BodyPies

Allison Waller (ghost edited: Glenn J Tattersall)

2019-03-12

The objective here is to overlay a series of piecharts on top of an existing image. This exercise was based on a microbiome sequence of the diversity of microbes (based on the sequence identification) and their abundance from different regions of skin on a human subject. The original data are removed for privacy and for pre-publication purposes, and random data are generated for this exercise.

Generate Random Abundance Data

Generate some random data. Actual values or units are not important here. In this case, 6 different body regions were measured so we generate 50 random numbers * 6 regions. Arbitrarily the mean = 0 and sd = 1.

```
dat<-as.data.frame(replicate(50, rnorm(6, mean=10, sd=1)))
```

Although not necessary, sometimes when creating random data you want to label the data. So here is code for creating random letters

Label the rows to correspond to the body regions from which these were derived

```
colnames(dat)<-replicate(50, paste(sample(LETTERS, 3, replace=TRUE), collapse=""))
```

We can also define the differnt rows on the basis of the region.

```
rownames(dat)<-c("ear","genital","gut","LH","mouth","nose")
head(dat)
```

```
##               JYS       NJS       OCB       NZL       KCB       RKK
## ear      9.367948  9.850507  9.370227  9.633007 11.055941  8.985264
## genital 10.169139 10.218784  9.823878  9.443310  8.574343  8.490574
## gut      9.891620  9.797156 11.163459  9.964733  9.810234 10.180573
## LH       9.475549  9.431359  8.885822  9.162601 10.343543 10.079127
## mouth   10.512541  9.824231  8.339991 11.202299  9.035959  9.766425
## nose     9.120390 10.310516 10.456761 10.033922  8.135807 10.667121
##               CDU       YGF       QFG       RHK       LQX       ELU
## ear     11.136154  7.883741  9.382495  9.326643 10.268598  9.233929
## genital  9.890831 10.702686 11.827909 10.778570  9.691769 10.232533
## gut     10.615784 10.663316 10.524699 10.869509  9.036197  8.948220
## LH       7.824949 10.402112 12.357182 10.801433  8.102194  9.251790
## mouth   10.841961 10.344860 10.221739  9.984354  9.080661 10.010881
## nose     9.660582 10.385474  9.359829 10.995331  7.058332 10.232478
##               XKW       UWL       HGN       TPW       TYX       CZO
## ear      9.159615  9.777337  8.289872  9.935120  8.908912 10.012750
## genital  7.964383 10.150577  9.369820 10.092535  8.603115 11.424642
## gut     11.126343 10.018129  9.864250  9.122209  9.140502 10.620763
## LH       8.934371 11.481066 10.910592  9.935300 10.695220  9.893130
## mouth    8.226417  9.190501 11.535424  8.512304 10.409869 11.189539
## nose    10.558256 11.154083  9.434187 10.984159 12.046398  8.812265
##               BVD       KYM       QJS       DLE       YCP       XBY
## ear      9.097433 11.368178 11.812236  9.217722  9.826867  9.208917
## genital  9.176122  8.187770 11.378467 10.967703  8.638963  9.201790
## gut      9.182190 10.303792  8.093335  9.022029 10.046217 10.621191
## LH       9.302914  9.453339  9.424067  9.740511  9.637432 10.713094
## mouth   10.755986  9.851919 10.274418 10.112893  8.877151 10.215698
## nose     9.575004  9.272923 10.319283 10.159701  9.271329 11.702225
##               IPS       MCY       RHC       ZQZ       SYY       MPQ
## ear     10.329477 10.414395 10.697292  9.733958  9.342026 10.608820
## genital  7.176713 10.943955  9.448259  9.966656  9.932715 10.735530
## gut     10.526951 11.548814 11.462928 10.717187  9.564636  9.344466
## LH      10.464014 10.760554 11.036522 10.725126  8.336383  9.838734
## mouth   10.924173  8.993162 11.383235 10.216162 10.714502  7.601534
## nose    10.222583 10.612048  9.557145 12.238977  9.481432  9.857364
##               FRU       SQY       DIC       JTX       MUZ       KWZ
## ear     11.762839 10.812792 10.832010 10.762449 10.133656 10.214842
## genital 10.919980 12.335518 10.039939  7.956277  9.848203 12.341368
## gut     10.923864  9.959205 10.963803  8.591557 10.415262  8.660630
## LH       9.126022 10.177470 10.276061  9.946564  8.309379 10.211710
## mouth   10.318929  8.839284  9.867311 10.715442 11.690022  9.142862
## nose     9.827585 11.976907 10.916965 10.007730 10.150779  9.990251
##               QYU       YVX       EQC       HTY       WGH       XHX
## ear      9.784051  9.602250 11.105240  9.015003 10.727926 10.711028
## genital 11.205704  9.001801  8.690637  9.420793  8.657295 11.484541
## gut      9.297117  9.408256  9.523190  9.053595 10.912767 10.390589
## LH      10.074648 11.347968 10.500404 11.329860  9.425461  7.429808
## mouth   11.592728 10.109740 10.610613  8.478329  9.760657 11.289029
## nose    10.360448 10.893637  8.827486  9.687599 10.001295  9.704214
##               WIH       MSV       ZCL       LSI      MLL       UGA
## ear     10.172415  8.029613  8.630429 10.596667 9.000261  8.858603
## genital 10.644968  9.523229 11.403698  7.985261 8.818383  9.688089
## gut     10.143900  9.415428  8.863259  9.655775 9.655973 12.149201
## LH       8.963733  7.696071  9.136487  9.339500 7.859895  9.508354
## mouth    9.506044 10.569830  9.499071 10.139568 9.199429  9.063186
## nose     9.058617 10.062690 10.536318 10.465250 9.746910  9.396683
##               SSI      PAV
## ear      9.278073 8.901188
## genital  9.654548 9.424648
## gut     11.560558 9.286115
## LH      10.062980 8.378807
## mouth    8.583548 8.915110
## nose     9.737079 8.952929
```

