formulas from above
a <- -0.0625
b <- 1.625
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 11)
# rescale
ar_grid_quad <- a * ar_x^2 + b * ar_x + c
# show values
kable(print(as_tibble(cbind(ar_x, ar_grid_quad))),
  caption = paste0(
    "Quadratic Fit of Three Equations and Three Unknowns\n",
    "Satisfies: f(0)=0, f(10)=10, f(2)=3"
  )
) %>%
  kable_styling_fc()
```

Quadratic Fit of Three Equations and Three Unknowns Satisfies: f(0)=0, f(10)=10, f(2)=3

ar_x	ar_grid_quad
0	0.0000
1	1.5625
2	3.0000
3	4.3125
4	5.5000
5	6.5625
6	7.5000
7	8.3125
8	9.0000
9	9.5625
10	10.0000

Second, as another example, suppose that $e = 0$, $f = 3.5$, $z = 0.5$, and $\alpha = 1.5$. Using a quadratic to fit these, we have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c 0.75 = a \cdot 0.5^2 + b \cdot 0.5 + c 3.5 = a \cdot 3.5^2 + b \cdot 3.5 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$0.75 = a \cdot 0.25 + b \cdot 0.5 3.5 = a \cdot 12.25 + b \cdot 3.5$$

Hence:

$$a = \frac{0.75 - 0.5b}{0.25} 3.5 = \frac{0.75 - 0.5b}{0.25} \cdot 12.25 + b \cdot 3.5 3.5 = 36.75 - 24.5b + 3.5b$$

And finally:

$$a = \frac{0.75 - 0.5 * 1.58333}{0.25} = -0.1666 b = \frac{36.75 - 3.5}{24.5 - 3.5} = 1.58333 c = 0$$

Generate the a, b and c points above for the quadratic function:

```r
# set values
e <- 0
f <- 3.5
z <- 0.5
alpha <- 1.5
# apply formulas from above
a <- -0.16666666
b <- 1.583333333
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 100000)
# rescale
ar_grid_quad <- a * ar_x^2 + b * ar_x + c
# show values
# cbind(ar_x, ar_grid_quad)
ar_x[which.min(abs(ar_grid_quad - 0.75))]
```

```
## [1] 0.500015
```

The exercises above are special cases of the formula we derive on this page: Formulas for Quadratic Parameters and Three Points.