Cheat Sheet

Yuehan Hu yh3291 & Wen Chen cw3229

Part I: Introduction to ggplot2

> Creating ggplot2

> Create a single line plot


```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_line()
```

· **ggplot()** creates a canvas to draw on.

· **data** is the data frame containing data for the plot. It contains columns named x_column and y_column.

· **aes()** matches columns of data to the aesthetics of the plot. Here, x_column is used for the x-axis and y_column for the y-axis.

· **geom_line()** adds a line geometry. That is, it draws a line plot connecting each data point in the dataset.

Geometries are visual representations of the data.
Attributes are fixed values of visual properties of geometries.
Aesthetics are values of visual properties of geometries that depend on data values.

> common aesthetic

· **x** set or map the x-axis coordinate

· **y** set or map the x-axis coordinate

· color set or map the color or edge color

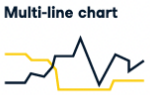
· fill set or map the interior (fill) color

· size set or map the size or width

· alpha set or map the transparency

> The most common visualizations

> Create a multi-line plot




```
ggplot(data, aes(x_column, y_column, color = color_column)) +
        geom_line()
```

Note: Swap the color aesthetic for the group aesthetic to make all lines the same color.

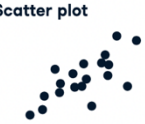
> Create an area chart


```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_area()
```

> Create a stacked area chart


```
ggplot(data, aes(x = x_column, y = y_column, fill=z_column)) +
        geom_area()
```

> Create a scatter plot



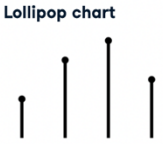

```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_point()
```

> Create a bar plot


```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_col()
```

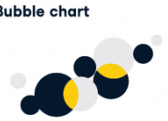
Note: Swap `geom_col()` **for** `geom_bar()` **to calculate the bar heights from counts of the x values.**

> Create a lollipop chart




```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_point() +
        geoxm_segment(aes(x = x_column, xend =  x_column, y = 0,
                          yend = y_column))
```

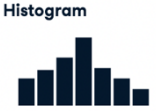
> Create a bubble plot




```
ggplot(data, aes(x = x_column, y = y_column, size =
        size_column)) +
        geom_point(alpha = 0.7) +
        scale_size_area()
```

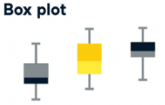
Note: In a bubble plot, " bubbles" can overlap, which can be solved by adjusting the transparency attribute, alpha. scale_size_area() **makes the points to be proportional to the values in** size_column**.**

> Create a histogram




```
ggplot(data, aes(x_column)) +
        geom_histogram(bins = 15)
```

> Create a box plot




```
ggplot(data, aes(x = x_column, y = y_column)) +
        geom_boxplot()
```

> Create a violin plot


```
ggplot(data, aes(x = x_column, y = y_column, fill =
        z_value)) +
        geom_violin()
```

> Create a density plot


```
ggplot(data, aes(x = x_column)) +
        geom_density()
```

Part II: ggalluvial & heat map

> Introduction to ggalluvial

> Description

The ggalluvial package is a ggplot2 extension for producing alluvial plots.
Alluvial plots use variable-width ribbons and stacked bar plots to represent multi-dimensional or repeated-measures data with categorical or ordinal variables.

> Five essential components

·**AIXS** A dimension (variable) along which the data are vertically grouped at a fixed horizontal position.

·**ALLUVIUM** Horizontal (x-) splines called alluvia span the width of the plot.

·**STRATUM** The groups at each axis are depicted as opaque blocks called strata.

·**LODE** The alluvia intersect the strata at lodes.

