MINNEAPOLIS PUBLIC SCHOOLS

R Markdown Template Demo

Demonstration of Package Functions

Written By: Morrigan Hughes

April 2024

Summary

The new `mpstheme` package allows users to readily and easily format any plot to MPS brand guidelines, and also includes PDF templates for scripted, scheduled, or ad-hoc reporting to stakeholders. A number of new **R** and LaTeX functions are introduced with this package and this document will cover them all in addition to a brief introduction to some base LaTeX commands, functions, and environments.

```
### New R Functions
## ggplot2 specific:
theme_mps()                # Adds the MPS theme to a ggplot object
theme_mps_donut()          # Adds the MPS donut plot theme to a ggplot object
title_mps("title")         # Adds the title to an R Markdown document by creating a
 ↪  ggplot object
geom_label_mps()           # Adds data labeling to a ggplot plot
mps_cols(color)            # For use with scale_*_manual() to select specific MPS colors
scale_color_mps(palette)   # Adds the requested palette to the color scale. See Figure 1
scale_fill_mps(palette)    # Adds the requested palette to the fill scale. See Figure 1


## R Markdown specific
# R Functions:
mps_caption(caption, type) # Adds the caption below a figure, table, or plot
mps_fonts()                # Loads the fonts if not autoloaded by library(mpstheme)
mps_tcolor(color, text)    # Change the color of text in an R Markdown doc

# LaTeX Functions
\notebox{content}          # Note box
\warnbox{content}          # Warning note box
\impbox{content}           # Important note box
\specbox{title}{content}   # Like previous boxes, but allows you to change box title.
```

Figure 1. New functions included with mpstheme.

Demonstration of common plots used by MPS REA and OFDA

MPS Color Palettes

NOTE: Teal color currently under review.

Figure 2. MPS color palettes

Data Setup

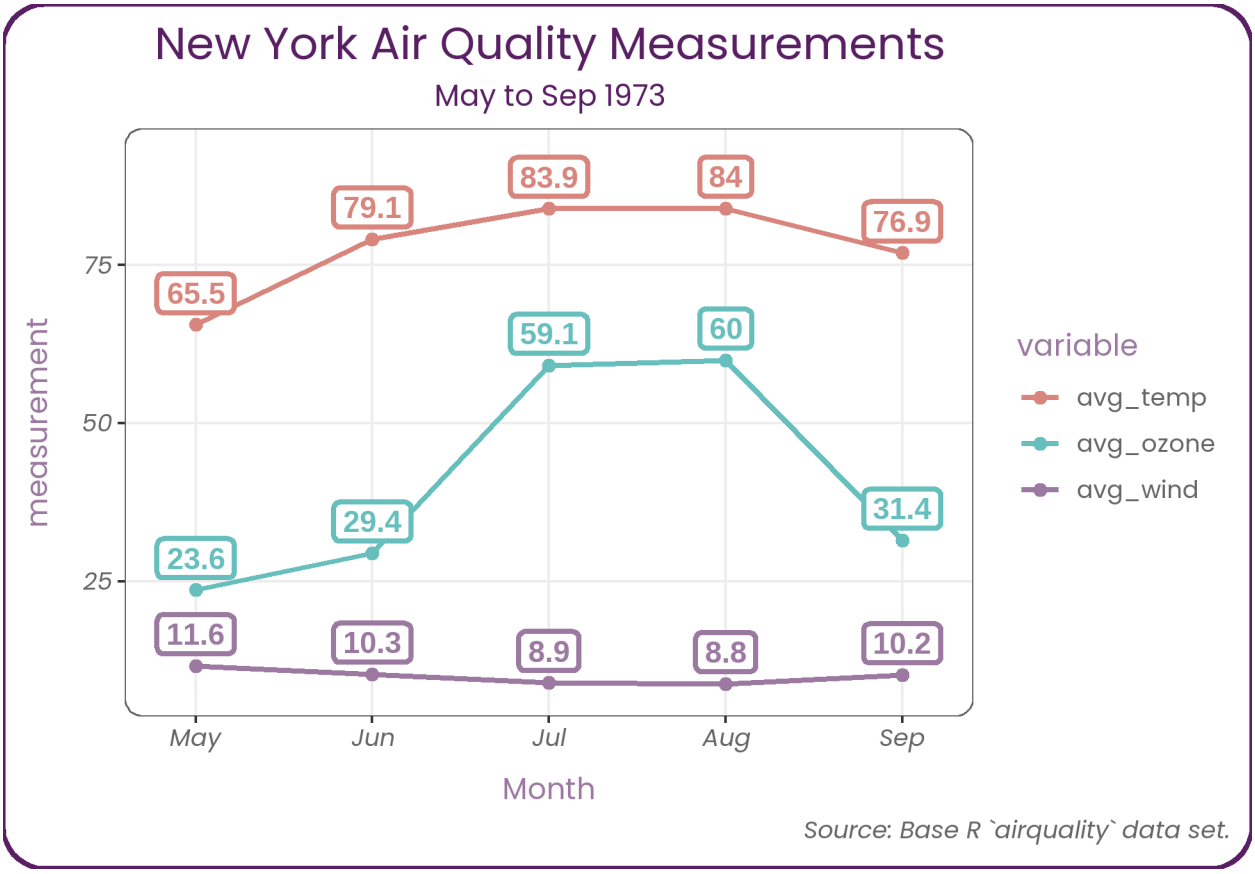
For demonstrating the various plots and functions included in the `mpstheme` package, we first need to setup our data for each of the different plots we're going to create.

```r
## Data for Line Chart
# Using inbuilt `airquality` dataset
aq <- airquality %>%
  group_by(Month) %>%
  summarise(
    avg_ozone = mean(Ozone, na.rm = TRUE),
    avg_wind = mean(Wind, na.rm = TRUE),
    avg_temp = mean(Temp, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  gather(avg_ozone, avg_wind, avg_temp,
         key = "variable", value = "measurement")

## Data for combined line and bar chart
# Leaving year as numeric will allow for scale_*_continuous arguments
lb <- tibble(
  year = c(2018:2023),
  group = c(.45, .47, .48, .46, .42, .41),
  overall = c(.6, .59, .6, .52, .51, .5)
)

## Data for the grouped vertical or horizontal bar plot
vb <- tibble(
  group = c("Group 1", "Group 2", "Group 3", "Group 4", "Group 5", "Group 6"),
  year_1 = c(.45, .34, .78, .9, .23, .25),
  year_2 = c(.56, .78, .9, .35, .26, .67),
  year_3 = c(.72, .8, .87, .45, .33, .5)
) %>%
  gather(year_1, year_2, year_3,
         key = "year", value = "result")

## Data for the stacked vertical bar plot
vbs <- tibble(
  Group = c("Group 1", "Group 2", "Group 3", "Group 4", "Group 5", "Group 6"),
  "On Track" = c(.5, .55, .6, .55, .57, .52),
  Focus = c(.1, .07, .06, .08, .09, .1),
  Priority = c(.4, .38, .34, .37, .34, .38)
) %>%
```

```
  gather("On Track", Focus, Priority,
         key = "On_Track_Status", value = "measurement")
# Factor the On_Track_Status values for plotting
vbs$On_Track_Status <- factor(vbs$On_Track_Status,
                              levels = c("On Track", "Focus", "Priority"))

## Data for the stacked horizontal bar plot
hbs <- tibble(
  Question = c("Question 1", "Question 2", "Question 3",
               "Question 4", "Question 5"),
  "Strongly Disagree" = c(.1, .09, .08, .07, .06),
  Disagree = c(.2, .21, .22, .23, .24),
  Neutral = c(.1, .11, .12, .13, .14),
  Agree = c(.3, .29, .28, .27, .26),
  "Strongly Agree" = c(.3, .3, .3, .3, .3)
) %>%
  gather("Strongly Disagree", Disagree, Neutral, Agree, "Strongly Agree",
         key = "Answer", value = "Percentage")
# Factor the Answer values for plotting
hbs$Answer <- factor(hbs$Answer,
                     levels = c("Strongly Agree", "Agree", "Neutral",
                                "Disagree", "Strongly Disagree"))

## Donut plot with one main point to highlight
dnt1 <- tibble(
  group = c("Group 1", "Group 2"),
  percent = c(.9, .1)
)
# Select the top percent for the annotation geom
dnt1_anno <- dnt1 %>%
  top_n(1, percent)

## Donut plot with multiple groups
dnt2 <- tibble(
  Ethnicity = as_factor(c("African American", "American Indian", "Asian",
                          "Hispanic", "White", "Two or More")),
  Percent = c(.1, .1, .2, .15, .4, .05)
)
```

