Visualize high dimensional data

Cory Whitney, Eduardo Fernandez, Thi Hoa Do, Marius Ruett

Contents

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 (R Core Team 2020).

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. 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 2020).

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

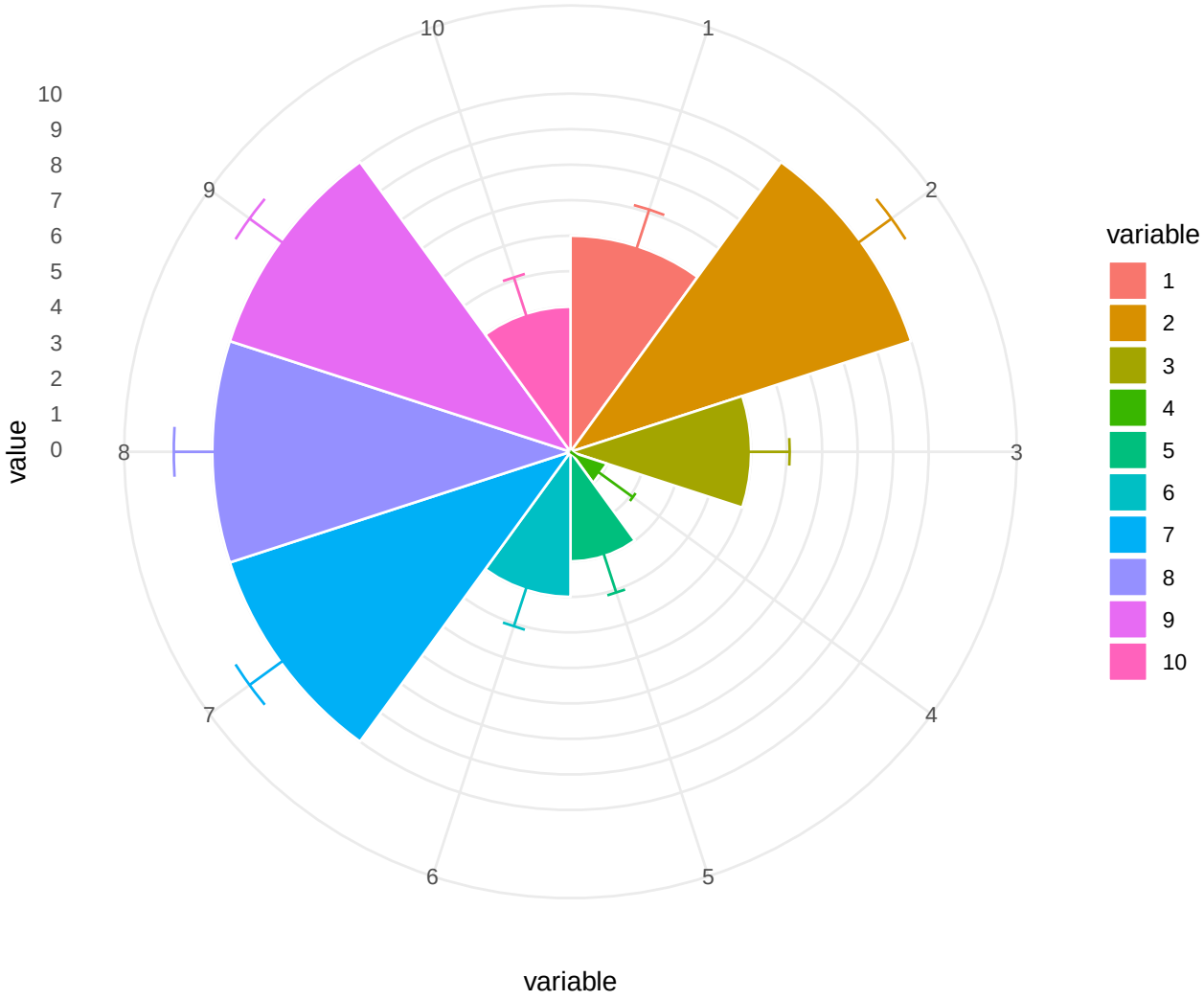
We also created a function to compute the standard error of the mean to represent some of the uncertainty in the data using the `sqrt` and `length` functions in base R and `var` from the `stats` library (R Core Team 2020).

```
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, Chang, et al. 2020).

```
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))
```

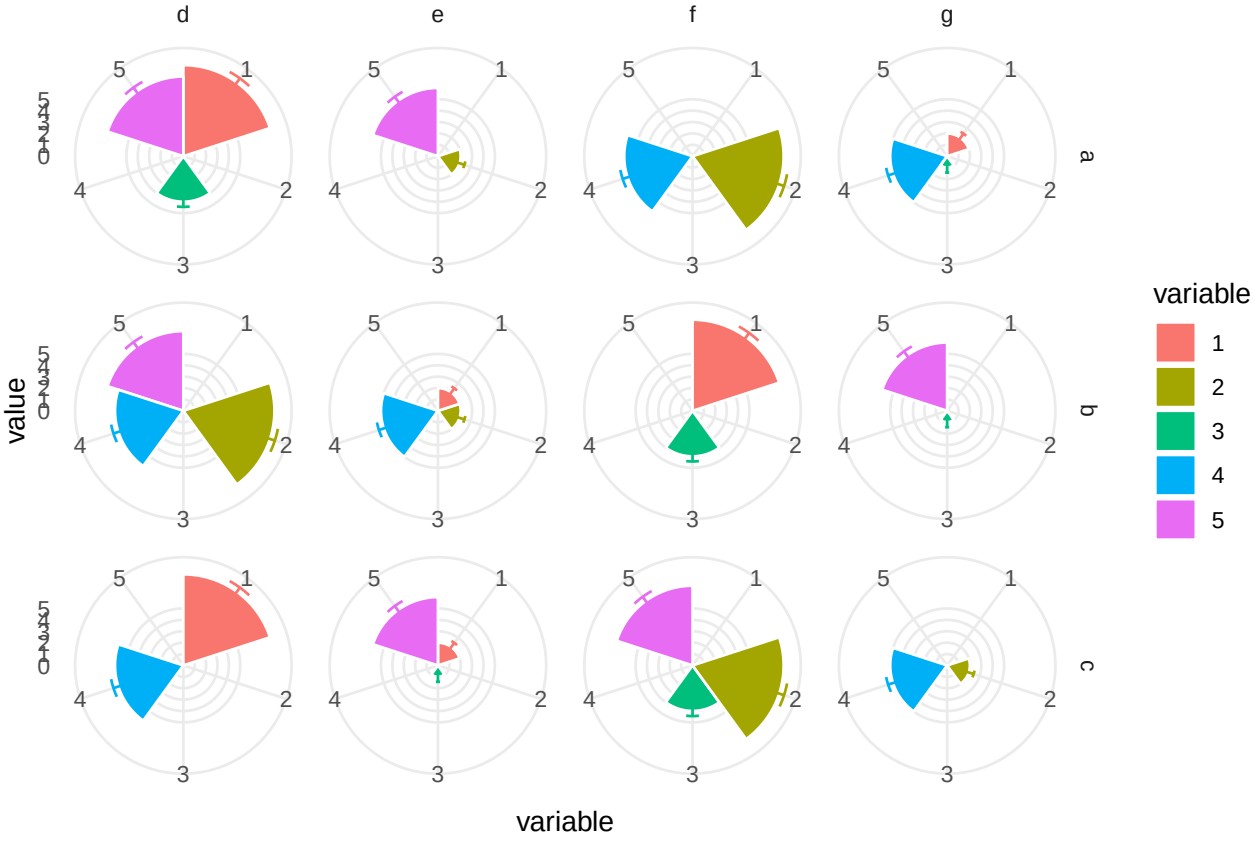
Plot radial plots with three factor variables.

```
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`.

