Introduction to ggplot

Example Code and Applications

- Why ggplot
- 1. Installation and Set-Up
- 2. Working Example Data
- 3. ggplot basics
- 4. Scales
- 5. Titles, subtitles, & labels: Basics
- 6. Titles, subtitles & labels: Extensions
- 7. Legends
- 8. Reference Lines
- 9. Facets
- 10. Themes
- 11. Combining Plots
- 12. Exporting Plots
- 13. Additional Resources:

Why ggplot

- Consists of an underlying grammar of graphics that allows for easy, structured, logical programming.
- Flexible, allows you to customize almost all aspects of the plot.
- Built in themes that polish the plot's appearance, and makes it more "publication ready."
- Open-sourced and collaborative, so there a number of packages written that extend ggplot to even further capabilities.

1. Installation and Set-Up

```
#install.packages("tidyverse") #only need to run once
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.0.3     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2. Working Example Data

This exercise will use the `storms` data set that comes with `dplyr` . (n=10,010)

```
head(storms)
```

```
## # A tibble: 6 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Amy    1975     6    27     0  27.5 -79   tropi… -1          25     1013
## 2 Amy    1975     6    27     6  28.5 -79   tropi… -1          25     1013
## 3 Amy    1975     6    27    12  29.5 -79   tropi… -1          25     1013
## 4 Amy    1975     6    27    18  30.5 -79   tropi… -1          25     1013
## 5 Amy    1975     6    28     0  31.5 -78.8 tropi… -1          25     1012
## 6 Amy    1975     6    28     6  32.4 -78.7 tropi… -1          25     1012
## # … with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

```
names(storms)
```

```
##  [1] "name"        "year"        "month"       "day"         "hour"
##  [6] "lat"         "long"        "status"      "category"    "wind"
## [11] "pressure"    "ts_diameter" "hu_diameter"
```

3. ggplot basics

- Overarching Idea: Using `ggplot` , we can specify different parts of the plot and combine them together using the "+" operator.
- `ggplot` Foundation:
    - `aes` : (aesthetics), a mapping between a visual cue and a variable. Examples include:
        - position (i.e., on the x and y axes)
        - color ("outside" color)
        - fill ("inside" color)
        - shape (of points)
        - line type
        - size
    - `geoms` : (geometric objects), the actual marks we put on a plot. Examples include:
        - points (geom_point)
        - lines (geom_line)
        - boxplot (geom_boxplot).

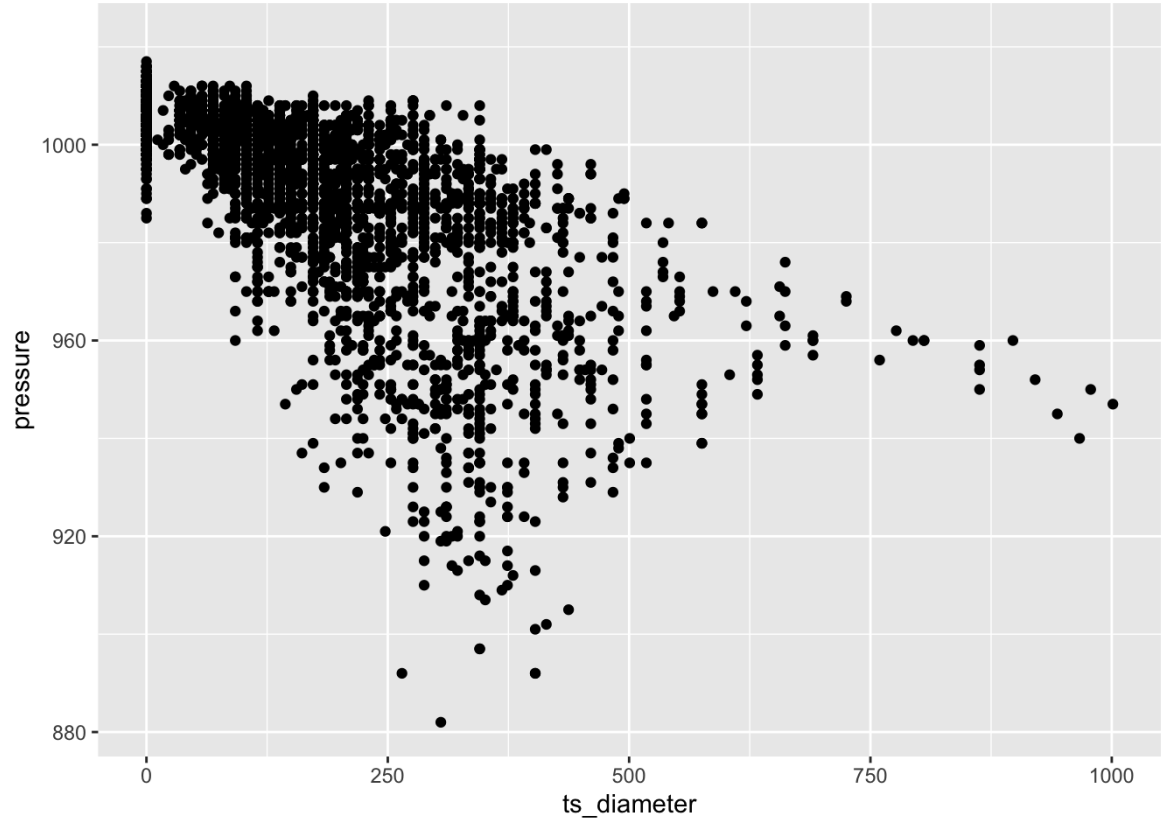
Example Plot 1

This is a simple example using just the foundational `ggplot` plot elements, **aesthetics** and **geoms**. Here, using the storms data, we have mapped `ts_diameter` (wind diameter) to our x variable and `pressure` (air pressure) to our y variable. We are using geom_point() for a scatter plot.

```
ex_plot1 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point()

ex_plot1
```

```
## Warning: Removed 6528 rows containing missing values (geom_point).
```



Note: the "warning" indicates that there were some *NAs* in the variables we were plotting

