Visualize high dimensional data

Cory Whitney, Eduardo Fernandez, Thi Hoa Do

Dealing with large data sets can sometimes be confusing. If you are working in spreadsheets the confusion can reach the point of existential crises bordering on pure chaos. Good visualization tools can help. Visualization can allow you to get an overview of your data. It can also help you report patterns and differences in your data.

Needless to say any aims objectives and hypotheses should be determined before any data is collected. Data visualization is a good time to get a clear sense for how your data looks, but is not the time to start making up hypotheses about it.

Here we demonstrate a few different approaches for data visualization. We do this for several types of high dimensional data using plotting functions from tidyverse libraries including ggplot2, plyr and dplyr among others in the [R programming language](https://www.r-project.org/) (R Core Team 2019).

## Radial bar plots

Plots of high dimensional data do not always need an x-axis to be easy to read. In this case we sometimes compress it to a point using [polar coordinates](https://ggplot2.tidyverse.org/reference/coord_polar.html). For showing off options for radial bar plots we created an example data set with a factor variable using the data.frame and sample functions in base R (R Core Team 2019).

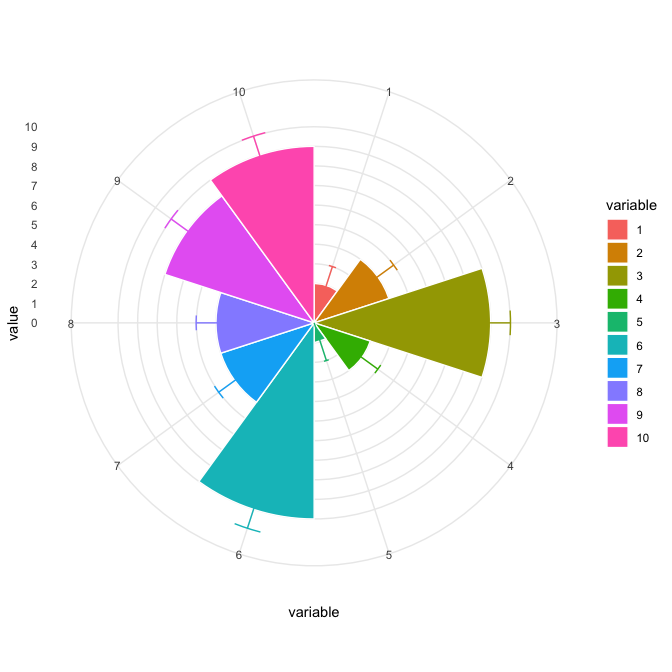
DF <- data.frame(variable = as.factor(1:10),  
 value = sample(10, replace = TRUE))

We also created a function to compute the standard error of the mean to represent some of the uncertainty in the data using the sqrt and length functions in base R and var from the stats library (R Core Team 2019).

se <- function(x) sqrt(var(x)/length(x))

We use the same data to create a radial bar plot using the functions above and the ggplot2 library (Wickham et al. 2019).

ggplot(DF, aes(variable, value, fill = variable)) +  
 geom\_bar(width = 1, stat = "identity", color = "white") +  
 geom\_errorbar(aes(ymin = value - se(DF$value),   
 ymax = value + se(DF$value),   
 color = variable),   
 width = .2) +   
 scale\_y\_continuous(breaks = 0:nlevels(DF$variable)) +  
 theme\_minimal() +  
 coord\_polar()



### Radial bar plots & multiple factor variables

Create a data set for radial plots with three factor variables.

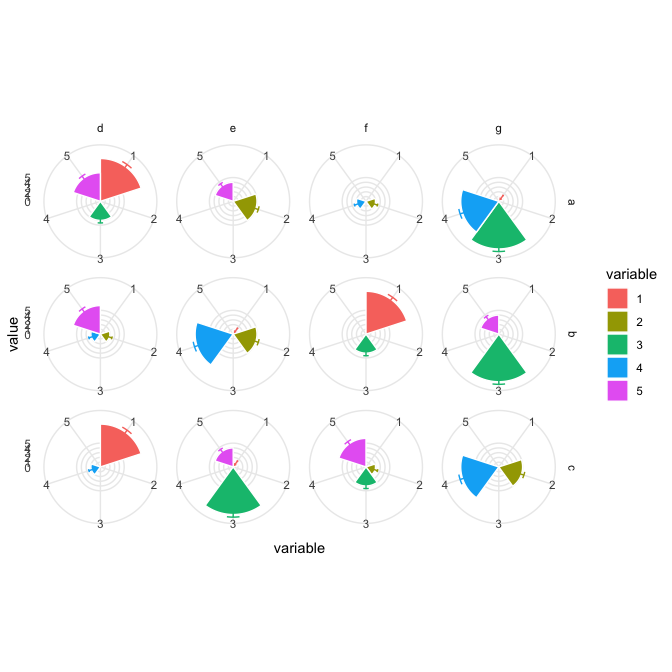
DF2 <- data.frame(name = rep(letters[1:3], length.out = 30),  
 variable = as.factor(1:5),  
 factor\_variable = rep(letters[4:7], length.out = 30),  
 value = sample(10, replace = TRUE))

