

# 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.

```

1  ### New R Functions
2  ## ggplot2 specific:
3  theme_mps()           # Adds the MPS theme to a ggplot object
4  theme_mps_donut()     # Adds the MPS donut plot theme to a ggplot object
5  title_mps("title")    # Adds the title to an R Markdown document by creating a
   ↪ ggplot object
6  geom_label_mps()      # Adds data labeling to a ggplot plot
7  mps_cols(color)       # For use with scale_*_manual() to select specific MPS colors
8  scale_color_mps(palette) # Adds the requested palette to the color scale. See Figure 1
9  scale_fill_mps(palette) # Adds the requested palette to the fill scale. See Figure 1
10
11
12 ## R Markdown specific
13 # R Functions:
14 mps_caption(caption, type) # Adds the caption below a figure, table, or plot
15 mps_fonts()               # Loads the fonts if not autoloading by require(mpstheme)
16 mps_tcolor(color, text)   # Change the color of text in an R Markdown doc
17
18 # LaTeX Functions
19 \notebox{content}         # Note box
20 \warnbox{content}         # Warning note box
21 \impbox{content}          # Important note box
22 \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.

```

1  ## Data for Line Chart
2  # Using inbuilt `airquality` dataset
3  aq <- airquality %>%
4    group_by(Month) %>%
5    summarise(
6      avg_ozone = mean(Ozone, na.rm = TRUE),
7      avg_wind = mean(Wind, na.rm = TRUE),
8      avg_temp = mean(Temp, na.rm = TRUE),
9      .groups = "drop"
10   ) %>%
11   gather(avg_ozone, avg_wind, avg_temp,
12          key = "variable", value = "measurement")
13
14  ## Data for combined line and bar chart
15  # Leaving year as numeric will allow for scale_*_continuous arguments
16  lb <- tibble(
17    year = c(2018:2023),
18    group = c(.45, .47, .48, .46, .42, .41),
19    overall = c(.6, .59, .6, .52, .51, .5)
20  )
21
22  ## Data for the grouped vertical or horizontal bar plot
23  vb <- tibble(
24    group = c("Group 1", "Group 2", "Group 3", "Group 4", "Group 5", "Group 6"),
25    year_1 = c(.45, .34, .78, .9, .23, .25),
26    year_2 = c(.56, .78, .9, .35, .26, .67),
27    year_3 = c(.72, .8, .87, .45, .33, .5)
28  ) %>%
29   gather(year_1, year_2, year_3,
30          key = "year", value = "result")
31
32  ## Data for the stacked vertical bar plot
33  vbs <- tibble(
34    Group = c("Group 1", "Group 2", "Group 3", "Group 4", "Group 5", "Group 6"),
35    "On Track" = c(.5, .55, .6, .55, .57, .52),
36    Focus = c(.1, .07, .06, .08, .09, .1),
37    Priority = c(.4, .38, .34, .37, .34, .38)
38  ) %>%

```

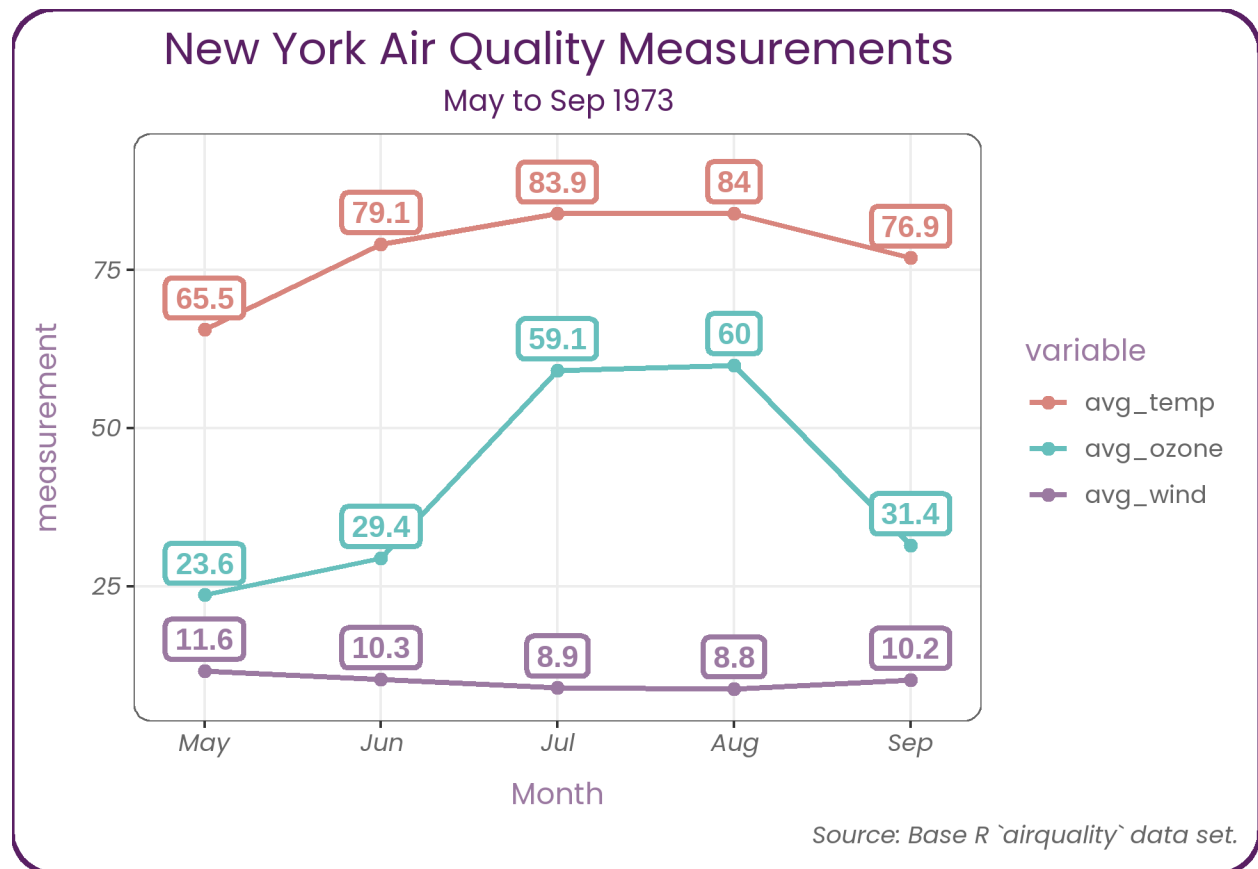
```