Figure 3. Setting up the data for the plots

Line Chart

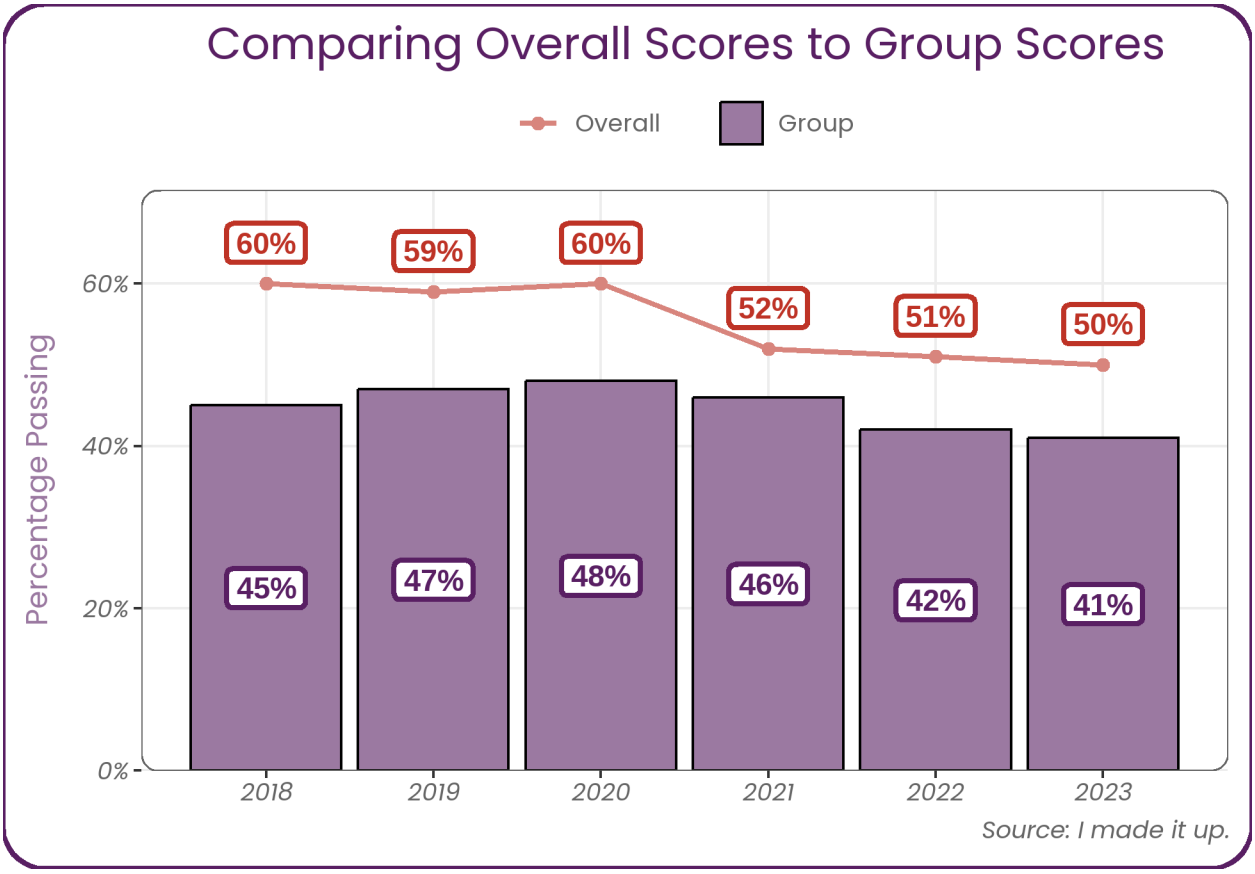


Plot 1. Line chart

```
ggplot(aq, aes(Month, measurement, color = variable)) +
  ## Add the required geoms
  geom_line(linewidth = 1) +
  geom_point(size = 2) +
  ## theme_mps to apply the MPS theme
  theme_mps() +
  ## Manually set the colors from MPS official colors
  ## Use breaks = to rearrange the order of items in the legend
  scale_color_manual(values = mps_cols("light red", "light teal", "light wine"),
                     breaks = c("avg_temp", "avg_ozone", "avg_wind")) +
  ## Use scale_x_* to set the labels to the month name
  # Use expand to add some padding to the x axis
  scale_x_continuous(labels = unique(month(aq$Month, label = TRUE)),
                     expand = expansion(mult = c(0.1, 0.1))) +
  ## Add the labels!
  # Position_nudge() allows for the labels to be moved above the lines
  geom_label_mps(mapping = aes(label = round(measurement, digits = 1),
```

```
                              y = measurement, color = variable),
                position = position_nudge(y = 5)) +
  ## Adjust the zoom of the plot to give more room to the labels
  # This could be done in a scale_y_* call if we had made one
  coord_cartesian(ylim = c(min(aq$measurement)*.9, max(aq$measurement)*1.1)) +
  ## Add titles, subtitles, and captions
  labs(
    title = "New York Air Quality Measurements",
    subtitle = "May to Sep 1973",
    caption = "Source: Base R `airquality` data set."
  )
```