Plot radial plots with three factor variables.

multi\_plot <- ggplot(DF2, aes(variable, value, fill = variable)) +  
 geom\_bar(width = 1, stat = "identity", color = "white") +  
 geom\_errorbar(aes(ymin = value - se(DF2$value),   
 ymax = value + se(DF2$value),   
 color = variable),   
 width = .2) +   
 scale\_y\_continuous(breaks = 0:nlevels(DF2$variable)) +  
 theme\_minimal() +  
 coord\_polar()

Plot with rows as names and columns as variables factor\_variable.

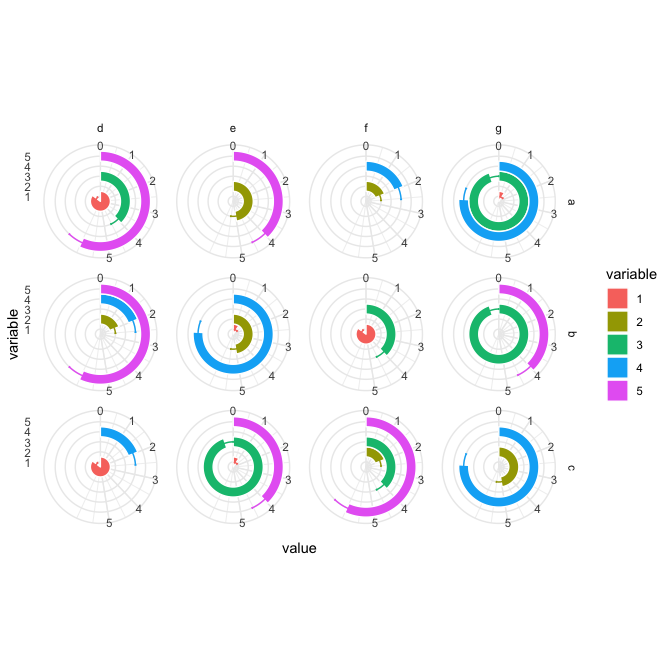
# Rows are name and columns are factor\_variable  
multi\_plot + facet\_grid(name ~ factor\_variable)



Plot with bars going around the circle.

# Rows are name and columns are factor\_variable  
multi\_plot +  
 coord\_polar(theta="y")+   
 facet\_grid(name ~ factor\_variable)

## Coordinate system already present. Adding new coordinate system, which will replace the existing one.



More on making polar bar plots from [this blog](http://rstudio-pubs-static.s3.amazonaws.com/72298_c1ba7f77276a4f27a0f375cadc9fac5d.html).

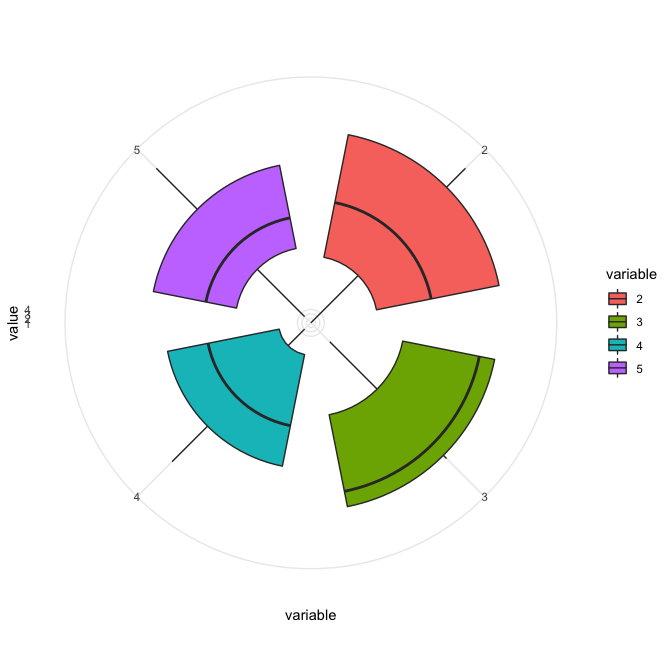
## Radial box plots

To show a radial box plot with a data set and grid with four factor variables and one continuous.

