Centering of the ternary balance color scale

Jonas Schöley

Fri Jan 5 14:37:01 2018

Contents

Setup 1
Non-balanced centered color scale 4
Balanced centered color scale 7

Setup

```
# Init ------------------------------------------------------------------

library(scales)
library(ggtern)

# Ternary grid ----------------------------------------------------------

# coordinates and labels for the centered gridlines of a ternary diagram
TernaryCentroidGrid <- function (centroid) {

  # centroid percent difference labels
  labels = seq(-1, 1, 0.1)
  labels = data.frame(
    L = labels[labels >= -centroid[1]][1:10],
    T = labels[labels >= -centroid[2]][1:10],
    R = labels[labels >= -centroid[3]][1:10]
  )

  # breaks of uncentered grid
  breaks = data.frame(
    L = labels$L + centroid[1],
    T = labels$T + centroid[2],
    R = labels$R + centroid[3]
  )

  # grid L
  gridL =
    data.frame(
      scale = 'L',
      centroid = ifelse(breaks$L == centroid[1], TRUE, FALSE),
      L_from = breaks$L,
      T_from = 1-breaks$L,
      R_from = 0,
      L_to = breaks$L,
      T_to = 0,
      R_to = 1-breaks$L
    )
```

```
  # grid T
  gridT =
    data.frame(
      scale = 'T',
      centroid = ifelse(breaks$T == centroid[2], TRUE, FALSE),
      L_from = 0,
      T_from = breaks$T,
      R_from = 1-breaks$T,
      L_to = 1-breaks$T,
      T_to = breaks$T,
      R_to = 0
    )

  # grid R
  gridR =
    data.frame(
      scale = 'R',
      centroid = ifelse(breaks$R == centroid[3], TRUE, FALSE),
      L_from = 1-breaks$R,
      T_from = 0,
      R_from = breaks$R,
      L_to = 0,
      T_to = 1-breaks$R,
      R_to = breaks$R
    )

  # grid line coordinates of uncentered grid
  grid = rbind(gridL, gridT, gridR)

  # grid line coordinates of centered grid
  cgrid = data.frame(
    grid[,1:2],
    prop.table(t(t(grid[,3:5])*(1/centroid)), margin = 1),
    prop.table(t(t(grid[,6:8])*(1/centroid)), margin = 1)
  )

  # breaks of centered grid
  cbreaks = data.frame(L = cgrid[cgrid$scale == 'L', 'L_from'],
                       T = cgrid[cgrid$scale == 'T', 'T_from'],
                       R = cgrid[cgrid$scale == 'R', 'R_from'])

  list(grid = grid, cgrid = cgrid,
       breaks = breaks, cbreaks = cbreaks, labels = labels)
}

# Color Mixture ------------------------------------------------------------

# color mixture of 3 polar vectors
GetMixture <- function (P, h_, c_, l_, contrast, color_space) {

  # generate primary colours starting with a hue value in [0, 360) and then
  # picking two equidistant points on the circumference of the colour wheel.
  # input hue in degrees, all further calculations in radians.
```

```
  phi = (h_*0.0174 + c(0, 2.09, 4.19)) %% 6.28

  # calculate the chroma matrix C by scaling the row proportions
  # of the input matrix P by the maximum chroma parameter.
  C = P*c_

  # the complex matrix Z represents each case (i) and group (j=1,2,3) specific
  # color in complex polar form with hue as angle and chroma as radius.
  Z = matrix(complex(argument = phi, modulus = c(t(C))), ncol = 3, byrow = TRUE)

  # adding up the rows gives the CIE-Lab (cartesian) coordinates
  # of the convex color mixture in complex form.
  z = rowSums(Z)
  # convert the cartesian CIE-Lab coordinates to polar CIE-Luv coordinates
  # and add lightness level.
  M = cbind(h = (Arg(z)*57.3)%%360, c = Mod(z), l = l_)

  # boost lightness and chroma contrast of balanced to unbalanced mixtures
  cfactor = rescale(M[,2], from = c(0, c_), to = c(1-contrast, 1))
  M[,3] = cfactor*M[,3]
  M[,2] = cfactor*M[,2]

  # convert the complex representation of the color mixture to
  # hex-srgb representation via the hcl (CIE-Luv) color space
  # or the hsv (polar RGB) color space
  if (color_space == 'hcl') {
    # expects h = [0, 360], c = [0, 200], l = c[0, 100]
    hexsrgb = hcl(h = M[,1], c = M[,2], l = M[,3],
                  alpha = 1, fixup = TRUE)
  }
  if (color_space == 'hsv') {
    # expects h = [0, 1], c = s = [0, 1], l = v = c[0, 1]
    hexsrgb = hsv(h = M[,1]/360, s = M[,2]/200, v = M[,3]/100,
                  alpha = 1)
  }

  # (centered) compositions, hcl values of mixtures and hexsrgb code
  result = data.frame(P, M[,1], M[,2], M[,3], hexsrgb,
                      row.names = NULL, check.rows = FALSE,
                      check.names = FALSE, stringsAsFactors = FALSE)
  colnames(result) = c('p1', 'p2', 'p3', 'h', 'c', 'l', 'hexsrgb')
  return(result)
}

# Data ------------------------------------------------------------------

# color space parameters
h_ = 0; c_ = 130; l_ = 70; contrast = 0

# the center of the data
center = c(0.7, 0.1, 0.2)

# centroid coordinates of legend surface
```

```
C <- tricolore:::GetCentroids(100)
# centered centroid coordinates of legend surface
cC <- data.frame(id = C[,1],
                 prop.table(t(t(C[,2:4])*1/center), 1))
# vertex coordinates of legend surface
V <- tricolore:::GetVertices(C)
# centered vertex coordinates of legend surface
cV <- data.frame(id = V[,1],
                 vertex = V[,2],
                 prop.table(t(t(V[,3:5])*1/center), 1))
# grid lines, breaks and labels
grids <- TernaryCentroidGrid(center)
```