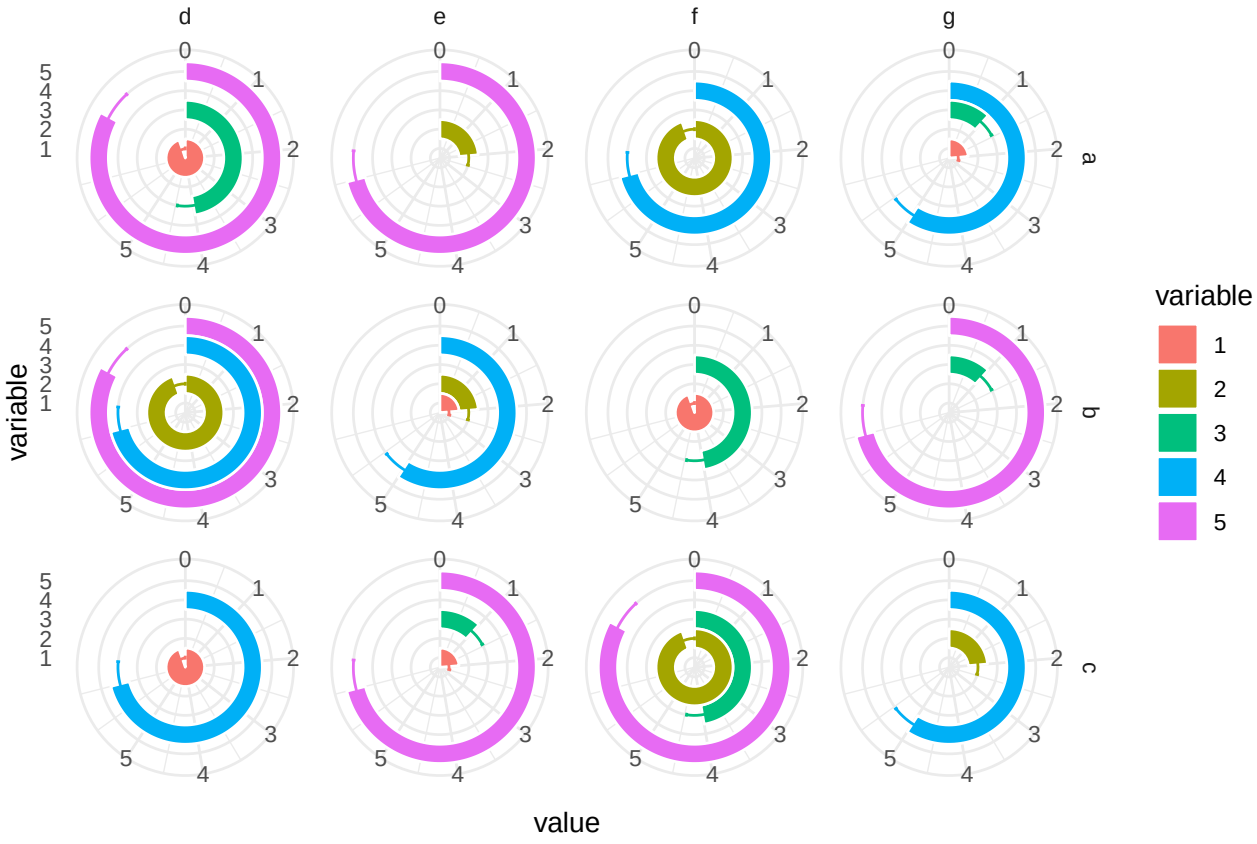
```
# Rows are name and columns are factor_variable
multi_plot + facet_grid(name ~ factor_variable)
```



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.

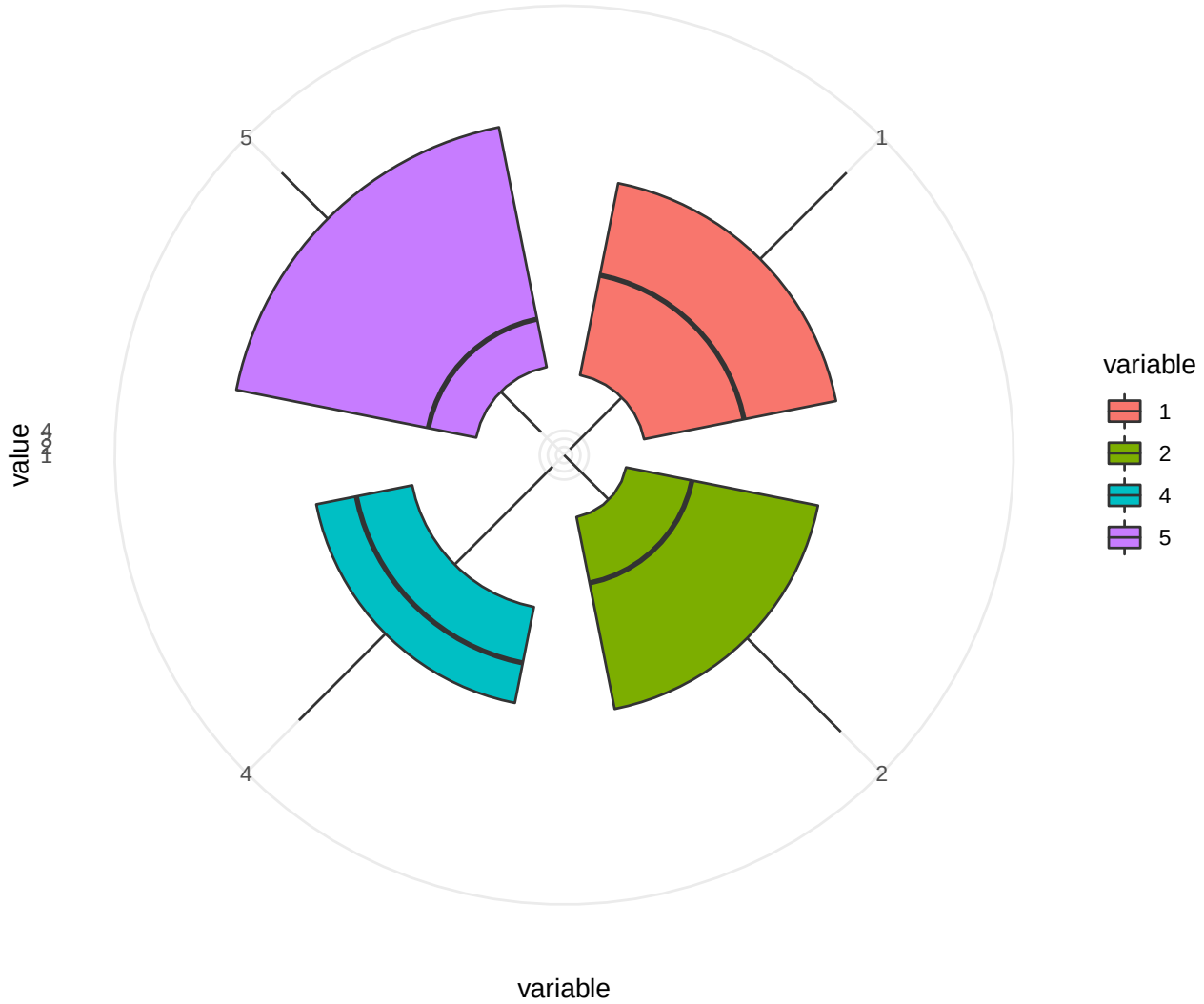
Radial box plots

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))
```

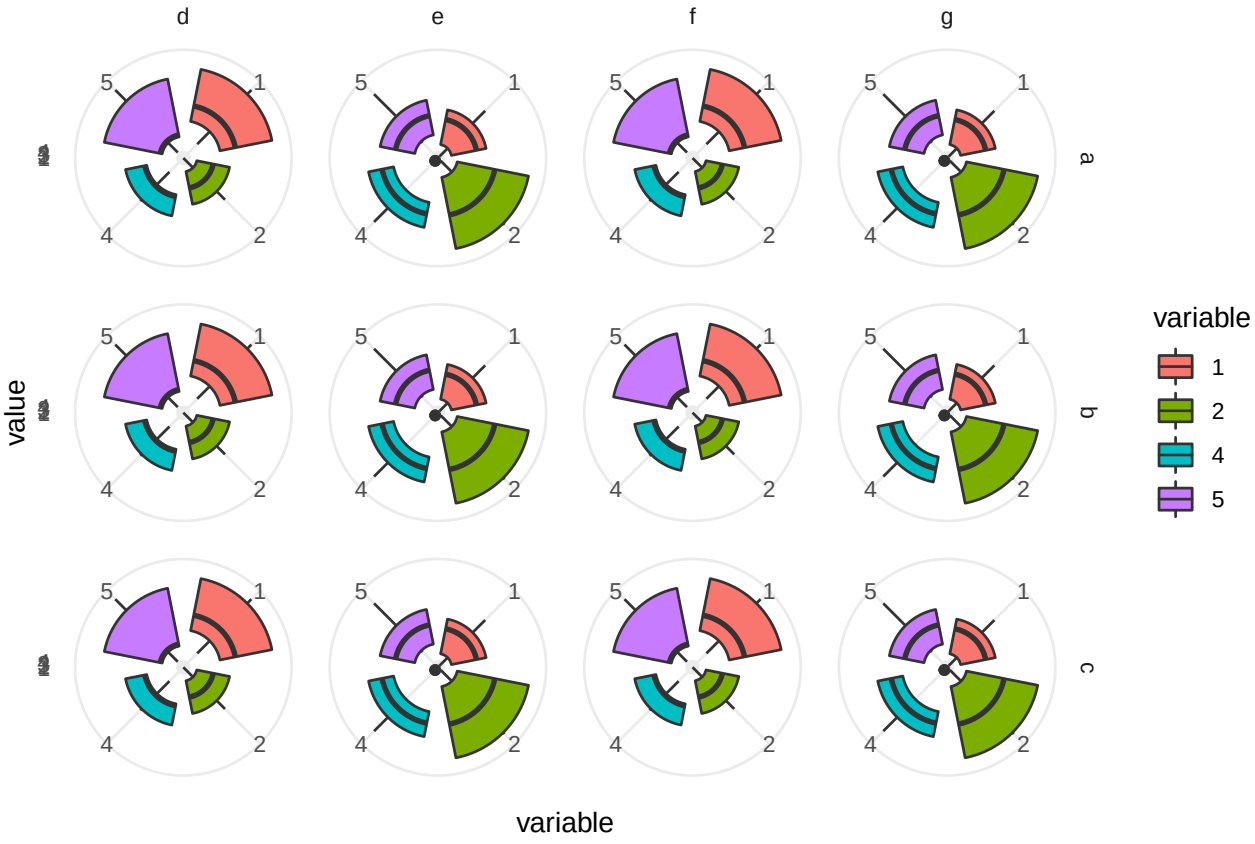
Plot the radial box plot with `ggplot2` functions `geom_boxplot()` and `coord_polar()` (Wickham, Chang, et al. 2020).