DF3 <- data.frame(name = rep(letters[1:3], length.out = 600),  
 variable = as.factor(sample(5, replace = TRUE)),  
 factor\_variable = rep(letters[4:7], length.out = 600),  
 variable3 = rep(letters[8:16], length.out = 600),  
 value = sample(50, replace = TRUE))

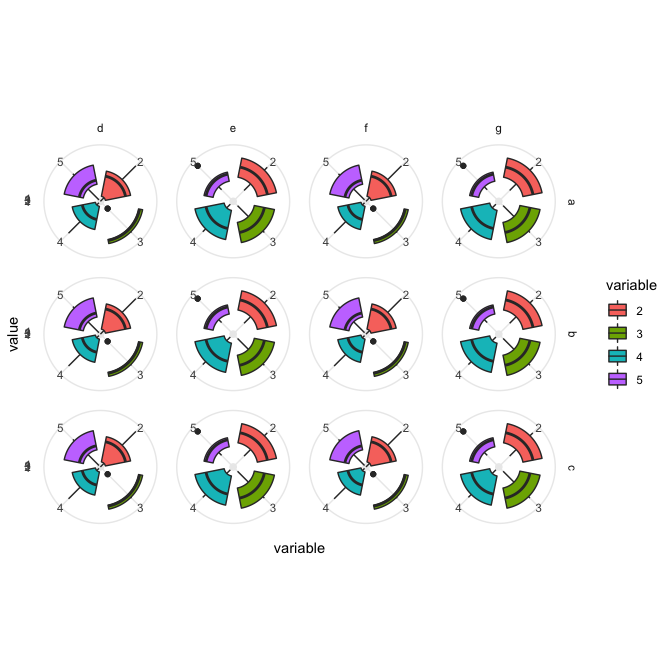
Plot the radial box plot with ggplot2 functions geom\_boxplot() and coord\_polar() (Wickham et al. 2019).

multi\_plot <- ggplot(data = DF3, aes(x=variable, y=value, fill=variable)) +  
 geom\_boxplot() +  
 scale\_y\_continuous(breaks = 0:nlevels(DF3$variable)) +  
 theme\_minimal() +  
 coord\_polar()  
  
#call the plot  
multi\_plot



Radial box plot with rows as names and columns as variables for factor\_variable.

multi\_plot + facet\_grid(name ~ factor\_variable)



### Radial box plots example using ToothGrowth data

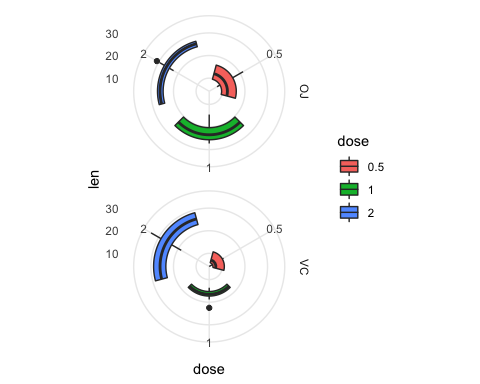
ToothGrowth$dose <- as.factor(ToothGrowth$dose)  
DF4 <- ToothGrowth  
head(DF4)

## len supp dose  
## 1 4.2 VC 0.5  
## 2 11.5 VC 0.5  
## 3 7.3 VC 0.5  
## 4 5.8 VC 0.5  
## 5 6.4 VC 0.5  
## 6 10.0 VC 0.5

box\_plot <- ggplot(DF4, aes(x=dose, y=len, group=dose)) +   
 geom\_boxplot(aes(fill=dose)) +  
 theme\_minimal()+  
 coord\_polar()

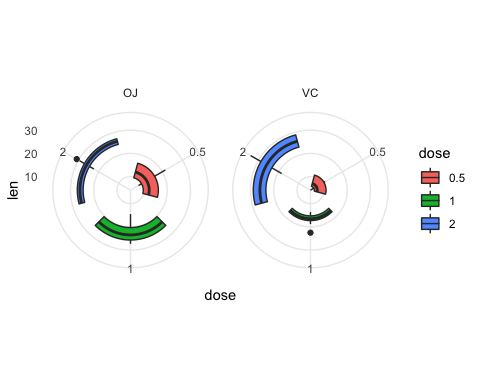
Split the radial box plot vertically

#   
box\_plot + facet\_grid(supp ~ .)



Split the radial box plot horizontally

box\_plot + facet\_grid(. ~ supp)



## Sunburst plot

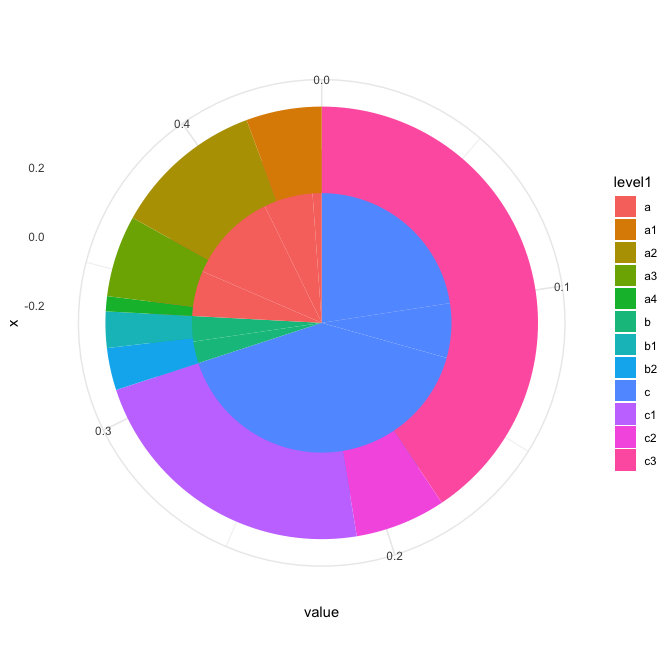
To demonstrate a sunburst-style bar plot confined to a circle we create small data set using data.frame.