Non-balanced centered color scale

```
# Non-balanced ------------------------------------------------------------

# mixed colors and associated polar coordinates
colC <- GetMixture(C[,-1], h_ = h_, c_ = c_, l_ = l_,
                   contrast = contrast, color_space = 'hcl')
```

```
ggplot(colC, aes(x = h, y = c)) +
  geom_point(aes(color = hexsrgb), show.legend = FALSE) +
  coord_polar(start = -2*pi/3, direction=1) +
  scale_x_continuous(breaks=seq(0, 360, by = 30),
                     expand=c(0, 0), lim=c(0, 360)) +
  scale_color_identity() +
  theme_minimal()
```

```
ggtern(data.frame()) +
  geom_polygon(aes(x = p1, y = p2, z = p3, group = id,
                   fill = rep(colC$hexsrgb, 3), color = rep(colC$hexsrgb, 3)),
               data = as.data.frame(V), show.legend = FALSE) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to),
               color = 'white', show.legend = FALSE,
               data = subset(grids$cgrid, centroid == FALSE)) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to, group = scale),
               lwd = 1.5, color = 'white', show.legend = FALSE,
               data = subset(grids$cgrid, centroid == TRUE)) +
  scale_L_continuous(breaks = grids$cbreaks$L,
                     labels = round(grids$labels$L, 2)) +
  scale_T_continuous(breaks = grids$cbreaks$T,
                     labels = round(grids$labels$T, 2)) +
  scale_R_continuous(breaks = grids$cbreaks$R,
                     labels = round(grids$labels$R, 2)) +
  scale_fill_identity() +
  scale_color_identity() +
  labs(x = 'Pri.', y = 'Sec.', z = 'Ter.') +
```