Line and Bar Chart

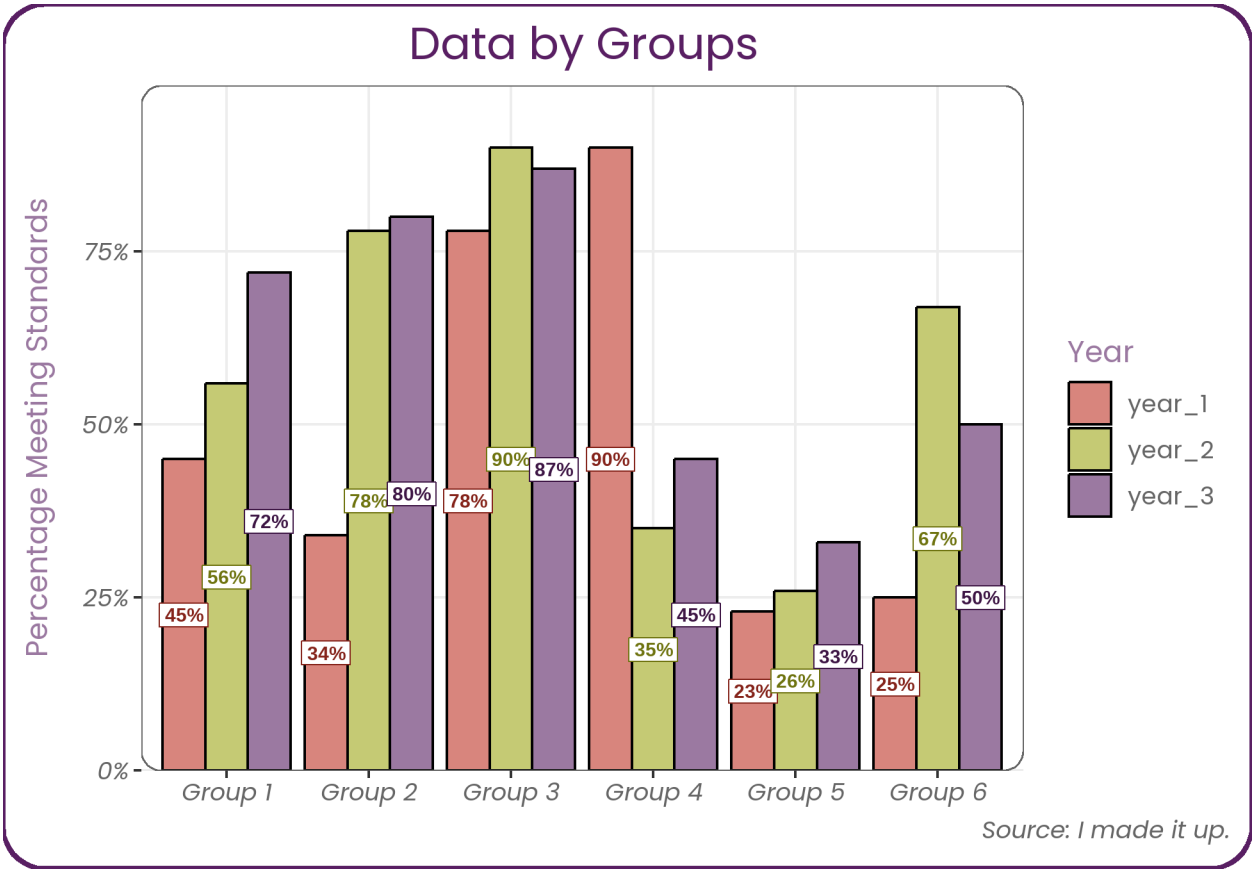


Plot 2. Line and bar chart

```
ggplot(lb) +
  ## Add the three required geoms
  # Adding a string to color and fill allows for the legend to be created
  geom_line(aes(x = year, y = overall, color = "Overall"), linewidth = 1) +
  geom_point(aes(x = year, y = overall, color = "Overall"), size = 2) +
  geom_col(aes(x = year, y = group, fill = "Group"), color = "black", linewidth = .5) +
  ## theme_mps applies the theme.
  # The optional argument moves the legend to the top.
  theme_mps(legend.position = "top") +
  ## Use labels = label_percent() to change the labels of the y axis
  # Use expand = expansion() to force the y axis origin down to the x axis line
  #     and to give room at the top of the y axis for the labels
  scale_y_continuous(labels = label_percent(),
                     expand = expansion(mult = c(0,0.1))) +
  ## Use scale_x_continuous to show all labels of the x axis
  scale_x_continuous(breaks = 2018:2023) +
  ## Add our MPS colors
```

```
  # Fill for the columns, color for the lines and points
  scale_fill_manual(values = mps_cols("light wine")) +
  scale_color_manual(values = mps_cols("light red")) +
  ## Label the geom_line points; use position_nudge to adjust the labels vertically
  geom_label_mps(mapping = aes(label = percent(overall), x = year, y = overall),
                 position = position_nudge(y = .05),
                 color = mps_cols("red")) +
  ## Label the geom_col columns; use position_stack to put labels in center of bars
  geom_label_mps(mapping = aes(label = percent(group), x = year, y = group),
                 position = position_stack(vjust = 0.5),
                 color = mps_cols("wine")) +
  ## Label the plot. Use color = NULL and fill = NULL to leave a simple legend
  labs(
    title = "Comparing Overall Scores to Group Scores",
    caption = "Source: I made it up.",
    color = NULL,
    fill = NULL,
    x = NULL,
    y = "Percentage Passing"
  )
```