Here is a thread about some more helpful options and scripts for making [sunbursts and donut plots](https://stackoverflow.com/questions/26748069/ggplot2-pie-and-donut-chart-on-same-plot).

DF5 <- data.frame(  
 'level1'=c('a', 'a', 'a', 'a', 'b', 'b', 'c', 'c', 'c'),   
 'level2'=c('a1', 'a2', 'a3', 'a4', 'b1', 'b2', 'c1', 'c2', 'c3'),   
 'value'=c(.025, .05, .027, .005, .012, .014, .1, .03, .18))

Create a sunburst-style bar plot confined to a circle

ggplot(DF5, aes(y=value)) +  
 geom\_bar(aes(fill=level1, x=0), width=.5, stat='identity') +   
 geom\_bar(aes(fill=level2, x=.25), width=.25, stat='identity') +   
 coord\_polar(theta='y') +   
 theme\_minimal()



## Spider plot

To demonstrate the spider plot data visualization we create the coord\_radar() function1 to obtain straight lines using match.arg() from base R (R Core Team 2019).

coord\_radar <-   
 function(theta = 'x', start = 0, direction = 1){  
 # input parameter sanity check  
 match.arg(theta, c('x', 'y'))  
   
 ggproto(  
 NULL, CoordPolar,   
 theta = theta, r = ifelse(theta == 'x', 'y', 'x'),  
 start = start, direction = sign(direction),  
 is\_linear = function() TRUE)  
 }

Create a factor, variable, and value to be plotted in the spider plot using base R functions (R Core Team 2019).

factor <- c(rep("A", 16), rep("B", 16))  
variable <- as.factor(c(1:16))  
value <- sample(c(1:10), 32, replace = T)

In order to neatly close the plot we add an empty level to the data set (a quasi-blank variable) which needs the same value as level 1. For this to work both factors (“A” and “B” in our case) need this correction.

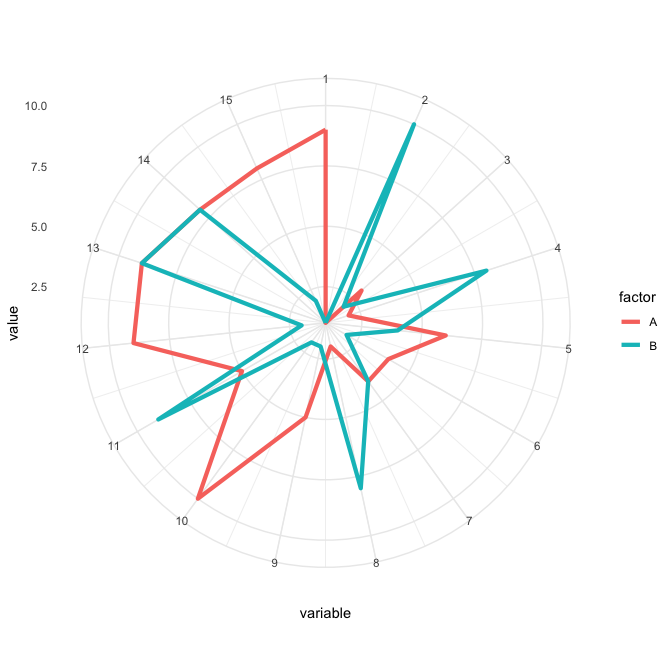
value[16] <- value[1]  
value[32] <- value[17]

We add the factor, variable, and value together with the blank variable to a data set using data.frame.

DF6 <- data.frame(factor = factor, variable = variable, value = value)

Plot with the ggplot2 library (Wickham et al. 2019).

ggplot(DF6, aes(as.numeric(DF6$variable), value, colour = factor)) +   
 coord\_radar() +  
 geom\_path(size = 1.5) + scale\_x\_continuous(breaks = c(1:15)) +  
 labs(x = "variable") +   
 theme\_minimal()