4. Scales

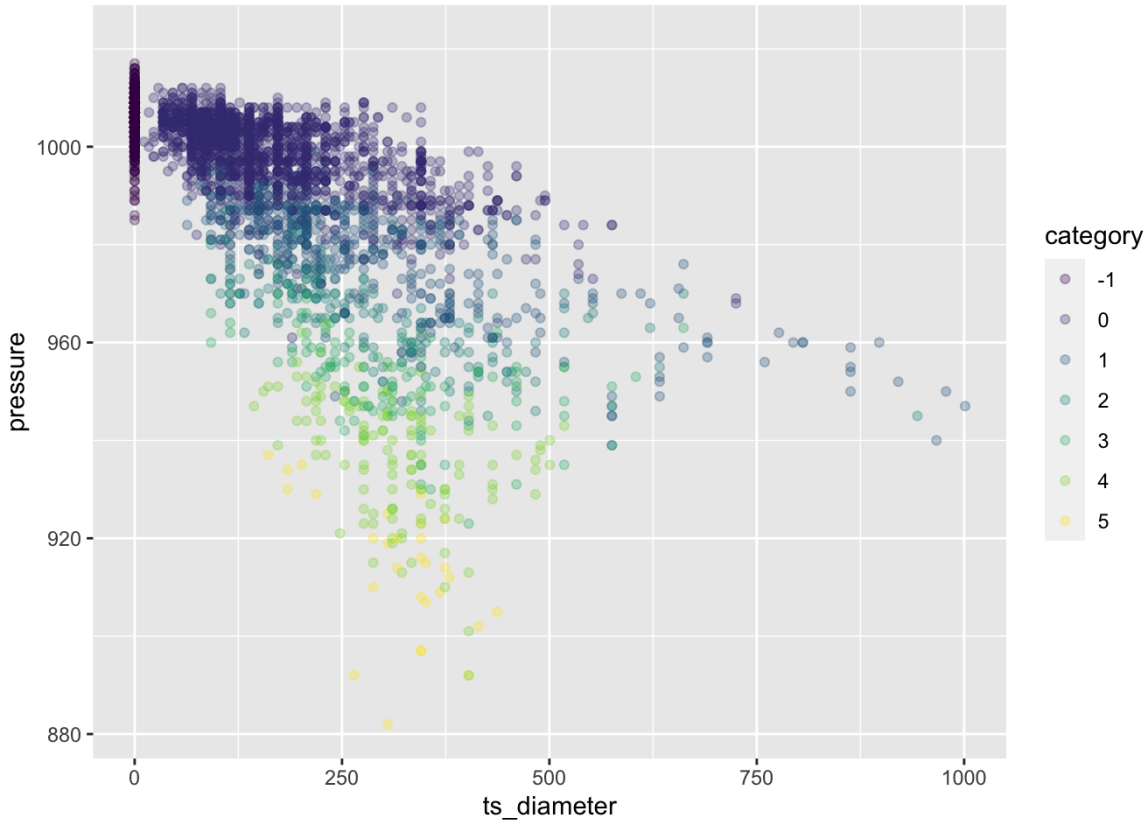
- Aesthetic mapping only says that a variable should be mapped to an aesthetic. It doesn't say how that should happen.
  - Ex. when mapping a variable to color with `aes(color = var3)`, you don't say what color scheme should be used.
- Describing what colors/shapes/sizes etc. to use is done by modifying the corresponding scale - In ggplot2, scales include:
  - Position
  - color, fill, and alpha
  - Size
  - Shape
  - Linetype
- Some scales can be changed within a geom() call, for example:
  - alpha: opacity of a geom with values ranging from 0 to 1. Lower values correspond to more transparent colors.
  - size: width in mm

Example Plot 2

In this first scale example plot, we examine some scales that can be changed from within a `geom()` call. Here, we first map the color of the points to the storm category variable, then we modify some scale attributes by changing the alpha to make the points more transparent, and decreasing the size of the points.

Of note, when you map a variable to an aesthetic, as we did here with color, ggplot automatically generates a legend for you and uses a default color scheme. A separate scale function, like I mentioned previously (and seen in the next example), is needed to customize the colors for the scale.

```
ex_plot2 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category), #also in this ex., we map color to storm category
             alpha = 0.3,
             size = 1.5)
ex_plot2
```



Example Plot 3

Other scales need to be changed in their own statement, following the `scale_<aesthetic>_<type>` naming scheme. For Example:

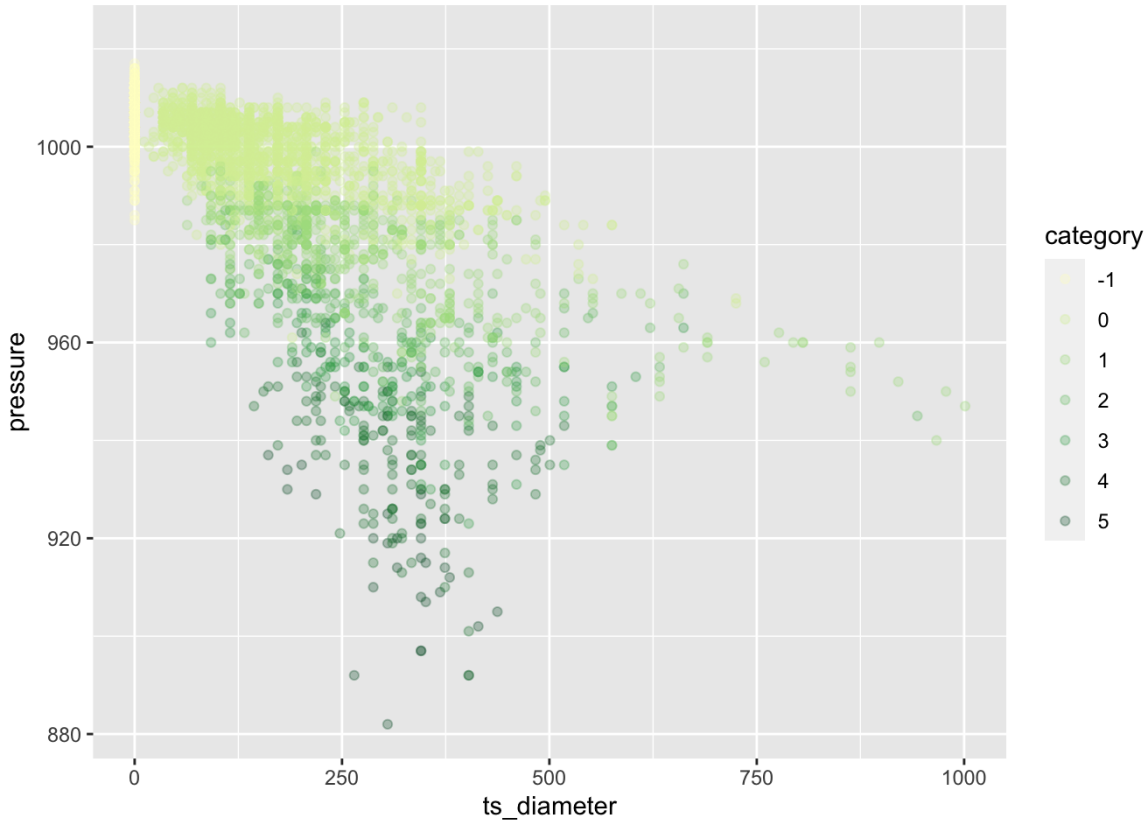
- specifying specific color scales
- changing number breaks/increments
- changing the specific shapes mapped to a variable

In this example plot, I use a neat color scale resource, the `RColorBrewer` package. The package includes color scales that use hand selected colors that are designed to work well in a wide variety of data situations. Also, a really nice feature of this package is that they have created several “color-blind friendly” schemes.