Vertical Grouped Bars

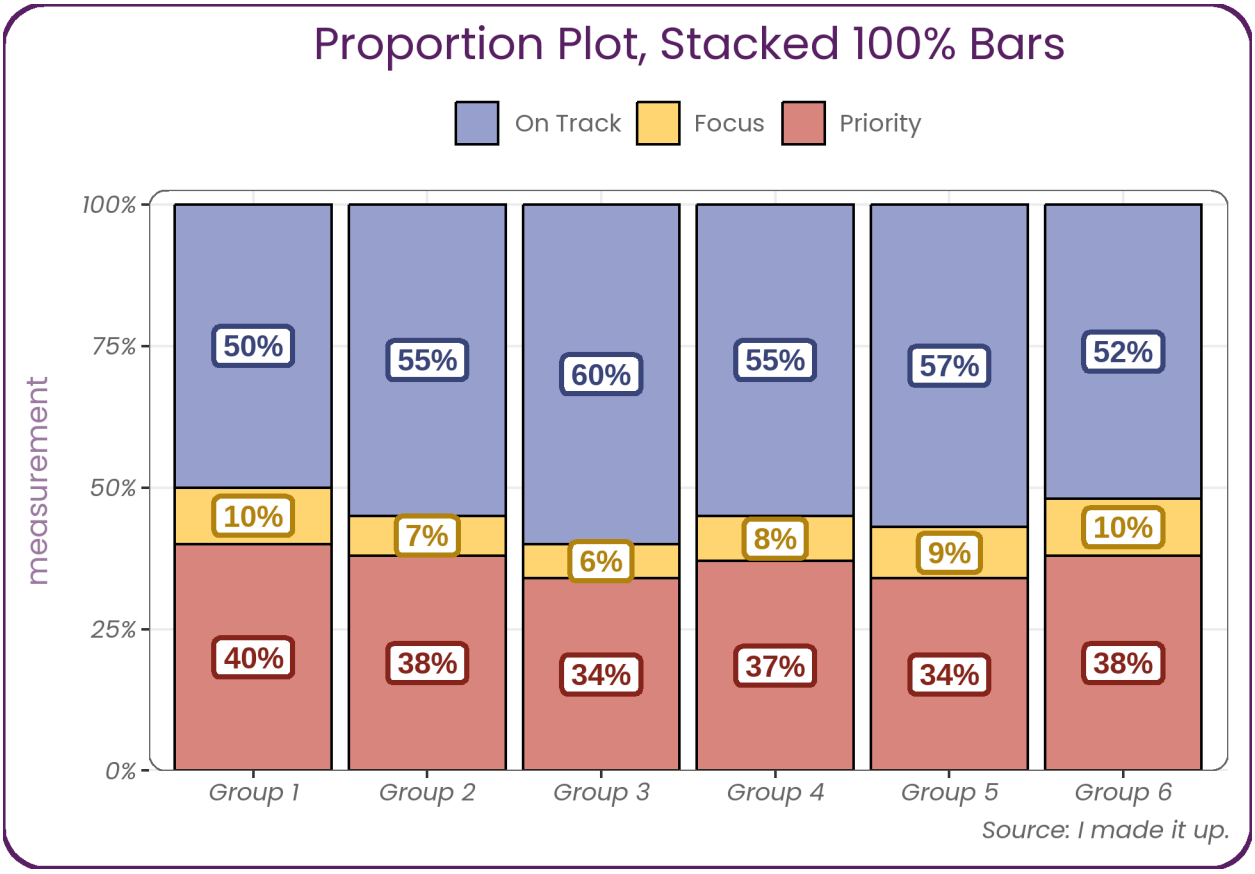


Plot 3. Grouped vertical bar plot

```
ggplot(vb, aes(group, result, fill = year)) +
  ## Add required geom
  # position_dodge() tells the geom to make grouped bars
  geom_col(position = position_dodge(width = 0.9), color = "black") +
  ## Add the theme
  theme_mps() +
  ## Set the y scale. Use label_percent() to make the axis formatted to percent
  # Use expand = expansion() to adjust the y axis start and end points
  scale_y_continuous(labels = label_percent(),
                     expand = expansion(mult = c(0,0.1))) +
  ## Set the labels for the bars.
  # The y = argument puts the bars at the middle of each bar.
  # Using vjust = would put the labels at the mid point of the group.
  geom_label_mps(mapping = aes(label = percent(result),
                               y = result*0.5,
                               x = group,
                               color = year),
```

```
                position = position_dodge(width = 0.9),
            #Reduce the text size to fit the bars
                size = 2.5,
            #Reduce the label size to fit the bars
                label.size = 0.25,
            #Remove the corner radius
                label.r = unit(0, "points"),
            #Adjust padding to get everything to fit
                label.padding = unit(2, "points")) +
  ## Use scale_*_mps() to set the color palette
  # scale_*_manual() allows for specific choices of colors if preferred
  scale_fill_mps("light") +
  # scale_color_mps(palette = "hot") +
  scale_color_mps("dark") +
  ## Add the labels. x = and y = NULL removes the axis labels
  labs(
    title = "Data by Groups",
    caption = "Source: I made it up.",
    fill = "Year",
    x = NULL,
    y = "Percentage Meeting Standards"
  )
```