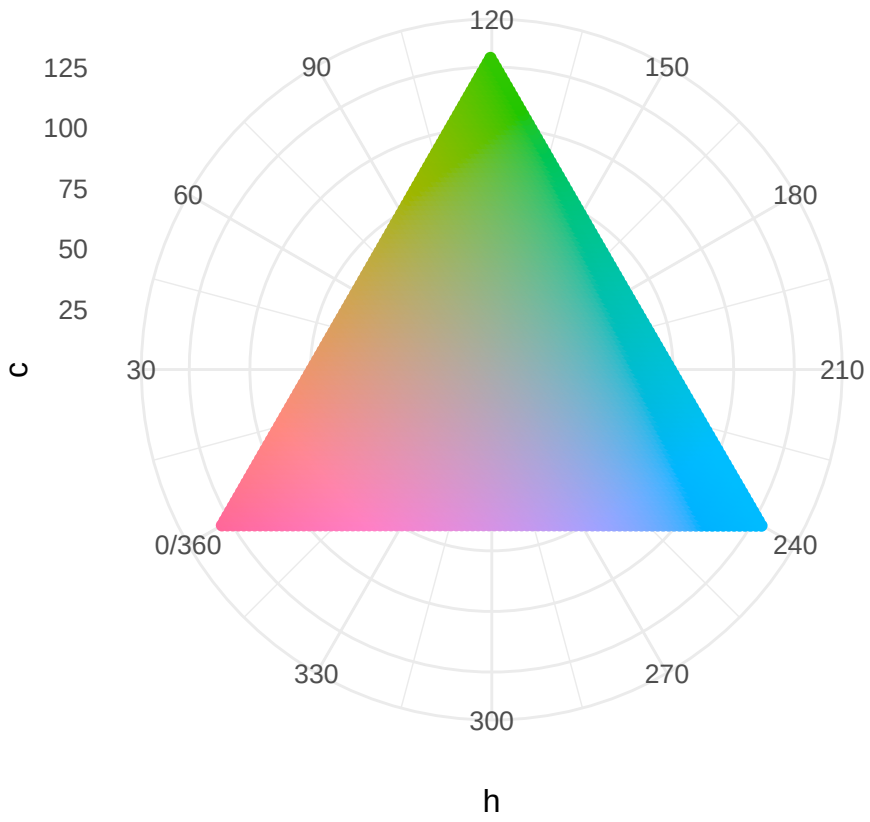

Figure 1: Sample space of the color mixtures for the non-balanced, centered color scale.