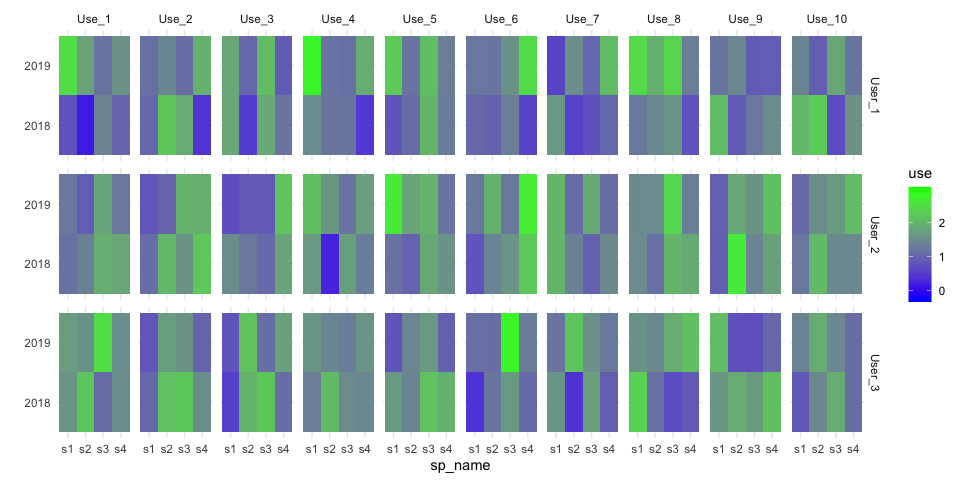
## Heat maps

Heat maps are another way of displaying multidimensional data in a single figure. We use the synthesized data from ethnobotanyR for this heat map example (Whitney 2019).

#create synthesized use data  
eb\_data <- data.frame(replicate(10,sample(rnorm(200, mean=1.5, sd=0.5))))  
names(eb\_data) <- gsub(x = names(eb\_data), pattern = "X", replacement = "Use\_")   
eb\_data$informant <- sample(c('User\_1', 'User\_2', 'User\_3'), 200, replace=TRUE)  
eb\_data$sp\_name <- sample(c('s1', 's2', 's3', 's4'), 200, replace=TRUE)  
eb\_data$year <- sample(c('2018', '2019'), 200, replace=TRUE)

We use the reshape library (Wickham 2018) to melt and geom\_tile() function from ggplot2 to plot the resulting heat map.

#reshape data for the plot  
ethno\_melt <- reshape::melt(eb\_data, id=c("informant","year", "sp\_name"))  
  
ggplot(ethno\_melt, aes(y = factor(year), x = factor(sp\_name))) +   
 geom\_tile(aes(fill = value)) + #heatmap  
 scale\_fill\_continuous(low = "blue", high = "green") + #use model result as color  
 facet\_grid(informant ~ variable) + #grid by factor  
 labs(fill='use') + #legend title  
 theme\_minimal()+  
 xlab("sp\_name") + ylab("")



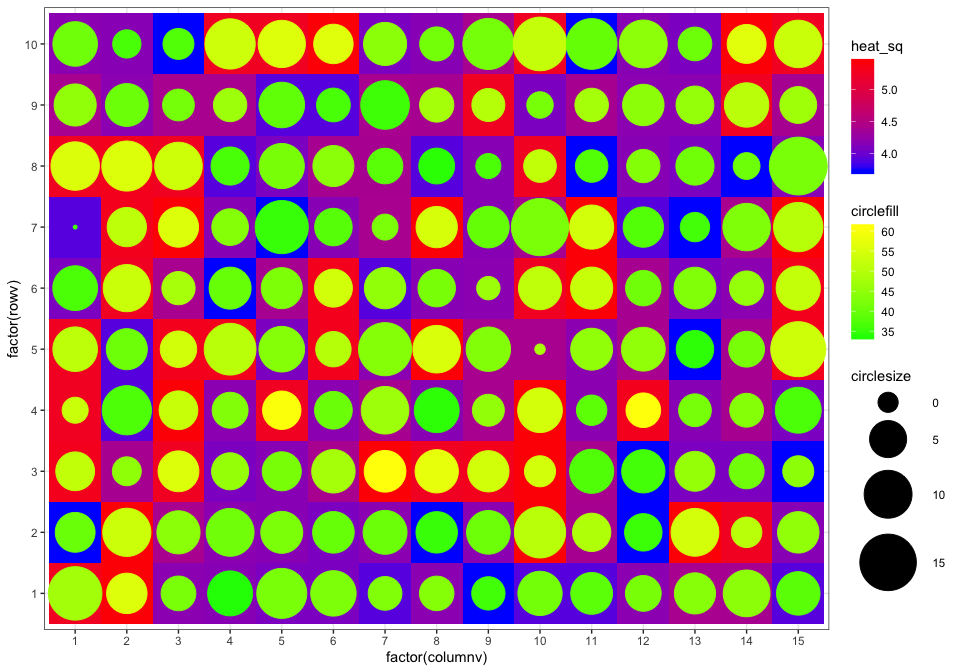
### Bubble graph & heat map

Here we use a combination of a bubble graph and a heat map to show several continuous variables in the same figure. We start by synthesizing data and conditions for bubble sizes and fill.

