grDevices

kejincai

9/10/2021

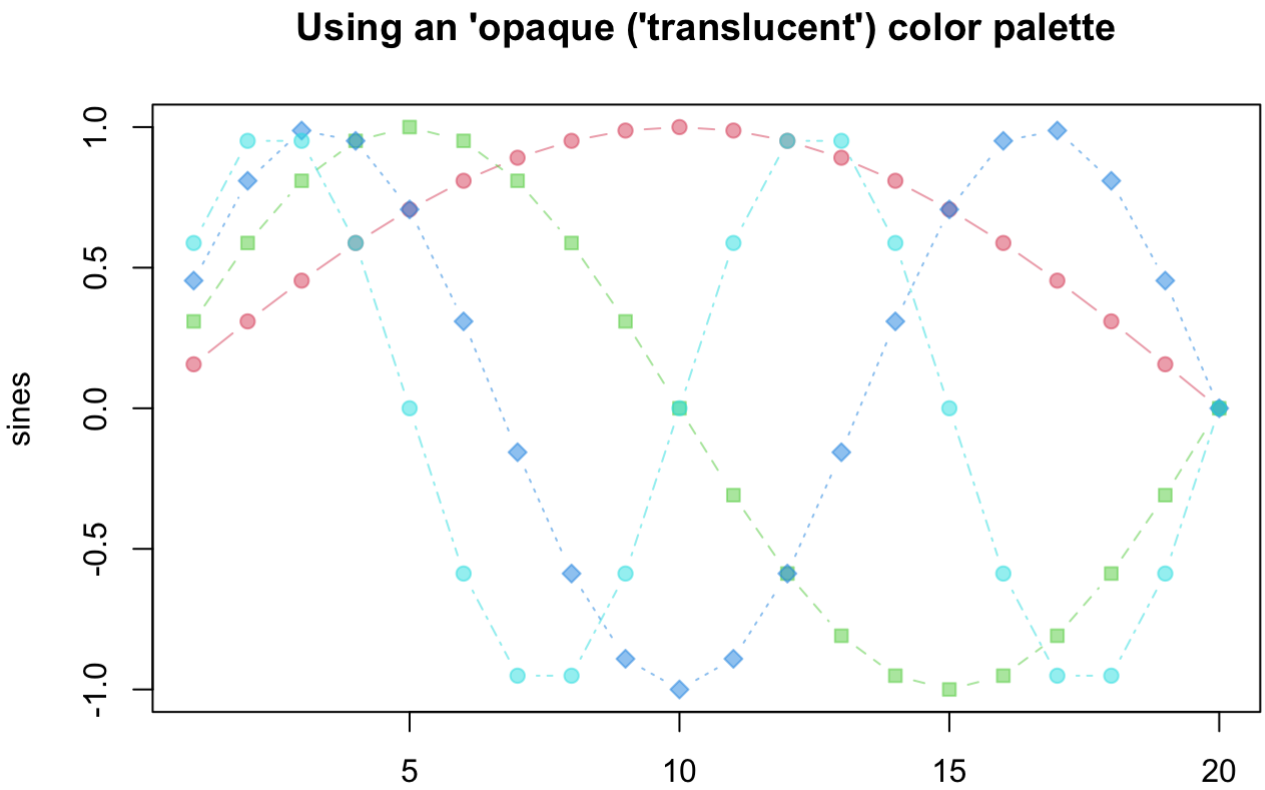
adjustcolor

Adjust or modify a vector of colors by “turning knobs” on one or more coordinates in (r,g,b,α) space, typically by up or down scaling them.

```
## Illustrative examples :
opal <- palette("default")
stopifnot(identical(adjustcolor(1:8,       0.75),
                    adjustcolor(palette(), 0.75)))
cbind(palette(), adjustcolor(1:8, 0.75))
```

```
##      [,1]      [,2]
## [1,] "black"   "#000000BF"
## [2,] "#DF536B" "#DF536BBF"
## [3,] "#61D04F" "#61D04FBF"
## [4,] "#2297E6" "#2297E6BF"
## [5,] "#28E2E5" "#28E2E5BF"
## [6,] "#CD0BBC" "#CD0BBCBF"
## [7,] "#F5C710" "#F5C710BF"
## [8,] "gray62"  "#9E9E9EBF"
```

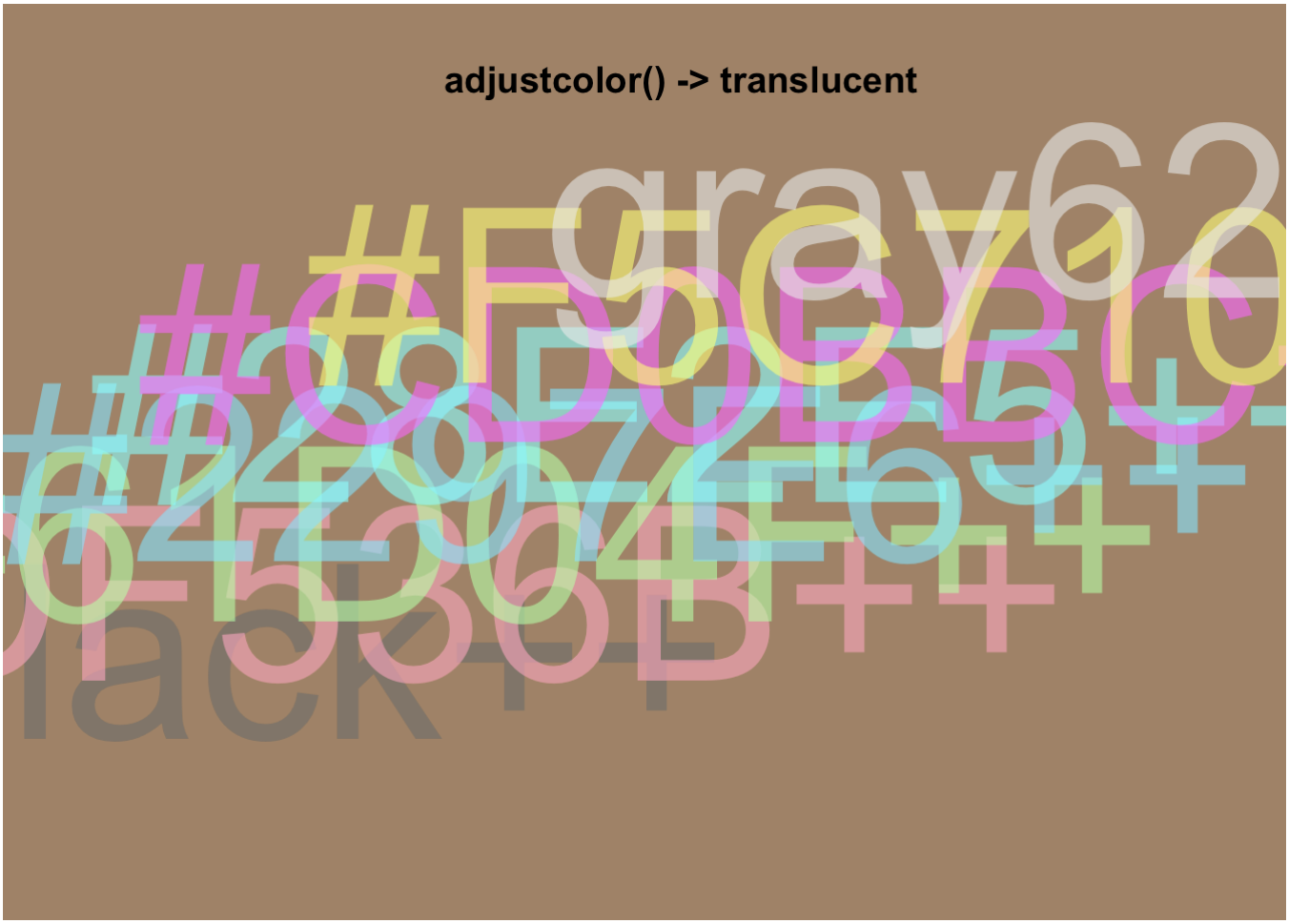
```
##  alpha = 1/2 * previous alpha --> opaque colors
x <- palette(adjustcolor(palette(), 0.5))

sines <- outer(1:20, 1:4, function(x, y) sin(x / 20 * pi * y))
matplot(sines, type = "b", pch = 21:23, col = 2:5, bg = 2:5,
        main = "Using an 'opaque ('translucent') color palette")
```




```
x. <- adjustcolor(x, offset = c(0.5, 0.5, 0.5, 0), # <- "more white"
                  transform = diag(c(.7, .7, .7, 0.6)))
cbind(x, x.)
```

```
##      x         x.
## [1,] "black"   "#80808099"
## [2,] "#DF536B" "#FFBACA99"
## [3,] "#61D04F" "#C3FFB799"
## [4,] "#2297E6" "#97E9FF99"
## [5,] "#28E2E5" "#9BFFFF99"
## [6,] "#CD0BBC" "#FF87FF99"
## [7,] "#F5C710" "#FFFF8B99"
## [8,] "gray62"  "#EEEEEE99"
```

```
op <- par(bg = adjustcolor("goldenrod", offset = -rep(.4, 4)), xpd = NA)
plot(0:9, 0:9, type = "n", axes = FALSE, xlab = "", ylab = "",
     main = "adjustcolor() -> translucent")
text(1:8, labels = paste0(x,"++"), col = x., cex = 8)
```




```
par(op)
```

axisTicks

Compute pretty axis scales and tick mark locations, the same way as traditional R graphics do it. This is interesting particularly for log scale axes.

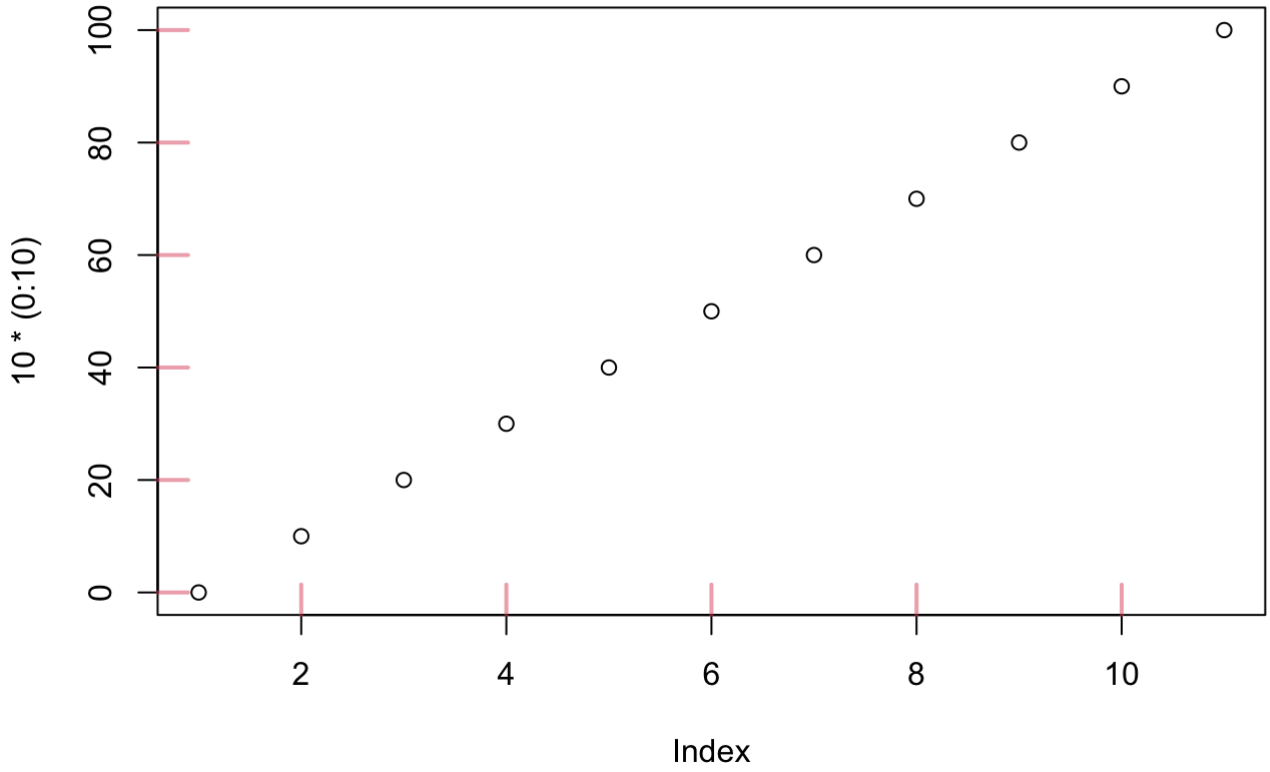
```
require("graphics")
plot(10*(0:10)); (pu <- par("usr"))
```

```
## [1]   0.6  11.4  -4.0 104.0
```

```
aX <- function(side, at, ...)
    axis(side, at = at, labels = FALSE, lwd.ticks = 2, col.ticks = 2,
         tck = 0.05, ...)
aX(1, print(xa <- axisTicks(pu[1:2], log = FALSE)))  # x axis
```

```
## [1]  2  4  6  8 10
```

```
aX(2, print(ya <- axisTicks(pu[3:4], log = FALSE)))  # y axis
```




```
## [1]   0  20  40  60  80 100
```

```
axisTicks(pu[3:4], log = FALSE, nint = 10)
```