39 gather("On Track", Focus, Priority,
40         key = "On_Track_Status", value = "measurement")
41 # Factor the On_Track_Status values for plotting
42 vbs$On_Track_Status <- factor(vbs$On_Track_Status,
43                               levels = c("On Track", "Focus", "Priority"))
44
45 ## Data for the stacked horizontal bar plot
46 hbs <- tibble(
47   Question = c("Question 1", "Question 2", "Question 3",
48               "Question 4", "Question 5"),
49   "Strongly Disagree" = c(.1, .09, .08, .07, .06),
50   Disagree = c(.2, .21, .22, .23, .24),
51   Neutral = c(.1, .11, .12, .13, .14),
52   Agree = c(.3, .29, .28, .27, .26),
53   "Strongly Agree" = c(.3, .3, .3, .3, .3)
54 ) %>%
55   gather("Strongly Disagree", Disagree, Neutral, Agree, "Strongly Agree",
56         key = "Answer", value = "Percentage")
57 # Factor the Answer values for plotting
58 hbs$Answer <- factor(hbs$Answer,
59                     levels = c("Strongly Agree", "Agree", "Neutral",
60                               "Disagree", "Strongly Disagree"))
61
62 ## Donut plot with one main point to highlight
63 dnt1 <- tibble(
64   group = c("Group 1", "Group 2"),
65   percent = c(.9, .1)
66 )
67 # Select the top percent for the annotation geom
68 dnt1_anno <- dnt1 %>%
69   top_n(1, percent)
70
71 ## Donut plot with multiple groups
72 dnt2 <- tibble(
73   Ethnicity = as_factor(c("African American", "American Indian", "Asian",
74                           "Hispanic", "White", "Two or More")),
75   Percent = c(.1, .1, .2, .15, .4, .05)
76 )

```

Figure 3. Setting up the data for the plots

## Line Chart



Plot 1. Line chart

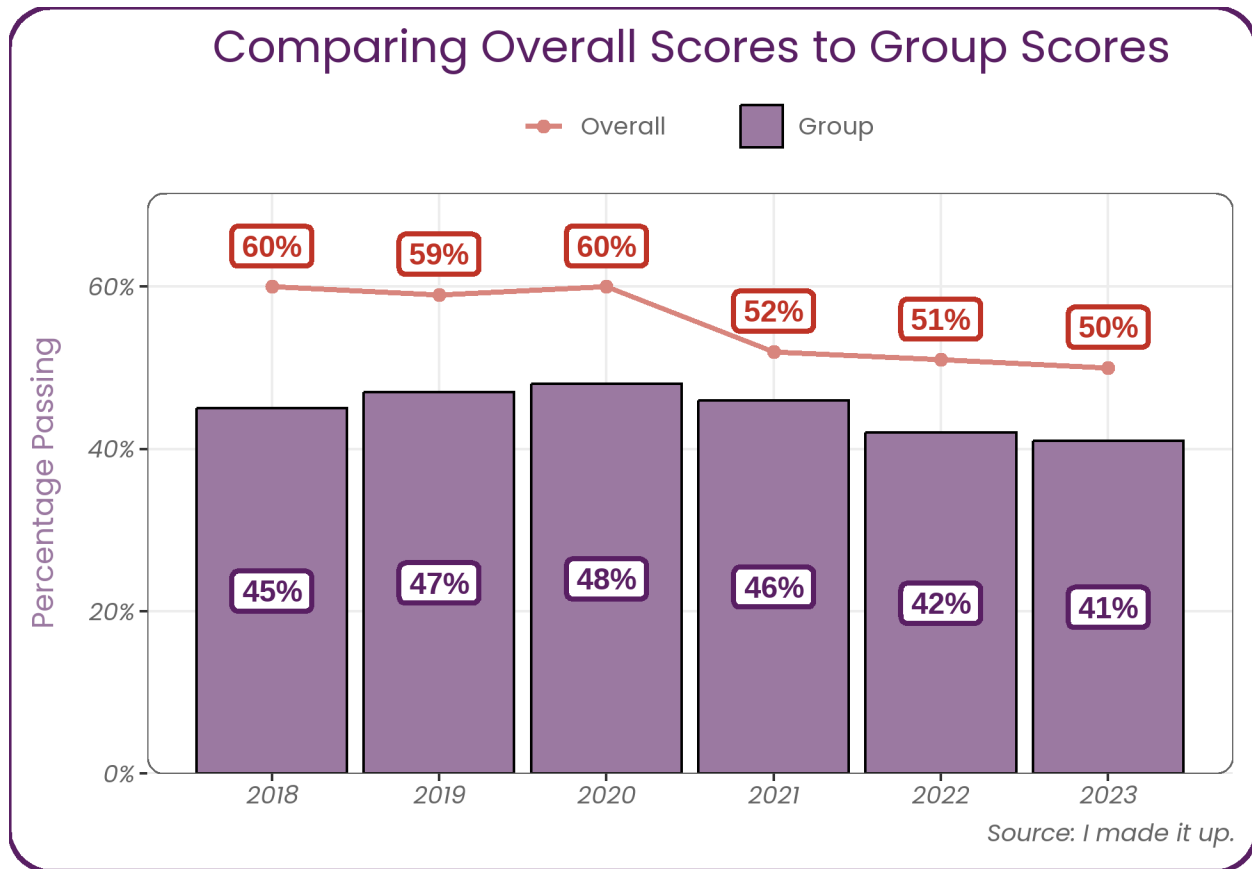
```