Note: In the next example, I use the *YlGn* scheme.

```
#install.packages("RColorBrewer") #only need to run once
library(RColorBrewer)

ex_plot3 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  scale_colour_brewer(palette = "YlGn")
ex_plot3
```



Custom Color Scales

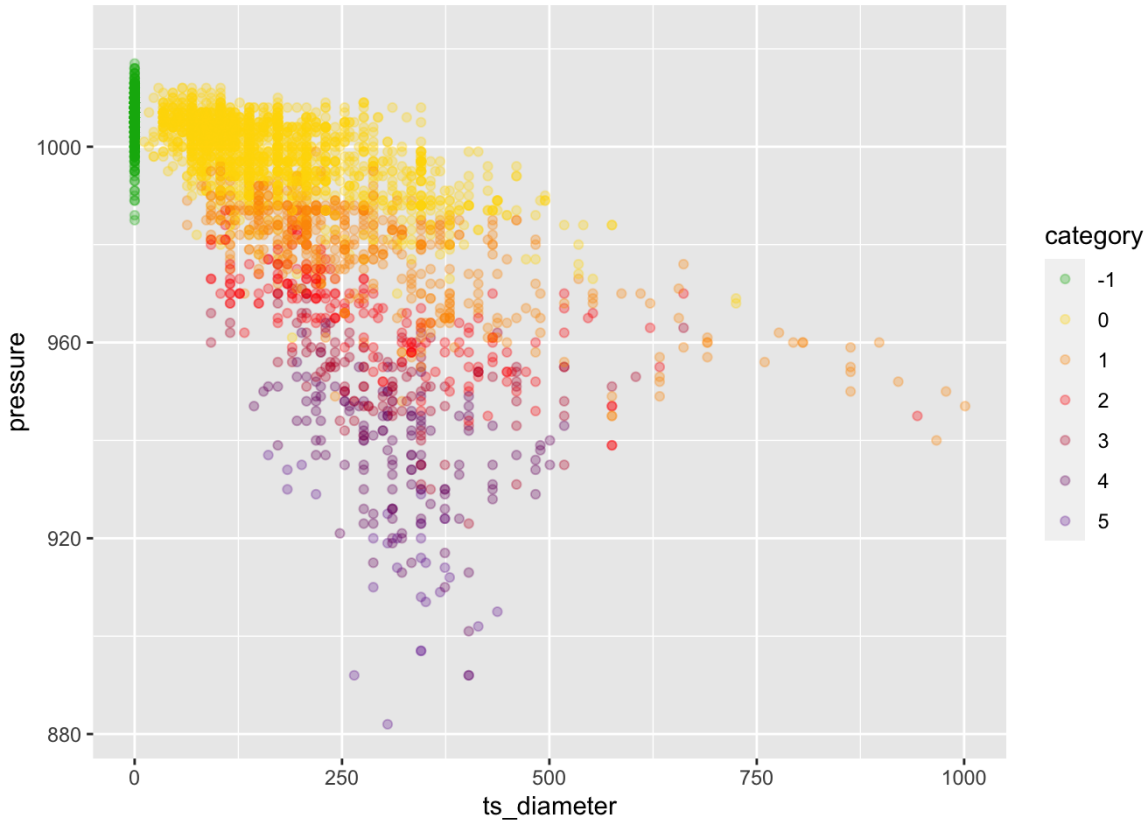
In addition to pre-existing color schemes, you can also generate custom color palettes using `scale_fill_manual` and `scale_color_manual`. Both of these functions accept actual color names like *"red"* or *"blue,"* or you can manually enter hex values for more intricate colors. I use a google chrome eye dropper extension, linked here (https://chrome.google.com/webstore/detail/eye-dropper/hmdcmlfkchdmnmnmheododdhjedfccka?hl=en).

Example Plot 4

In the plot I just created, the yellow and light green colors are kind of hard to see, so I may want to change that. For example, I can change the color scheme of our plot to match the a storm weather radar color scheme, since this is storm data, after all(!). In this example plot, I do that using the google chrome eye dropper and `scale_color_manual` function.

```
ex_plot4 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1")) #pass in color codes

ex_plot4
```



Note: - Order of colors codes passed in matches the order of the mapped variables:

- Category -1 = #00b10d
- Category 0 = #ffd800
- Etc.

For **continuous variables**, you can add a `breaks=c()` to indicate where you want the color change to occur:

- Ex: `scale_color_manual(values = c("red", "blue", "green"), breaks = c(0, 5, 10))`

5. Titles, subtitles, & labels: Basics

Main functions for adding titles, subtitles & labels:

- `Labs(x = , y = , title = , subtitle = , caption = )`
- `ggtitle(title = , subtitle = )`
- `scale_y/x_continuous/discrete(name =)`

Titles and Subtitles:

- `ggtitle()` function
- `labs()` function

Labels:

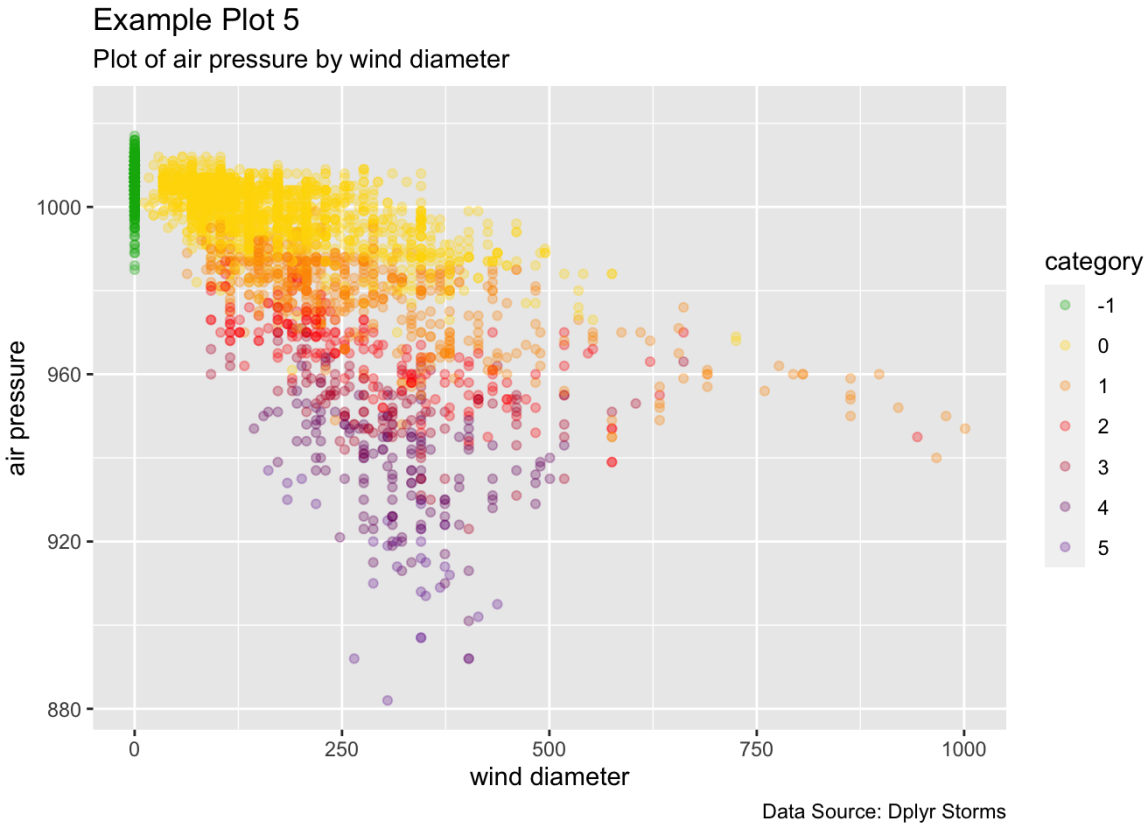
- `labs()` function
- `scale_x_continuous()`
- `scale_x_discrete()`
- `scale_y_continuous()`
- `scale_y_discrete()`