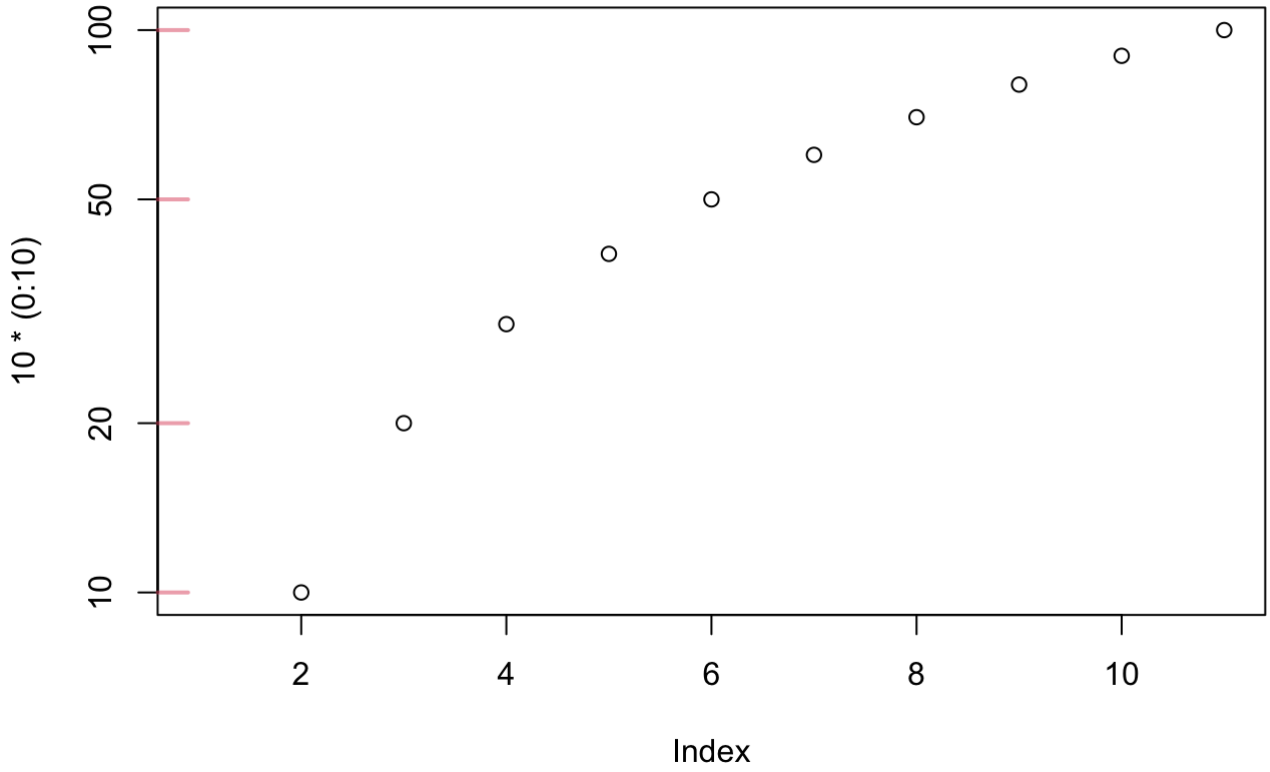
```
##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

```
plot(10*(0:10), log = "y"); (pu <- par("usr"))
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 y value <= 0 omitted from
## logarithmic plot
```

```
## [1]  0.60 11.40  0.96  2.04
```

```
aX(2, print(ya <- axisTicks(pu[3:4], log = TRUE)))  # y axis
```

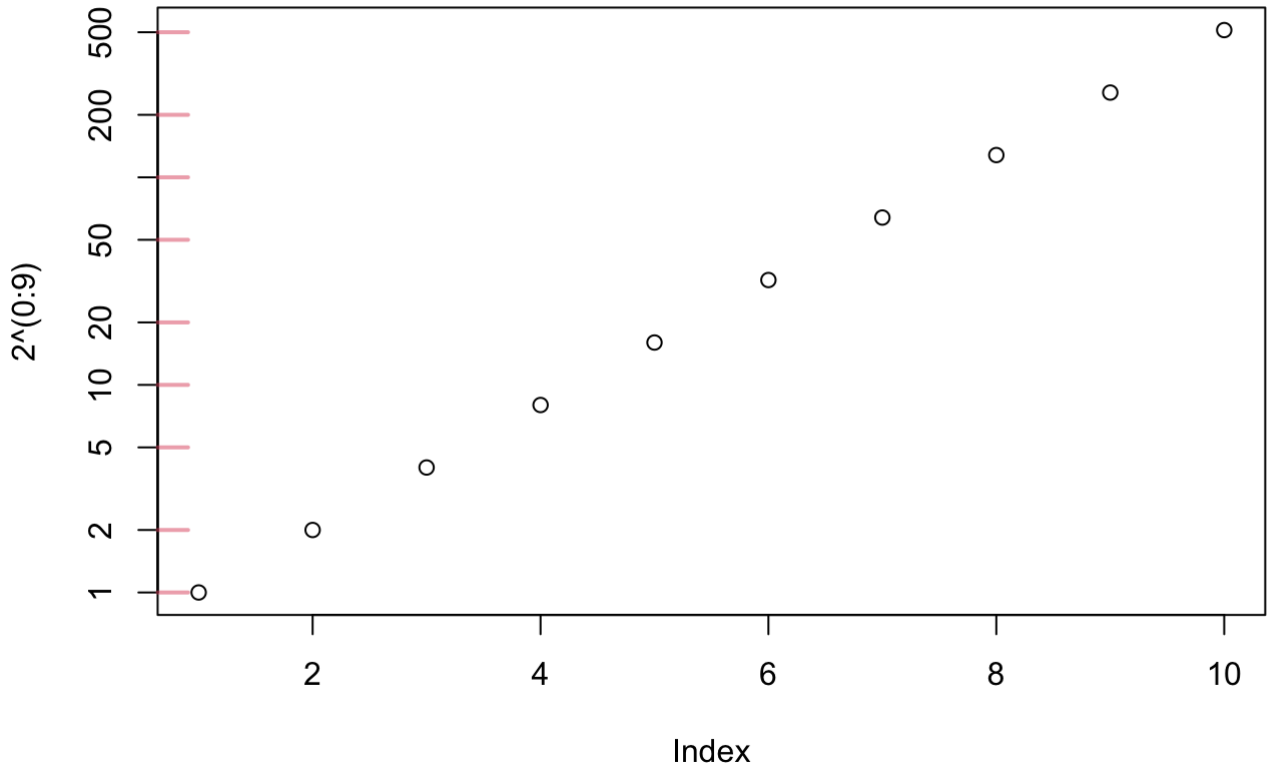



```
## [1]  10  20  50 100
```

```
plot(2^(0:9), log = "y"); (pu <- par("usr"))
```

```
## [1]  0.6400000 10.3600000 -0.1083708  2.8176408
```

```
aX(2, print(ya <- axisTicks(pu[3:4], log = TRUE)))  # y axis
```

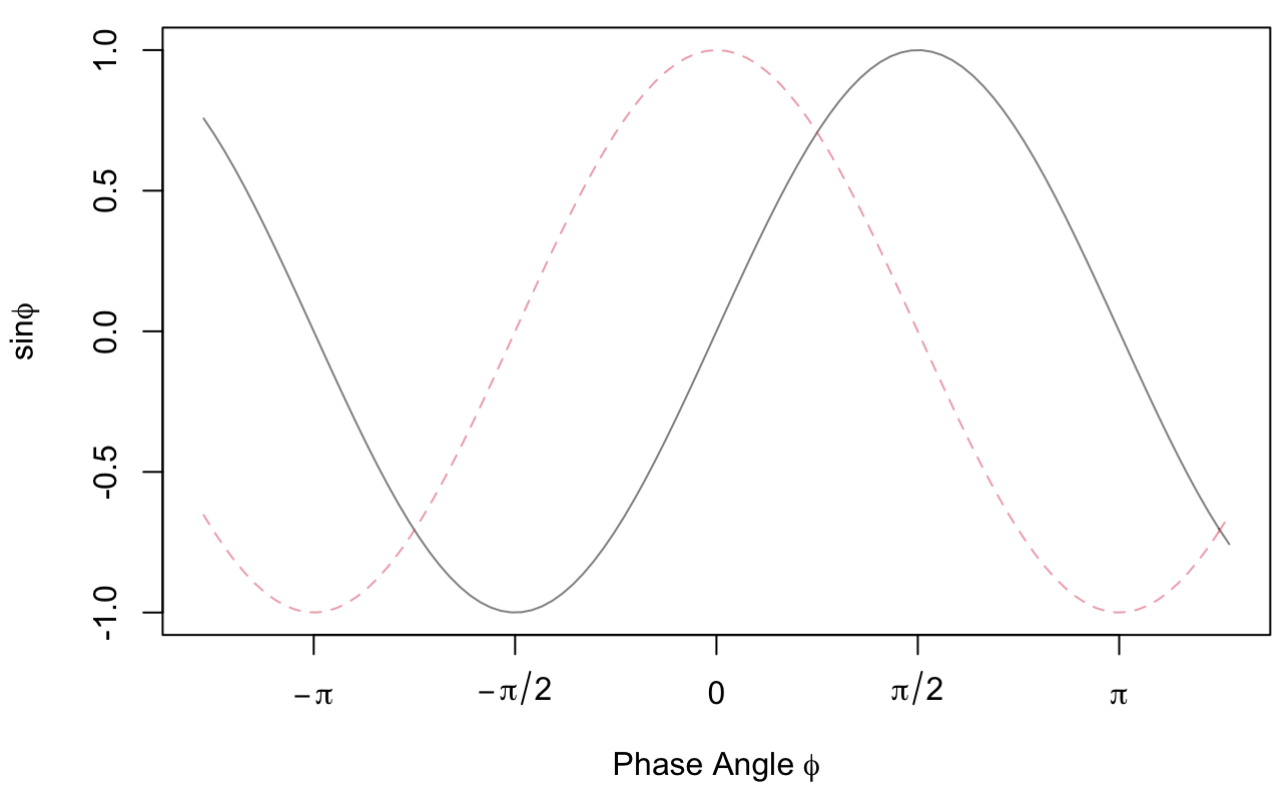