Vertical Stacked Bars

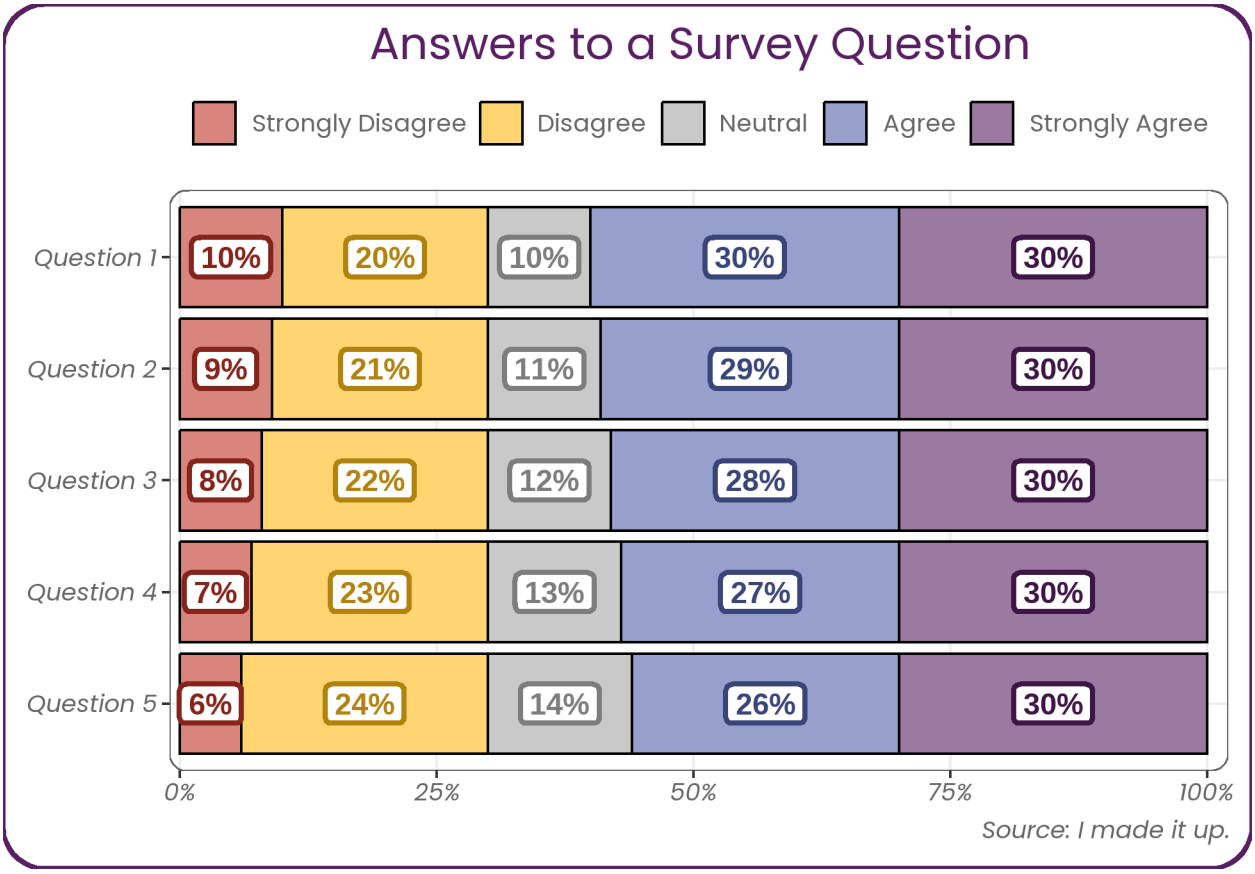


Plot 4. Stacked vertical bar plot

```
ggplot(vbs, aes(Group, measurement, fill = On_Track_Status)) +
  ## Add the required geom
  geom_col(color = "black") +
  ## scale_*_manual() to set specific colors for the statuses
  scale_fill_manual(values = mps_cols("light blue", "light yellow", "light red")) +
  scale_color_manual(values = mps_cols("dark blue", "dark yellow", "dark red")) +
  ## Add the theme. Optional argument moves the legend to the top.
  theme_mps(legend.position = "top") +
  ## Set the scale to use labeled percentages, rather than decimals
  scale_y_continuous(labels = label_percent(),
                     expand = expansion(mult = c(0,0.025))) +
  ## Now add the data labels
  geom_label_mps(mapping = aes(label = percent(measurement, accuracy = 1),
                               y = measurement,
                               color = On_Track_Status),
                 position = position_stack(vjust = 0.5)) +
  ## And the plot labels
```

```
  labs(
    title = "Proportion Plot, Stacked 100% Bars",
    caption = "Source: I made it up.",
    fill = NULL,
    x = NULL
  )
```

Horizontal Stacked Bars

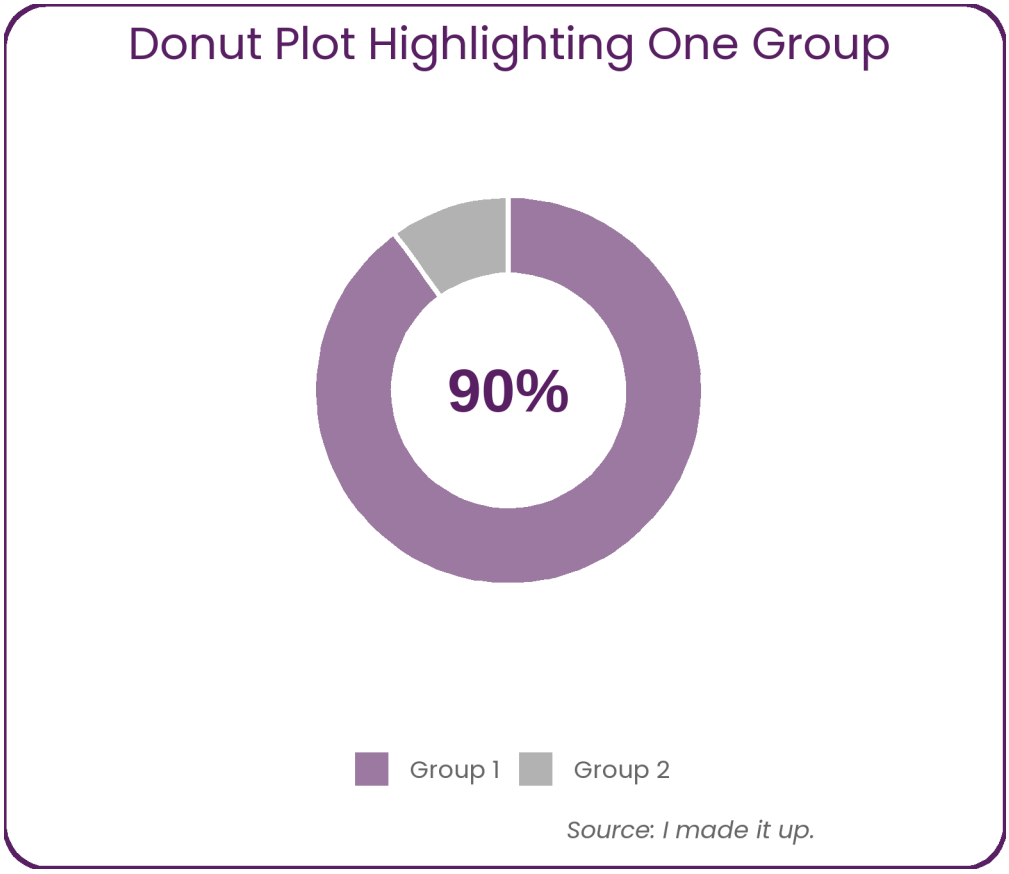


Plot 5. Stacked horizontal bar plot

```
ggplot(hbs, aes(Percentage, fct_rev(Question), fill = Answer)) +
  ## Add the required geom
  geom_col(color = "black") +
  ## Use scale_x_* to set the labels to percentage and adjust the limits
  scale_x_continuous(labels = label_percent(),
                     expand = expansion(mult = c(0.01,0.02))) +
  ## Use scale_*_manual to set the specific colors to use
  # Use guide_legend(reverse = TRUE) to reverse the legend so it
  #     matches the order of the plot
  scale_fill_manual(values = mps_cols("light wine", "light blue", "light gray",
                                      "light yellow", "light red"),
                    guide = guide_legend(reverse = TRUE)) +
  scale_color_manual(values = mps_cols("dark wine", "dark blue", "dark gray",
                                       "dark yellow", "dark red")) +
  ## Add the theme with optional argument to move legend to the top
  theme_mps(legend.position = "top") +
  ## Add the labels, and use position_stack() to put the labels in the bars
```

```
  geom_label_mps(mapping = aes(label = percent(Percentage, accuracy = 1),
                               y = Question,
                               color = Answer),
                 position = position_stack(vjust = 0.5)) +
  ## Add plot labels
  labs(
    title = "Answers to a Survey Question",
    caption = "Source: I made it up.",
    fill = NULL,
    y = NULL,
    x = NULL
  )
```