Example Plot 5

In this example, I added the `labs()` function to my plot for all plot titles, subtitles, captions and axes labels.

```
ex_plot5 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 5",
       subtitle = "Plot of air pressure by wind diameter", caption = "Data Source: Dplyr Storms")

ex_plot5
```



Alternative coding options: -

```
ggtitle(title = "Example Plot 5", subtitle = "Plot of air pressure by wind diameter") -
scale_y_continuous(name = "air pressure") + scale_x_continuous(name = "wind diameter")
```

6. Titles, subtitles & labels: Extensions

You can also use `theme()` to change the face, font, and size of a title, subtitle or label, as seen in Example Plot 6.

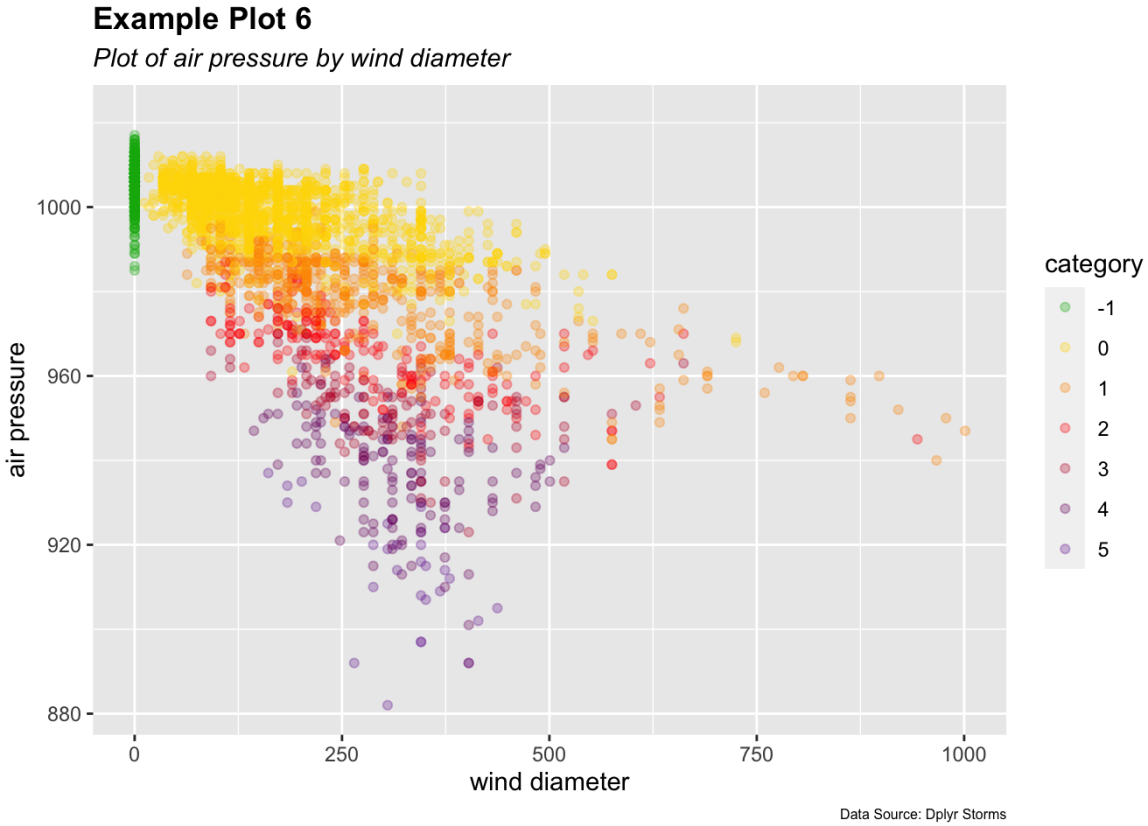
- Ex. `theme(plot.title = element_text(color = "", size = , face = "")`

Example Plot 6

In this example, using one theme function, I changed the face of the title to bold, the face of the subtitle to italics, and decreased the size of the caption to 6pt.

```
ex_plot6 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 6",
       subtitle = "Plot of air pressure by wind diameter", caption = "Data Source: Dplyr Storms") +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 6))

ex_plot6
```



7. Legends

Our plot currently has the default legend that ggplot generates any time you map an aesthetic to a variable. We can also customize a legend using separate ggplot functions.

- **Changing Legend title:**
  - From within the `lab()` function, specify color = “legend name” (or fill = , shape = , etc.)
- **Changing Legend Labels:**
  - Use the respective scale() function `scale_color_manual()`, `scale_color_discrete()`, `scale_color_continuous()` etc.
- **Changing Legend Background Color:**
  - From within the `theme()` function, use `legend.background = element_rect(fill = "<color>")`
- **Changing Legend Position:**
  - From within the `theme()` function, use `legend.position = “top/bottom/bottom left”`

This is a far from exhaustive list of legend capabilities.