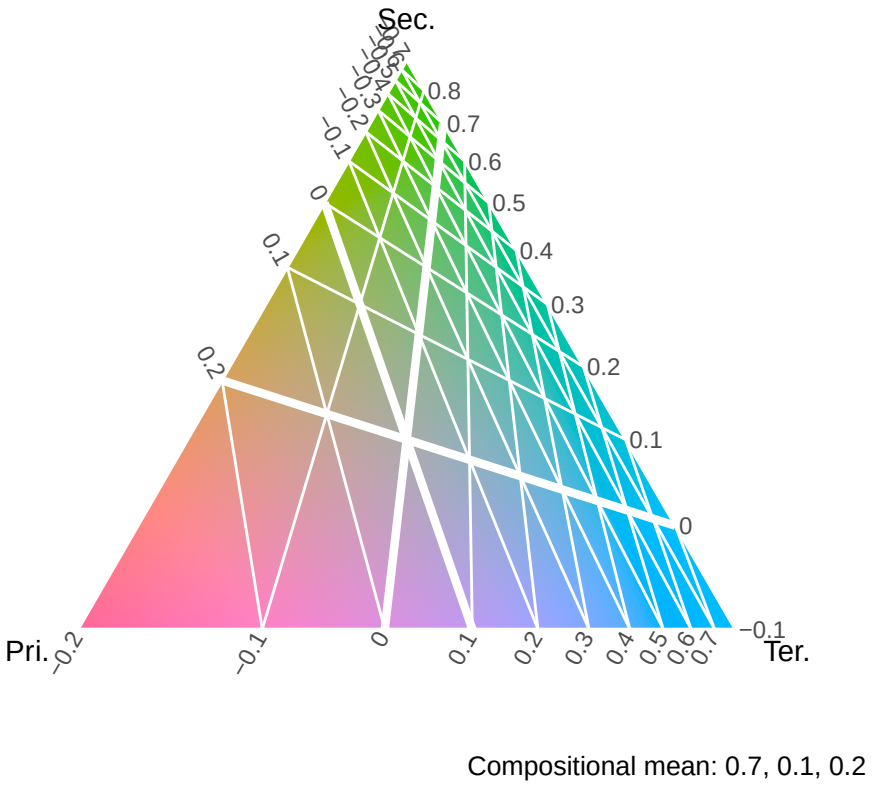

Figure 2: Color key for the non-balanced, centered color scale on a centered grid.

```
  labs(caption = paste0('Compositional mean: ',
                        paste0(round(center, 2), collapse = ', '))) +
  theme_noticks()
```

```
# mixed colors and associated polar coordinates
colcC <- GetMixture(cC[,-1], h_ = 0, c_ = 130, l_ = 70,
                    contrast = 0, color_space = 'hcl')
```

```
ggtern(data.frame()) +
  geom_polygon(aes(x = p1, y = p2, z = p3, group = id,
                   fill = rep(colcC$hexsrgb, 3), color = rep(colcC$hexsrgb, 3)),
               data = as.data.frame(V), show.legend = FALSE) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to),
               color = 'white', show.legend = FALSE,
               data = subset(grids$grid, centroid == FALSE)) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to, group = scale),
               lwd = 1.5, color = 'white', show.legend = FALSE,
               data = subset(grids$grid, centroid == TRUE)) +
  scale_L_continuous(breaks = grids$breaks$L,
                     labels = round(grids$labels$L, 2)) +
  scale_T_continuous(breaks = grids$breaks$T,
                     labels = round(grids$labels$T, 2)) +
```