```
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

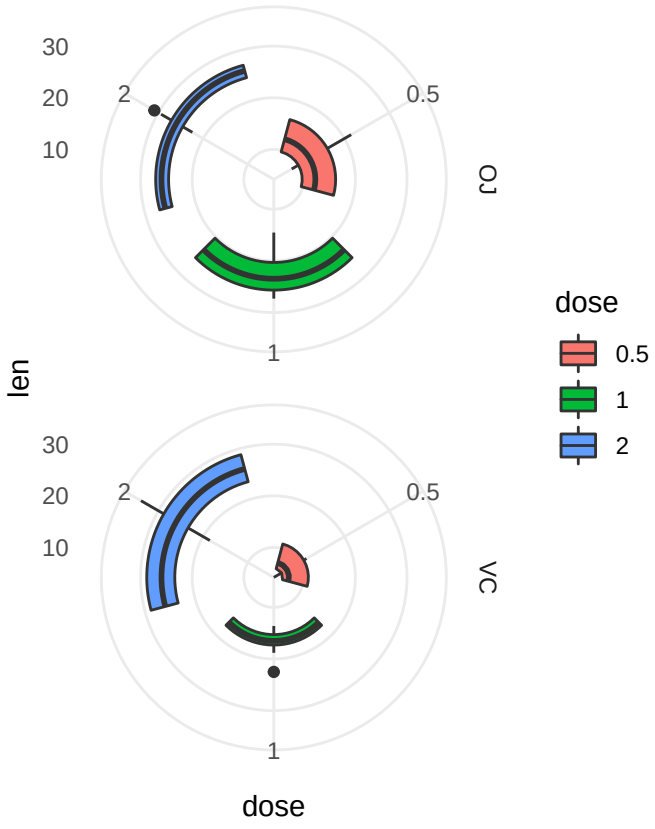
```
ToothGrowth$dose <- as.factor(ToothGrowth$dose)
DF4 <- ToothGrowth
head(DF4)
```

```
##    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()
```

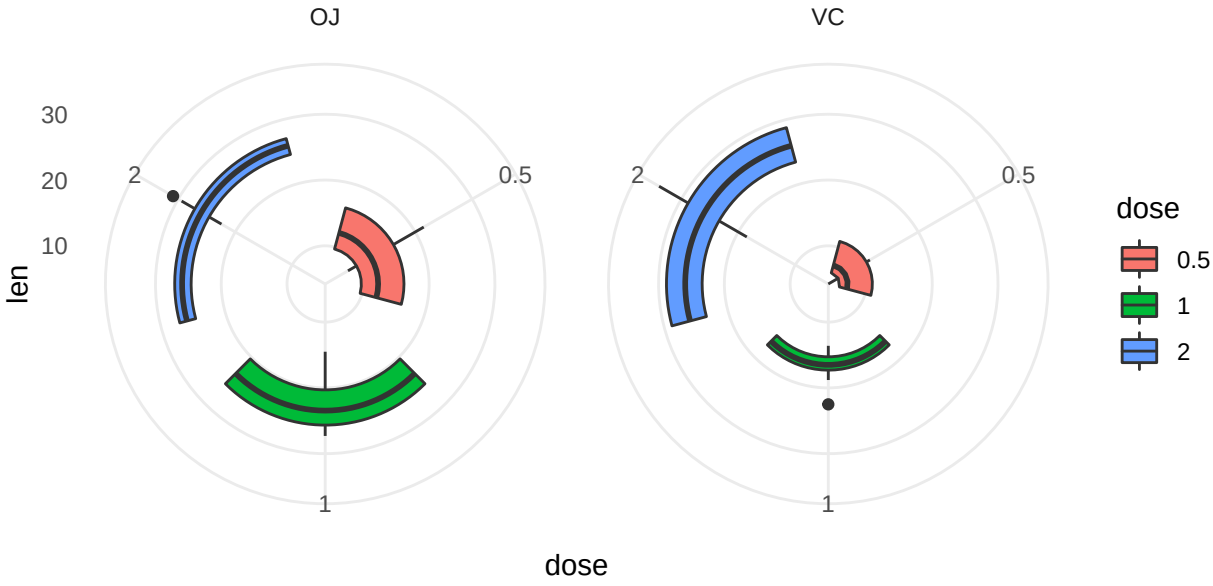
Split the radial box plot vertically

```
#
box_plot + facet_grid(supp ~ .)
```



Split the radial box plot horizontally

```
box_plot + facet_grid(. ~ supp)
```



Sunburst plot

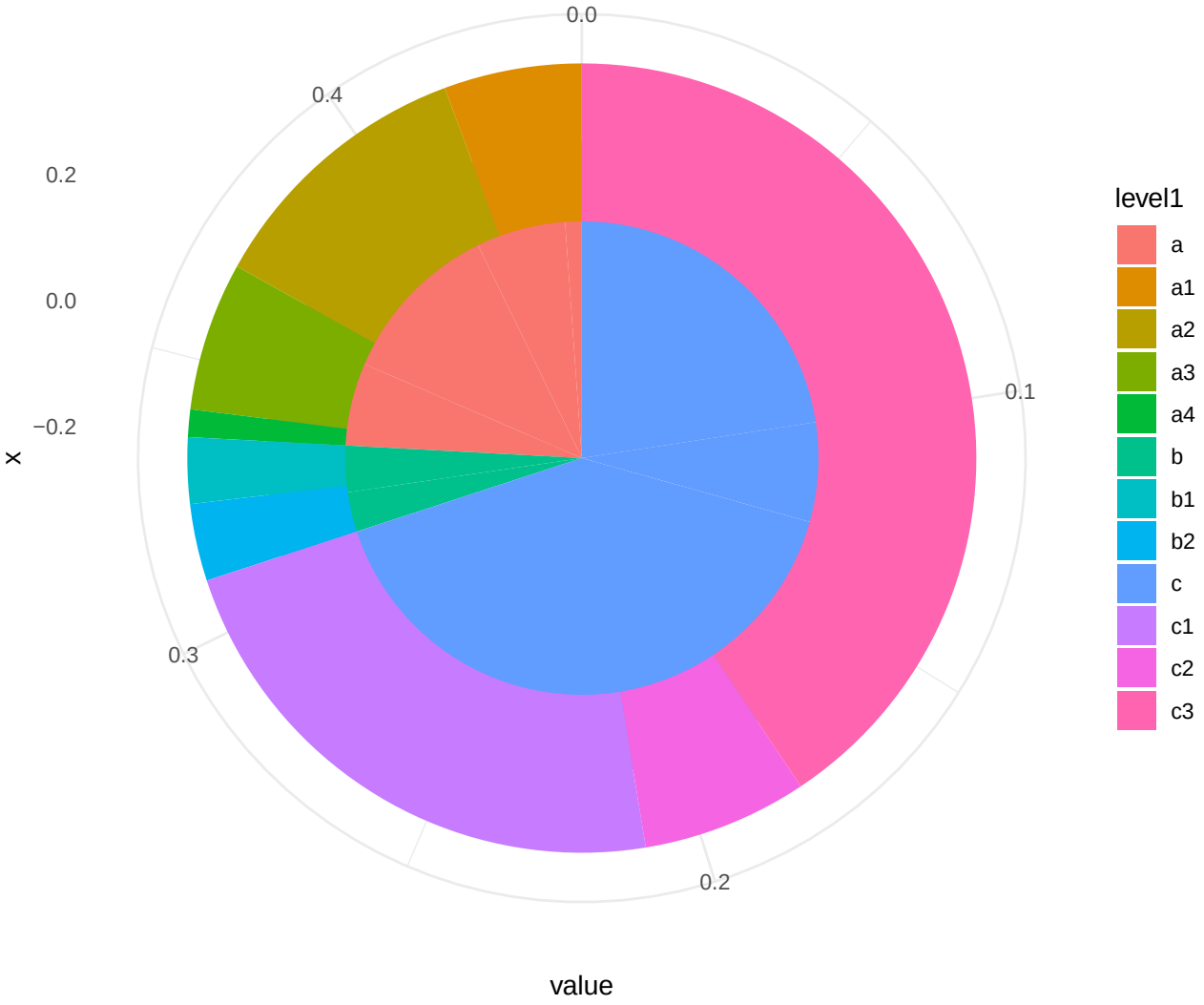
To demonstrate a sunburst-style bar plot confined to a circle we create small data set using `data.frame`.