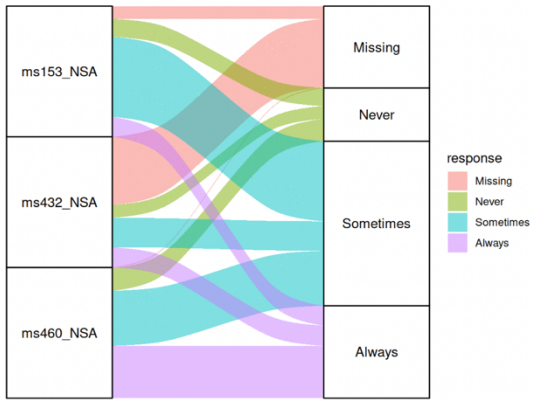
·**FLOW** The segments of the alluvia between pairs of adjacent axes are flows.

> Basic Alluvial Plot

```
ggplot(df, aes(axis1 = x1, axis2 = x2, y = Freq)) +

geom_alluvium() +

geom_stratum() +

geom_text(stat = “stratum”, aes(label = paste(after_stat(stratum))))
```

> Example

```
# install.packages("ggalluvial")
library(ggalluvial)

ggplot(data = vaccinations,
     aes(axis1 = survey, axis2 = response, y = freq)) +
 geom_alluvium(aes(fill = response)) +
 geom_stratum() +
 geom_text(stat = "stratum",
         aes(label = after_stat(stratum))) +
 scale_x_discrete(limits = c("Survey", "Response"),
              expand = c(0.15, 0.05)) +
 theme_void()
```

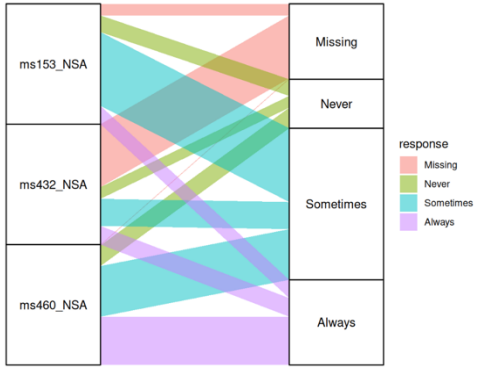


> Curve types

The type of flows of the plot area can be customized with the curve_type argument of the geom_alluvium function. The default value is "xspline", which produces approximation splines using four points per curve.

> Linear

```
ggplot(data = vaccinations,
     aes(axis1 = survey, axis2 = response, y = freq)) +
 geom_alluvium(aes(fill = response),
           curve_type = "linear") +
 geom_stratum() +
 geom_text(stat = "stratum",
         aes(label = after_stat(stratum))) +
 scale_x_discrete(limits = c("Survey", "Response"),
              expand = c(0.15, 0.05)) +
 theme_void()
```



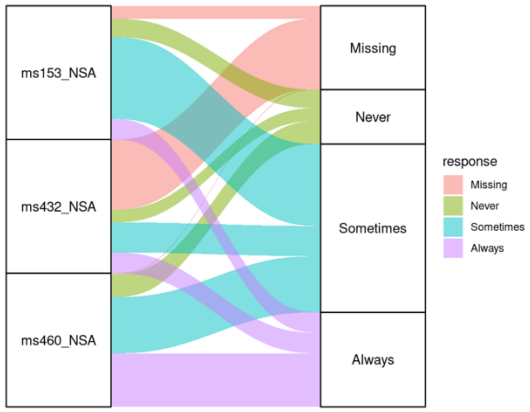
> Cubic

```
ggplot(data = vaccinations,
     aes(axis1 = survey, axis2 = response, y = freq)) +
 geom_alluvium(aes(fill = response),
           curve_type = "cubic") +
 geom_stratum() +
 geom_text(stat = "stratum",
         aes(label = after_stat(stratum))) +
 scale_x_discrete(limits = c("Survey", "Response"),
              expand = c(0.15, 0.05)) +
 theme_void()
```

> Curve types

> Quintic

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response),
                curve_type = "quintic") +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```



> Sine

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response),
                curve_type = "sine") +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```

> Arctangent

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response),
                curve_type = "arctangent") +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```

> Sigmoid

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response),
                curve_type = "sigmoid") +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```