```
## [1]   1   2   5  10  20  50 100 200 500
```

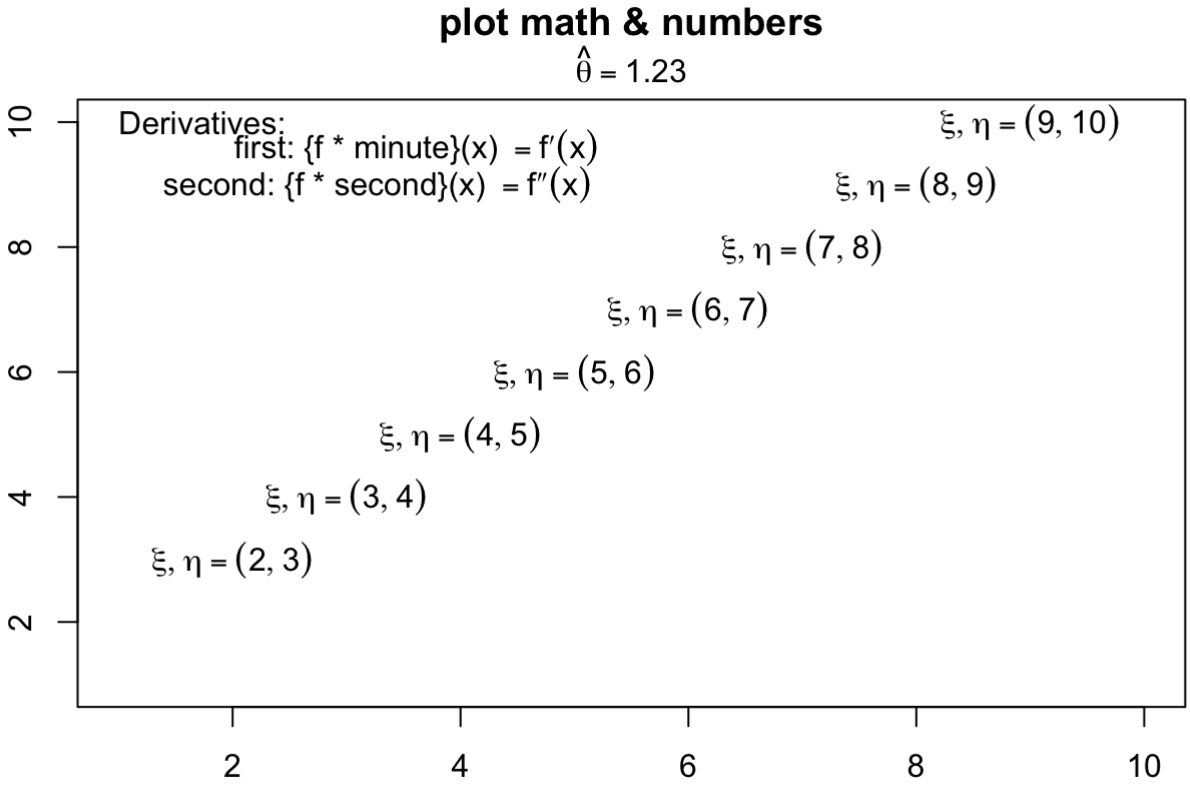
plotmath

If the text argument to one of the text-drawing functions (text, mtext, axis, legend) in R is an expression, the argument is interpreted as a mathematical expression and the output will be formatted according to TeX-like rules. Expressions can also be used for titles, subtitles and x- and y-axis labels (but not for axis labels on persp plots).

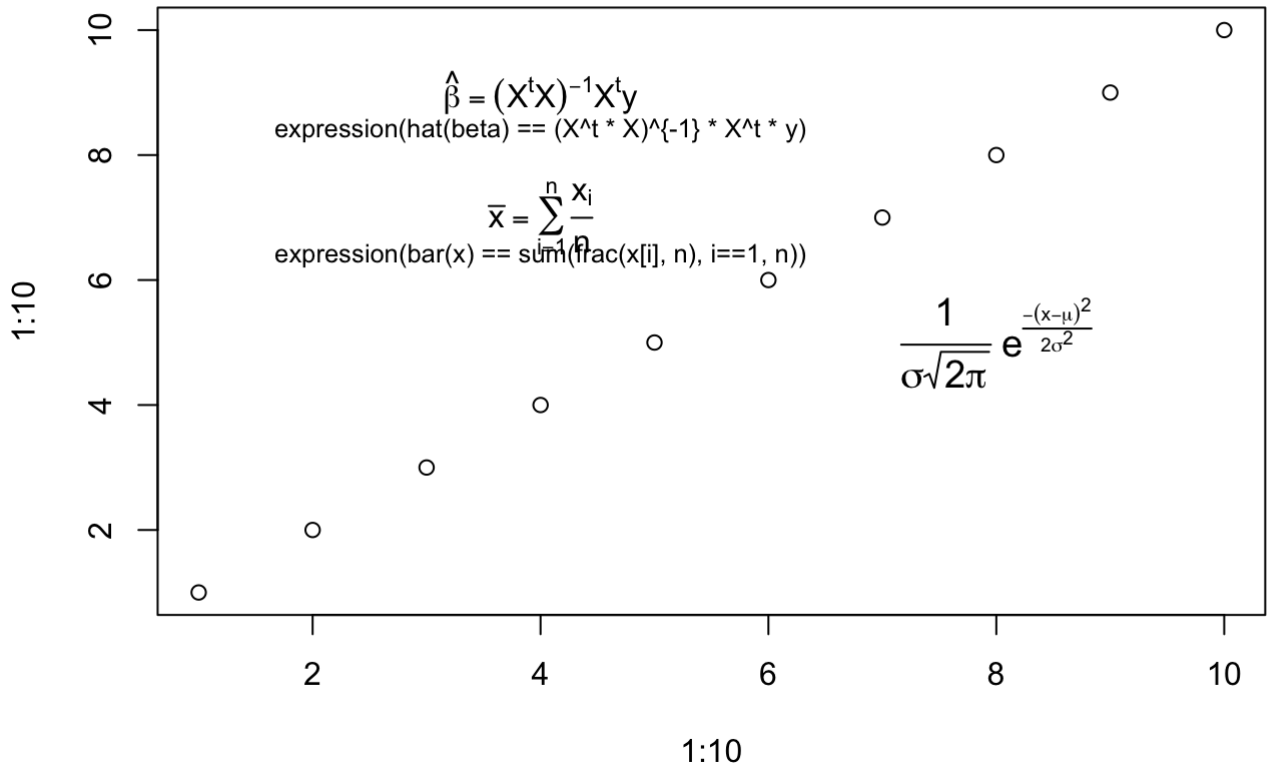
```
require(graphics)

x <- seq(-4, 4, length.out = 101)
y <- cbind(sin(x), cos(x))
matplot(x, y, type = "l", xaxt = "n",
        main = expression(paste(plain(sin) * phi, "  and  ",
                                plain(cos) * phi)),
        ylab = expression("sin" * phi, "cos" * phi), # only 1st is taken
        xlab = expression(paste("Phase Angle ", phi)),
        col.main = "blue")
axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
     labels = expression(-pi, -pi/2, 0, pi/2, pi))
```




```
## How to combine "math" and numeric variables :
plot(1:10, type="n", xlab="", ylab="", main = "plot math & numbers")
theta <- 1.23 ; mtext(bquote(hat(theta) == .(theta)), line= .25)
for(i in 2:9)
    text(i, i+1, substitute(list(xi, eta) == group("(",list(x,y),")"),
                            list(x = i, y = i+1)))
## note that both of these use calls rather than expressions.
##
text(1, 10,  "Derivatives:", adj = 0)
text(1, 9.6, expression(
 "             first: {f * minute}(x) " == {f * minute}(x)), adj = 0)
text(1, 9.0, expression(
 "     second: {f * second}(x) "        == {f * second}(x)), adj = 0)
```




```
plot(1:10, 1:10)
text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
text(4, 8.4, "expression(hat(beta) == (X^t * X)^{-1} * X^t * y)",
     cex = .8)
text(4, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
text(4, 6.4, "expression(bar(x) == sum(frac(x[i], n), i==1, n))",
     cex = .8)
text(8, 5, expression(paste(frac(1, sigma*sqrt(2*pi)), " ",
                            plain(e)^{frac(-(x-mu)^2, 2*sigma^2)})),
     cex = 1.2)
```



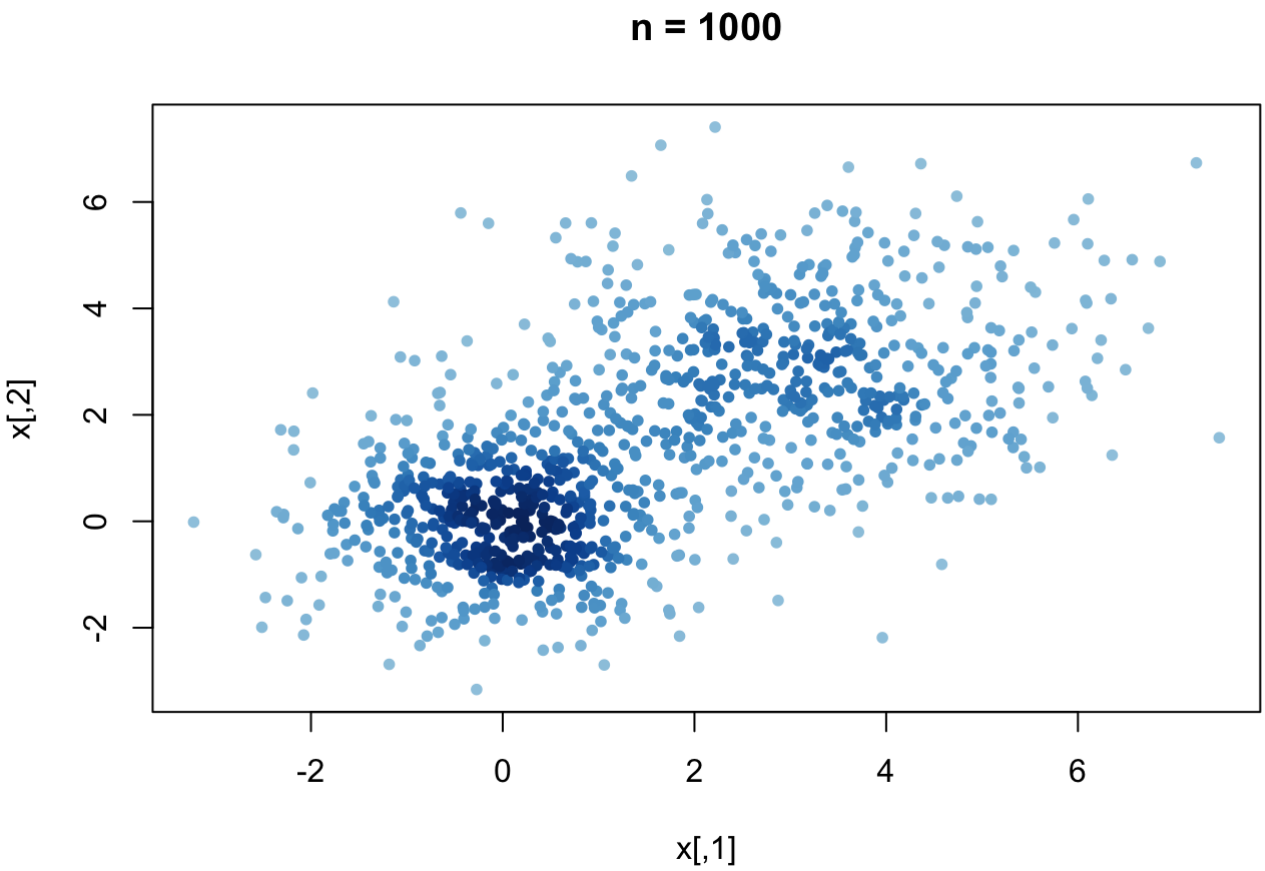

```
## some other useful symbols
plot.new(); plot.window(c(0,4), c(15,1))
text(1, 1, "universal", adj = 0); text(2.5, 1,  "\\042")
text(3, 1, expression(symbol("\042")))
text(1, 2, "existential", adj = 0); text(2.5, 2,  "\\044")
text(3, 2, expression(symbol("\044")))
text(1, 3, "suchthat", adj = 0); text(2.5, 3,  "\\047")
text(3, 3, expression(symbol("\047")))
text(1, 4, "therefore", adj = 0); text(2.5, 4,  "\\134")
text(3, 4, expression(symbol("\134")))
text(1, 5, "perpendicular", adj = 0); text(2.5, 5,  "\\136")
text(3, 5, expression(symbol("\136")))
text(1, 6, "circlemultiply", adj = 0); text(2.5, 6,  "\\304")
text(3, 6, expression(symbol("\304")))
text(1, 7, "circleplus", adj = 0); text(2.5, 7,  "\\305")
text(3, 7, expression(symbol("\305")))
text(1, 8, "emptyset", adj = 0); text(2.5, 8,  "\\306")
text(3, 8, expression(symbol("\306")))
text(1, 9, "angle", adj = 0); text(2.5, 9,  "\\320")
text(3, 9, expression(symbol("\320")))
text(1, 10, "leftangle", adj = 0); text(2.5, 10,  "\\341")
text(3, 10, expression(symbol("\341")))
text(1, 11, "rightangle", adj = 0); text(2.5, 11,  "\\361")
text(3, 11, expression(symbol("\361")))
```

universal	\042	∀
existential	\044	∃
suchthat	\047	∍
therefore	\134	∴
perpendicular	\136	⊥
circlemultiply	\304	⊗
circleplus	\305	⊕
emptyset	\306	∅
angle	\320	∠
leftangle	\341	⟨
rightangle	\361	⟩

densCols

densCols produces a vector containing colors which encode the local densities at each point in a scatterplot.

```
x1  <- matrix(rnorm(1e3), ncol = 2)
x2  <- matrix(rnorm(1e3, mean = 3, sd = 1.5), ncol = 2)
x   <- rbind(x1, x2)

dcols <- densCols(x)
graphics::plot(x, col = dcols, pch = 20, main = "n = 1000")
```



colors

Returns the built-in color names which R knows about.