Here is a thread about some more helpful options and scripts for making sunbursts and donut plots.

```
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 2020).

```
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 2020).

```
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.

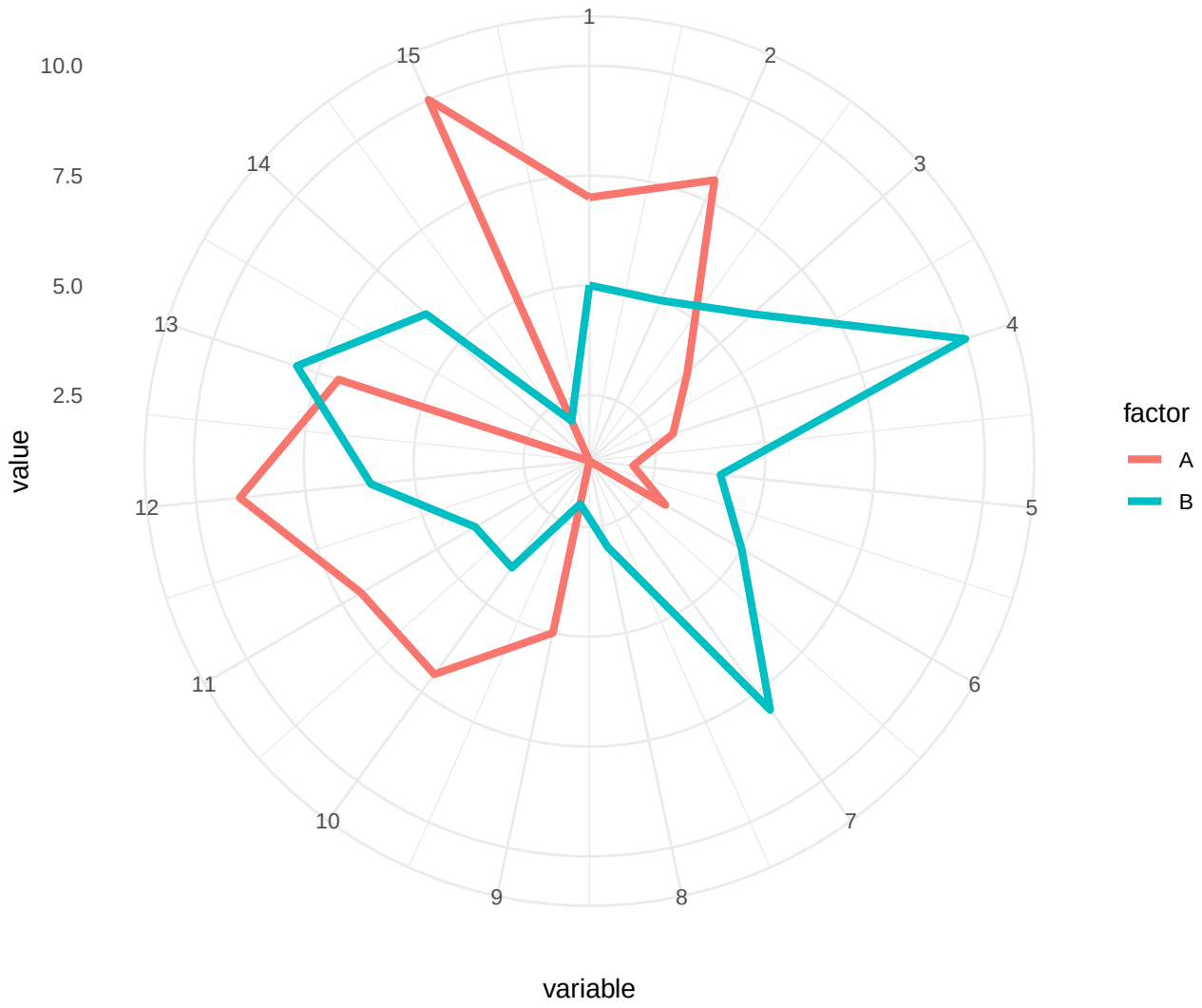
```
value[16] <- value[1]
value[32] <- value[17]
```

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, Chang, et al. 2020).

```
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

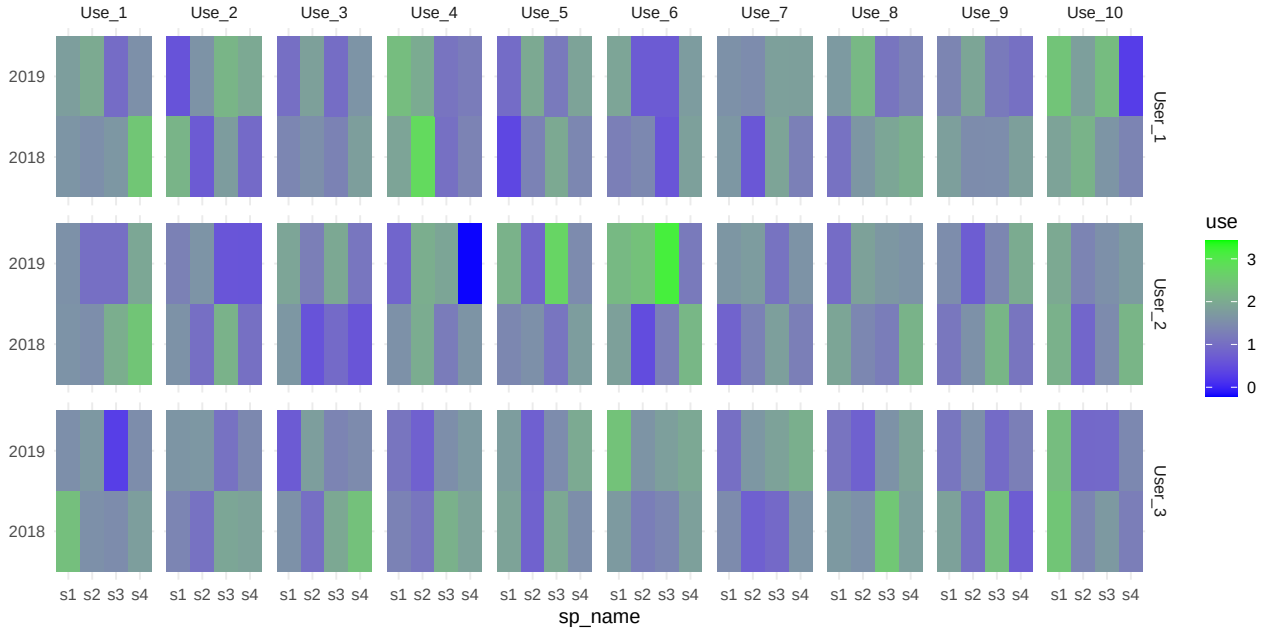
Heat maps are another way of displaying multidimensional data in a single figure. We use the synthesized data from `ethnobotanyR` for this heat map example (Whitney 2021).

```
#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("")
```