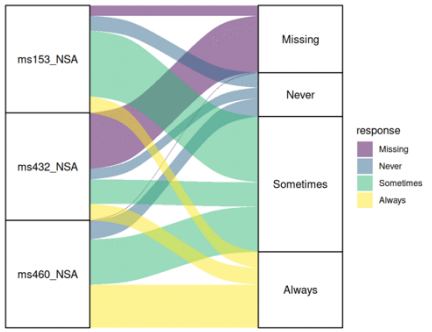
> Color customization

> Fill Color

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = "red")) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  scale_fill_viridis_d()
  theme_void()
```

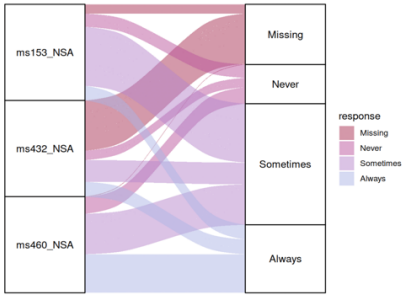
> Color palette

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  scale_fill_viridis_d() +
  theme_void()
```



> Custom colors

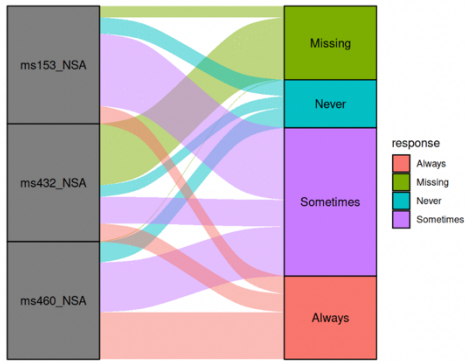
```
colors <- hcl.colors(4, "Red-Blue")
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  scale_fill_manual(values = colors) +
  theme_void()
```



> Color customization

> Stratum Color

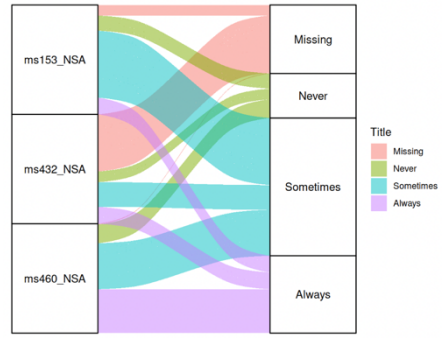
```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum(aes(fill = response)) +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```



> Legend customization

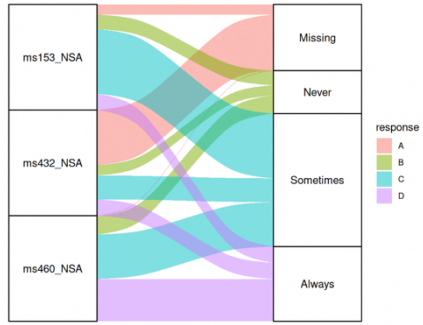
> Legend title

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void() +
  guides(fill = guide_legend(title = "Title"))
```



> Legend key labels

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void() +
  scale_fill_hue(labels = c("A", "B", "C", "D"))
```



> Remove the legend

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response)) +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void() +
```

> Introduction to heat map

> > Description

A heap map in ggplot2 can be created with geom_tile, passing the categorical variables to x and y arguments and the continuous variable to fill argument of aes.

x: position on the X axis
y: position on the Y axis
fill: the numeric value that will be translated in a color

> Example

Given a numerical matrix you will need to transform it into a data frame that ggplot2 can understand. For that purpose, you can use the melt function from reshape package.

```
# install.packages("reshape")
library(reshape)

# Data
set.seed(8)
m <- matrix(round(rnorm(200), 2), 10, 10)
colnames(m) <- paste("Col", 1:10)
rownames(m) <- paste("Row", 1:10)

# Transform the matrix in long format
df <- melt(m)
colnames(df) <- c("x", "y", "value")
```