```
col <- colors()
head(col, 50)
```

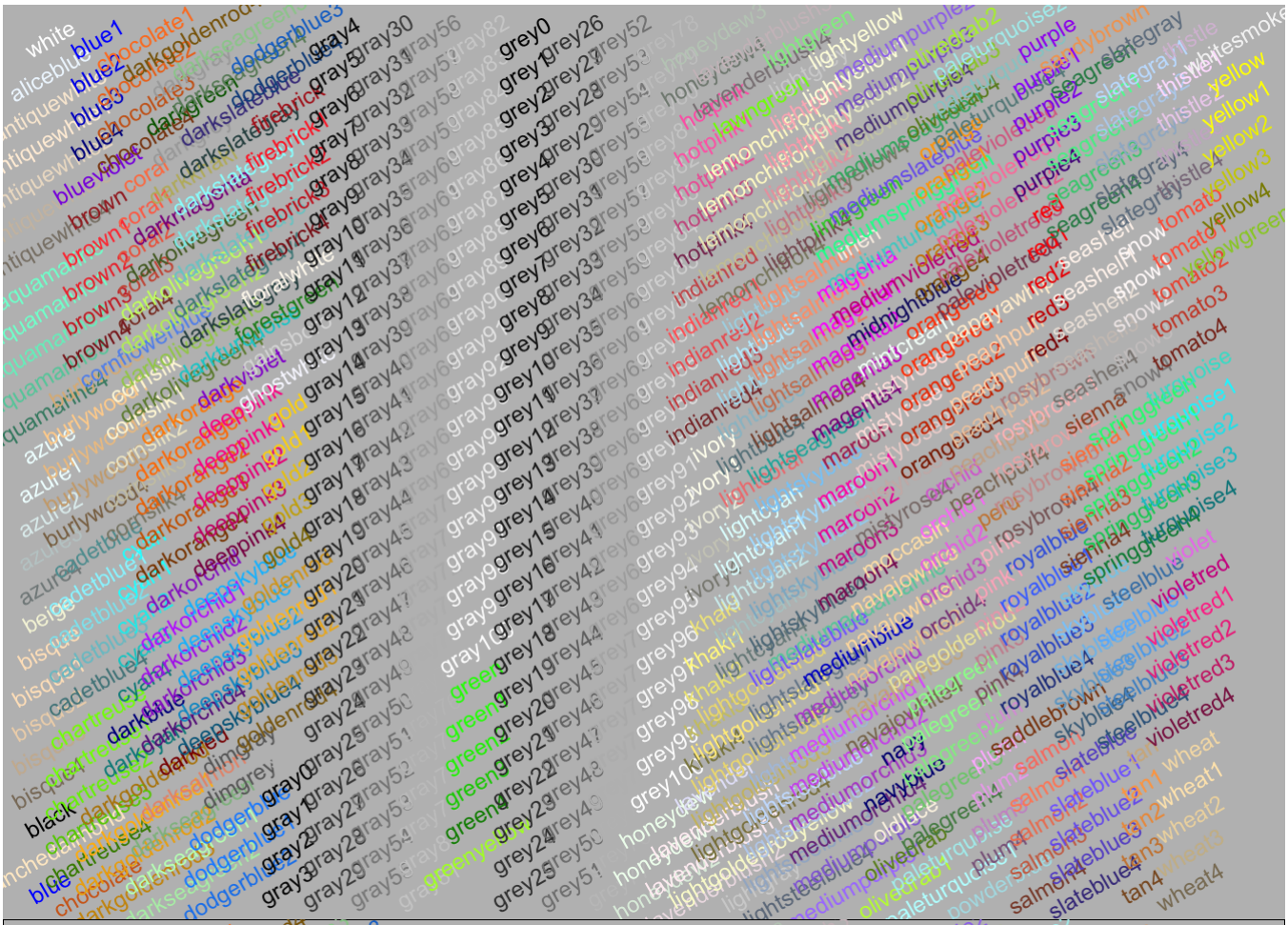
```
##  [1] "white"           "aliceblue"       "antiquewhite"    "antiquewhite1"
##  [5] "antiquewhite2"   "antiquewhite3"   "antiquewhite4"   "aquamarine"
##  [9] "aquamarine1"     "aquamarine2"     "aquamarine3"     "aquamarine4"
## [13] "azure"           "azure1"          "azure2"          "azure3"
## [17] "azure4"          "beige"           "bisque"          "bisque1"
## [21] "bisque2"         "bisque3"         "bisque4"         "black"
## [25] "blanchedalmond" "blue"            "blue1"           "blue2"
## [29] "blue3"           "blue4"           "blueviolet"      "brown"
## [33] "brown1"          "brown2"          "brown3"          "brown4"
## [37] "burlywood"       "burlywood1"      "burlywood2"      "burlywood3"
## [41] "burlywood4"      "cadetblue"       "cadetblue1"      "cadetblue2"
## [45] "cadetblue3"      "cadetblue4"      "chartreuse"      "chartreuse1"
## [49] "chartreuse2"     "chartreuse3"
```

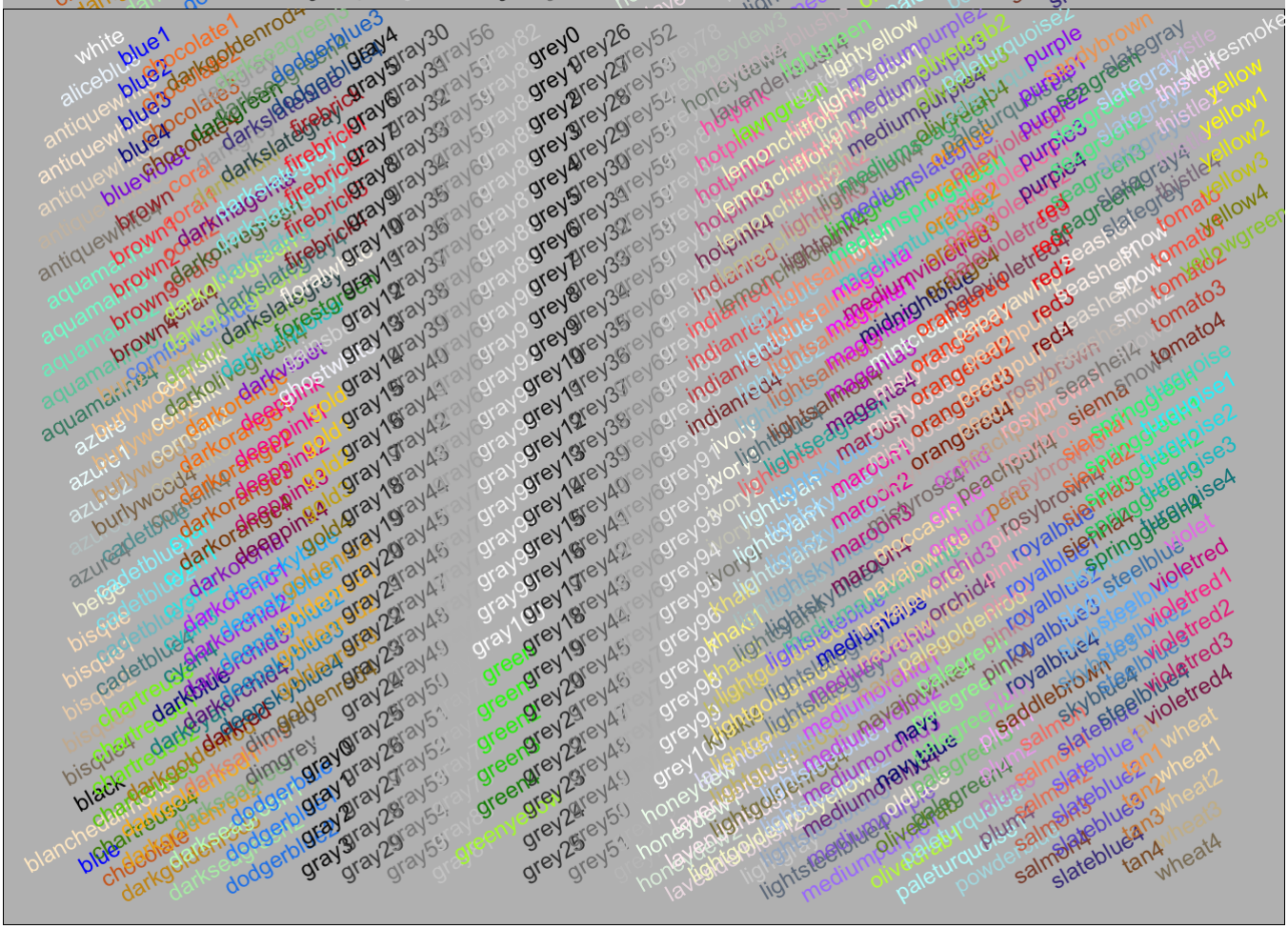
```
demo("colors")
```

```
##
##
##  demo(colors)
##  ---- ~~~~~~
##
## > ### ----------- Show (almost) all named colors ----------------------
## >
## > ## 1) with traditional 'graphics' package:
## > showCols1 <- function(bg = "gray", cex = 0.75, srt = 30) {
## +     m <- ceiling(sqrt(n <- length(cl <- colors())))
## +     length(cl) <- m*m; cm <- matrix(cl, m)
## +     ##
## +     require("graphics")
## +     op <- par(mar=rep(0,4), ann=FALSE, bg = bg); on.exit(par(op))
## +     plot(1:m,1:m, type="n", axes=FALSE)
## +     text(col(cm), rev(row(cm)), cm,  col = cl, cex=cex, srt=srt)
## + }
##
## > showCols1()
```

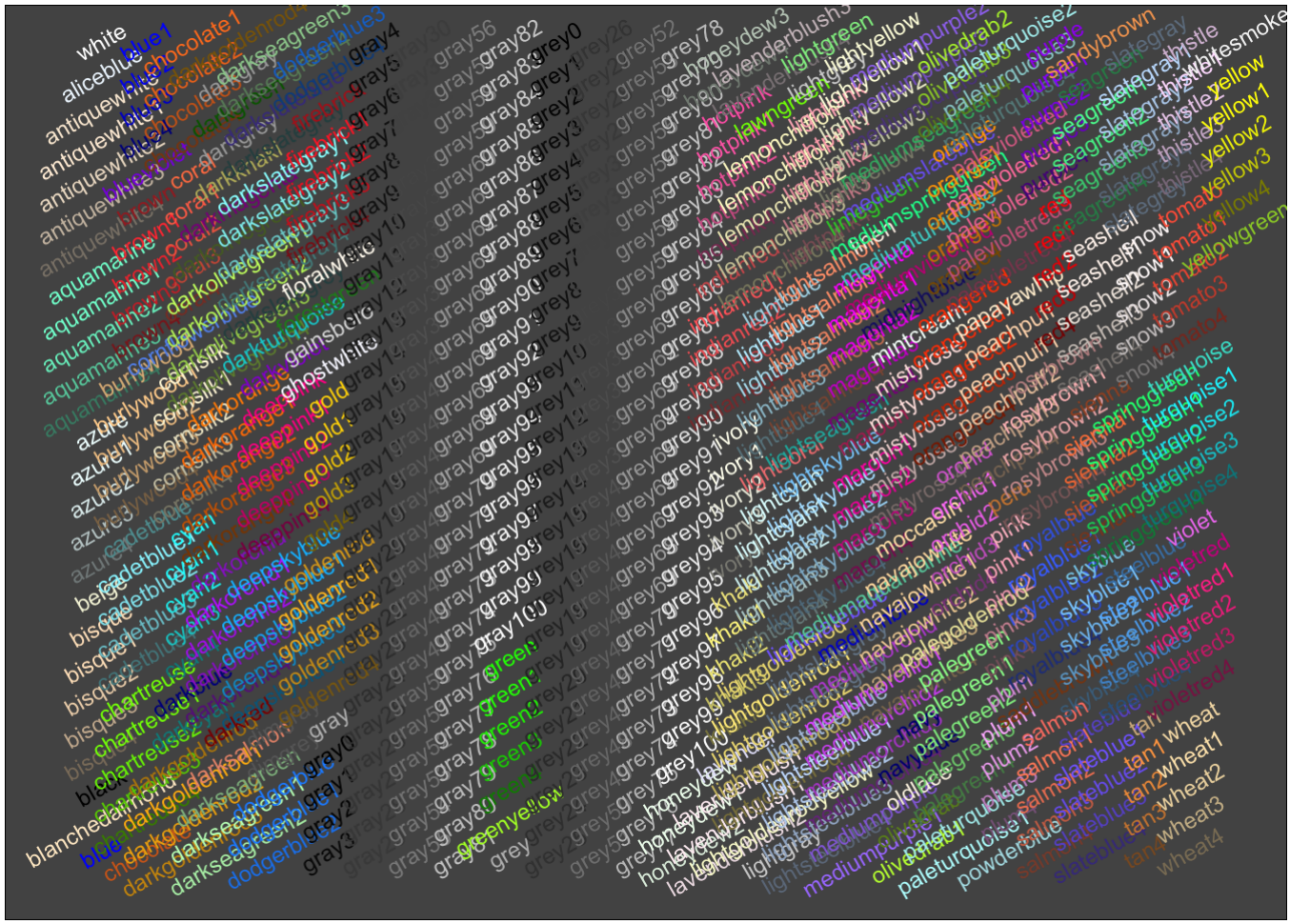
```
##
## > ## 2) with 'grid' package:
## > showCols2 <- function(bg = "grey", cex = 0.75, rot = 30) {
## +     m <- ceiling(sqrt(n <- length(cl <- colors())))
## +     length(cl) <- m*m; cm <- matrix(cl, m)
## +     ##
## +     require("grid")
## +     grid.newpage(); vp <- viewport(width = .92, height = .92)
## +     grid.rect(gp=gpar(fill=bg))
## +     grid.text(cm, x = col(cm)/m, y = rev(row(cm))/m, rot = rot,
## +               vp=vp, gp=gpar(cex = cex, col = cm))
## + }
##
## > showCols2()
```