#set heat bubble parameters  
heat\_sq <- sample(c(rnorm (10, 5,1)), 150, replace = T)  
circlefill <- heat\_sq \* 10 + rnorm (length (heat\_sq), 0, 3)  
circlesize <- heat\_sq \* 1.5 + rnorm (length (heat\_sq), 0, 3)  
  
#synthesize heat bubble data  
D7 <- data.frame (rowv = rep (1:10, 15), columnv = rep(1:15, each = 10),  
 heat\_sq, circlesize, circlefill)

As above we use geom\_tile() from the ggplot2 library to plot this as a heat map. In addition we use geom\_point() to put bubbles on the heat map to more continuous variables by adjusting size and color.

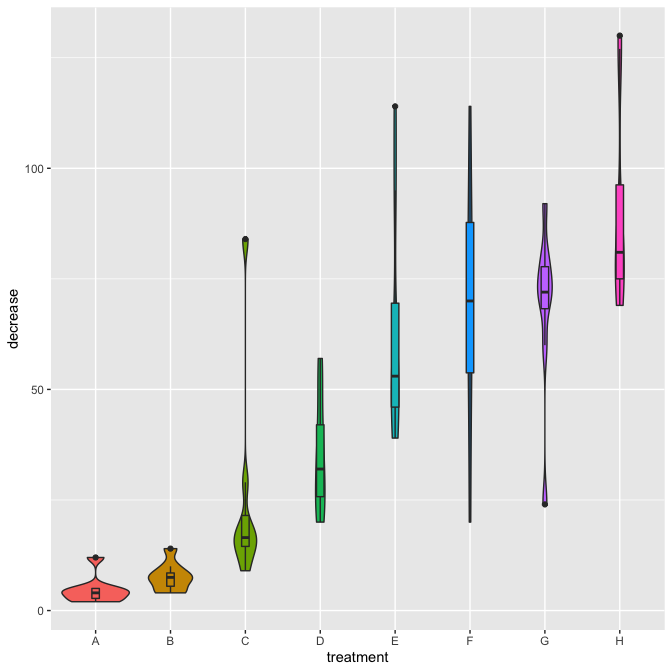
ggplot(D7, aes(y = factor(rowv), x = factor(columnv))) +   
 geom\_tile(aes(fill = heat\_sq)) +   
 scale\_fill\_continuous(low = "blue", high = "red")+   
 geom\_point(aes(colour = circlefill, size =circlesize)) +   
 scale\_color\_gradient(low = "green", high = "yellow")+   
 scale\_size(range = c(1, 20))+   
 theme\_bw()



## Violin & box plot overlays

Here we use the OrchardSprays data to run the example from the tidyverse [Violin plot](https://ggplot2.tidyverse.org/reference/geom_violin.html) examples (Wickham 2017).

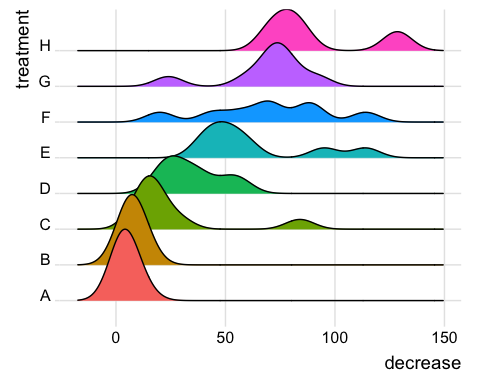
ggplot(OrchardSprays, aes(y=decrease, x=treatment, fill=treatment))+  
 geom\_violin()+  
 geom\_boxplot(width=0.1)+  
 theme(legend.position = "none")



## Ridge line plot

A variation on the example from [edav](https://edav.info/ridgeline.html) using the ggridges library (Wilke 2018).

ggplot(OrchardSprays, aes(x=decrease,y=treatment,fill=treatment))+  
 geom\_density\_ridges\_gradient(scale=2) + theme\_ridges()+  
 theme(legend.position = "none")



More examples on the rdrr.io [CRAN](https://rdrr.io/cran/ggridges/man/geom_ridgeline_gradient.html) website.

## Visualizing uncertainty

Here we demonstrate various graphical options to visualize uncertainty intervals of outcomes of Monte Carlo simulations.

