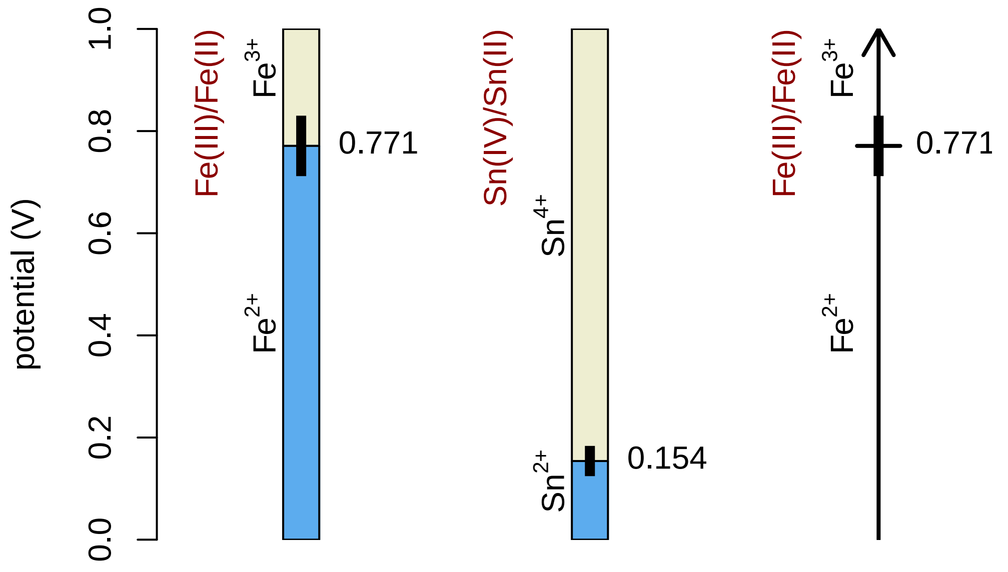
ladder_pot.R

davidharvey

Sat Feb 27 13:48:53 2016

```r
# function to plot ladder diagram for redox couples
# pot_list: list of standard state reduction potential, in order
#           from most negative to most positive; default values are for
#           Sn4+/Sn2+ redox couple
# electrons: list giving number of electrons for each standard state
#             potential; defaults to single value of 2 for the
#             Sn4+/Sn2+ redox couple
# pot_axis: logical; defaults to FALSE but TRUE draws potential axis
# pot_limit: limits for potential axis; defaults to -2 to 2
# 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 redox couple to add as title for
#          plot; defaults to NULL, which supresses  title
# labels: vector of labels for legend; defaults to labels
#         for Sn4+/Sn2+ redox couple
# 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_pot = function(pot_list = c(0.154),
                      electrons = c(2),
                      pot_axis = FALSE,
                      pot_limit = c(-2, 2),
                      type = "arrow",
                      shade = "color",
                      buffer = FALSE,
                      species = NULL,
                      labels = c(expression("Sn"^"4+"),
                                 expression("Sn"^"2+")),
                      locate = 2,
                      overlay = FALSE){

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

  pots = pot_list
  n = length(pots)
  limits = c(pot_limit[2], pots, pot_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.electrons = 0
#   for (i in 1:(n)) {
#     num.electrons = num.electrons + electrons[i]
#     labels[i + 1] = eval(substitute(expression(alpha[I]),
#                                 list(I = num.electrons)))
#   }
# }

# 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 (pot_axis == FALSE) {
                           potax = "n"
                           potlabel = "E (V)"
                           potaxis = ""
                         } else {
                            potax = "s"
                            potlabel = ""
                            potaxis = "potential (V)"
                         }
  plot(NULL, xlim = c(0,14), ylim = c(pot_limit[1],pot_limit[2]),
                            type = "n", xaxt = "n", yaxt = potax,
                            bty = "n", xlab = "", ylab = potaxis,
                            xaxs = "i", yaxs = "i")
  text(locate + 0.25, pot_limit[2] - (pot_limit[2] - pot_limit[1])/25,
       potlabel, pos = 4)
  }

if (type == "arrow") {
    Arrows(locate, pot_limit[1], locate, pot_limit[2], lwd = 2,
           arr.type = "simple")
    segments(x0 = rep(locate - 0.3, n), y0 = pots,
             x1 = rep(locate + 0.3, n), y1 = pots, 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 = ""))
}
```

```r
    for (i in 1:n) {
      text(x = locate + 0.25, y = pots[i],
           labels = pots[i], pos = 4)
    }
    for (i in 1:(n + 1)){
      text(x = locate - 0.35, y = (limits[i + 1] + limits[i])/2,
           labels[i], srt = 90, pos = 3)
    }
    if (buffer == TRUE) {
      if (n == 1) {
        segments(x0 = locate, y0 = pots - 0.05916/electrons, x1 = locate,
                 y1 = pots + 0.05916/electrons, lwd = 5, lend = "butt")
      } else { for (i in 1:n) {
        if (i %% 2 == 0){
          segments(x0 = locate + 0.05, y0 = pots[i] - 0.05916/electrons[i],
                   x1 = locate + 0.05, y1 = pots[i] + 0.05916/electrons[i],
                   lwd = 5, lend = "butt")
        } else {
          segments(x0 = locate - 0.05, y0 = pots[i] - 0.05916/electrons[i],
                   x1 = locate - 0.05, y1 = pots[i] + 0.05916/electrons[i],
                   lwd = 5, lend = "butt")
        }
      }
      }
    }
  if (is.null(species) == FALSE) {
    text(x = locate - 1, y = pot_limit[2], species, pos = 2,
         srt = 90, col = "darkred")
  }
}

# code to test

ladder_pot(type = "strip",
           pot_limit = c(0,1),
           labels = c(expression("Fe"^"3+"),
                      expression("Fe"^"2+")),
           buffer = TRUE,
           pot_list = 0.771, electrons = 1,
           species = "Fe(III)/Fe(II)", pot_axis = TRUE)
ladder_pot(type = "strip", pot_limit = c(0,1), overlay = TRUE,
           labels = c(expression("Sn"^"4+"), expression("Sn"^"2+")),
           pot_list = 0.154, electrons = 2, buffer = TRUE, locate = 6,
           species = "Sn(IV)/Sn(II)")
ladder_pot(type = "arrow",
           pot_limit = c(0,1),
           labels = c(expression("Fe"^"3+"),
                      expression("Fe"^"2+")),
           buffer = TRUE,
           pot_list = 0.771, electrons = 1,
           species = "Fe(III)/Fe(II)", overlay = TRUE, locate = 10)
```



potential (V)
0.0
0.2
0.4
0.6
0.8
1.0
Fe(III)/Fe(II)
Fe^{3+}
Fe^{2+}
0.771
Sn(IV)/Sn(II)
Sn^{4+}
Sn^{2+}
0.154
Fe(III)/Fe(II)
Fe^{3+}
Fe^{2+}
0.771