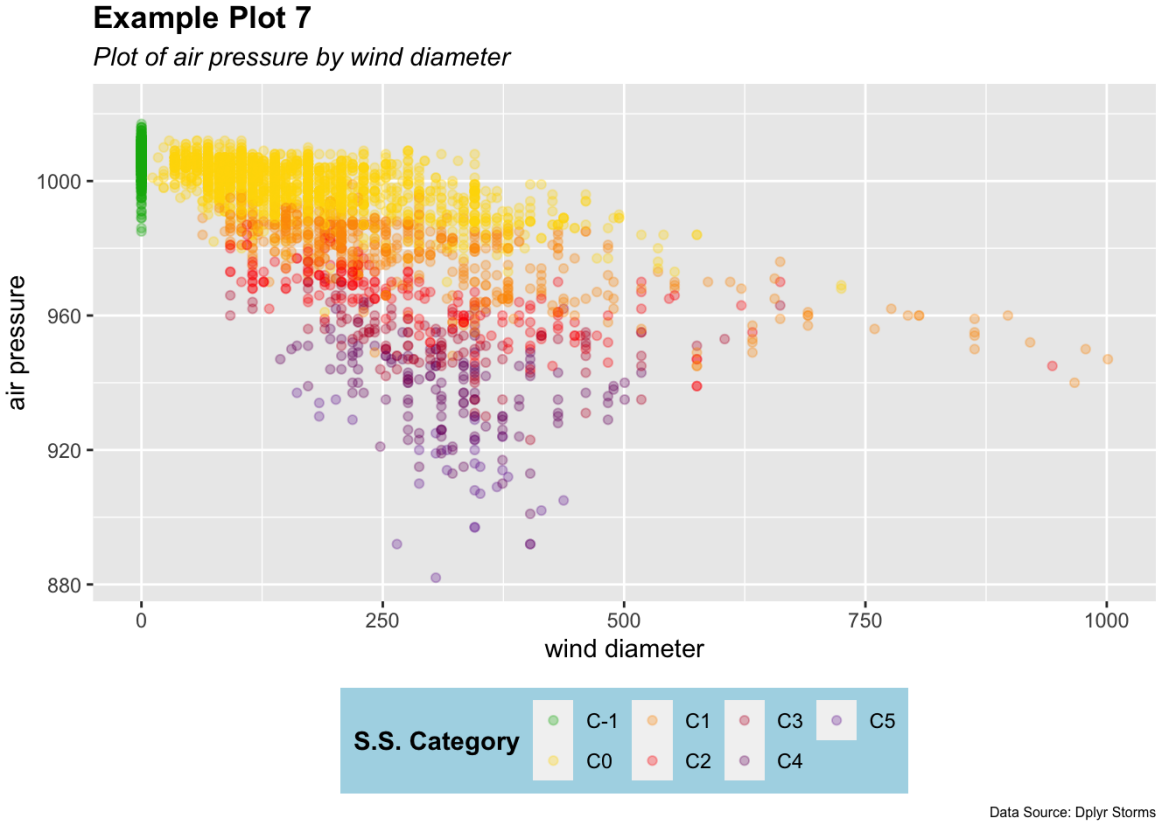
Example Plot 7

In this example, I manipulated 5 aspects of the default legend:

1. I changed the legend level labels to include a C for category.
2. I changed the the legend title to S.S. category to reflect the official name of the category system, *Saffir Simpson*.
3. I then changed three aspects of the legend within the `theme()` function:
    a. the color of the legend background to light blue
    b. the position of the legend to the bottom
    c. the face of the legend title to bold

```
ex_plot7 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1"),
                    labels = c("C-1", "C0", "C1", "C2", "C3", "C4", "C5")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 7",
       subtitle = "Plot of air pressure by wind diameter",
       caption = "Data Source: Dplyr Storms",
       color = "S.S. Category") +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 6),
        legend.background = element_rect(fill = "light blue"),
        legend.position = "bottom",
        legend.title = element_text(face = "bold"))

ex_plot7
```



8. Reference Lines

Reference lines, smoothers, other regression componenents are really easy to add: Just include an additional `geom()` !

There are 4 `geoms()` commonly used for these plot elements:

- `geom_vline()` for vertical reference lines
- `geom_hline()` for horizontal reference lines
- `geom_abline()` for reference lines with a slope and intercept. (note this is usually the go-to function for including a reference regression line)
- `geom_smooth()` for adding a smoother on top of your data

Example Plot 8

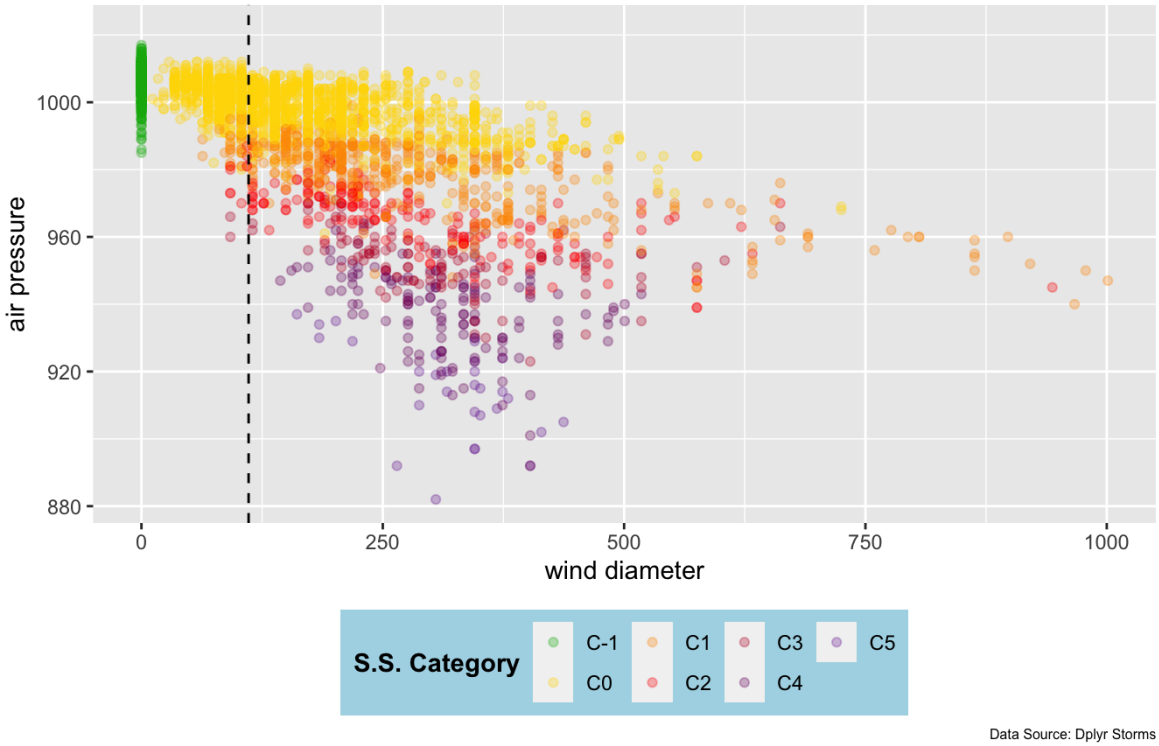
In this example, I will be leveraging information about the Saffir Simpson storm category system used in this plot to add an informative reference line. According to the S.S. system, storm damages become devastating or worse as categories >= 3. This corresponds to a wind speed of 111 mph. So, lets add a reference line at wind diameter (x axis) = 111. We accomplish this with a `geom_vline()`.

```
ex_plot8 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  geom_vline(xintercept = 111, linetype = "dashed") + #linetype = dashed, default is solid
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1"),
                    labels = c("C-1", "C0", "C1", "C2", "C3", "C4", "C5")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 8",
       subtitle = "Plot of air pressure by wind diameter",
       caption = "Data Source: Dplyr Storms",
       color = "S.S. Category") +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 6),
        legend.background = element_rect(fill = "light blue"),
        legend.position = "bottom",
        legend.title = element_text(face = "bold"))

ex_plot8
```

Example Plot 8

Plot of air pressure by wind diameter



Data Source: Dplyr Storms

9. Facets

Faceting is `ggplot` 's way for partitioning a plot into a matrix of panels, where each panel shows a different subset of the data.