```
## Loading required package: grid
```





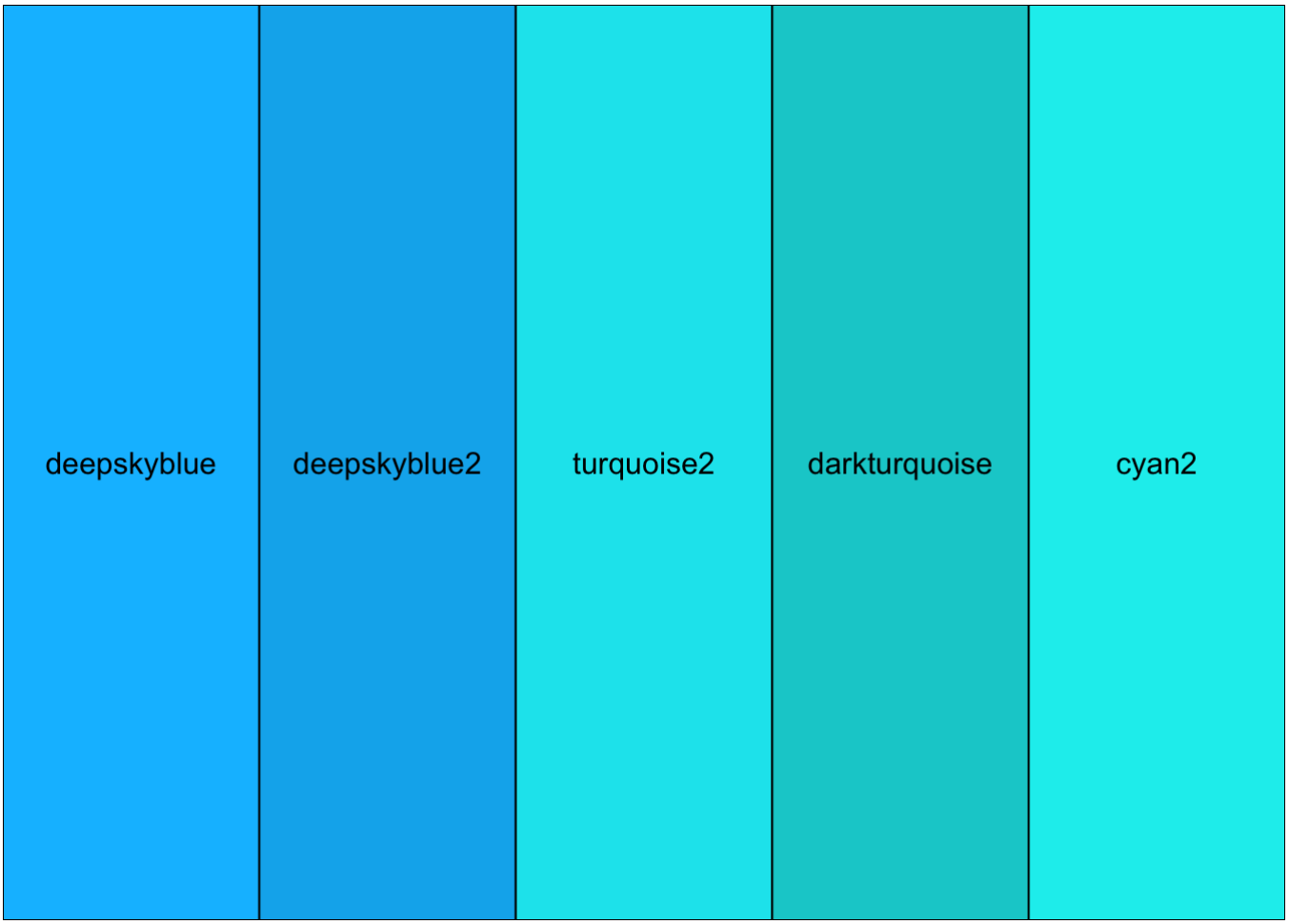
```
##
## > showCols2(bg = "gray33")
```


white
aliceblue
antiquewhite
antiquewhite1
antiquewhite2
antiquewhite3
antiquewhite4
aquamarine
aquamarine1
aquamarine2
aquamarine3
aquamarine4
azure
azure1
azure2
azure3
azure4
beige
bisque
bisque1
bisque2
bisque3
bisque4
black
blanchedalmond
blue
blue1
blue2
blue3
blue4
brown
brown1
brown2
brown3
brown4
burlywood
burlywood1
burlywood2
burlywood3
burlywood4
cadetblue
cadetblue1
cadetblue2
cadetblue3
cadetblue4
chartreuse
chartreuse1
chartreuse2
chartreuse3
chartreuse4
chocolate
chocolate1
chocolate2
chocolate3
chocolate4
coral
coral1
coral2
coral3
coral4
cornflowerblue
cornsilk
cornsilk1
cornsilk2
cornsilk3
cornsilk4
cyan
cyan1
cyan2
cyan3
cyan4
darkblue
darkcyan
darkgoldenrod
darkgoldenrod1
darkgoldenrod2
darkgoldenrod3
darkgoldenrod4
darkgray
darkgreen
darkkhaki
darkmagenta
darkolivegreen
darkolivegreen1
darkolivegreen2
darkolivegreen3
darkolivegreen4
darkorange
darkorange1
darkorange2
darkorange3
darkorange4
darkorchid
darkorchid1
darkorchid2
darkorchid3
darkorchid4
darkred
darksalmon
darkseagreen
darkseagreen1
darkseagreen2
darkseagreen3
darkseagreen4
darkslateblue
darkslategray
darkslategray1
darkslategray2
darkslategray3
darkslategray4
darkturquoise
darkviolet
deeppink
deeppink1
deeppink2
deeppink3
deeppink4
deepskyblue
deepskyblue1
deepskyblue2
deepskyblue3
deepskyblue4
dimgray
dodgerblue
dodgerblue1
dodgerblue2
dodgerblue3
dodgerblue4
firebrick
firebrick1
firebrick2
firebrick3
firebrick4
floralwhite
forestgreen
gainsboro
ghostwhite
gold
gold1
gold2
gold3
gold4
goldenrod
goldenrod1
goldenrod2
goldenrod3
goldenrod4
gray
gray0
gray1
gray2
gray3
gray4
gray5
gray6
gray7
gray8
gray9
gray10
gray11
gray12
gray13
gray14
gray15
gray16
gray17
gray18
gray19
gray20
gray21
gray22
gray23
gray24
gray25
gray26
gray27
gray28
gray29
gray30
gray31
gray32
gray33
gray34
gray35
gray36
gray37
gray38
gray39
gray40
gray41
gray42
gray43
gray44
gray45
gray46
gray47
gray48
gray49
gray50
gray51
gray52
gray53
gray54
gray55
gray56
gray57
gray58
gray59
gray60
gray61
gray62
gray63
gray64
gray65
gray66
gray67
gray68
gray69
gray70
gray71
gray72
gray73
gray74
gray75
gray76
gray77
gray78
gray79
gray80
gray81
gray82
gray83
gray84
gray85
gray86
gray87
gray88
gray89
gray90
gray91
gray92
gray93
gray94
gray95
gray96
gray97
gray98
gray99
gray100
green
green1
green2
green3
green4
greenyellow
grey
grey0
grey1
grey2
grey3
grey4
grey5
grey6
grey7
grey8
grey9
grey10
grey11
grey12
grey13
grey14
grey15
grey16
grey17
grey18
grey19
grey20
grey21
grey22
grey23
grey24
grey25
grey26
grey27
grey28
grey29
grey30
grey31
grey32
grey33
grey34
grey35
grey36
grey37
grey38
grey39
grey40
grey41
grey42
grey43
grey44
grey45
grey46
grey47
grey48
grey49
grey50
grey51
grey52
grey53
grey54
grey55
grey56
grey57
grey58
grey59
grey60
grey61
grey62
grey63
grey64
grey65
grey66
grey67
grey68
grey69
grey70
grey71
grey72
grey73
grey74
grey75
grey76
grey77
grey78
grey79
grey80
grey81
grey82
grey83
grey84
grey85
grey86
grey87
grey88
grey89
grey90
grey91
grey92
grey93
grey94
grey95
grey96
grey97
grey98
grey99
grey100
honeydew
honeydew1
honeydew2
honeydew3
honeydew4
hotpink
hotpink1
hotpink2
hotpink3
hotpink4
indianred
indianred1
indianred2
indianred3
indianred4
ivory
ivory1
ivory2
ivory3
ivory4
khaki
khaki1
khaki2
khaki3
khaki4
lavender
lavenderblush
lavenderblush1
lavenderblush2
lavenderblush3
lavenderblush4
lawngreen
lemonchiffon
lemonchiffon1
lemonchiffon2
lemonchiffon3
lemonchiffon4
lightblue
lightblue1
lightblue2
lightblue3
lightblue4
lightcoral
lightcyan
lightcyan1
lightcyan2
lightcyan3
lightcyan4
lightgoldenrod
lightgoldenrod1
lightgoldenrod2
lightgoldenrod3
lightgoldenrod4
lightgoldenrodyellow
lightgray
lightgreen
lightpink
lightpink1
lightpink2
lightpink3
lightpink4
lightsalmon
lightsalmon1
lightsalmon2
lightsalmon3
lightsalmon4
lightseagreen
lightskyblue
lightskyblue1
lightskyblue2
lightskyblue3
lightskyblue4
lightslateblue
lightslategray
lightsteelblue
lightsteelblue1
lightsteelblue2
lightsteelblue3
lightsteelblue4
lightyellow
lightyellow1
lightyellow2
lightyellow3
lightyellow4
limegreen
linen
magenta
magenta1
magenta2
magenta3
magenta4
maroon
maroon1
maroon2
maroon3
maroon4
mediumaquamarine
mediumblue
mediumorchid
mediumorchid1
mediumorchid2
mediumorchid3
mediumorchid4
mediumpurple
mediumpurple1
mediumpurple2
mediumpurple3
mediumpurple4
mediumseagreen
mediumslateblue
mediumspringgreen
mediumturquoise
mediumvioletred
midnightblue
mintcream
mistyrose
mistyrose1
mistyrose2
mistyrose3
mistyrose4
moccasin
navajowhite
navajowhite1
navajowhite2
navajowhite3
navajowhite4
navy
navyblue
oldlace
olivedrab
olivedrab1
olivedrab2
olivedrab3
olivedrab4
orange
orange1
orange2
orange3
orange4
orangered
orangered1
orangered2
orangered3
orangered4
orchid
orchid1
orchid2
orchid3
orchid4
palegoldenrod
palegreen
palegreen1
palegreen2
palegreen3
palegreen4
paleturquoise
paleturquoise1
paleturquoise2
paleturquoise3
paleturquoise4
palevioletred
palevioletred1
palevioletred2
palevioletred3
palevioletred4
papayawhip
peachpuff
peachpuff1
peachpuff2
peachpuff3
peachpuff4
peru
pink
pink1
pink2
pink3
pink4
plum
plum1
plum2
plum3
plum4
powderblue
purple
purple1
purple2
purple3
purple4
red
red1
red2
red3
red4
rosybrown
rosybrown1
rosybrown2
rosybrown3
rosybrown4
royalblue
royalblue1
royalblue2
royalblue3
royalblue4
saddlebrown
salmon
salmon1
salmon2
salmon3
salmon4
sandybrown
seagreen
seagreen1
seagreen2
seagreen3
seagreen4
seashell
seashell1
seashell2
seashell3
seashell4
sienna
sienna1
sienna2
sienna3
sienna4
skyblue
skyblue1
skyblue2
skyblue3
skyblue4
slateblue
slateblue1
slateblue2
slateblue3
slateblue4
slategray
slategray1
slategray2
slategray3
slategray4
snow
snow1
snow2
snow3
snow4
springgreen
springgreen1
springgreen2
springgreen3
springgreen4
steelblue
steelblue1
steelblue2
steelblue3
steelblue4
tan
tan1
tan2
tan3
tan4
thistle
thistle1
thistle2
thistle3
thistle4
tomato
tomato1
tomato2
tomato3
tomato4
turquoise
turquoise1
turquoise2
turquoise3
turquoise4
violet
violetred
violetred1
violetred2
violetred3
violetred4
wheat
wheat1
wheat2
wheat3
wheat4
whitesmoke
yellow
yellow1
yellow2
yellow3
yellow4
yellowgreen

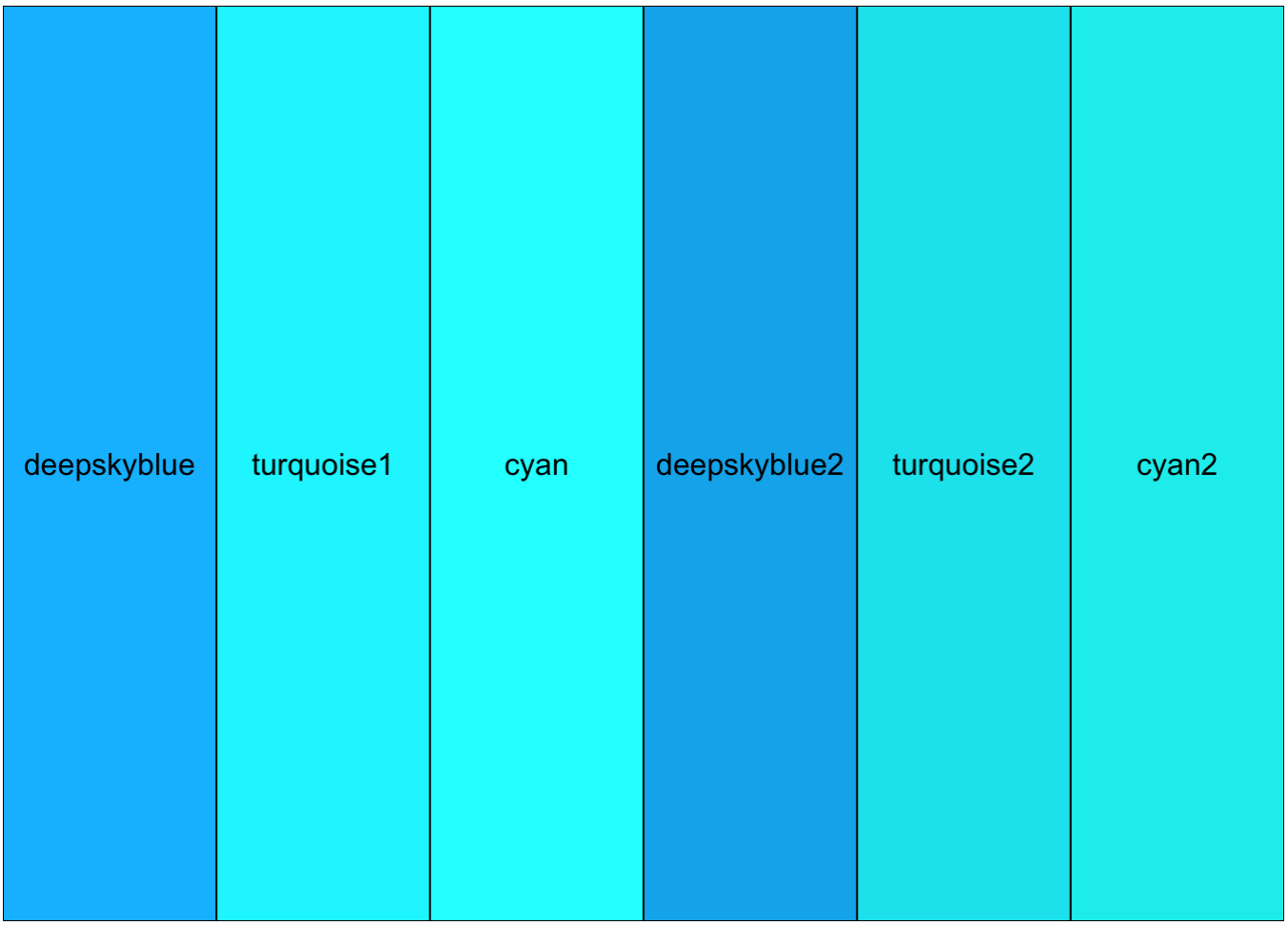
```
##
## > ###
## >
## > ##' @title Comparing Colors
## > ##' @param col
## > ##' @param nrow
## > ##' @param ncol
## > ##' @param txt.col
## > ##' @return the grid layout, invisibly
## > ##' @author Marius Hofert, originally
## > plotCol <- function(col, nrow=1, ncol=ceiling(length(col) / nrow),
## +                     txt.col="black") {
## +     stopifnot(nrow >= 1, ncol >= 1)
## +     if(length(col) > nrow*ncol)
## +         warning("some colors will not be shown")
## +     require(grid)
## +     grid.newpage()
## +     gl <- grid.layout(nrow, ncol)
## +     pushViewport(viewport(layout=gl))
## +     ic <- 1
## +     for(i in 1:nrow) {
## +         for(j in 1:ncol) {
## +             pushViewport(viewport(layout.pos.row=i, layout.pos.col=j))
## +             grid.rect(gp= gpar(fill=col[ic]))
## +             grid.text(col[ic], gp=gpar(col=txt.col))
## +             upViewport()
## +             ic <- ic+1
## +         }
## +     }
## +     upViewport()
## +     invisible(gl)
## + }
##
## > ## A Chocolate Bar of colors:
## > plotCol(c("#CC8C3C", paste0("chocolate", 2:4),
## +           paste0("darkorange", c("",1:2)), paste0("darkgoldenrod", 1:2),
## +           "orange", "orange1", "sandybrown", "tan1", "tan2"),
## +         nrow=2)
```


#CC8C3C
chocolate2
chocolate3
chocolate4
darkorange
darkorange1
darkorange2
arkgoldenrod
darkgoldenrod
orange
orange1
sandybrown
tan1
tan2

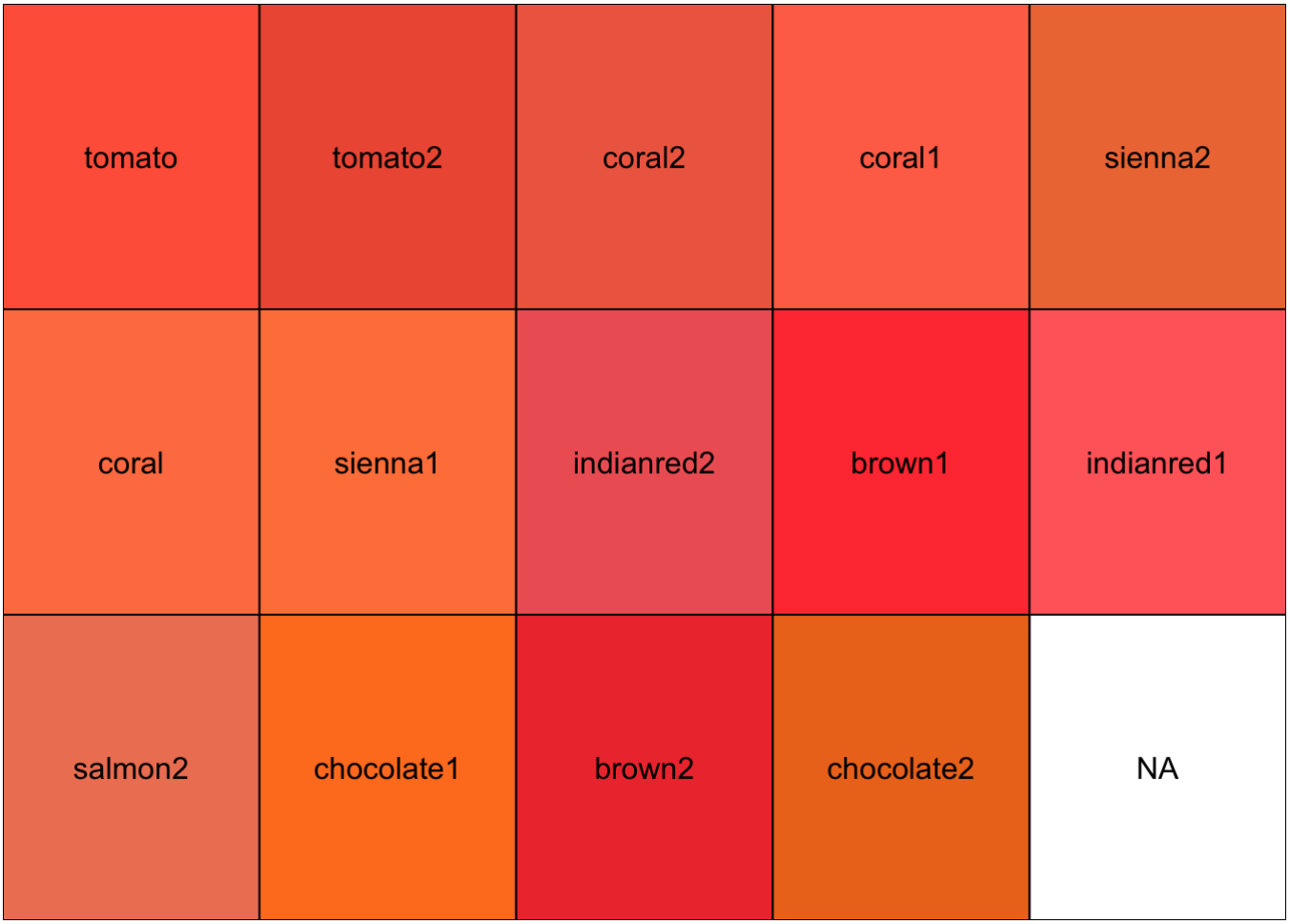
```
##
## > ##' Find close R colors() to a given color {original by Marius Hofert)
## > ##' using Euclidean norm in (HSV / RGB / ...) color space
## > nearRcolor <- function(rgb, cSpace = c("hsv", "rgb255", "Luv", "Lab"),
## +                        dist = switch(cSpace, "hsv" = 0.10, "rgb255" = 30,
## +                        "Luv" = 15, "Lab" = 12))
## + {
## +     if(is.character(rgb)) rgb <- col2rgb(rgb)
## +     stopifnot(length(rgb <- as.vector(rgb)) == 3)
## +     Rcol <- col2rgb(.cc <- colors())
## +     uniqC <- !duplicated(t(Rcol)) # gray9 == grey9 (etc)
## +     Rcol <- Rcol[, uniqC] ; .cc <- .cc[uniqC]
## +     cSpace <- match.arg(cSpace)
## +     convRGB2 <- function(Rgb, to)
## +         t(convertColor(t(Rgb), from="sRGB", to=to, scale.in=255))
## +     ## the transformation,  rgb{0..255} --> cSpace :
## +     TransF <- switch(cSpace,
## +                      "rgb255" = identity,
## +                      "hsv" = rgb2hsv,
## +                      "Luv" = function(RGB) convRGB2(RGB, "Luv"),
## +                      "Lab" = function(RGB) convRGB2(RGB, "Lab"))
## +     d <- sqrt(colSums((TransF(Rcol) - as.vector(TransF(rgb)))^2))
## +     iS <- sort.list(d[near <- d <= dist])# sorted: closest first
## +     setNames(.cc[near][iS], format(zapsmall(d[near][iS]), digits=3))
## + }
##
## > nearRcolor(col2rgb("tan2"), "rgb")
##         0.0         21.1         25.8         29.5
##       "tan2"       "tan1" "sandybrown"    "sienna1"
##
## > nearRcolor(col2rgb("tan2"), "hsv")
##       0.0000       0.0410       0.0618       0.0638       0.0667       0.0766
##       "tan2"    "sienna2"     "coral2"    "tomato2"       "tan1"      "coral"
##       0.0778       0.0900       0.0912       0.0918
##    "sienna1" "sandybrown"     "coral1"     "tomato"
##
## > nearRcolor(col2rgb("tan2"), "Luv")
##         0.00         7.42         7.48        12.41        13.69
##       "tan2"       "tan1" "sandybrown"    "orange3"    "orange2"
##
## > nearRcolor(col2rgb("tan2"), "Lab")
##         0.00         5.56         8.08        11.31
##       "tan2"       "tan1" "sandybrown"       "peru"
##
## > nearRcolor("#334455")
##          0.0867
## "darkslategray"
##
## > ## Now, consider choosing a color by looking in the
## > ## neighborhood of one you know :
## >
## > plotCol(nearRcolor("deepskyblue", "rgb", dist=50))
```



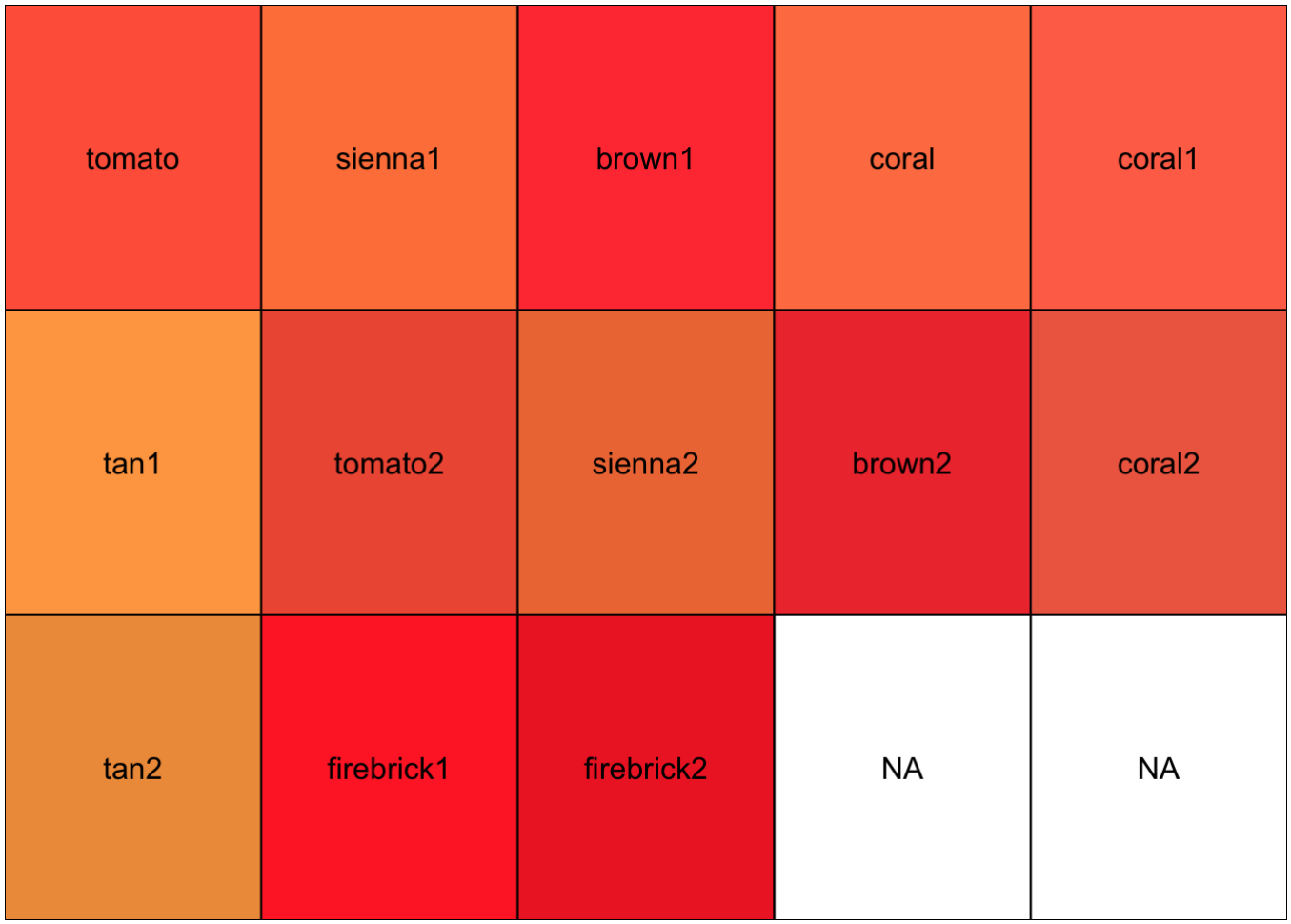