Create a Relative Abundance Plot using scatterpie

You'll need to install 3 libraries: "png", "grid", and "scatterpie"

```r
library(png)
library(grid)
library(scatterpie)
```

```
## Loading required package: ggplot2
```

First load in your png or jpg graphic upon which you want to plot your scatterplots. If you use the **as.raster()** you can convert the loaded image file into a rasterised image that readily plots.

```r
img <- readPNG( "aac_human_body.png")
plot(as.raster(img))
```

To use your graphic within the R ggplot environment, you will need to convert it to a grob, or specifically into a rasterGrob.

This allows you to include the png within the ggplot space. If you change the width or height, you can alter the aspect ratio. Keeping width and height = 1 retains the image aspect ratio. You can use the x and y options to place your graphic left/right or up/down within the plotting space.

We will set width to 0.8 to make the graphic a little more svelt:

```r
g<-rasterGrob(img, width=unit(0.8,"npc"), height=unit(1,"npc"), interpolate = FALSE)
```

We'll now use this rasterGrob below in the ggplot call, but first we need to decide where we want our scatter pie charts to go on top of this rasterGrob.

We do this by adding x, y, and radius coordinates to determine the placement (x,y) and size (radius) of pie charts. We'll call these imX, imY, and r and add them individually to the working data.frame, dat:

```r
dat$imX<-c(2, 1.5, 1.75, 0.3, 1.5, 1.5)
dat$imY<-c(5.5, 1, 2, 1, 4.8, 5.2)
dat$radius<-0.3
```

These numbers above are note necessarily intuitive, and you may need to tinkering with them. But one thing to understand is that x,y = 0,0 corresponds to the lower left of the plot window, while x,y = maxX, maxY corresponds to the upper right corner of the window.

So, when you create the final ggplot geom below, you will need to set your x and y limits to xlim(0,3) and ylim(0,6). At the moment, I don't follow how you ascertain what these values should be and why they are not simply 0 to 1. The help for scatterpie is terrible, thus making it difficult to figure some reasons out.

Create colour palette for the pie chart

R has a built in list of colours that you can see by running the **colors()** function:

```r
length(colors())
```

```
## [1] 657
```

```r
head(colors())
```

```
## [1] "white"         "aliceblue"     "antiquewhite"  "antiquewhite1"
## [5] "antiquewhite2" "antiquewhite3"
```

Use the **grep()** function to search for any entry that mentions grey or gray in order to create a list of only colours. There are > 200 colours that are technically gray, so we will create a new vector of colors, called allcolors that represents all the colours except the greyscales. Then we draw a random sample of 35 colours from allcolours.