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

```

```
18         y = measurement, color = variable),
19         position = position_nudge(y = 5)) +
20     ## Adjust the zoom of the plot to give more room to the labels
21     # This could be done in a scale_y_* call if we had made one
22     coord_cartesian(ylim = c(min(aq$measurement)*.9, max(aq$measurement)*1.1)) +
23     ## Add titles, subtitles, and captions
24     labs(
25         title = "New York Air Quality Measurements",
26         subtitle = "May to Sep 1973",
27         caption = "Source: Base R `airquality` data set."
28     )
```

## Line and Bar Chart



Plot 2. Line and bar chart

```

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

```

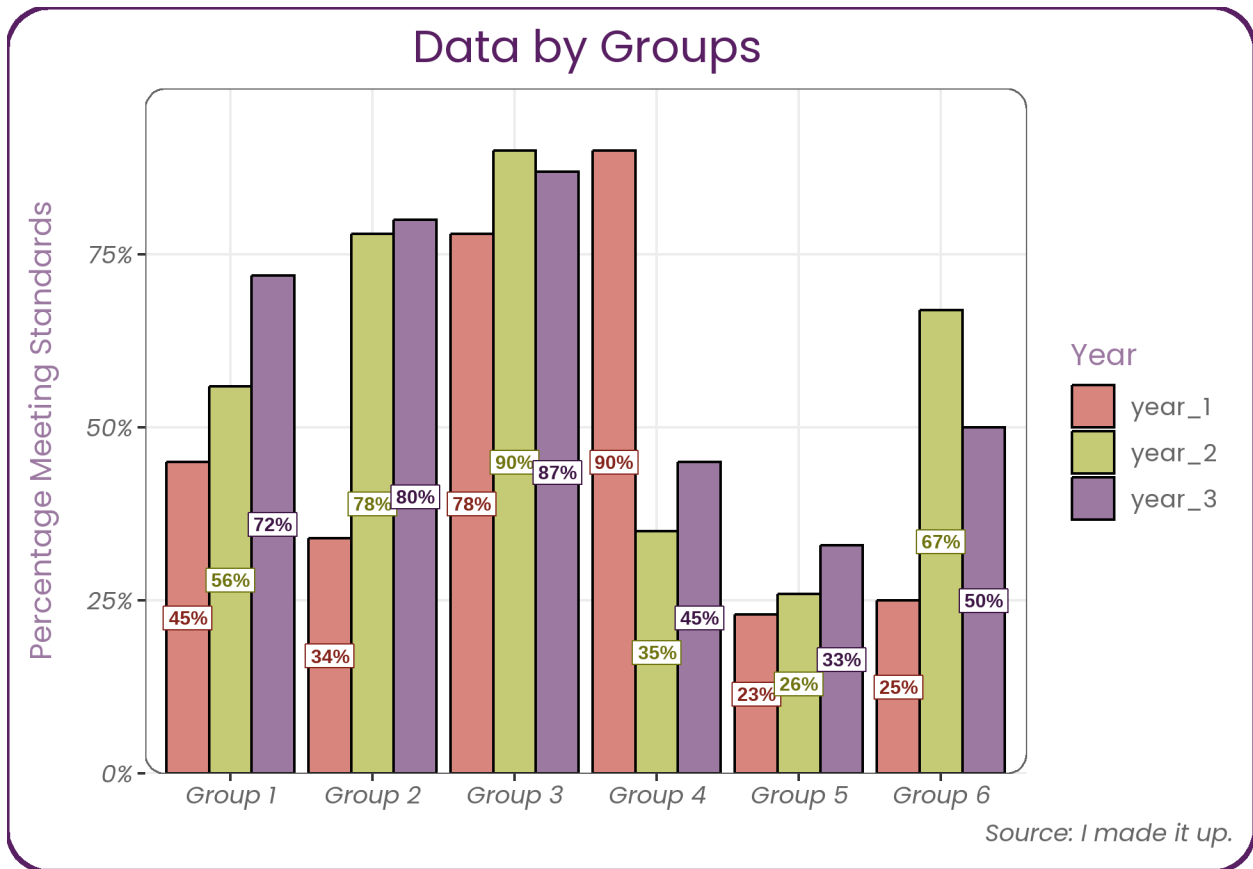
```

18  # Fill for the columns, color for the lines and points
19  scale_fill_manual(values = mps_cols("light wine")) +
20  scale_color_manual(values = mps_cols("light red")) +
21  ## Label the geom_line points; use position_nudge to adjust the labels vertically
22  geom_label_mps(mapping = aes(label = percent(overall), x = year, y = overall),
23                position = position_nudge(y = .05),
24                color = mps_cols("red")) +
25  ## Label the geom_col columns; use position_stack to put labels in center of bars
26  geom_label_mps(mapping = aes(label = percent(group), x = year, y = group),
27                position = position_stack(vjust = 0.5),
28                color = mps_cols("wine")) +
29  ## Label the plot. Use color = NULL and fill = NULL to leave a simple legend
30  labs(
31    title = "Comparing Overall Scores to Group Scores",
32    caption = "Source: I made it up.",
33    color = NULL,
34    fill = NULL,
35    x = NULL,
36    y = "Percentage Passing"
37  )

```



## Vertical Grouped Bars



Plot 3. Grouped vertical bar plot

