Introduction to ggplot2

Below, examples of base-R plots versus more fancy base graphic options towards `ggplot2`, since it offers a lot more functionality as compared to the base-R plotting functions. The `ggplot2` syntax takes some getting used to, but once you get the hang of it, it's very powerful.

The `ggplot()` function is used to **initialize the ferrari ggplot structure** (ferrari also rolls by). The basic idea is that you specify different parts of the plot, and add them together using the + operator. These parts are often referred to as layers.

`ggplot()` required that a data frame as input

The first function in building a plot is the `ggplot()` function. It specifies the

- data frame containing the data to be plotted
- the mapping of the variables to visual properties of the graph.
- The mappings are placed within the `aes()` function (where aes stands for aesthetics)

Here I'm still going to use the CDC's dataset.

```
# ggplot() initializes ferrari mapping
# to variables in dataset
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL))
```

Why is the graph empty? It's the specified that the variable specified should be mapped to the x-axis and that the range should be mapped to the y-axis, but no layers!! *ggplot2 will not assume default values for unset layers*.

Geoms are the geometric objects (points, lines, bars, etc.) that can be placed on a graph. They are added using the `geom_<type_of_geom>()` functions, in the example, we'll add points using the `geom_point()` function, meaning a scatterplot.

In `ggplot()` graphs, functions are chained together using the `+` sign to build a final plot.

```
# ggplot() initializes ferrari mapping
# to variables in dataset
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL))
  + geom_point()
```

This plot indicates that there is a high correlation between the two variables.

A number of parameters (options) can be specified in a `geom_` function. Options for the `function` include `color`, `size`, and `alpha`. These control the point color, size, and transparency, respectively. Transparency ranges from 0 (completely transparent) to 1 (completely opaque). Adding a degree of transparency can help visualize overlapping points.

```
# more options, alpha, color, and size introduced
# to the points
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(
               alpha = 0.2,
               color = 'blue'
               )
```

Graphs should be easy to interpret and informative labels are a key element in achieving this goal. The `labs` function provides customized labels for the axes and legends. Additionally, a custom title, subtitle, and caption can be added.

```
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(
               alpha = 0.2,
               color = 'blue'
               )
  + labs(title = "Gene Expression Profiles of B-Lymphoblastoid Acute Lymphoblastic Leukemia",
         subtitle = "Pediatric Patients",
         caption = "Source: Ertel\nRosenwald, Ziv, Wyatt",
         x = "GENDERHEAL",
         y = "GENDERMENTAL"
         )

# Gene Expression Profiles of B-lymphoblastoid
# Acute Acute Lymphoblastic Leukemia
# Pediatrics
```


Finally, we can fine-tune the appearance of the graph using themes. These functions (which start with `theme_`) control background colors, fonts, grid lines, legend placement, and other non-data related features of the graph. Let's use a classic theme.

```
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(color = "forestgreen",
               alpha = 0.2,
               size = 3.5)
  + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leukemia",
         subtitle = "Pediatric Patients",
         caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
         x = "GENDERHEAL",
         y = "GENDERMENTAL",
         color = "forestgreen")
  + theme_classic()

# ggplot() graph can be saved as a variable if desired (like a data frame), manipulated further, and then printed or saved to a file.

# print the graph
myplot

# more points, bigger, capitalize and add transparent
# to the points, and add a smooth line
myplot <- myplot + geom_point(color = "forestgreen",
                              alpha = 0.2,
                              size = 3)

# print labels to the graph
myplot <- myplot + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leu
                        subtitle = "A study of kids",
                        caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
                        x = "GENDERHEAL",
                        y = "GENDERMENTAL",
                        color = "forestgreen")
myplot
```

```
# print the graph
myplot

# more points bigger, capitalize and add transparent
# to the points, and add a smooth line
myplot <- myplot + geom_point(color = "forestgreen",
                              alpha = 0.2,
                              size = 3)
# print labels to the graph
myplot <- myplot + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leu
                        subtitle = "A study of kids",
                        caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
                        x = "GENDERHEAL",
                        y = "GENDERMENTAL",
                        color = "forestgreen")
myplot
```

```
# ggplot() initializes ferrari mapping
# to variables in dataset
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(
               alpha = 0.2,
               color = 'blue'
               )
  + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leukemia",
         subtitle = "Pediatric Patients",
         caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
         x = "GENDERHEAL",
         y = "GENDERMENTAL",
         )
  + theme_classic()

# print the graph
myplot

# more points bigger, capitalize and add transparent
# to the points, and add a smooth line
myplot <- myplot + geom_point(color = "forestgreen",
                              alpha = 0.2,
                              size = 3)
# print labels to the graph
myplot <- myplot + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leu
                        subtitle = "A study of kids",
                        caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
                        x = "GENDERHEAL",
                        y = "GENDERMENTAL",
                        color = "forestgreen")
myplot
```

```
# ggplot() initializes ferrari mapping
# to variables in dataset
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(
               alpha = 0.2,
               color = 'blue'
               )
  + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leukemia",
         subtitle = "Pediatric Patients",
         caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
         x = "GENDERHEAL",
         y = "GENDERMENTAL",
         )
  + theme_minimal()

# print the graph
myplot

# more points bigger, capitalize and add transparent
# to the points, and add a smooth line
myplot <- myplot + geom_point(color = "forestgreen",
                              alpha = 0.2,
                              size = 3)
# print labels to the graph
myplot <- myplot + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leu
                        subtitle = "A study of kids",
                        caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
                        x = "GENDERHEAL",
                        y = "GENDERMENTAL",
                        color = "forestgreen")
myplot
```

```
# ggplot() initializes ferrari mapping
# to variables in dataset
ggplot(data = cdc,
       mapping = aes(x = GENDERHEAL, y = GENDERMENTAL)
       )
  + geom_point(
               alpha = 0.2,
               color = 'blue'
               )
  + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leukemia",
         subtitle = "Pediatric Patients",
         caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
         x = "GENDERHEAL",
         y = "GENDERMENTAL",
         )
  + theme_bw()

# print the graph
myplot

# more points bigger, capitalize and add transparent
# to the points, and add a smooth line
myplot <- myplot + geom_point(color = "forestgreen",
                              alpha = 0.2,
                              size = 3)
# print labels to the graph
myplot <- myplot + labs(title = "Gene Expression Profiles of B-lymphoblastoid Acute Lymphoblastic Leu
                        subtitle = "A study of kids",
                        caption = "Source: Ertel\nRosenwald, Ziv,Wyatt",
                        x = "GENDERHEAL",
                        y = "GENDERMENTAL",
                        color = "forestgreen")
myplot
```