Donut Plots - Highlight

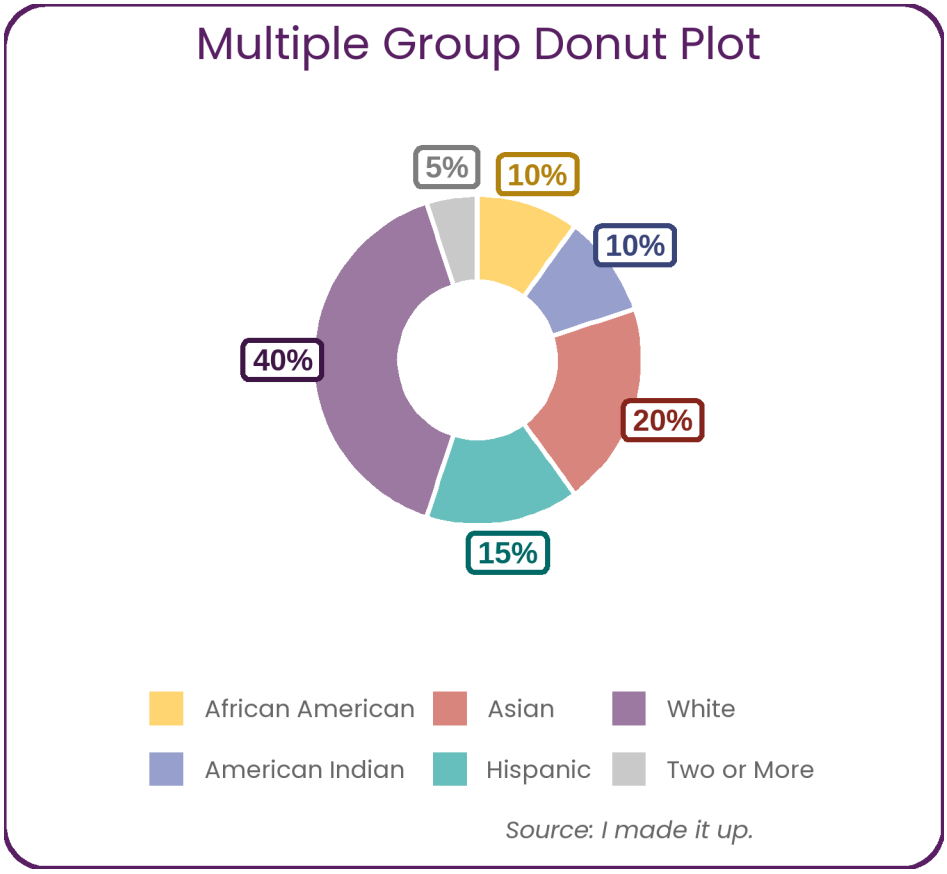


Plot 6. Donut plot highlighting one variable

```
ggplot(dnt1, aes(x = 1, y = percent, fill = group)) +
  # Donut plots are just column plots translated to polar coords
  geom_col(position = "fill", color = "white", linewidth = 1) +
  coord_polar(theta = "y", direction = -1) +
  # xlim() sets the inner and outer size of the ring
  xlim(c(-.75, 2)) +
  ## Add the center of the plot annotation
  # TODO Considering developing a geom_text_donut() function
  geom_text(
    data = dnt1_anno,
    # Set x = to same as minimum xlim, and y = 0 to center the annotation
    aes(x = -.75, y = 0, label = percent(percent)),
    inherit.aes = FALSE,
    fontface = "bold",
    color = mps_cols("wine"),
    size = 8
  ) +
```

```
  ## Add the scale_*_manual to set the group colors
  scale_fill_manual(values = mps_cols("light wine", "gray")) +
  ## Add the donut-specific theme
  theme_mps_donut() +
  ## Add the plot labels
  labs(
    title = "Donut Plot Highlighting One Group",
    caption = "Source: I made it up.",
    fill = NULL
  )
```

Donut Plots - Multiple Groups



Plot 7. Normal donut plot

```
ggplot(dnt2, aes(x = 1, y = Percent, fill = Ethnicity)) +
  geom_col(position = "fill", color = "white", linewidth = 1) +
  coord_polar(theta = "y", direction = -1) +
  xlim(c(-0.25, 2)) +
  scale_fill_manual(values = mps_cols("light yellow", "light blue", "light red",
                                      "light teal", "light wine", "light gray")) +
  scale_color_manual(values = mps_cols("dark yellow", "dark blue", "dark red",
                                       "dark teal", "dark wine", "dark gray")) +
  geom_label_mps(mapping = aes(label = percent(Percent),
                               y = Percent,
                               x = 1.75,
                               color = Ethnicity),
                 position = position_stack(vjust = 0.5)) +
  theme_mps_donut() +
  labs(
    title = "Multiple Group Donut Plot",
    caption = "Source: I made it up.",
```

```
    fill = NULL
  )
```

Demonstration of Additional Functionality

Kable Tables

To display tabular data, you should use the `kable` and `kableExtra` packages. Below is the same data as displayed in Plot 5, this time displayed in an MPS-themed table.

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Question 1	30.0%	30.0%	10.0%	20.0%	10.0%
Question 2	30.0%	29.0%	11.0%	21.0%	9.0%
Question 3	30.0%	28.0%	12.0%	22.0%	8.0%
Question 4	30.0%	27.0%	13.0%	23.0%	7.0%
Question 5	30.0%	26.0%	14.0%	24.0%	6.0%

Table 1. Plot 5 data as a data table.

```
hbs %>%
  mutate(Percentage = percent_format()(Percentage)) %>%
  spread(Answer, Percentage) %>%
  kable(
    format = "latex", # Comment this line out while viewing table in-line
    booktabs = TRUE,
    align = "lccccc",
    linesep = ""
  ) %>%
  kable_paper(lightable_options = "striped") %>% # For viewing inline while working
  kable_styling(stripe_color = "mpswine!5", latex_options = c("scale_down",
   ↪  "HOLD_position", "striped")) %>%
  row_spec(row = 0, color = mps_cols("wine"))
```

Plot, Table, and Figure Captions

You may have noticed from the above plots, tables, and figures that each is captioned with an iterative plot, figure, or table number. Those plot captions are generated with the `mps_caption(caption, type)` function. Acceptable types for that function are "plots", "tables", and "figures". Each type will begin their count from 1 and iterate every time the function is called throughout the document.