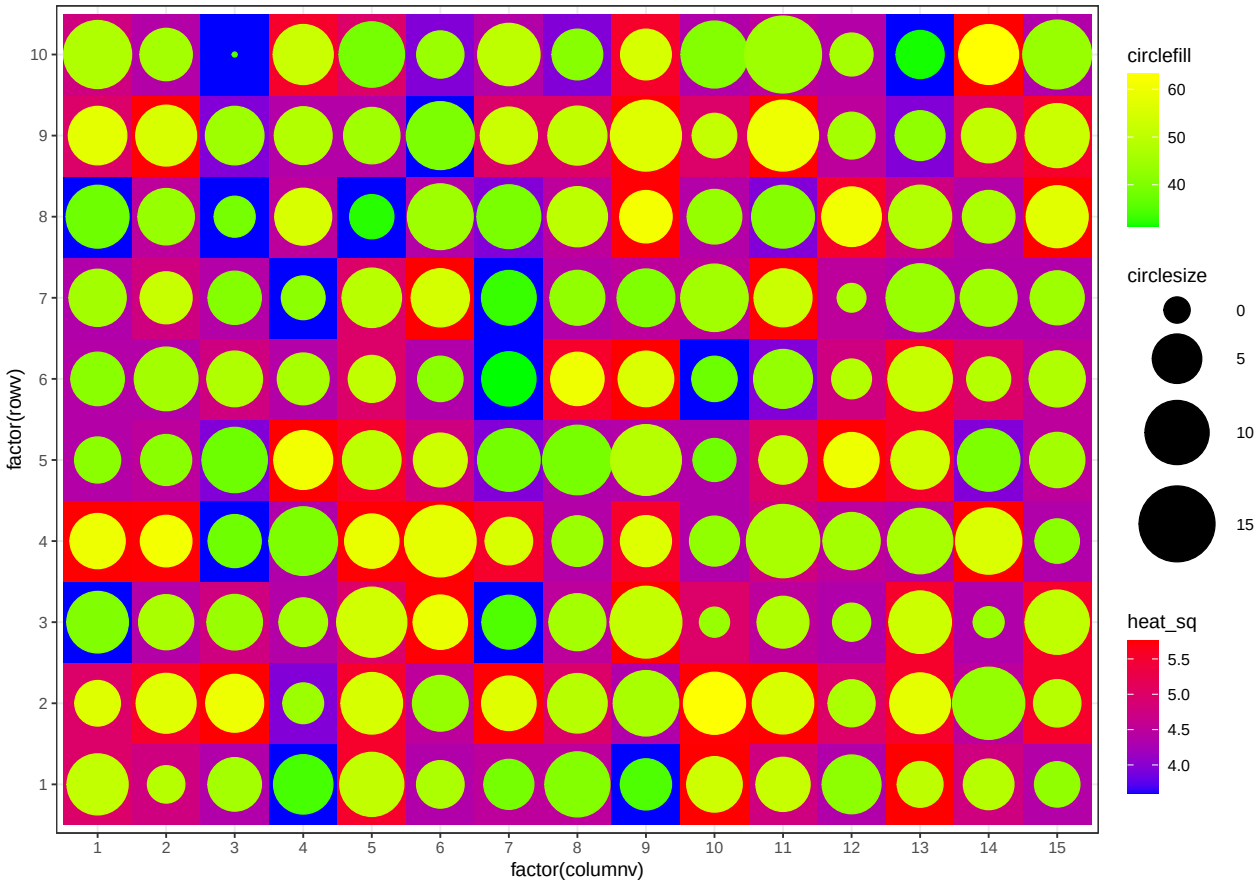
Bubble graph & heat map

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.

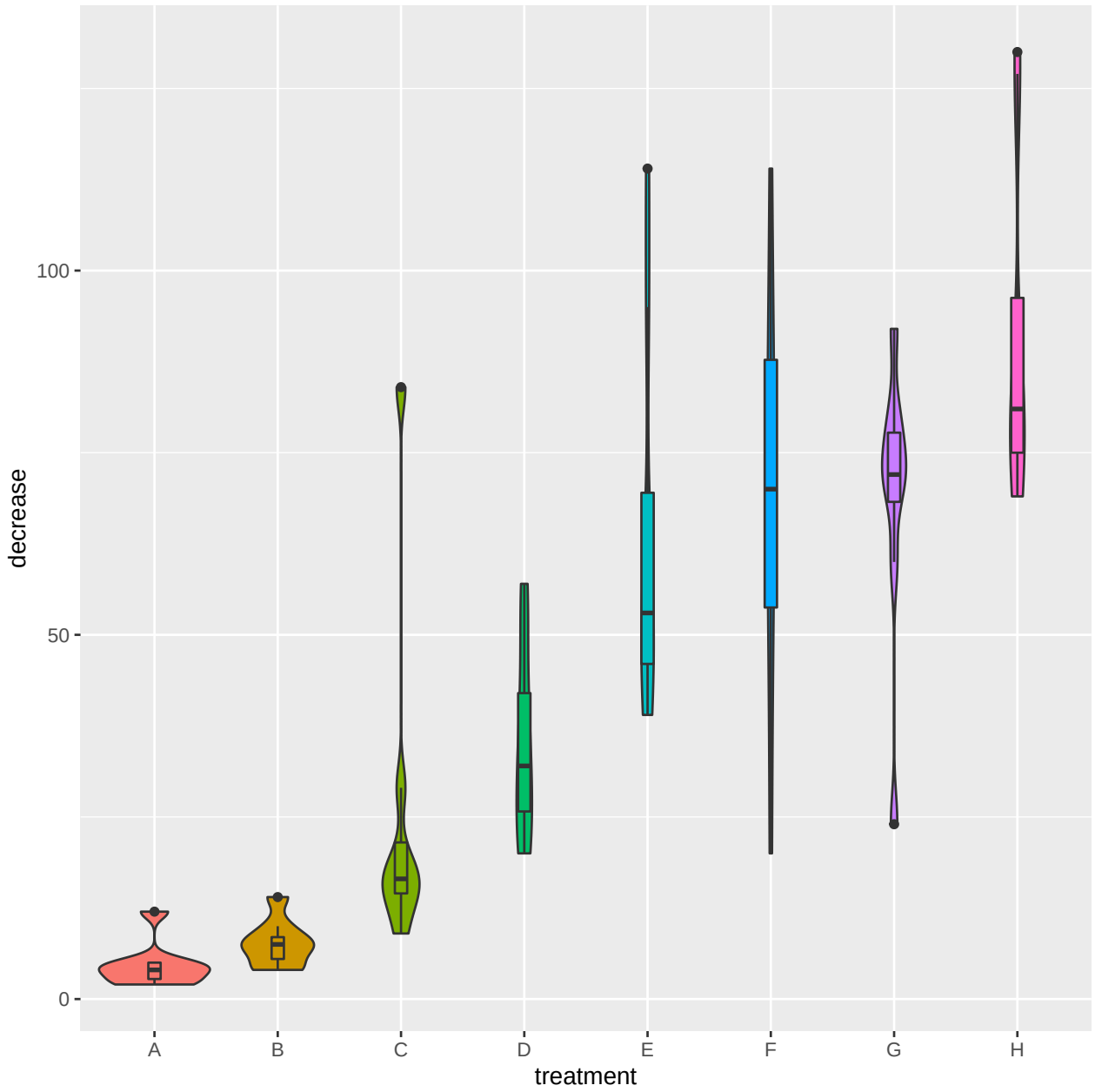
```
ggplot(D7, aes(y = factor(rowv),  x = factor(columnv))) +
  geom_tile(aes(fill = heat_sq)) +
  scale_fill_continuous(low = "blue", high = "red")+
  geom_point(aes(colour = circlefill,  size =circlesize))  +
  scale_color_gradient(low = "green",   high = "yellow")+
  scale_size(range = c(1, 20))+
  theme_bw()
```



Violin & box plot overlays

Here we use the `OrchardSprays` data to run the example from the `tidyverse` Violin plot examples (Wickham 2019).

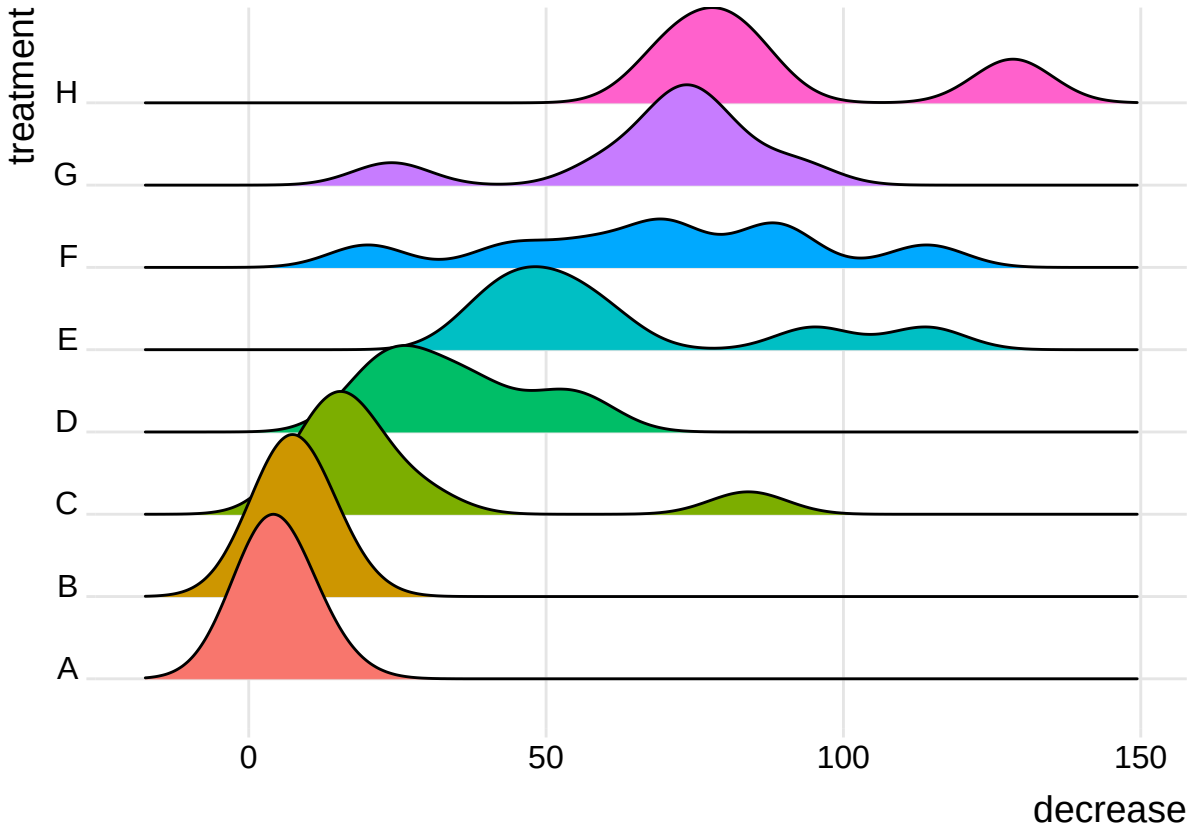
```
ggplot(OrchardSprays, aes(y=decrease, x=treatment, fill=treatment))+
  geom_violin()+
  geom_boxplot(width=0.1)+
  theme(legend.position = "none")
```