```

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

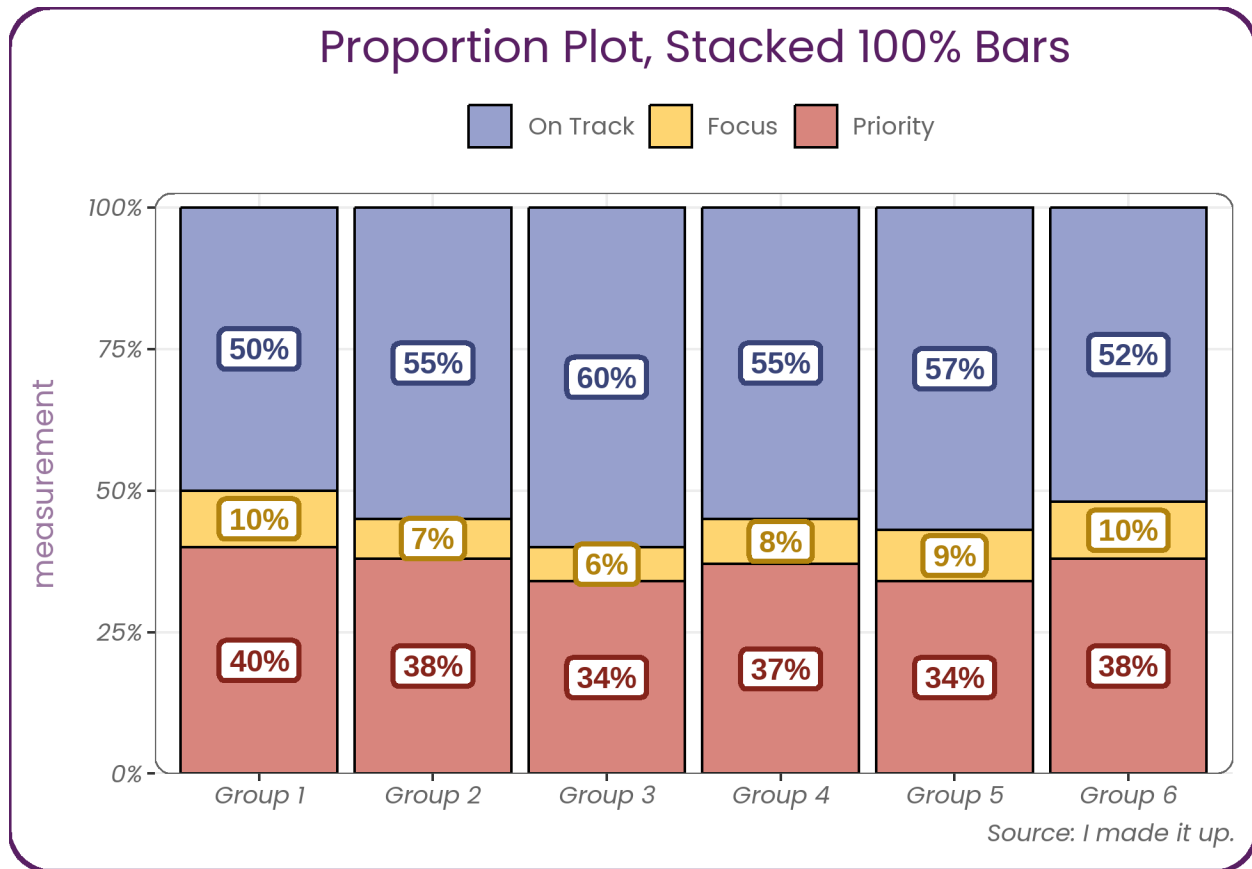
```

```

18     position = position_dodge(width = 0.9),
19     #Reduce the text size to fit the bars
20     size = 2.5,
21     #Reduce the label size to fit the bars
22     label.size = 0.25,
23     #Remove the corner radius
24     label.r = unit(0, "points"),
25     #Adjust padding to get everything to fit
26     label.padding = unit(2, "points")) +
27 ## Use scale_*_mps() to set the color palette