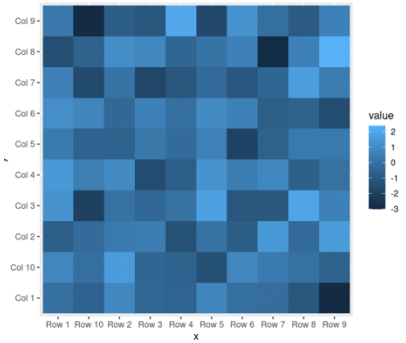
x	y	value
Row 1	Col 1	-0.08
Row 2	Col 1	0.84
Row 3	Col 1	-0.46
Row 4	Col 1	-0.55
Row 5	Col 1	0.74
Row 6	Col 1	-0.11

1–6 of 100 rows

> Heat map with geom_tile

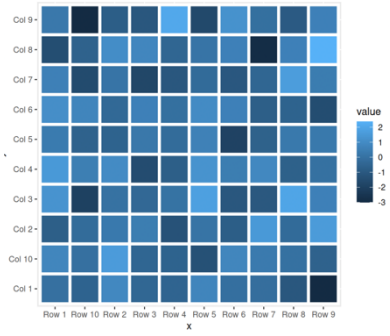
> Square tiles

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  coord_fixed()
```



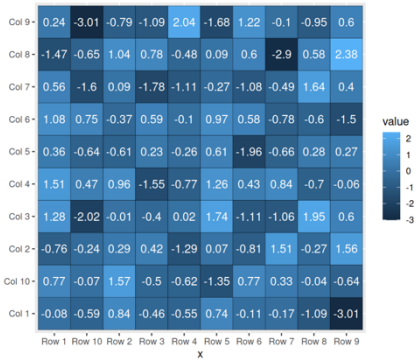
> Border customization

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "white",
            lwd = 1.5,
            linetype = 1) +
  coord_fixed()
```