Ridge line plot

A variation on the example from edav using the `ggridges` library (Wilke 2020).

```
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 website.

Visualizing uncertainty

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

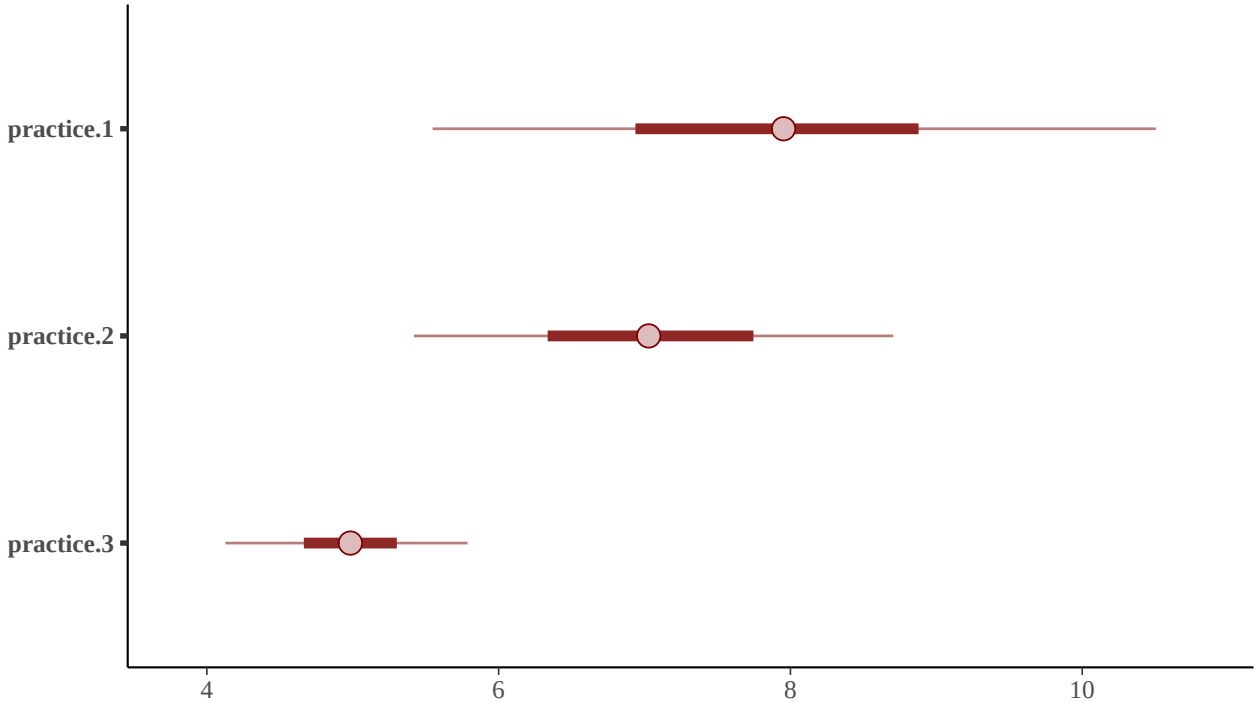
Assume a data set of yield distributions of three different farming practices:

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

We can use the function `mcmc_intervals()`or `mcmc_areas()`from `bayesplot` library to plot the data set (Gabry and Mahr 2021).

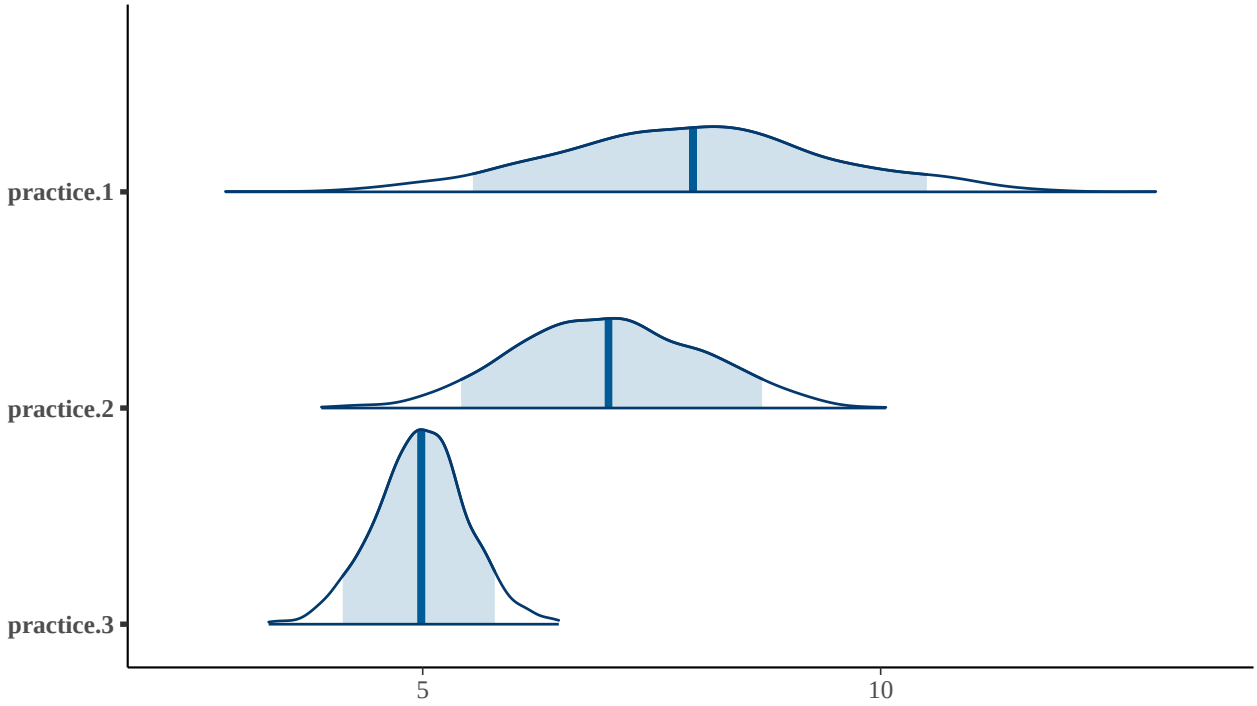
with `mcmc_intervals()`

