ladder_pL.R

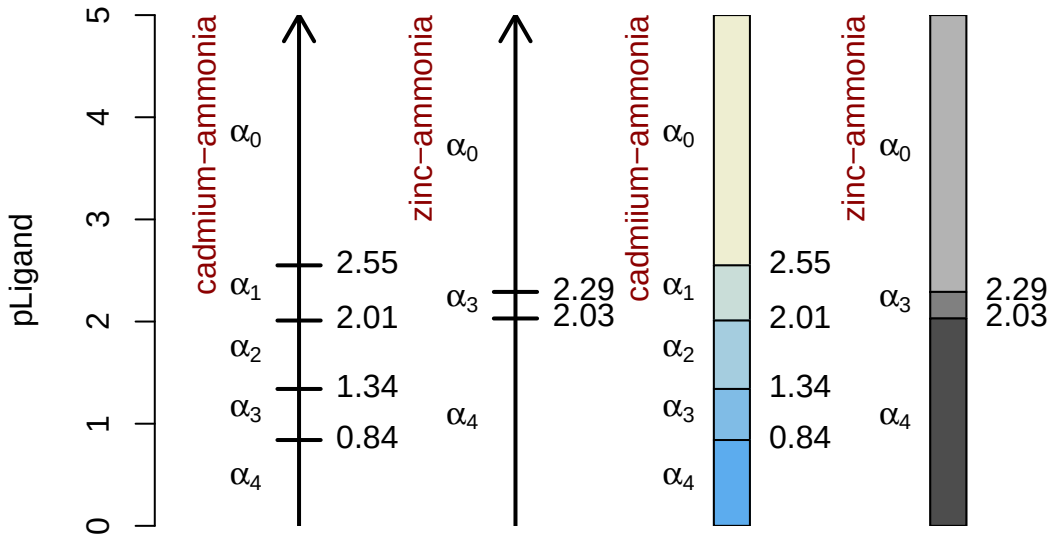
davidharvey

Sat Feb 27 13:49:37 2016

```
# function to plot ladder diagram for metal-ligand complexes
# pk_list: list of pK values, in order of fewest ligands to most
#           ligands; default values are for cadmium-ammine complexes
# ligands: list giving number of ligands added for each pK value;
#           defaults to vector of (1, 1, 1, 1) for the four stepwise
#           cadmium-amine complexes
# pL_axis: logical; defaults to FALSE but TRUE draws pLigand axis
# pL_limit: limits for pLigand axis; defaults to 0 to 14
# type: the type of ladder diagram; options are "arrow," which is the
#       default, or "strip"
# buffer: logical; defaults to FALSE, but TRUE will add buffer regions
# species: option to enter name of weak acid to add as title for plot;
#          defaults to NULL, which supresses title
# labels: option to enter vector of labels for legend; defaults to
#         NULL, which uses a default legend
# locate: x-axis location of arrow or center of strip; defaults to 2,
#         which is practical lower limit; increase in steps of three
#         will separate diagrams; practical upper limit is 12
# overlay: logical; defaults to FALSE, but setting to TRUE allows for
#          adding a new ladder diagram

library(shape)

ladder_pL = function(pk_list = c(2.55, 2.01, 1.34, 0.84),
                     ligands = c(1, 1, 1, 1),
                     pL_axis = FALSE,
                     pL_limit = c(0, 14),
                     type = "arrow",
                     shade = "color",
                     buffer = FALSE,
                     species = NULL,
                     labels = NULL,
                     locate = 2,
                     overlay = FALSE){

  # initial set-up; creates vector of limits for adding labels;
  # creates counter, n, for the number of alpha values;
  # sets colors for strip version of ladder diagram

  pks = pk_list/ligands
  n = length(pks)
  limits = c(pL_limit[2], pks, pL_limit[1])
  if (shade == "color") {
  col.func = colorRampPalette(c("lightyellow2", "steelblue2"))
  colors = col.func(n + 1)
  } else {
    col.func = colorRampPalette(c("gray70", "gray30"))
```

```
  colors = col.func(n + 1)
}

# creates default set of alpha labels if labels are not provided

if (is.null(labels) == TRUE) {
  labels = rep(0, n + 1)
  labels[1] = expression(alpha[0])
  num.ligands = 0
  for (i in 1:(n)) {
    num.ligands = num.ligands + ligands[i]
    labels[i + 1] = eval(substitute(expression(alpha[I]),
                                list(I = num.ligands)))
  }
}

# routines for plotting the ladder diagrams for each possible set
# of options: new or overlay; arrow or strip; with or without
# pH axis, and with or without buffer regions

if (overlay == FALSE) {if (pL_axis == FALSE) {
                            pLax = "n"
                            pLlabel = "pLigand"