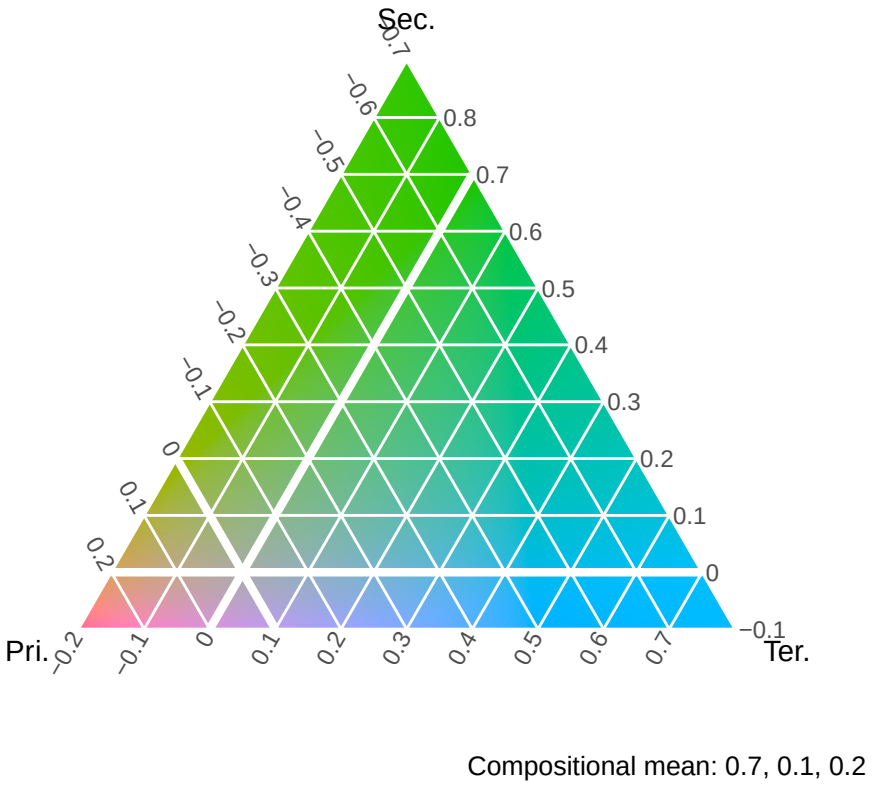

Figure 3: Color key for the non-balanced, centered color scale on a non-centered grid.

```
  scale_R_continuous(breaks = grids$breaks$R,
                     labels = round(grids$labels$R, 2)) +
  scale_fill_identity() +
  scale_color_identity() +
  labs(x = 'Pri.', y = 'Sec.', z = 'Ter.') +
  labs(caption = paste0('Compositional mean: ',
                        paste0(round(center, 2), collapse = ', '))) +
  theme_noticks()
```

Balanced centered color scale

```
# Balanced ---------------------------------------------------------------

# inverse center weighted centroids
wC <- t(t(C[,-1])*(1-center))
# mixed colors and associated polar coordinates
colwC <- GetMixture(wC, h_ = h_, c_ = c_, l_ = l_,
                    contrast = contrast, color_space = 'hcl')
```

```
ggplot(colwC, aes(x = h, y = c)) +
  geom_point(aes(color = hexsrgb), show.legend = FALSE) +
  coord_polar(start = -2*pi/3, direction=1) +
  scale_x_continuous(breaks=seq(0, 360, by = 30), expand=c(0, 0), lim=c(0, 360)) +
  scale_color_identity() +
```