```
##
## > plotCol(nearRcolor("deepskyblue", dist=.1))
```



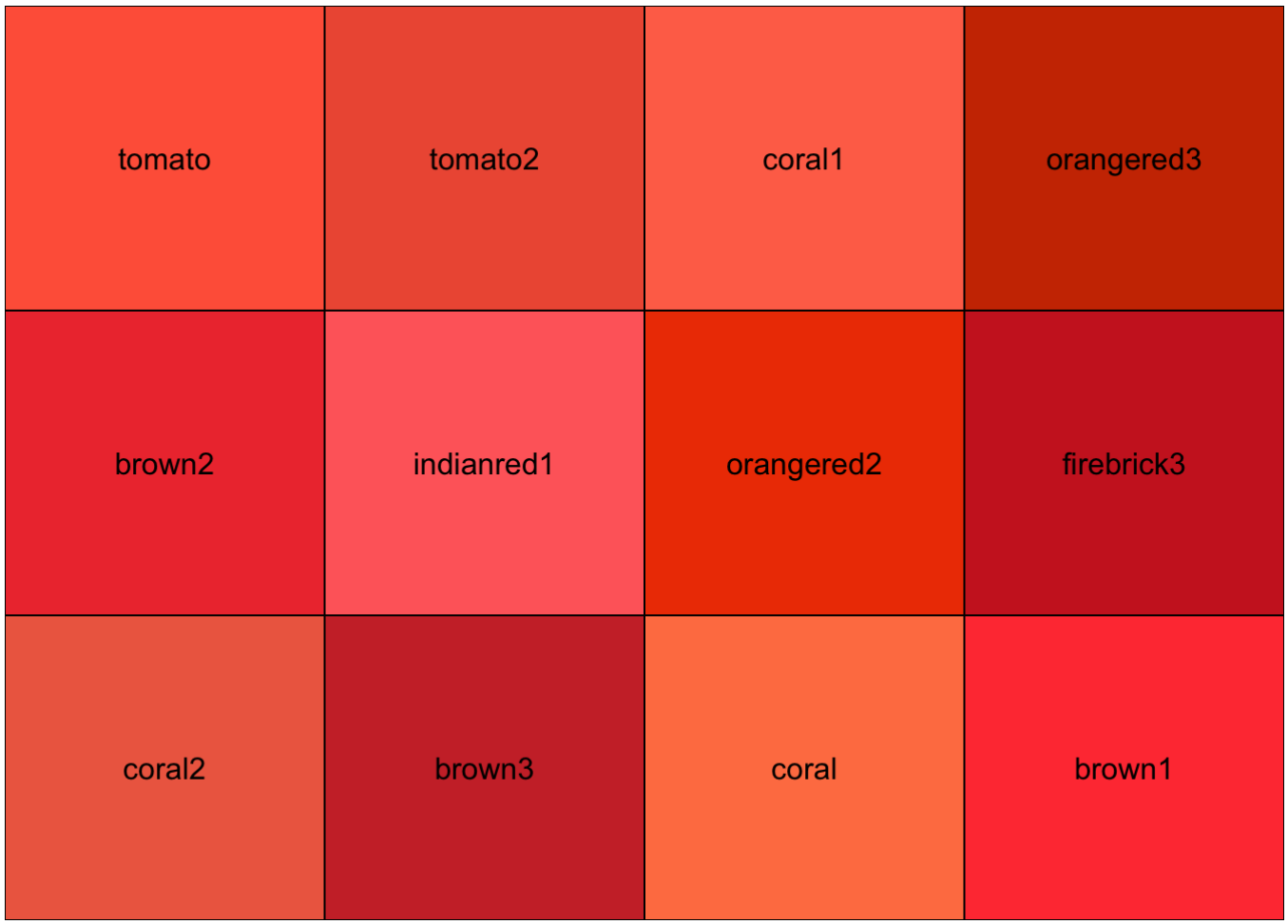
```
##
## > plotCol(nearRcolor("tomato", "rgb", dist= 50), nrow=3)
```



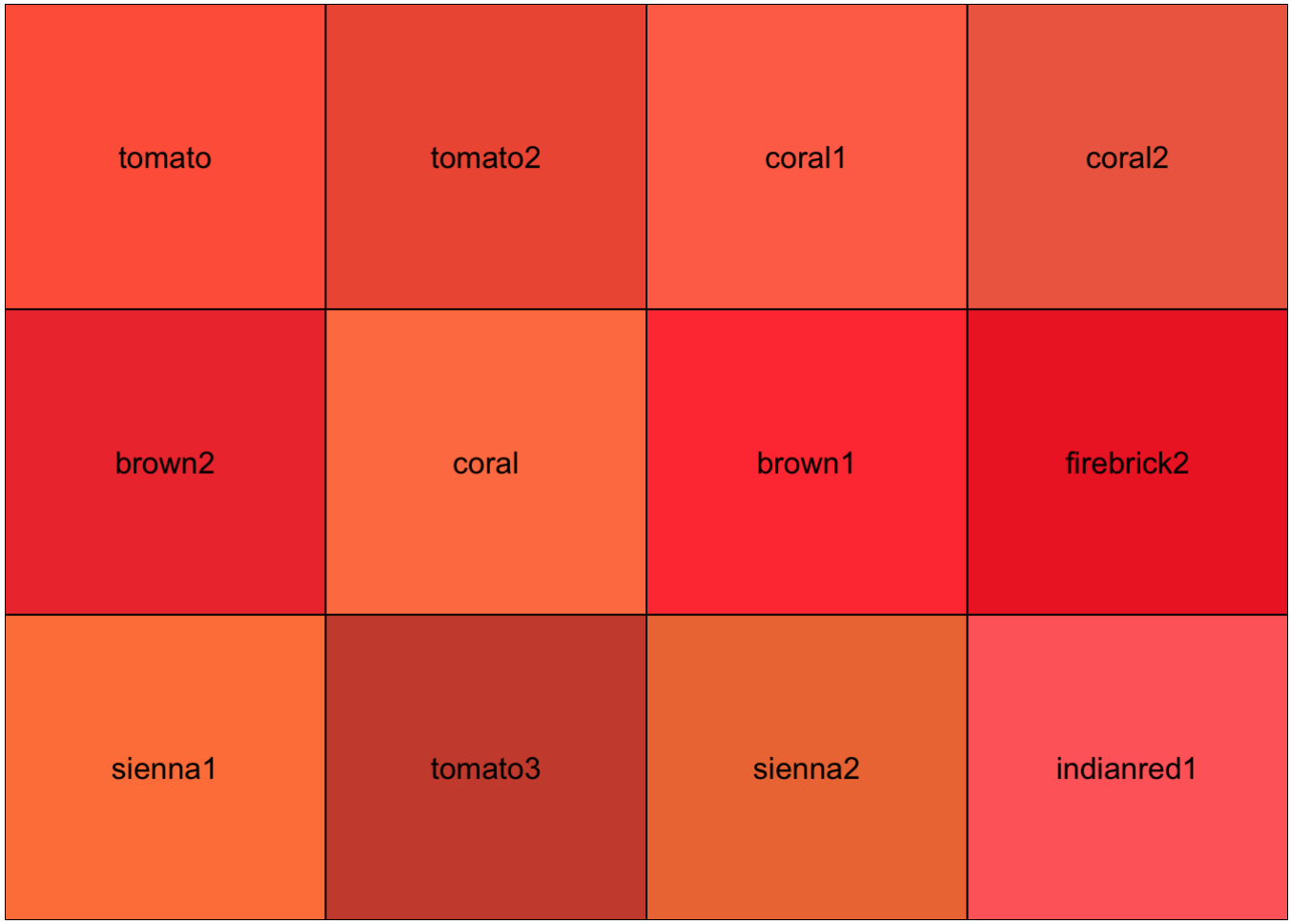