28 # scale_*_manual() allows for specific choices of colors if preferred
29 scale_fill_mps("light") +
30 # scale_color_mps(palette = "hot") +
31 scale_color_mps("dark") +
32 ## Add the labels. x = and y = NULL removes the axis labels
33 labs(
34     title = "Data by Groups",
35     caption = "Source: I made it up.",
36     fill = "Year",
37     x = NULL,
38     y = "Percentage Meeting Standards"
39 )

```

## Vertical Stacked Bars



Plot 4. Stacked vertical bar plot

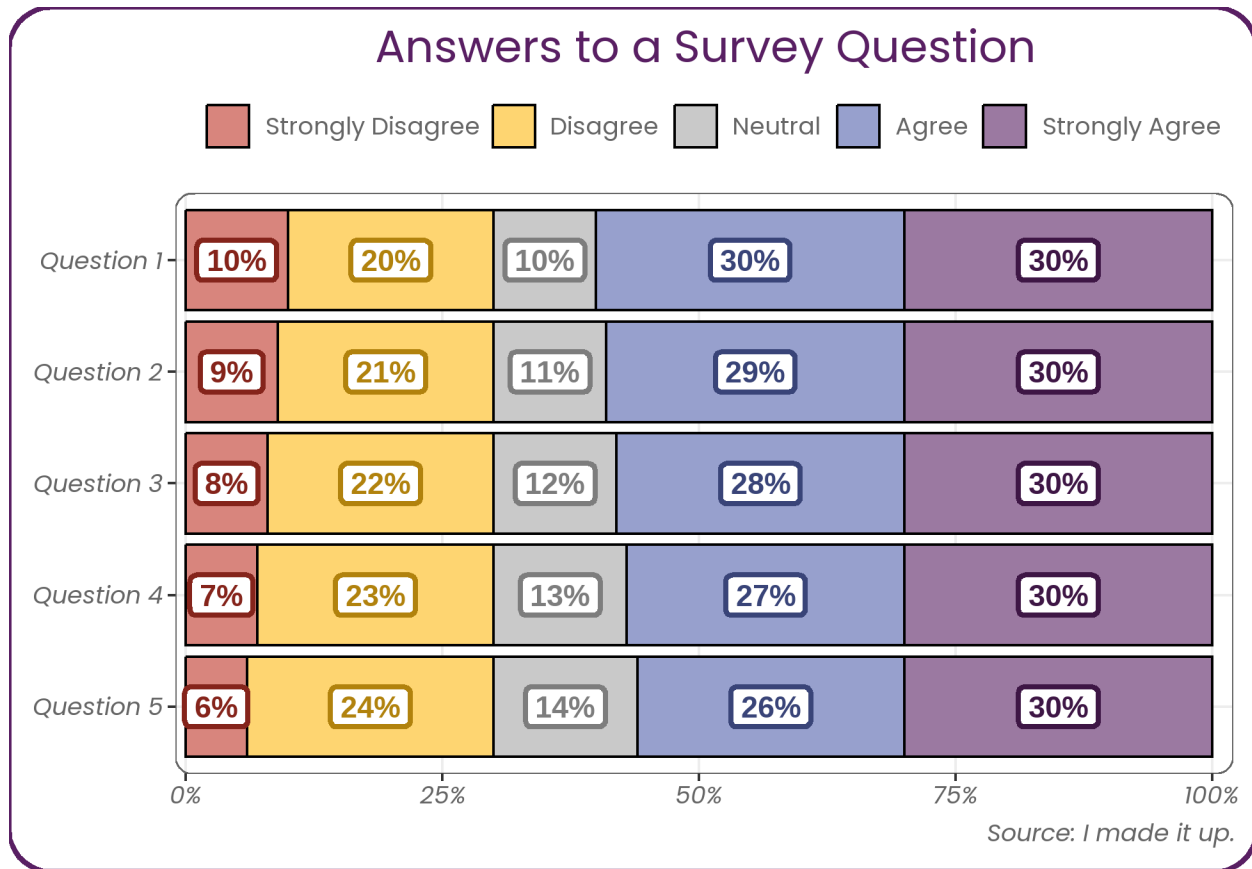
```