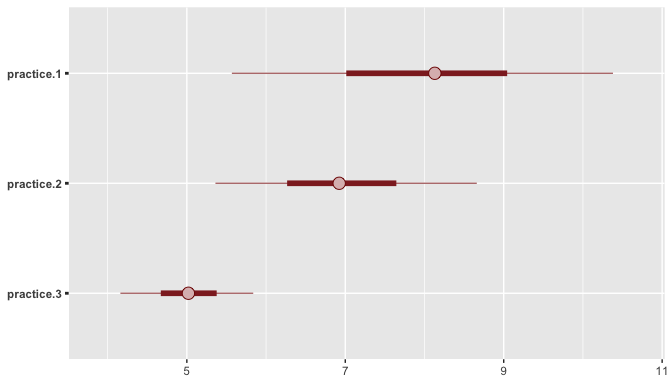
Assume a data set of yield distributions of three different farming practices:

test<- data.frame("practice 1" = rnorm(1000,8,1.5), "practice 2" = rnorm(1000,7,1), "practice 3" = rnorm(1000, 5, 0.5))

We can use the function mcmc\_intervals()or mcmc\_areas()from bayesplot library to plot the data set (Gabry and Mahr 2019)

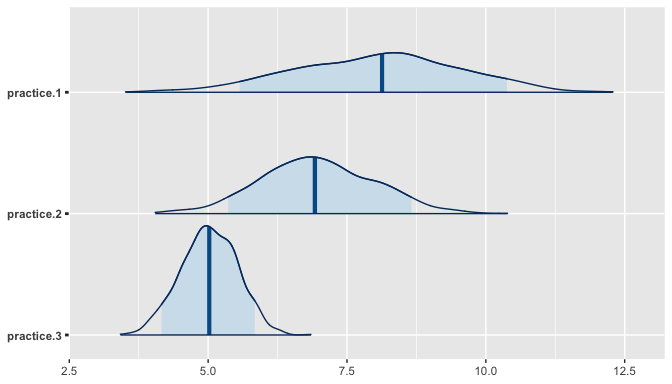
### with mcmc\_intervals()

color\_scheme\_set("red")  
mcmc\_intervals(test,prob = 0.5,prob\_outer = 0.9,point\_est = "median")



### with mcmc\_areas()

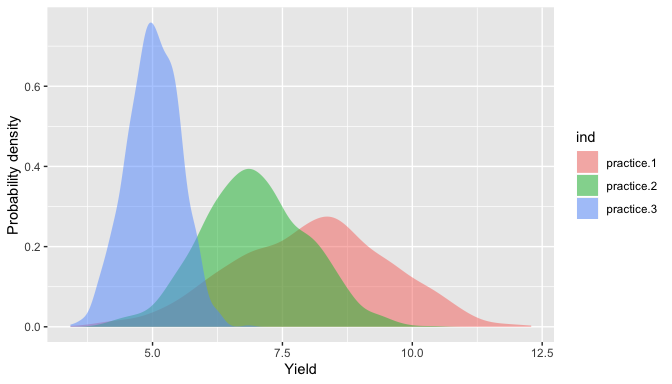
color\_scheme\_set("blue")  
mcmc\_areas(test,prob = 0.9,point\_est = "median")



### Comparative density curves

We can also use geom\_density()in ggplot2 to compare the spread of different distributions (Wickham et al. 2019):

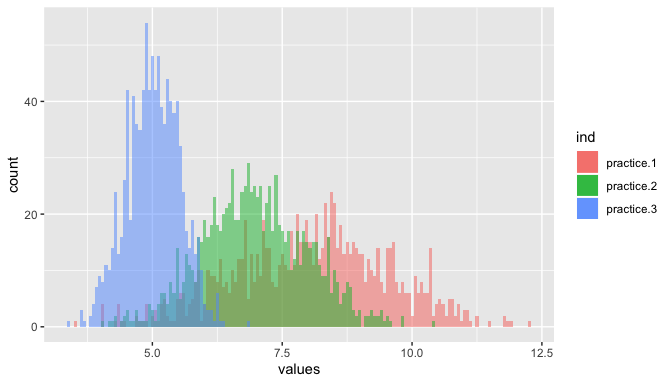
stacked\_test <- stack(test)  
ggplot(stacked\_test,aes(x=values,group=ind,fill=ind ))+  
 geom\_density(colour=NA,alpha=.5)+  
 ylab("Probability density")+  
 xlab("Yield")



### Comparative histogram

Use ggplot2 geom\_histogram()function to show the histogram of the data in comparison:

ggplot(stacked\_test,aes(x=values))+   
 geom\_histogram(data=subset(stacked\_test,ind =='practice.1'),aes(fill = ind), alpha = 0.5, bins = 150)+   
 geom\_histogram(data=subset(stacked\_test,ind == 'practice.2'),aes(fill = ind), alpha = 0.5, bins = 150) +  
 geom\_histogram(data=subset(stacked\_test,ind == 'practice.3'),aes(fill = ind), alpha = 0.5, bins = 150)



## Bar plot

Here demonstrate an option to visualize a set of variables with multiple attributes.