```
##
## > plotCol(nearRcolor("tomato", "hsv", dist=.12), nrow=3)
```




```
##
## > plotCol(nearRcolor("tomato", "Luv", dist= 25), nrow=3)
```



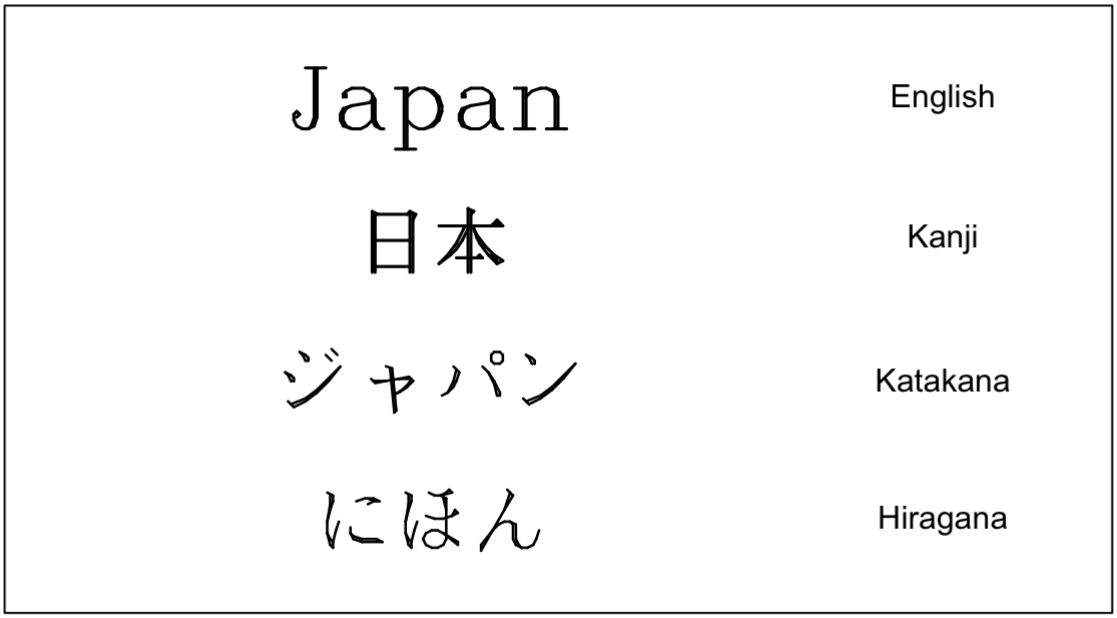
```
##
## > plotCol(nearRcolor("tomato", "Lab", dist= 18), nrow=3)
```



Japanese Returns the built-in color names which R knows about.