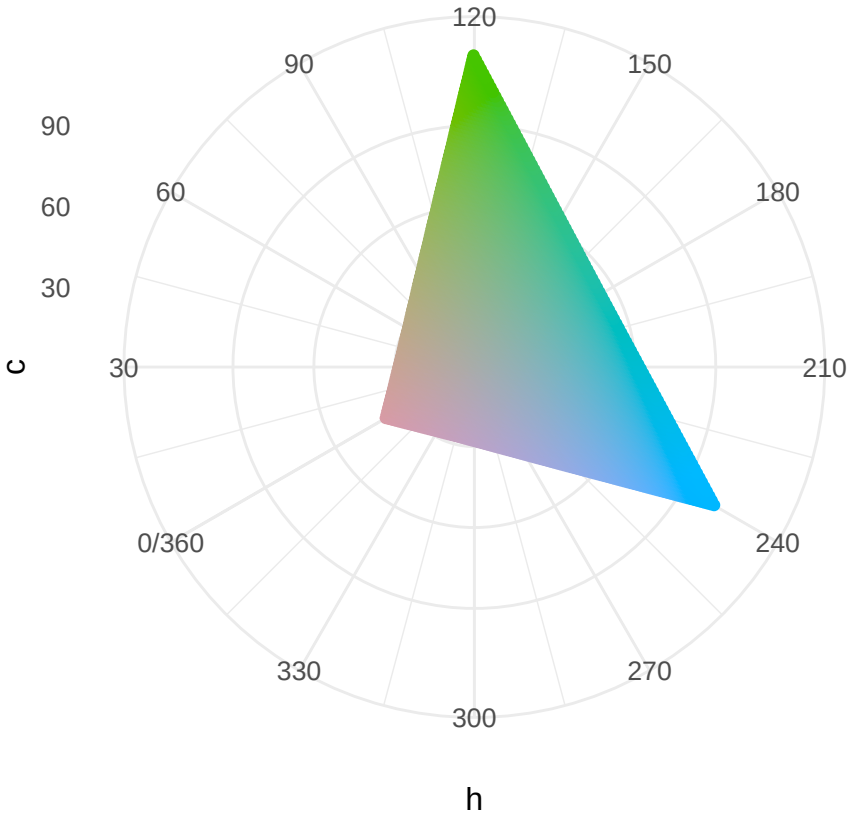

Figure 4: Sample space of the color mixtures for the balanced, centered color scale.

```
  theme_minimal()
```

```
ggtern(data.frame()) +
  geom_polygon(aes(x = p1, y = p2, z = p3, group = id,
                   fill = rep(colwC$hexsrgb, 3), color = rep(colwC$hexsrgb, 3)),
               data = cV, show.legend = FALSE) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to),
               color = 'white', show.legend = FALSE,
               data = subset(grids$cgrid, centroid == FALSE)) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to, group = scale),
               lwd = 1.5, color = 'white', show.legend = FALSE,
               data = subset(grids$cgrid, centroid == TRUE)) +
  scale_L_continuous(breaks = grids$cbreaks$L,
                     labels = round(grids$labels$L, 2)) +
  scale_T_continuous(breaks = grids$cbreaks$T,
                     labels = round(grids$labels$T, 2)) +
  scale_R_continuous(breaks = grids$cbreaks$R,
                     labels = round(grids$labels$R, 2)) +
  scale_fill_identity() +
  scale_color_identity() +
  labs(x = 'Pri.', y = 'Sec.', z = 'Ter.') +
```