```
color_scheme_set("red")
mcmc_intervals(test,prob = 0.5,prob_outer = 0.9,point_est = "median")
```



with `mcmc_areas()`

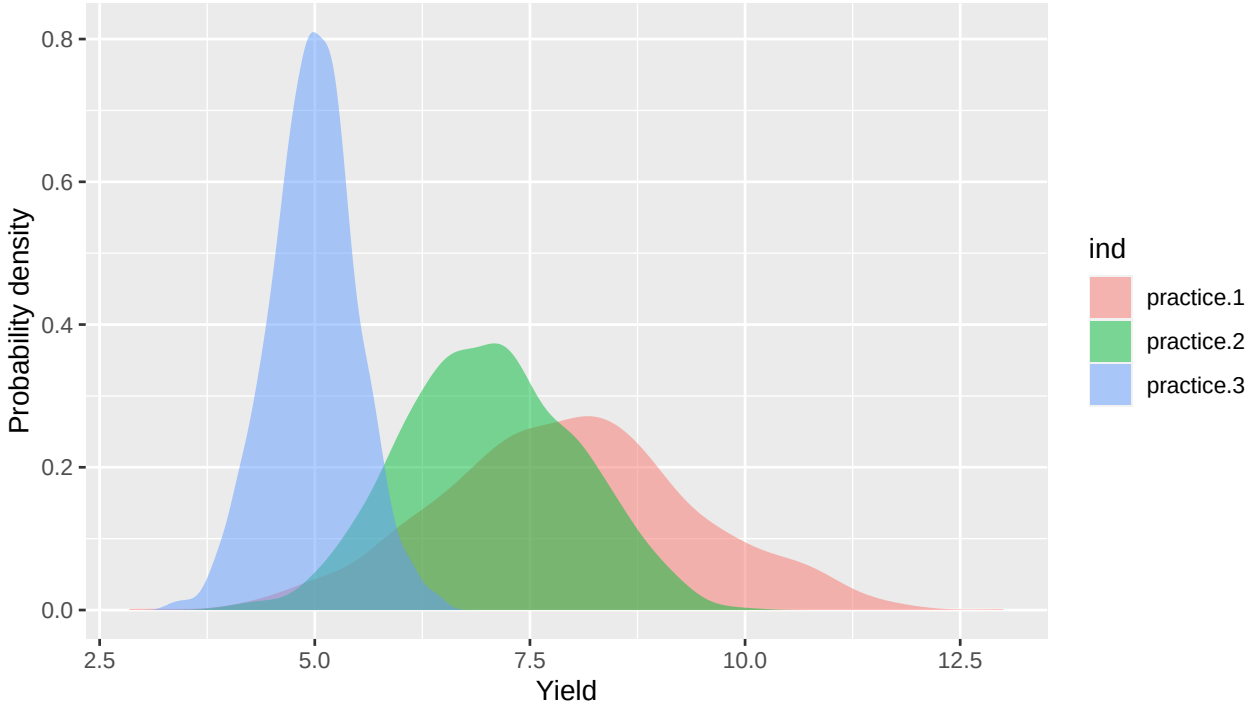
```
color_scheme_set("blue")
mcmc_areas(test,prob = 0.9,point_est = "median")
```



Comparative density curves

We can also use `geom_density()`in `ggplot2` to compare the spread of different distributions (Wickham, Chang, et al. 2020):

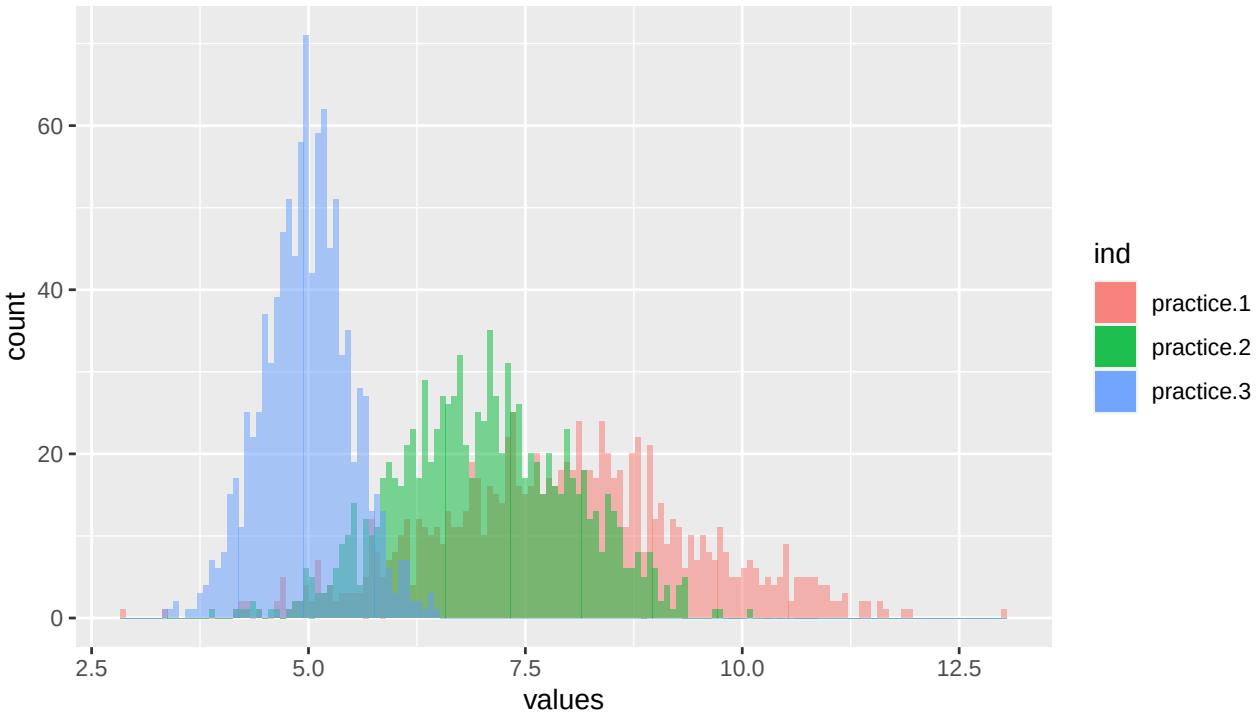
```
stacked_test <- stack(test)

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.

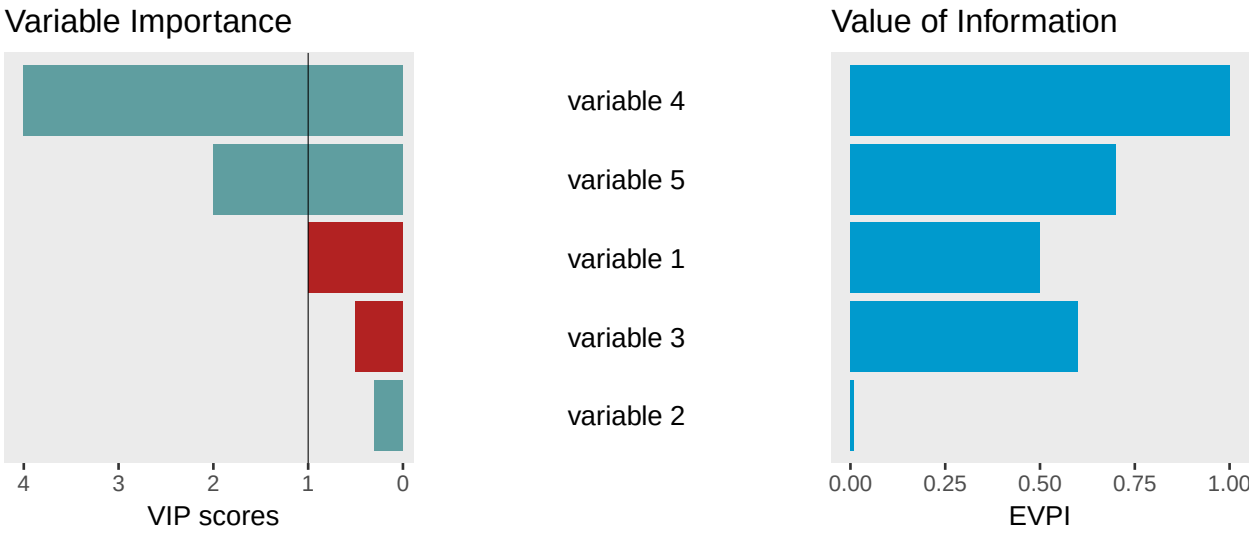
First we create element plots for the combined plot:

```
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")
```



Box plot probability distributions

Here we plot partially transparent probability distributions with box plots in the foreground. Input data is an example such as that from a Monte Carlo simulation in the `decisionSupport` package (Luedeling, Goehring, and Schiffers 2020).

```
test.data <- data.frame("decision1" = rnorm(1000,150,20), "decision2" = rnorm(1000,120,20), "decision3"
```

Use `dplyr` (Wickham, François, et al. 2020) to select data for the various hypothetical decisions and the `geom_boxploth` and `geom_density` functions of `ggplot2` to generate plots (**???**):

Option 1

```
data_plot_decision1 <-
  dplyr::select(test.data, starts_with("decision1")) %>%
  stack(drop = FALSE)

data_plot_decision1$values <- as.numeric(data_plot_decision1$values)

plot_decision1 <-
  ggplot(data_plot_decision1,
         aes(x = values, y = ind, fill = ind)) +
  geom_density(aes(y = ..count..), alpha = 0.5) +
  scale_fill_manual(
    labels = ("Decision1"),
    values = ("blue3") ,
    guide = "legend") +
  geom_boxploth(aes(x = values, y = 2.5),
                width = 5, fill = "blue3") +
  scale_x_continuous(
    labels = scales::dollar_format(suffix = "Euro",
                                   prefix = ""),
    limits = c(0, 280)) +
  ylim(breaks = c(0, 24)) +
  annotate("text", x = 10, y = 15,
    label = 'atop(bold("Decision1"))',
    size = 4,
    parse = TRUE) +
```

```
  theme(
    axis.title.x = element_text(color = "white"),
    axis.title.y = element_text(),
    axis.text.x = element_text(color = "white"),
    axis.ticks.x = element_line(),
    legend.title = element_blank(),
    legend.position = "none",
    legend.text = element_text(size = 12),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12, face = "bold"),
    axis.line = element_line(colour = "black"),
    panel.background = element_blank()) +
  xlab("Net benefit") +
  ylab("Frequency")
```