Create an example data frame: Assume that we have five independent variables that are involved in the model to predict yield (above example). After running a regression analysis and performing value of information analysis, we get a data set with three attributes for each variable:

1. VIP score: Show the strength of the relationship.
2. Correlation coefficient: show the direction of the relationship
3. Expected value of perfect information (EVPI): additional gain in yield when having more information on particular variable

ob<-data.frame("variable"=c("variable 1","variable 2","variable 3","variable 4","variable 5"),   
 "Variable\_Importance"=c(1,0.3,0.5,4,2),   
 "Coefficient"=c(-1.5,0.6,-0.2,2.7,0.9),   
 "Value\_of\_information"=c(0.5,0.01,0.6,1,0.7), stringsAsFactors = TRUE)

### Cow plot

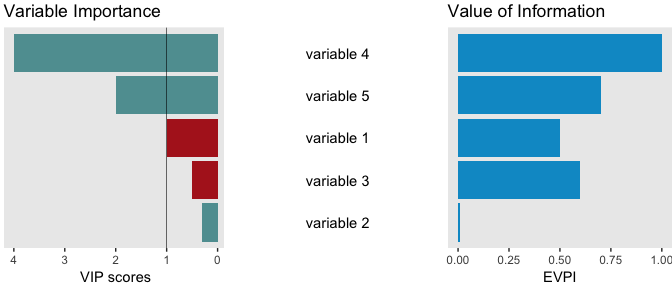
We can use cowplot to represent all the attributes in one single plot.

First we create element plots for the combined plot:

ob$Category[ob$Coefficient > 0] = "cadetblue"  
ob$Category[ob$Coefficient < 0] = "firebrick"  
  
ob$variable <- factor(ob$variable, levels = ob[order(ob$Variable\_Importance),"variable"])  
  
p <- ggplot(ob,aes(x=variable,y=Variable\_Importance))+  
 geom\_bar(aes(fill=ob$Category),stat ="identity")+   
 ggtitle("Variable Importance")+  
 ylab("VIP scores")+  
 xlab(NULL)+  
 scale\_fill\_manual(values = c("cadetblue","firebrick","grey"))+  
 theme(axis.title.y =element\_text(color="black", size=10),   
 axis.text.y = element\_blank(),   
 axis.ticks.y = element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 plot.margin = unit(c(1,-1,1,0), "mm")) +  
 geom\_hline(yintercept = 1, size=0.2)+  
 theme(legend.position = "none")+  
 scale\_y\_reverse() +   
 coord\_flip()  
  
q <- ggplot(data = ob, aes(x = ob$variable, y = ob$Value\_of\_information))+  
 geom\_bar(fill = "deepskyblue3",stat = "identity") +   
 ggtitle("Value of Information") +   
 ylab("EVPI")+   
 xlab(NULL)+  
 theme(axis.title.y = element\_text(color="black", size=10),   
 axis.text.y = element\_blank(),   
 axis.ticks.y = element\_blank(),  
 panel.grid=element\_blank(),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 plot.margin = unit(c(1,0,1,-1), "mm")) +  
 coord\_flip()  
  
g.mid <- ggplot(ob,aes(x=1,y=ob$variable))+geom\_text(aes(label=ob$variable))+  
 geom\_segment(aes(x=0,xend=0,yend=ob$variable))+  
 geom\_segment(aes(x=0,xend=0,yend=ob$variable))+  
 ggtitle("")+  
 ylab(NULL)+  
 scale\_x\_continuous(expand=c(0,0),limits=c(1.0,1.0))+  
 theme(axis.title=element\_blank(),  
 panel.grid=element\_blank(),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 panel.background=element\_blank(),  
 axis.text.x=element\_text(size=10, color=NA),  
 axis.ticks.x=element\_line(size=10, color=NA),  
 plot.margin = unit(c(1,0,1,0), "mm"))  
  
gg1 <- ggplot\_gtable(ggplot\_build(p))  
gg2 <- ggplot\_gtable(ggplot\_build(q))  
gg.mid <- ggplot\_gtable(ggplot\_build(g.mid))

After generating all element plots, use cow plot to put everything together:

cowplot::plot\_grid(gg1,gg.mid,gg2, ncol = 3, align = "h")



## Notes

1The coord\_radar() function was taken from the question “Closing the lines in a ggplot2 radar / spider chart” from stackoverflow website. <https://stackoverflow.com/questions/28898143/closing-the-lines-in-a-ggplot2-radar-spider-chart>

## References

Gabry, Jonah, and Tristan Mahr. 2019. *Bayesplot: Plotting for Bayesian Models*. <https://CRAN.R-project.org/package=bayesplot>.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

Whitney, Cory. 2019. *EthnobotanyR: Calculate Quantitative Ethnobotany Indices*. <https://github.com/CWWhitney/ethnobotanyR>.

Wickham, Hadley. 2017. *Tidyverse: Easily Install and Load the ’Tidyverse’*. <https://CRAN.R-project.org/package=tidyverse>.

———. 2018. *Reshape: Flexibly Reshape Data*. <https://CRAN.R-project.org/package=reshape>.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, and Hiroaki Yutani. 2019. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. <https://CRAN.R-project.org/package=ggplot2>.

Wilke, Claus O. 2018. *Ggridges: Ridgeline Plots in ’Ggplot2’*. <https://CRAN.R-project.org/package=ggridges>.