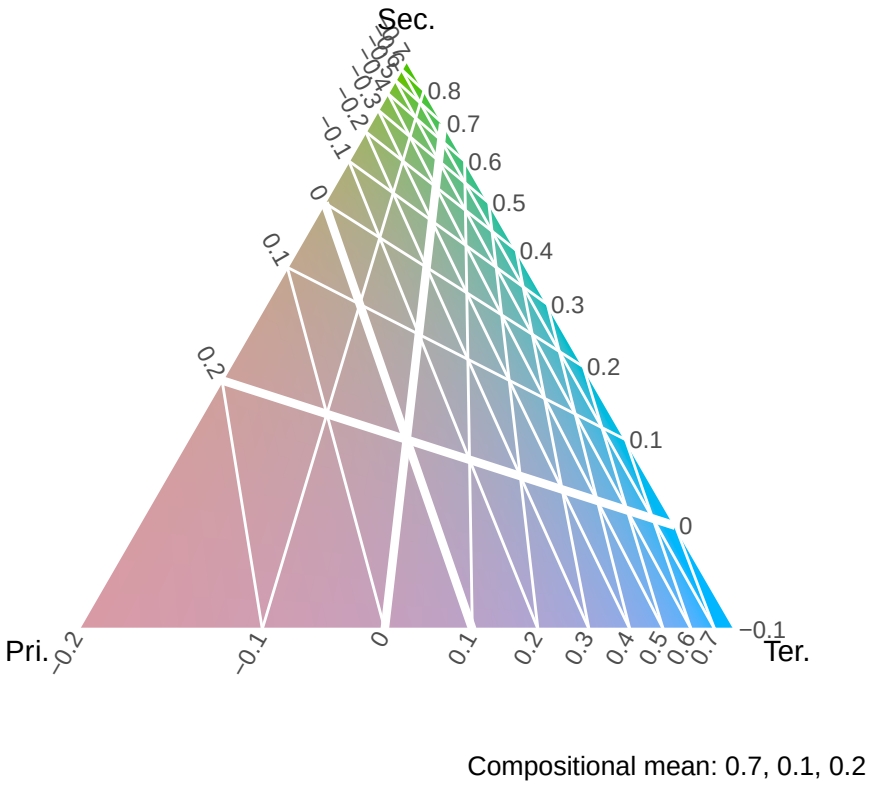

Figure 5: Color key for the balanced, centered color scale on a centered grid.

```
  labs(caption = paste0('Compositional mean: ',
                        paste0(round(center, 2), collapse = ', '))) +
  theme_noticks()
```

```
ggtern(data.frame()) +
  geom_polygon(aes(x = p1, y = p2, z = p3, group = id,
                   fill = rep(colwC$hexsrgb, 3), color = rep(colwC$hexsrgb, 3)),
               data = as.data.frame(V), show.legend = FALSE) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to),
               color = 'white', show.legend = FALSE,
               data = subset(grids$grid, centroid == FALSE)) +
  geom_segment(aes(x = L_from, xend = L_to,
                   y = T_from, yend = T_to,
                   z = R_from, zend = R_to, group = scale),
               lwd = 1.5, color = 'white', show.legend = FALSE,
               data = subset(grids$grid, centroid == TRUE)) +
  scale_L_continuous(breaks = grids$breaks$L,
                     labels = round(grids$labels$L, 2)) +
  scale_T_continuous(breaks = grids$breaks$T,
                     labels = round(grids$labels$T, 2)) +
  scale_R_continuous(breaks = grids$breaks$R,
                     labels = round(grids$labels$R, 2)) +
  scale_fill_identity() +
```

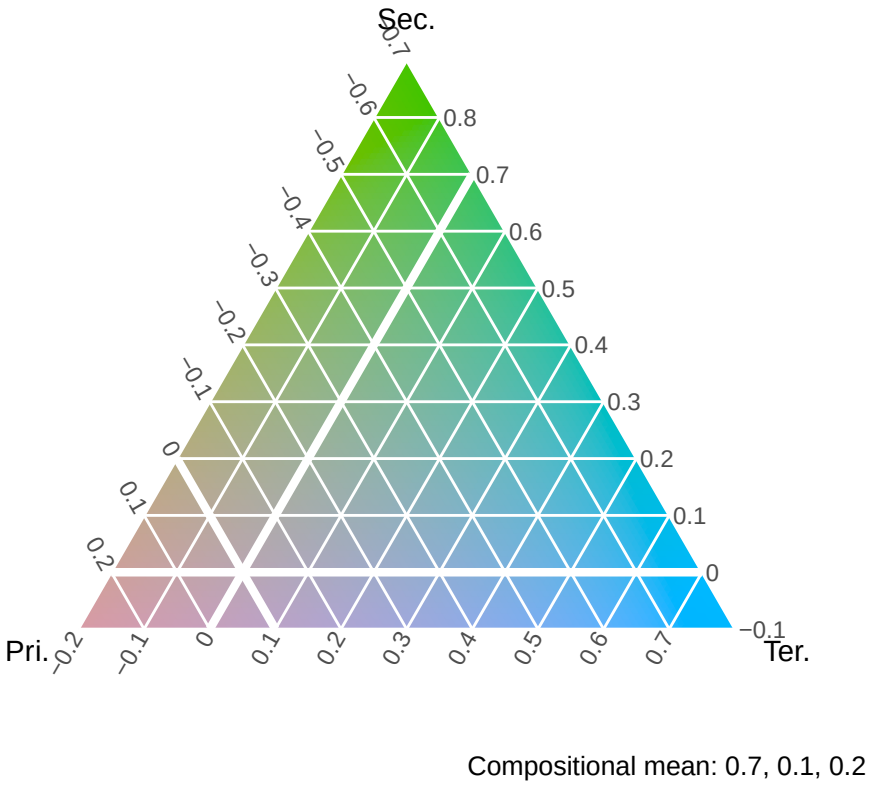

Figure 6: Color key for the balanced, centered color scale on a non-centered grid.

```
  scale_color_identity() +
  labs(x = 'Pri.', y = 'Sec.', z = 'Ter.') +
  labs(caption = paste0('Compositional mean: ',
                        paste0(round(center, 2), collapse = ', '))) +
  theme_noticks()
```