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

```

```
18 labs(  
19   title = "Proportion Plot, Stacked 100% Bars",  
20   caption = "Source: I made it up.",  
21   fill = NULL,  
22   x = NULL  
23 )
```

## Horizontal Stacked Bars



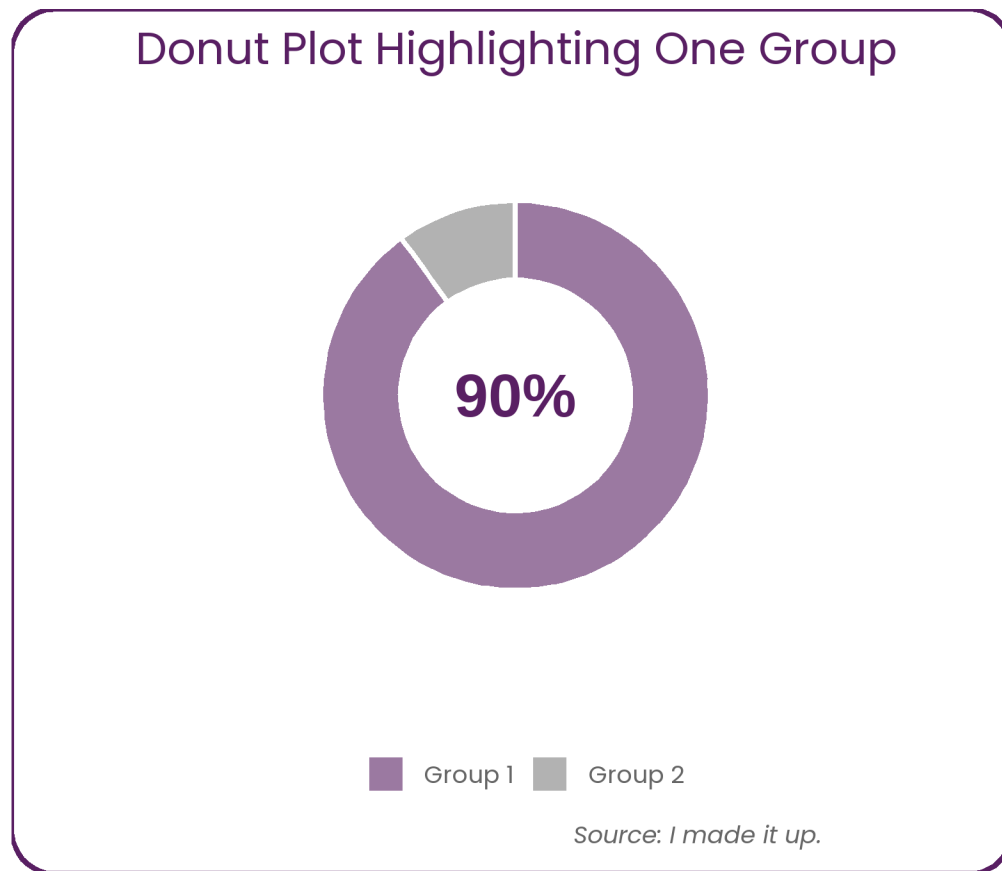
Plot 5. Stacked horizontal bar plot