2 functions for creating facets: - `facet_wrap()` : define subsets as the levels of a single grouping variable - `facet_grid()` : define subsets as the crossing of two grouping variables

Goal: Facilitates comparison among subsets

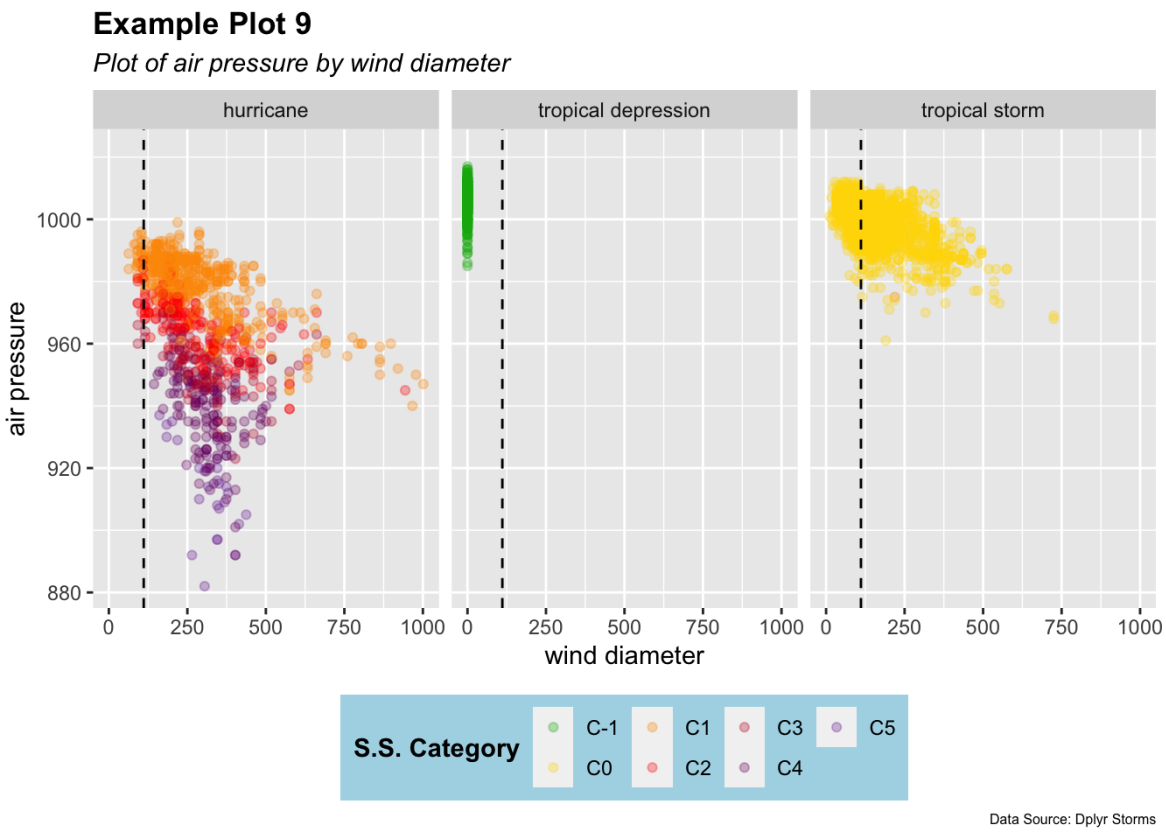
Visual elements of facets can be changed from within the `theme()` function using the `strip.background()` function. Here, you can change the color of the outline of the facet label, the fill of the facet label box, and the type of outline line used.

Example Plot 9

In this example, suppose I want to know how the relationship between wind diameter and air pressure by storm category varies based on the storm status (tropical depression, tropical storm, or hurricane). I can do this using the facet_wrap() function on the status variable, below.

```
ex_plot9 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  geom_vline(xintercept = 111, linetype = "dashed") +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1"),
                    labels = c("C-1", "C0", "C1", "C2", "C3", "C4", "C5")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 9",
       subtitle = "Plot of air pressure by wind diameter",
       caption = "Data Source: Dplyr Storms",
       color = "S.S. Category") +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 6),
        legend.background = element_rect(fill = "light blue"),
        legend.position = "bottom",
        legend.title = element_text(face = "bold")) +
  facet_wrap(~ status)

ex_plot9
```



Reording Facets

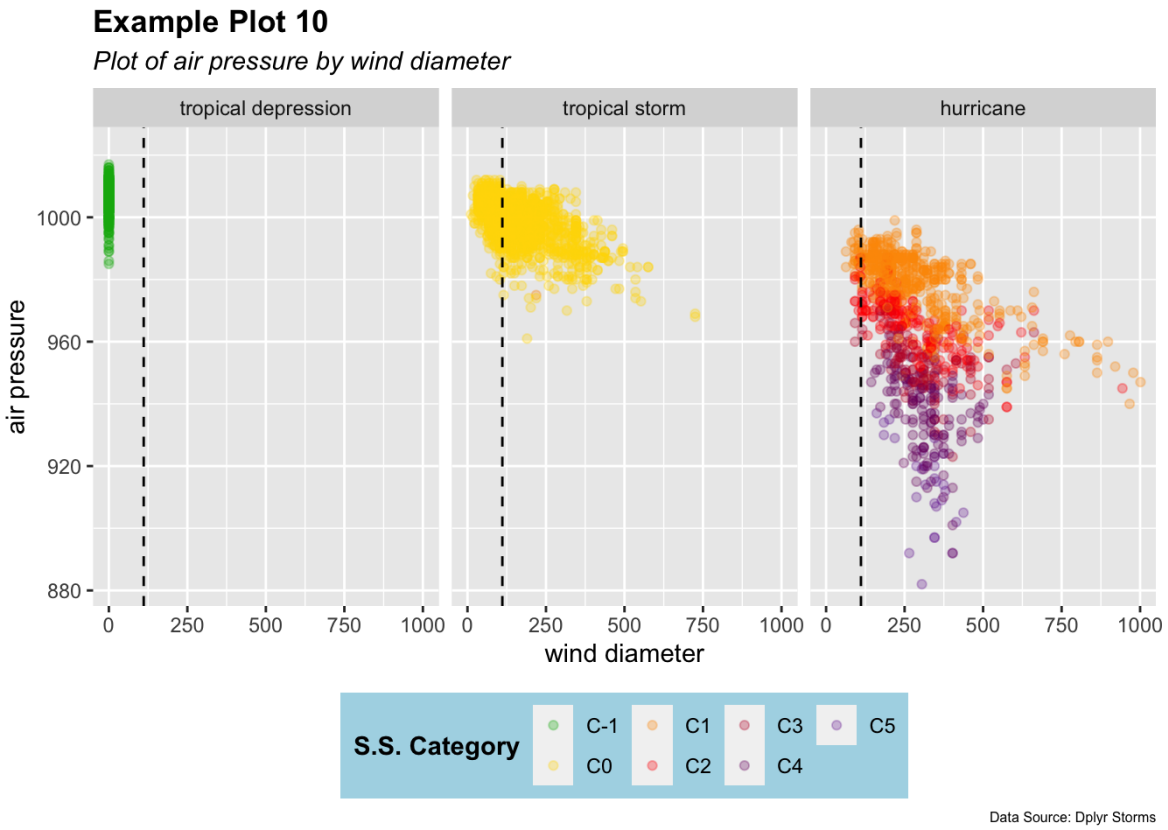
We can also re-order the facets. For example, suppose we wanted to reorder the storm statuses to reflect increasing severity (first tropical depression, then tropical storm, last hurricane).