Option 2

```
data_plot_decision2 <-
  dplyr::select(test.data, starts_with("decision2")) %>%
  stack(drop = FALSE)
data_plot_decision2$values <- as.numeric(data_plot_decision2$values)

plot_decision2 <-
  ggplot(data_plot_decision2, aes(x = values, y = ind, fill = ind)) +
  geom_density(aes(y = ..count..), alpha = 0.5) +
  scale_fill_manual(
    labels = ("Decision2"),
    values = ("red3") ,
    guide = "legend") +
  geom_boxploth(aes(x = values, y = 2.5), width = 5, fill = "red3") +
  scale_x_continuous(
    labels = scales::dollar_format(suffix = "Euro", prefix = ""),
    limits = c(0, 280)) +
  ylim(breaks = c(0, 24)) +
  annotate(
    "text",
    x = 10,
    y = 15,
    label = 'atop(bold("Decision2"))',
    size = 4,
    parse = TRUE) +
  theme(
    axis.title.x = element_text(color = "white"),
    axis.title.y = element_text(),
    axis.text.x = element_text(color = "white"),
    axis.ticks.x = element_line(),
    legend.title = element_blank(),
    legend.position = "none",
    legend.text = element_text(size = 12),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12, face = "bold"),
    axis.line = element_line(colour = "black"),
    panel.background = element_blank()) +
  xlab("Net benefit") +
```

```
  ylab("Frequency")
```

Option 3

```
data_plot_decision3 <-
  dplyr::select(test.data, starts_with("decision3")) %>%
  stack(drop = FALSE)
data_plot_decision3$values <- as.numeric(data_plot_decision3$values)

plot_decision3 <-
  ggplot(data_plot_decision3, aes(x = values, y = ind, fill = ind)) +
  geom_density(aes(y = ..count..), alpha = 0.5) +
  scale_fill_manual(
    labels = ("Decision3"),
    values = ("green3") ,
    guide = "legend") +
  geom_boxploth(aes(x = values, y = 2.5), width = 5, fill = "green3") +
  scale_x_continuous(
    labels = scales::dollar_format(suffix = "Euro", prefix = ""),
    limits = c(0, 280)) +
  ylim(breaks = c(0, 24)) +
  annotate(
    "text",
    x = 10,
    y = 15,
    label = 'atop(bold("Decision3"))',
    size = 4,
    parse = TRUE) +
  theme(
    axis.title.x = element_text(color = "white"),
    axis.title.y = element_text(),
    axis.text.x = element_text(color = "white"),
    axis.ticks.x = element_line(),
    legend.title = element_blank(),
    legend.position = "none",
    legend.text = element_text(size = 12),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12, face = "bold"),
    axis.line = element_line(colour = "black"),
    panel.background = element_blank()) +
  xlab("Net benefit") +
  ylab("Frequency")
```

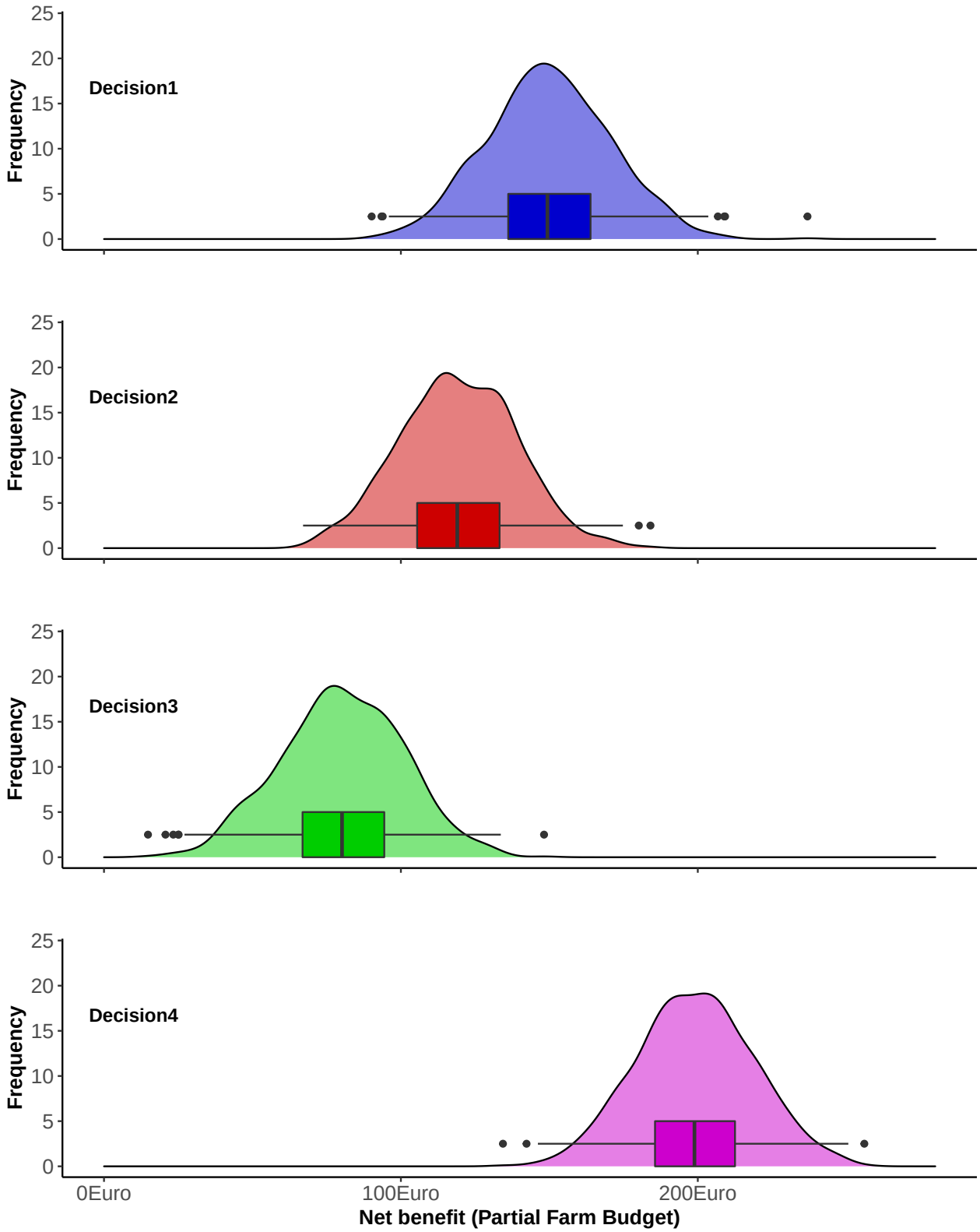
Option 4

```
data_plot_decision4 <-
  dplyr::select(test.data, starts_with("decision4")) %>%
  stack(drop = FALSE)

data_plot_decision4$values <- as.numeric(data_plot_decision4$values)
plot_decision4 <-
  ggplot(data_plot_decision4, aes(x = values, y = ind, fill = ind)) +
  geom_density(aes(y = ..count..), alpha = 0.5) +
  scale_fill_manual(
    labels = ("Decision4"),
```

```
    values = ("magenta3") ,
    guide = "legend") +
  geom_boxploth(aes(x = values, y = 2.5), width = 5, fill = "magenta3") +
  scale_x_continuous(
    labels = scales::dollar_format(suffix = "Euro", prefix = ""),
    limits = c(0, 280)) +
  ylim(breaks = c(0, 24)) +
  annotate(
    "text",
    x = 10,
    y = 15,
    label = 'atop(bold("Decision4"))',
    size = 4,
    parse = TRUE) +
  theme(
    axis.title.x = element_text(),
    axis.title.y = element_text(),
    legend.title = element_blank(),
    axis.ticks.x = element_line(),
    legend.position = "none",
    legend.text = element_text(size = 12),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12, face = "bold"),
    axis.line = element_line(colour = "black"),
    panel.background = element_blank()
  ) +
  xlab("Net benefit (Partial Farm Budget)") +
  ylab("Frequency")
```

Merge data into overlays with box plots.

```
merged_distribution_boxplots <-
  ggarrange(
    plot_decision1,
    plot_decision2,
    plot_decision3,
    plot_decision4,
    ncol = 1,
    nrow = 4
  )

merged_distribution_boxplots
```



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. 2021. *Bayesplot: Plotting for Bayesian Models*. https://mc-stan.org/bayesplot/.

Luedeling, Eike, Lutz Goehring, and Katja Schiffers. 2020. *DecisionSupport: Quantitative Support of Decision Making Under Uncertainty*. http://www.worldagroforestry.org/.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Whitney, Cory. 2021. *EthnobotanyR: Calculate Quantitative Ethnobotany Indices*. https://CRAN.R-project.org/package=ethnobotanyR.

Wickham, Hadley. 2018. *Reshape: Flexibly Reshape Data*. http://had.co.nz/reshape.

———. 2019. *Tidyverse: Easily Install and Load the Tidyverse*. https://CRAN.R-project.org/package=tidyverse.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2020. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2020. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wilke, Claus O. 2020. *Ggridges: Ridgeline Plots in Ggplot2*. https://wilkelab.org/ggridges.