```

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

```
18 geom_label_mps(mapping = aes(label = percent(Percentage, accuracy = 1),
19                               y = Question,
20                               color = Answer),
21               position = position_stack(vjust = 0.5)) +
22 ## Add plot labels
23 labs(
24   title = "Answers to a Survey Question",
25   caption = "Source: I made it up.",
26   fill = NULL,
27   y = NULL,
28   x = NULL
29 )
```

## Donut Plots – Highlight



*Plot 6. Donut plot highlighting one variable*

```

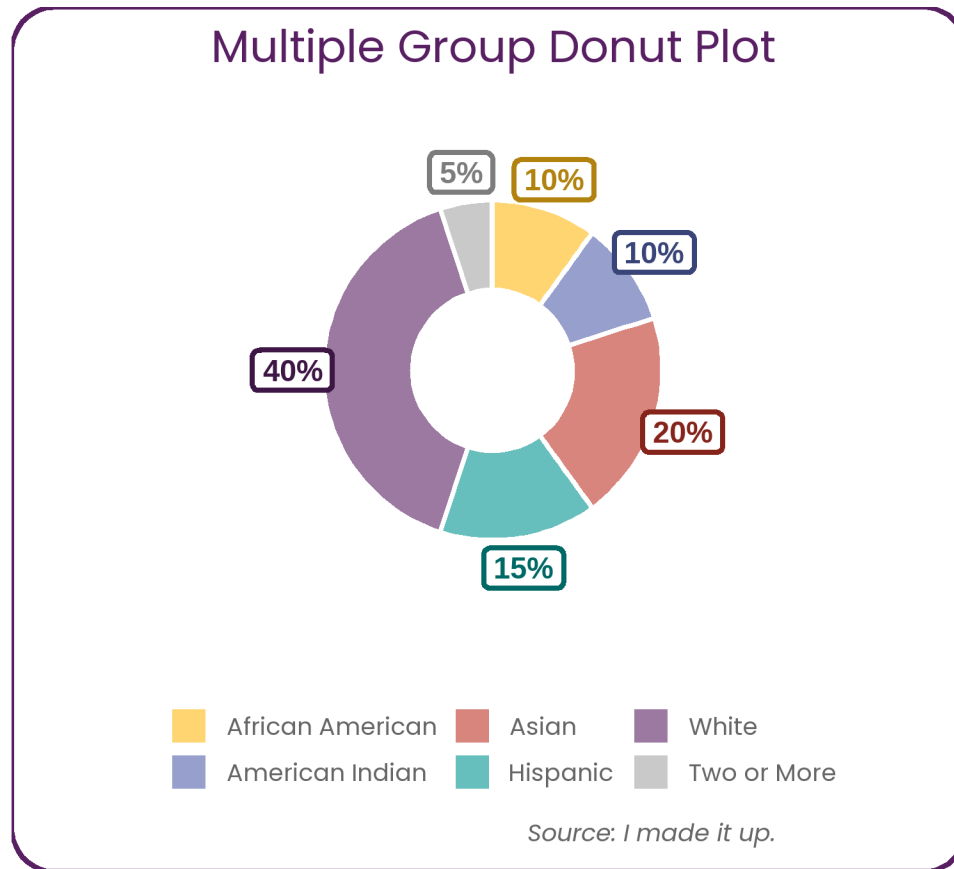
1  ggplot(dnt1, aes(x = 1, y = percent, fill = group)) +
2    # Donut plots are just column plots translated to polar coords
3    geom_col(position = "fill", color = "white", linewidth = 1) +
4    coord_polar(theta = "y", direction = -1) +
5    # xlim() sets the inner and outer size of the ring
6    xlim(c(-.75, 2)) +
7    ## Add the center of the plot annotation
8    # TODO Considering developing a geom_text_donut() function
9    geom_text(
10     data = dnt1_anno,
11     # Set x = to same as minimum xlim, and y = 0 to center the annotation
12     aes(x = -.75, y = 0, label = percent(percent)),
13     inherit.aes = FALSE,
14     fontface = "bold",
15     color = mps_cols("wine"),
16     size = 8
17   ) +

```

```
18  ## Add the scale_*_manual to set the group colors
19  scale_fill_manual(values = mps_cols("light wine", "gray")) +
20  ## Add the donut-specific theme
21  theme_mps_donut() +
22  ## Add the plot labels
23  labs(
24    title = "Donut Plot Highlighting One Group",
25    caption = "Source: I made it up.",
26    fill = NULL
27  )
```