> Adding the values

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  geom_text(aes(label = value), color = "white", size = 4) +
  coord_fixed()
```

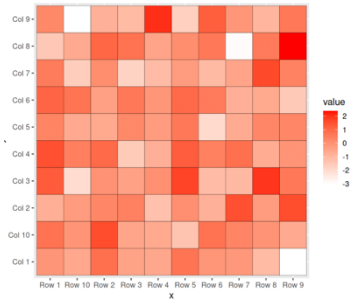


> Color palette

> scale_fill_gradient

This function allows changing the colors, setting a lower and a higher color to represent the values of the heat map.

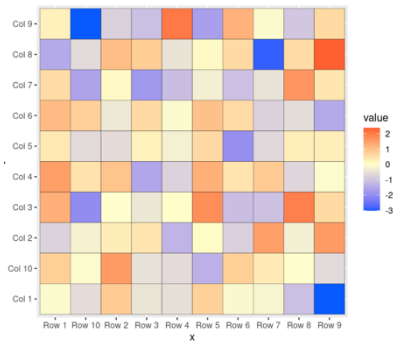
```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  scale_fill_gradient(low = "white", high = "red") +
  coord_fixed()
```



> scale_fill_gradient2

If you want to add a mid-color you can use scale_fill_gradient2, which includes the mid argument.

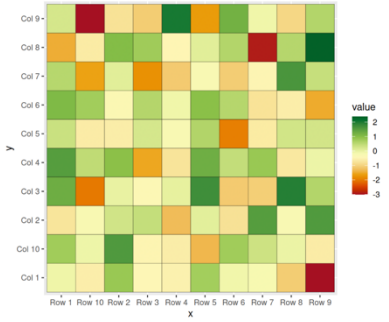
```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  scale_fill_gradient2(low = "#075AFF",
                       mid = "#FFFFCC",
                       high = "#FF0000") +
  coord_fixed()
```



> scale_fill_gradientn

Finally, you can also use a custom color palette with scale_fill_gradientn, which allows passing *n* colors to the colors argument. In this example we are passing 20 colors of the "RdYlGn" palette.

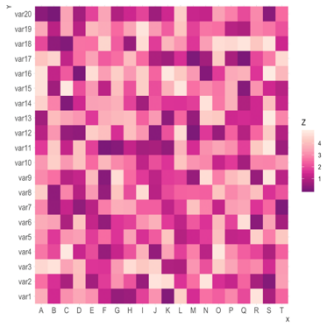
```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  scale_fill_gradientn(colors = hcl.colors(20, "RdYlGn")) +
  coord_fixed()
```



> scale_fill_distiller

to provide a ColorBrewer palette

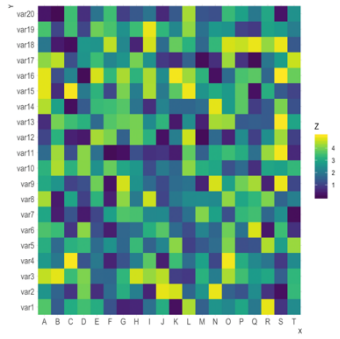
```
library(ggplot2)
library(hrbrthemes)
ggplot(data, aes(X, Y, fill= Z)) +
  geom_tile() +
  scale_fill_distiller(palette = "RdPu") +
  theme_ipsum()
```



> scale_fill_viridis

to use Viridis. Do not forget discrete=FALSE for a continuous variable.

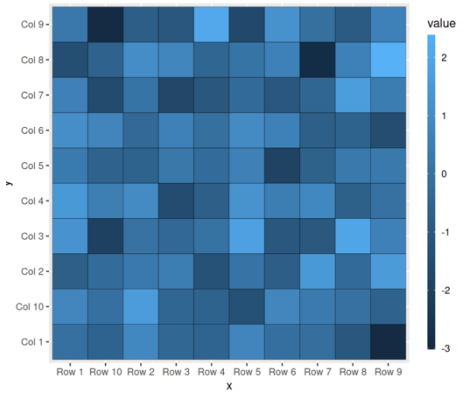
```
library(viridis)
ggplot(data, aes(X, Y, fill= Z)) +
  geom_tile() +
  scale_fill_viridis(discrete=FALSE) +
  theme_ipsum()
```



> Legend customization

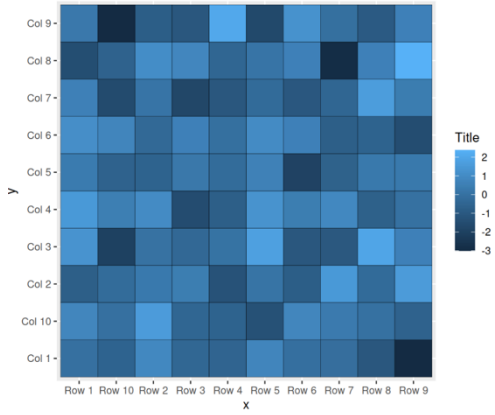
> Width and height

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  coord_fixed() +
  guides(fill = guide_colourbar(barwidth = 0.5,
                                barheight = 20))
```



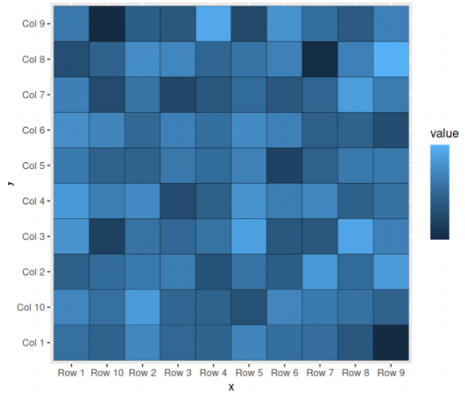
> Change the title

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  coord_fixed() +
  guides(fill = guide_colourbar(title = "Title"))
```



> Remove the labels and the ticks

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  coord_fixed() +
  guides(fill = guide_colourbar(label = FALSE,
                                ticks = FALSE))
```



> Remove the legend

```
ggplot(df, aes(x = x, y = y, fill = value)) +
  geom_tile(color = "black") +
  coord_fixed() +
  theme(legend.position = "none")
```