We can do that using the `factor(var, levels = c("level1", "level2", … , "leveln"))` convention. See plot clode below.

Example Plot 10

```
ex_plot10 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  geom_vline(xintercept = 111, linetype = "dashed") +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1"),
                    labels = c("C-1", "C0", "C1", "C2", "C3", "C4", "C5")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 10",
       subtitle = "Plot of air pressure by wind diameter",
       caption = "Data Source: Dplyr Storms",
       color = "S.S. Category") +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 6),
        legend.background = element_rect(fill = "light blue"),
        legend.position = "bottom",
        legend.title = element_text(face = "bold")) +
  facet_wrap(~ factor(status, levels = c("tropical depression", "tropical storm", "hurricane")))

ex_plot10
```



10. Themes

ggplot has several build-in themes that you can apply to your visualization. A comprehensive list can be found here, with images of how each theme displays. Examples include:

- `theme_gray()` : The signature ggplot2 theme with a grey background and white gridlines, designed to put the data forward yet make comparisons easy.
- `theme_bw()` : The classic dark-on-light ggplot2 theme. May work better for presentations displayed with a projector.
- `theme_linedraw()` : A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawing. Serves a purpose similar `theme_bw()` . Note that this theme has some very thin lines (<< 1 pt) which some journals may

refuse.

- `theme_minimal()` : A minimalistic theme with no background annotations.
- `theme_classic()` : A classic-looking theme, with x and y axis lines and no gridlines.

Even more themes can be used from external packages such as `ggthemes` (https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/) and `ggthmr` (https://github.com/Mikata-Project/ggthemr).

Example Plot 11

Using the ggthemes package, we can apply the `theme_bw()` theme:

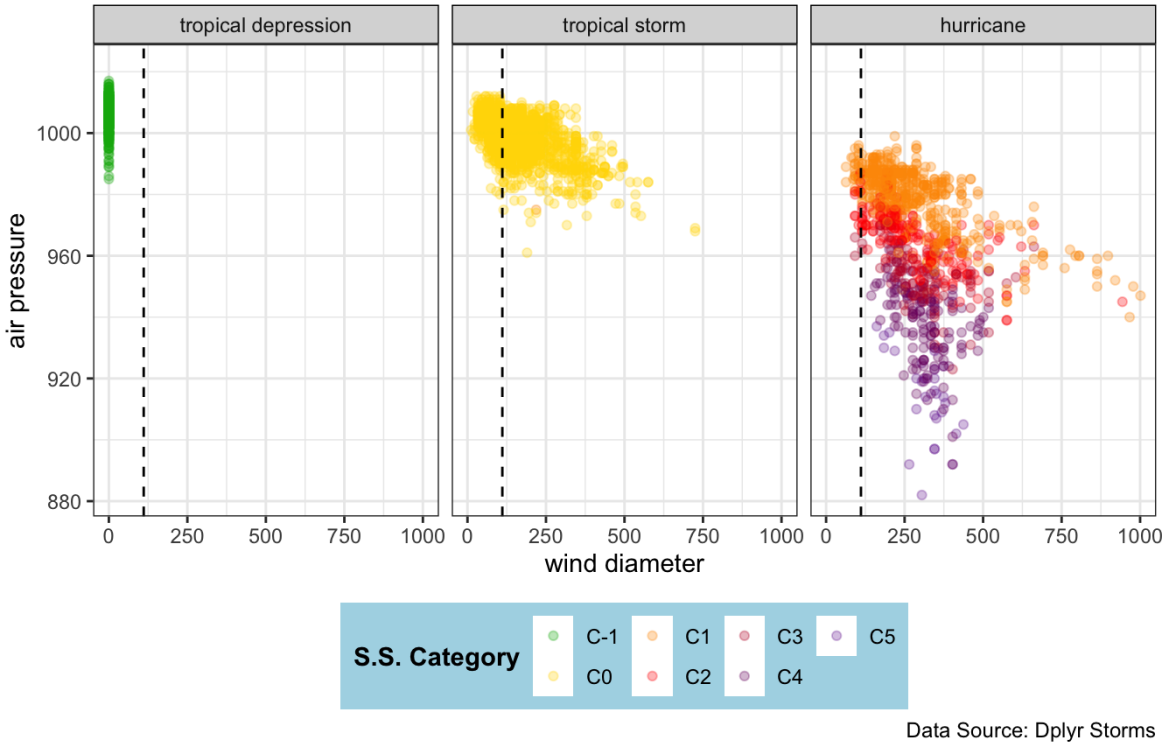
```
#install.packages("ggthemes") #only need to run once
library(ggthemes)


ex_plot11 <- ggplot(data = storms, aes(x = ts_diameter, y = pressure)) +
  geom_point(aes(color = category),
             alpha = 0.3,
             size = 1.5) +
  geom_vline(xintercept = 111, linetype = "dashed") +
  scale_color_manual(values=c("#00b10d", "#ffd800", "#ff9901", "#ff0100", "#be0032",
                    "#79016d", "#7a2fa1"),
                    labels = c("C-1", "C0", "C1", "C2", "C3", "C4", "C5")) +
  labs(x = "wind diameter", y = "air pressure", title = "Example Plot 11",
       subtitle = "Plot of air pressure by wind diameter",
       caption = "Data Source: Dplyr Storms",
       color = "S.S. Category") +
  theme_bw() +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(size = 9),
        legend.background = element_rect(fill = "light blue"),
        legend.position = "bottom",
        legend.title = element_text(face = "bold"),
        strip.background = element_rect(color = "black", size = 0.5),) +
  facet_wrap(~ factor(status, levels = c("tropical depression", "tropical storm", "hurricane")))

ex_plot11
```

Example Plot 11

Plot of air pressure by wind diameter



Data Source: Dplyr Storms

Note: I've found that applying the overall theme first (`theme_bw`) is good practice, because sometimes it has elements that might over-ride the general `theme()` call.

11. Combining Plots

Sometimes, you want a **figure** that has **multiple plots** combined.