## Donut Plots – Multiple Groups



Plot 7. Normal donut plot

```

1 ggplot(dnt2, aes(x = 1, y = Percent, fill = Ethnicity)) +
2   geom_col(position = "fill", color = "white", linewidth = 1) +
3   coord_polar(theta = "y", direction = -1) +
4   xlim(c(-0.25, 2)) +
5   scale_fill_manual(values = mps_cols("light yellow", "light blue", "light red",
6     "light teal", "light wine", "light gray")) +
7   scale_color_manual(values = mps_cols("dark yellow", "dark blue", "dark red",
8     "dark teal", "dark wine", "dark gray")) +
9   geom_label_mps(mapping = aes(label = percent(Percent),
10     y = Percent,
11     x = 1.75,
12     color = Ethnicity),
13     position = position_stack(vjust = 0.5)) +
14   theme_mps_donut() +
15   labs(
16     title = "Multiple Group Donut Plot",
17     caption = "Source: I made it up.",

```

```
18     fill = NULL
19 )
```

## 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.*

```

1 hbs %>%
2   mutate(Percentage = percent_format()(Percentage)) %>%
3   spread(Answer, Percentage) %>%
4   kable(
5     format = "latex", # Comment this line out while viewing table in-line
6     booktabs = TRUE,
7     align = "lccccc",
8     linesep = ""
9   ) %>%
10  kable_paper(lightable_options = "striped") %>% # For viewing inline while working
11  kable_styling(stripe_color = "mpswine!5", latex_options = c("scale_down",
12    ↪ "HOLD_position", "striped")) %>%
13  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
<code>mps_tcolor('red', 'red')</code>	red
<code>mps_tcolor('orange', 'orange')</code>	orange
<code>mps_tcolor('yellow', 'yellow')</code>	yellow
<code>mps_tcolor('green', 'green')</code>	green
<code>mps_tcolor('teal', 'teal')</code>	teal
<code>mps_tcolor('blue', 'blue')</code>	blue
<code>mps_tcolor('darkblue', 'darkblue')</code>	darkblue
<code>mps_tcolor('wine', 'wine')</code>	wine
<code>mps_tcolor('gray', 'gray')</code>	gray

Table 2. Inline text colors when ‘mps = TRUE’.

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

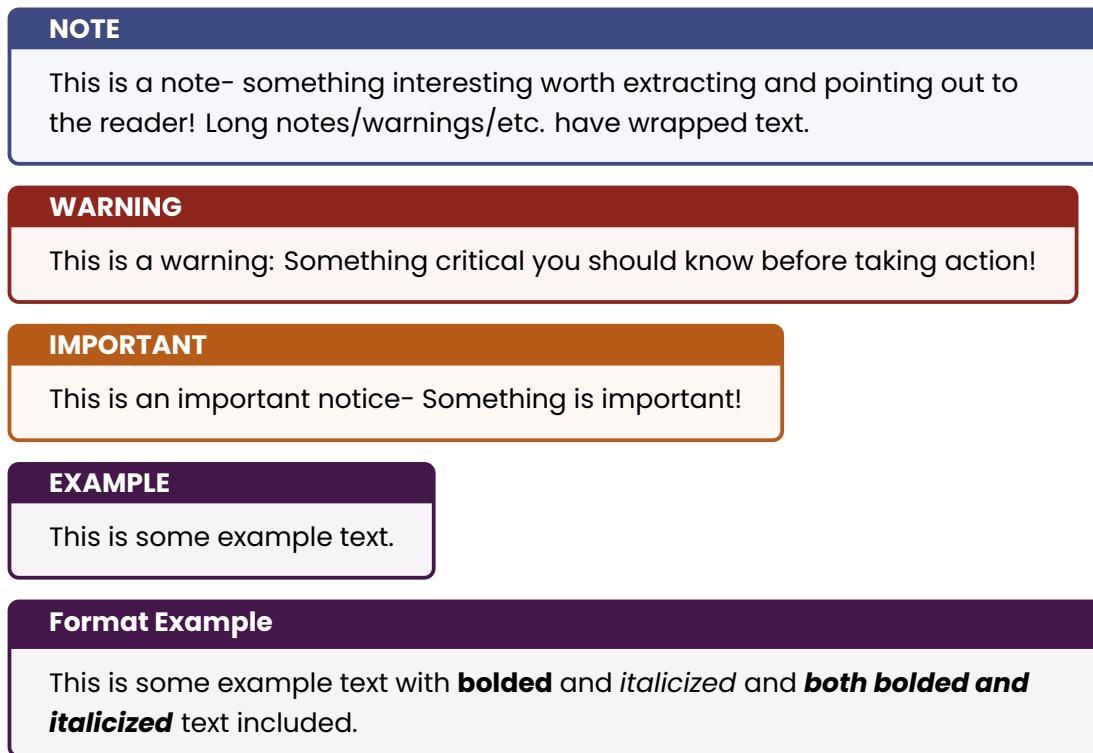


Figure 4. Custom LaTeX color box command outputs.

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

Table 4. Additional LaTeX commands.