The `mps_caption()` function works by pasting LaTeX and R functions together during document knitting. Part of the function of the caption includes a LaTeX call to reduce the vertical height between the caption and the line before it. Sometimes [particularly in the case of wide tables which are scaled down to fit the width of the report] this reduction in vertical space is too much, or not enough. You can adjust the vertical space between the caption and the plot, figure, or table by adding `\vspace{distance}` before `mps_caption()`. Distances should be in points [pt] and can be any positive or negative number.

NOTE

The default distance set in the plot caption function is -25pt. When adjusting the distance of your caption, start with 5 point increments to make your life easier.

Text Colors

Additionally, there is a new function available in the form of `mps_tcolor(color, text, mps = TRUE)`. This allows you to change the color of inline text to any of the colors listed in the tables below. When `mps = TRUE`, the default, you can use any MPS colors for inline text, and when `mps = FALSE`, you can use any of the base LaTeX colors. The two tables below have been created with basic `markdown` syntax, rather than the full `kable` package due to the need to run the `mps_tcolor()` functions.

Function	Result
`mps_tcolor('red', 'red')`	red
`mps_tcolor('orange', 'orange')`	orange
`mps_tcolor('yellow', 'yellow')`	yellow
`mps_tcolor('green', 'green')`	green
`mps_tcolor('teal', 'teal')`	teal
`mps_tcolor('blue', 'blue')`	blue
`mps_tcolor('darkblue', 'darkblue')`	darkblue
`mps_tcolor('wine', 'wine')`	wine
`mps_tcolor('gray', 'gray')`	gray

Table 2. Inline text colors when 'mps = TRUE'.

Function	Result
`mps_tcolor('black', 'black', mps = FALSE)`	black
`mps_tcolor('blue', 'blue', mps = FALSE)`	blue
`mps_tcolor('brown', 'brown', mps = FALSE)`	brown
`mps_tcolor('cyan', 'cyan', mps = FALSE)`	cyan
`mps_tcolor('darkgray', 'darkgray', mps = FALSE)`	darkgray
`mps_tcolor('gray', 'gray', mps = FALSE)`	gray
`mps_tcolor('green', 'green', mps = FALSE)`	green
`mps_tcolor('lightgray', 'lightgray', mps = FALSE)`	lightgray
`mps_tcolor('lime', 'lime', mps = FALSE)`	lime
`mps_tcolor('magenta', 'magenta', mps = FALSE)`	magenta
`mps_tcolor('olive', 'olive', mps = FALSE)`	olive
`mps_tcolor('orange', 'orange', mps = FALSE)`	orange
`mps_tcolor('pink', 'pink', mps = FALSE)`	pink
`mps_tcolor('purple', 'purple', mps = FALSE)`	purple
`mps_tcolor('red', 'red', mps = FALSE)`	red
`mps_tcolor('teal', 'teal', mps = FALSE)`	teal
`mps_tcolor('violet', 'violet', mps = FALSE)`	violet
`mps_tcolor('white', 'white', mps = FALSE)`	
`mps_tcolor('yellow', 'yellow', mps = FALSE)`	yellow

Table 3. Inline text colors when mps = FALSE.

Special Callout Boxes

There are additionally four new LaTeX commands one can use in your **R Markdown** document. These commands allow you to create special callout boxes of different types. The note box is designed for something interesting or some additional context for something in your report, and uses MPS Blue as the base color. The warn box is designed for some sort of warning before the end user takes an action, or something to keep in mind about data being displayed, and uses MPS Red as the base color. The important box is for important notices and uses MPS Orange as the base color. And the special box allows the creator to set a specific title to the box as well as the box content and uses MPS Wine as the base color.

- **Command**
  - `\notebox{content}`
  - `\warnbox{content}`
  - `\impbox{content}`
  - `\specbox{title}{content}`

NOTE

This is a note- something interesting worth extracting and pointing out to the reader! Long notes/warnings/etc. have wrapped text.

WARNING

This is a warning: Something critical you should know before taking action!

IMPORTANT

This is an important notice- Something is important!

EXAMPLE

This is some example text.

Format Example

This is some example text with **bolded** and *italicized* and ***both bolded and italicized*** text included.

Figure 4. Custom LaTeX color box command outputs.

Additional Notes

In addition to all the functionality above, these documents accept a variety of LaTeX commands. For example, you can center text or callout boxes, or whatever else by adding `\begin{center}` before the text/item to be centered and `\end{center}` at the end. If you need an additional space before or after something in text blocks, you can use `\space` since two consecutive spaces is treated as a new line in **R Markdown**. This is most often useful when adding inline code since the MPS font, *Poppins*, interacts a little strangely with monospace fonts.

Some useful LaTeX commands:

Command	Description
`\begin{center} \end{center}`	Centering of some text.
`\begin{flushright} \end{flushright}`	Right-justified text.
`\space`	Adds an extra space where needed.
`\newpage`	Adds a page break.
`\textbf{text}`	Bold text inside LaTeX commands.
`\textit{text}`	Italic text inside LaTeX commands.

Table 4. Additional LaTeX commands.