The R packages, `gridExtra` (https://cran.r-project.org/web/packages/gridExtra/index.html) and `cowplot` (https://cran.r-project.org/web/packages/cowplot/vignettes/introduction.html#:~:text=2020%2D12%2D15,or%20mix%20plots%20with%20images) can do this!

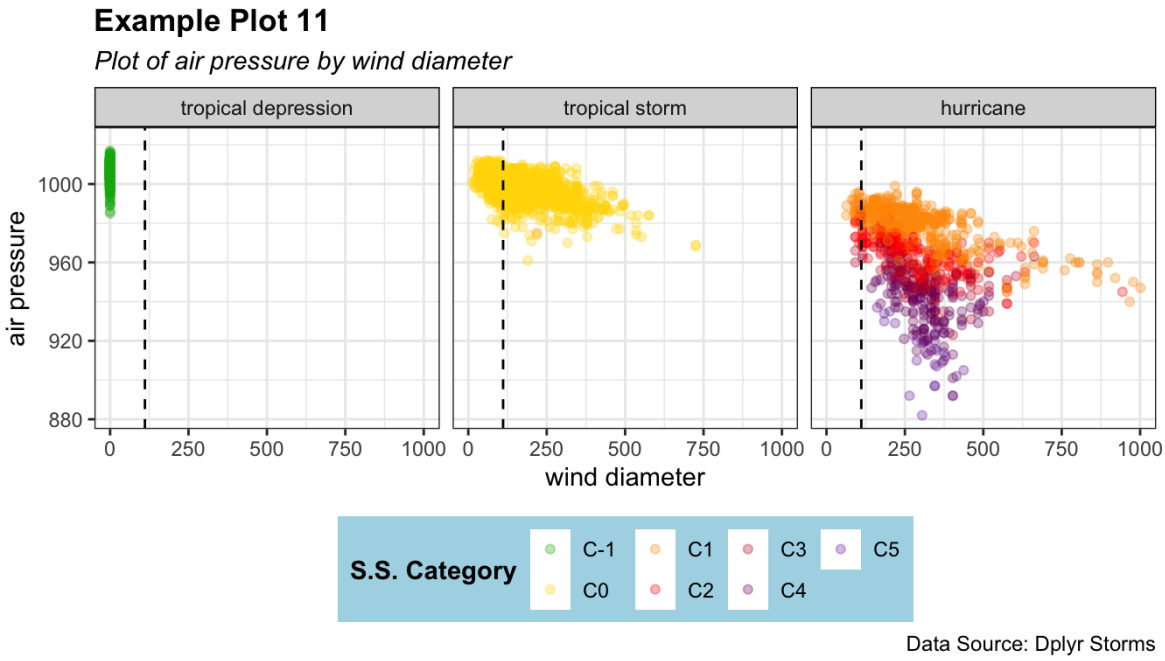
In this example, we will use `cowplot`. `Cowplot` operates on an (X, Y) coordinate plane where X and Y go from 0 to 1. The `draw_plot` function (seen below) takes on 5 arguments, the plot to be included in the figure, the x and y coordinates for the bottom left corner of that plot, and the width and height for the plot.

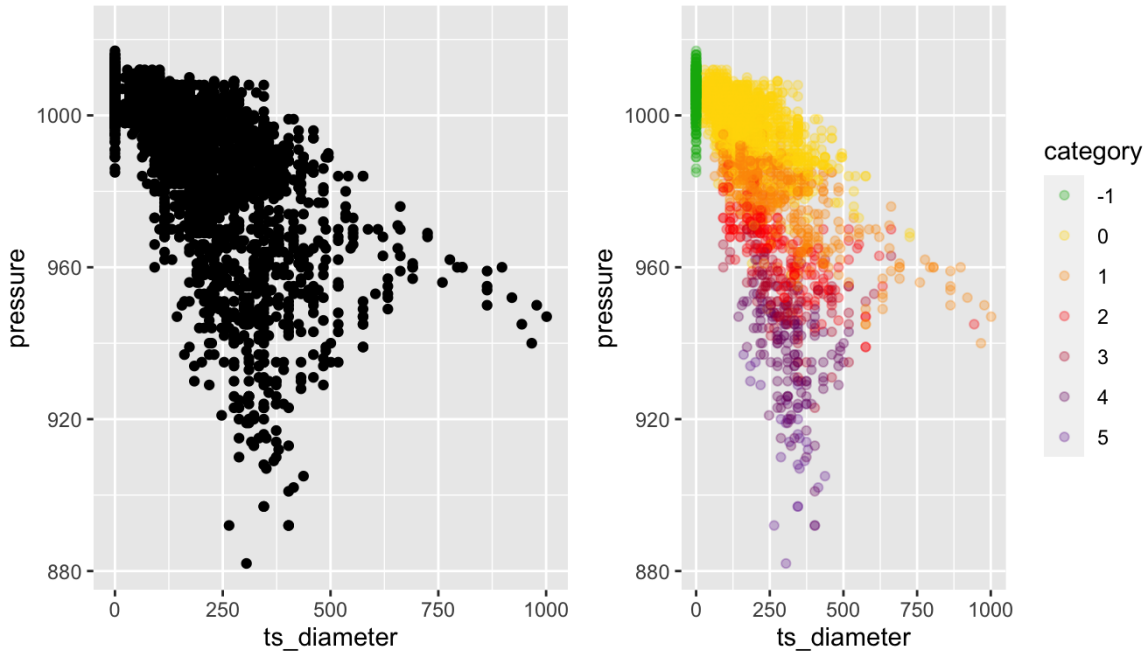
Example Plot 12

In this example, `ex_plot1's` bottom left corner will begin at the origin (x and y are set to 0) and will take up half of the x axis and y axis in width and height, respectively (height and width are set to 0.5).

```
#install.packages("cowplot") only need to run this once
library(cowplot)

ggdraw() +
  draw_plot(ex_plot11, x = 0, y = 0.5, width = 1, height = 0.5) +
  draw_plot(ex_plot1, x = 0, y = 0, width = 0.5, height = 0.5) +
  draw_plot(ex_plot4, x = 0.5, y = 0, width = 0.5, height = 0.5)
```





12. Exporting Plots

Once you have a plot or figure that you're ready to use, you can export it to multiple formats using one of **3 main methods**.

- From within R Studio: export > save as Image or PDF -Advantage: you can specify extension (.jpeg, .png, etc.) and the exact width and height.
- In R code: Use dev system:

```
#png('~/Desktop/rplot.png', height = 500, width = 500)
ex_plot11
dev.off()
```

- From within R Studio: zoom plot > drag corner to adjust sizing > right click > save image as
  - Advantage: allows you to play with sizing "live"
  - Disadvantage: unknown dimensions

13. Additional Resources:

- *Modern Data Science with R* (Baumer, Kaplan, Horton)
- Dr. Ben Baumer's Graphics with `ggplot` Github lecture as part of a Smith College course linked here (https://beanumber.github.io/sds192/index.html)
- `ggplot` cheat-sheet here (https://www.rstudio.com/resources/cheatsheets/)