                            pLaxis = ""
                           } else {
                             pLax = "s"
                             pLlabel = ""
                             pLaxis = "pLigand"
                           }
  plot(NULL, xlim = c(0,14), ylim = c(pL_limit[1],pL_limit[2]),
                             type = "n", xaxt = "n", yaxt = pLax,
                             bty = "n", xlab = "", ylab = pLaxis,
                             xaxs = "i", yaxs = "i")
  text(locate + 0.25, pL_limit[2] - (pL_limit[2] - pL_limit[1])/25,
       pLlabel, pos = 4)
  }

if (type == "arrow") {
    Arrows(locate, pL_limit[1], locate, pL_limit[2], lwd = 2,
           arr.type = "simple")
    segments(x0 = rep(locate - 0.3, n), y0 = pks,
             x1 = rep(locate + 0.3, n), y1 = pks, lwd = 2)
} else if (type == "strip") {
    for (i in 1:(n + 1)) {
      filledrectangle(mid = c(locate, (limits[i] + limits[i + 1])/2),
                      wx = 0.5, wy = limits[i + 1] - limits[i],
                      col = colors[i], lcol = "black")
    }
} else {
    return(paste(type, " is not an option.", sep = ""))
}
  for (i in 1:n) {
    text(x = locate + 0.25, y = pks[i],
```

```
          labels = pks[i], pos = 4)
    }
    for (i in 1:(n + 1)){
      text(x = locate - 0.25, y = (limits[i + 1] + limits[i])/2,
           labels[i], pos = 2)
    }
    if (buffer == TRUE) {
      if (n == 1) {
        segments(x0 = locate, y0 = pks - 1/ligands, x1 = locate,
                 y1 = pks + 1/ligands, lwd = 5, lend = "butt")
      } else { for (i in 1:n) {
        if (i %% 2 == 0){
          segments(x0 = locate + 0.05, y0 = pks[i] - 1/ligands[i],
                   x1 = locate + 0.05, y1 = pks[i] + 1/ligands[i],
                   lwd = 5, lend = "butt")
        } else {
          segments(x0 = locate - 0.05, y0 = pks[i] - 1/ligands[i],
                   x1 = locate - 0.05, y1 = pks[i] + 1/ligands[i],
                   lwd = 5, lend = "butt")
        }
      }
      }
    }
  if (is.null(species) == FALSE) {
    text(x = locate - 1, y = pL_limit[2], species, pos = 2,
         srt = 90, col = "darkred")
  }
}

# code to test

ladder_pL(pL_axis = TRUE, type = "arrow", species = "cadmium-ammonia",
          locate = 2, pL_limit = c(0, 5))
ladder_pL(type = "arrow", locate = 5, overlay = TRUE,
          pk_list = c(6.87, 2.03), ligands = c(3, 1),
          species = "zinc-ammonia", pL_limit = c(0, 5))
ladder_pL(type = "strip", species = "cadmiium-ammonia", locate = 8,
          pL_limit = c(0, 5), shade = "color", overlay = TRUE)
ladder_pL(type = "strip", locate = 11, overlay = TRUE,
          pL_limit = c(0, 5), pk_list = c(6.87, 2.03), ligands = c(3, 1),
          species = "zinc-ammonia", shade = "gray")
```


pLigand
0
1
2
3
4
5
cadmium−ammonia
α0
α1
α2
α3
α4
2.55
2.01
1.34
0.84
zinc−ammonia
α0
α3
α4
2.29
2.03
cadmiium−ammonia
α0
α1
α2
α3
α4
2.55
2.01
1.34
0.84
zinc−ammonia
α0
α3
α4
2.29
2.03