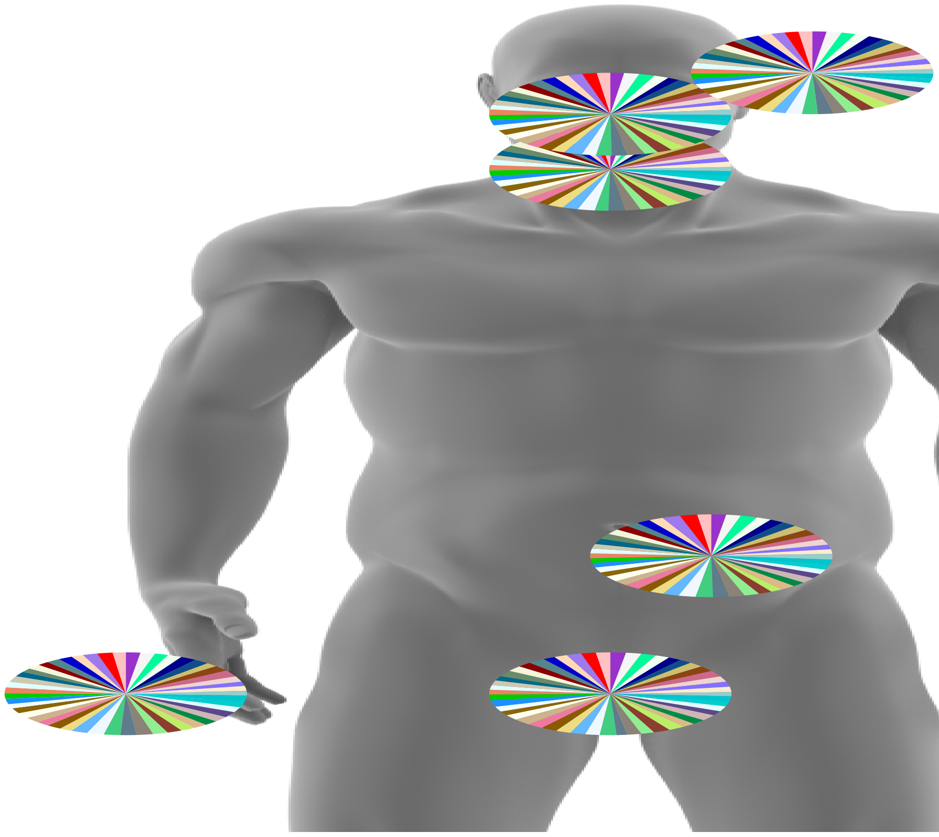
```r
length(grep('gr(a|e)y', grDevices::colors()))
```

```
## [1] 224
```

```r
allcolors = grDevices::colors()[grep('gr(a|e)y', grDevices::colors(), invert = T)]
colors35<-sample(allcolors,35)
```

Now, finally, the plot!

```r
ggplot(dat)+
  annotation_custom(g, xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=Inf)+
  geom_scatterpie(aes(x=imX, y=imY,r=radius),
                data=dat, cols=colnames(dat)[1:50],color=NA) +
  scale_fill_manual(values=sample(allcolors,50)) +
  scale_x_continuous(expand=c(0,0), lim=c(0,3)) +
  scale_y_continuous(expand=c(0,0), lim=c(0,6)) +
  #annotate("text", x=label_df$Xloc, y=label_df$Yloc+0.4,
  #         label= label_df$label)+
  theme(legend.position="none",
        panel.background = element_rect(fill = "transparent") # bg of the panel
        , plot.background = element_rect(fill = "transparent") # bg of the plot
        , panel.grid.major = element_blank() # get rid of major grid
        , panel.grid.minor = element_blank(), # get rid of minor grid
        line = element_blank(),
        text = element_blank(),
        title = element_blank()
  )
```



Finally save the document.

```
# ggsave('Fig4v2_wholebody_cutLH_cutbody_pies_col4.png', height=10, width = 5, units = 'in',dpi=300)
```