```
require(graphics)

plot(1:9, type = "n", axes = FALSE, frame.plot = TRUE, ylab = "",
     main = "example(Japanese)", xlab = "using Hershey fonts")
par(cex = 3)
Vf <- c("serif", "plain")

text(4, 2, "\\#J244b\\#J245b\\#J2473", vfont = Vf)
text(4, 4, "\\#J2538\\#J2563\\#J2551\\#J2573", vfont = Vf)
text(4, 6, "\\#J467c\\#J4b5c", vfont = Vf)
text(4, 8, "Japan", vfont = Vf)
par(cex = 1)
text(8, 2, "Hiragana")
text(8, 4, "Katakana")
text(8, 6, "Kanji")
text(8, 8, "English")
```

example(Japanese)



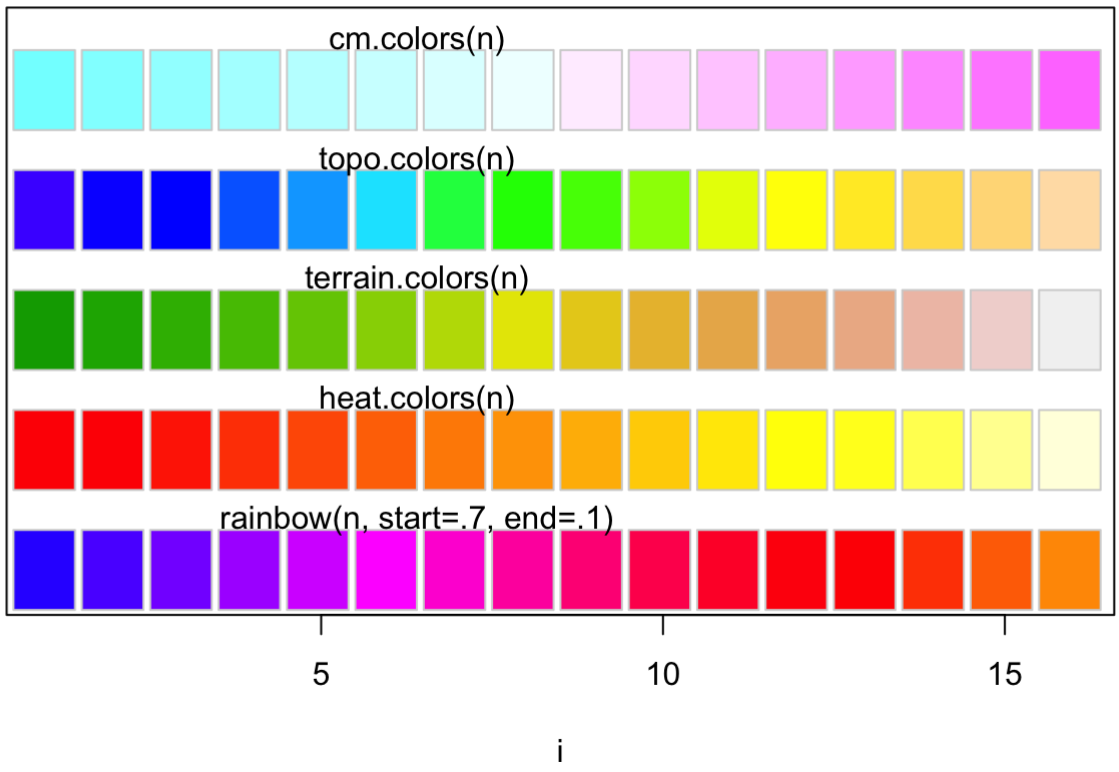
using Hershey fonts

Palettes

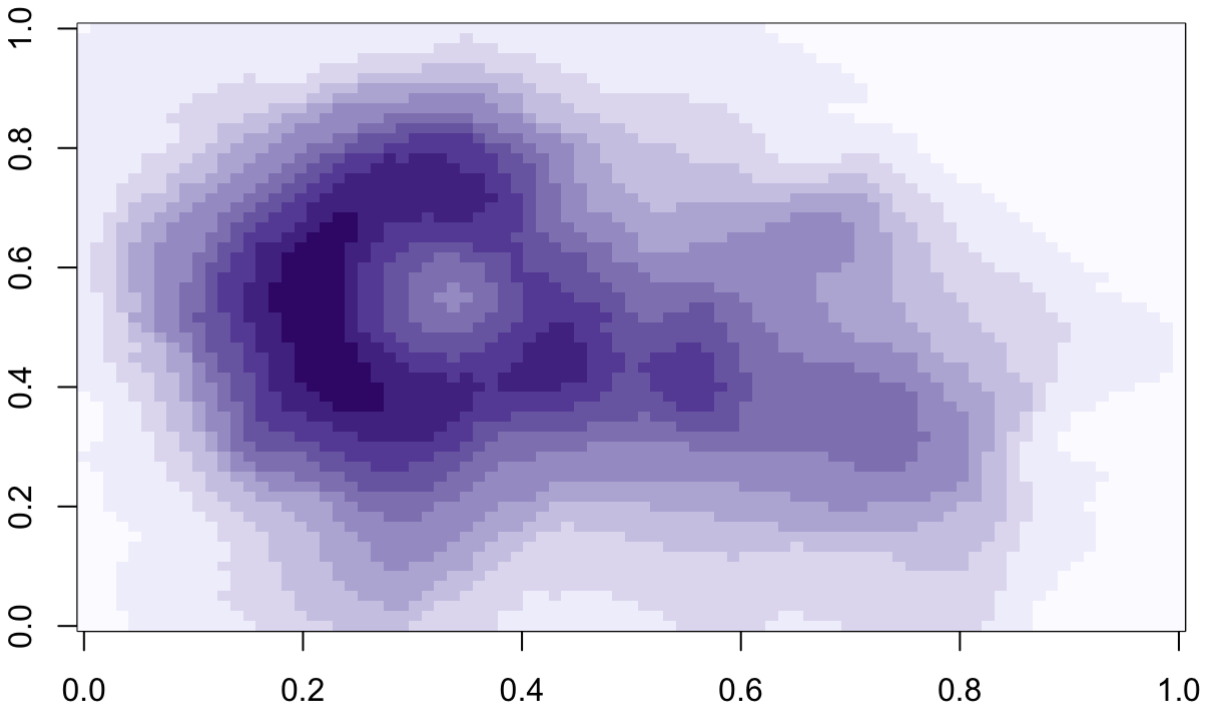
Create a vector of n contiguous colors.

```
require("graphics")

# color wheels in RGB/HSV and HCL space
par(mfrow = c(2, 2))
pie(rep(1, 12), col = rainbow(12), main = "RGB/HSV")
pie(rep(1, 12), col = hcl.colors(12, "Set 2"), main = "HCL")
par(mfrow = c(1, 1))
```

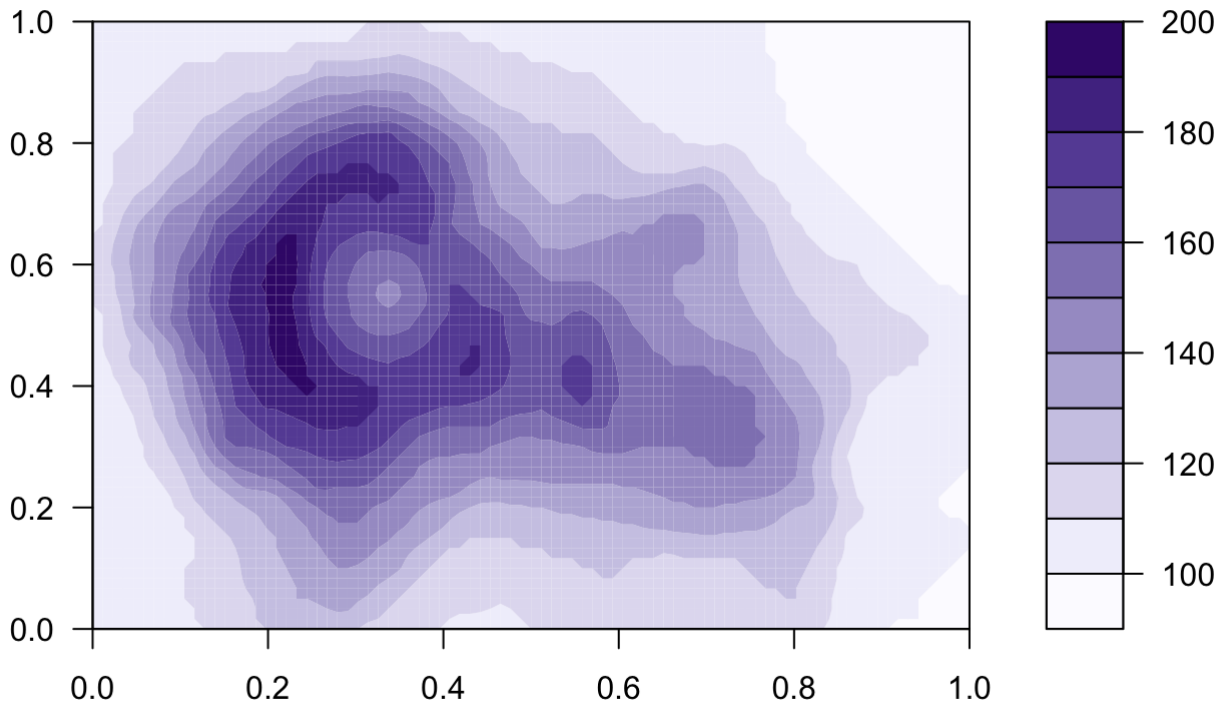

```
## color swatches for RGB/HSV palettes
demo.pal <-
  function(n, border = if (n < 32) "light gray" else NA,
           main = paste("color palettes;  n=", n),
           ch.col = c("rainbow(n, start=.7, end=.1)", "heat.colors(n)",
                      "terrain.colors(n)", "topo.colors(n)",
                      "cm.colors(n)"))
{
    nt <- length(ch.col)
    i <- 1:n; j <- n / nt; d <- j/6; dy <- 2*d
    plot(i, i+d, type = "n", yaxt = "n", ylab = "", main = main)
    for (k in 1:nt) {
        rect(i-.5, (k-1)*j+ dy, i+.4, k*j,
             col = eval(str2lang(ch.col[k])), border = border)
        text(2*j,  k * j + dy/4, ch.col[k])
    }
}
demo.pal(16)
```


color palettes; n= 16
cm.colors(n)
topo.colors(n)
terrain.colors(n)
heat.colors(n)
rainbow(n, start=.7, end=.1)
5
10
15
i

```
## color swatches for HCL palettes
hcl.swatch <- function(type = NULL, n = 5, nrow = 11,
  border = if (n < 15) "black" else NA) {
    palette <- hcl.pals(type)
    cols <- sapply(palette, hcl.colors, n = n)
    ncol <- ncol(cols)
    nswatch <- min(ncol, nrow)

    par(mar = rep(0.1, 4),
        mfrow = c(1, min(5, ceiling(ncol/nrow))),
        pin = c(1, 0.5 * nswatch),
        cex = 0.7)

    while (length(palette)) {
        subset <- 1:min(nrow, ncol(cols))
        plot.new()
        plot.window(c(0, n), c(0, nrow + 1))
        text(0, rev(subset) + 0.1, palette[subset], adj = c(0, 0))
        y <- rep(subset, each = n)
        rect(rep(0:(n-1), n), rev(y), rep(1:n, n), rev(y) - 0.5,
             col = cols[, subset], border = border)
        palette <- palette[-subset]
        cols <- cols[, -subset, drop = FALSE]
    }

    par(mfrow = c(1, 1), mar = c(5.1, 4.1, 4.1, 2.1), cex = 1)
}
# hcl.swatch()
# hcl.swatch("qualitative")
# hcl.swatch("sequential")
# hcl.swatch("diverging")
# hcl.swatch("divergingx")

## heat maps with sequential HCL palette (purple)
image(volcano, col = hcl.colors(11, "purples", rev = TRUE))
```




```
filled.contour(volcano, nlevels = 10,
               color.palette = function(n, ...)
                   hcl.colors(n, "purples", rev = TRUE, ...))
```




```
## list available HCL color palettes
hcl.pals("qualitative")
```

```
## [1] "Pastel 1" "Dark 2"   "Dark 3"   "Set 2"    "Set 3"    "Warm"     "Cold"
## [8] "Harmonic" "Dynamic"
```

```
hcl.pals("sequential")
```

```
##  [1] "Grays"         "Light Grays"   "Blues 2"       "Blues 3"
##  [5] "Purples 2"     "Purples 3"     "Reds 2"        "Reds 3"
##  [9] "Greens 2"      "Greens 3"      "Oslo"          "Purple-Blue"
## [13] "Red-Purple"    "Red-Blue"      "Purple-Orange" "Purple-Yellow"
## [17] "Blue-Yellow"   "Green-Yellow"  "Red-Yellow"    "Heat"
## [21] "Heat 2"        "Terrain"       "Terrain 2"     "Viridis"
## [25] "Plasma"        "Inferno"       "Rocket"        "Mako"
## [29] "Dark Mint"     "Mint"          "BluGrn"        "Teal"
## [33] "TealGrn"       "Emrld"         "BluYl"         "ag_GrnYl"
## [37] "Peach"         "PinkYl"        "Burg"          "BurgYl"
## [41] "RedOr"         "OrYel"         "Purp"          "PurpOr"
## [45] "Sunset"        "Magenta"       "SunsetDark"    "ag_Sunset"
## [49] "BrwnYl"        "YlOrRd"        "YlOrBr"        "OrRd"
## [53] "Oranges"       "YlGn"          "YlGnBu"        "Reds"
## [57] "RdPu"          "PuRd"          "Purples"       "PuBuGn"
## [61] "PuBu"          "Greens"        "BuGn"          "GnBu"
## [65] "BuPu"          "Blues"         "Lajolla"       "Turku"
## [69] "Hawaii"        "Batlow"
```

```
hcl.pals("diverging")
```

```
##  [1] "Blue-Red"      "Blue-Red 2"    "Blue-Red 3"    "Red-Green"
##  [5] "Purple-Green"  "Purple-Brown"  "Green-Brown"   "Blue-Yellow 2"
##  [9] "Blue-Yellow 3" "Green-Orange"  "Cyan-Magenta"  "Tropic"
## [13] "Broc"          "Cork"          "Vik"           "Berlin"
## [17] "Lisbon"        "Tofino"
```

```
hcl.pals("divergingx")
```

```
##  [1] "ArmyRose" "Earth"    "Fall"     "Geyser"   "TealRose" "Temps"
##  [7] "PuOr"     "RdBu"     "RdGy"     "PiYG"     "PRGn"     "BrBG"
## [13] "RdYlBu"   "RdYlGn"   "Spectral" "Zissou 1" "Cividis"  "Roma"
```