Now that we have the required aesthetics, let's add some extras like color to the plot. We can `color` **the points or lines** based on a variable, or specify a color. Let's color the points by `GENDERMENTAL`, and specify a number of colors that will be used as well as the color to specify. Note, the **`legend.title` automatically pulled for us!**

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype)
```

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17))
```

Alternatively, we could color based on subtypes by changing to `color` instead. Let's try something different and type `colors` and **`genotype` identified on the plot**. To do this we can assign `color` to `genotype`. `aesthetics` the column header, so that each subtype is plotted with a different shaped data point. Add in `shape` = `c(21)`type to your aesthetic and see how it changes your plot.

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17),
                                                  size=c(2,3))
```

The **size of the data points** are quite small. We can adjust that within the `geom_point()`. Again, but does **not** need to be included in the `aesthetics`, but it does need to be included in the `scale`, rather than mapping it to a variable. Let's try a `size` of `2` and `3` and specify a corresponding `size` for the size of the data points.

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17),
                                                  size=c(2,3))
```

The labels on the y- and x-axis are also quite small and hard to read. To change them size, we need to add an additional **theme layer**. The `ggplot2` theme system handles non-data plot elements such as:

- Axis label aesthetics
- Plot background
- Gridline appearance
- Legend appearance

There are built-in themes we can use (i.e. `theme_bw()`) that mostly change the background/overall colours, but we can also fine-tune individual parts of the theme ourselves. We can do this by adding a `theme()` function, and passing in arguments for the things we wish to change. Or we can add on a `theme_bw()`.

Let's add a bit of both `theme_bw()`. On the axis labels or the tick labels get any larger by changing theme?

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17),
                                                  size=c(2,3))
  theme_bw()
```

Not in this case! But we can set arbitrarily using `theme()`. We can change it ourselves. Since we are adding this layer on top of what is already in place, any features we change will override what is set in the `theme_bw()`. Here we'll **increase the size of the axes labels and axes tick labels to be 1.5 times the default size.** When modifying the size of text we need to specify that we want it to be a specific `unit` of `rel()` relative to the default size. Let's also set the `legend.position` to `"bottom"` in order to receive greater space for the actual data (needs a bit of room). Let's set the `legend.text` to be `size=8` in order to make the text smaller, rather than keeping it at the default size. Note, the `legend.text` does not need to be nested in `element` in order to receive greater space for the actual data (needs a bit of room).

Let's add a subtitle to a graph for the first time! Add a subtitle using a function which specifies the graphical format you intend on creating, i.e. `ggtitle()`, `xlab()`, etc. Within the function you will need to specify a `character` or `character vector` of length one (one element). This will be taken as the device that you want to write to.

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17),
                                                  size=c(2,3))
  theme_bw()
```

If you want to modify the size and resolution of the image you will need to add in the appropriate parameters as required by the function when you initialize. Then we plot the image to the device, using the `ggplot` subtitle that we just created.

```
ggplot(cdc,aesthetics) +
  geom_point(aesthetics) + scale_y_discr, + scale(limits, color = genotype,
                                                  shape=c(15,16,17),
                                                  size=c(2,3))
  theme_bw()
```

Finally, draw the "device" on file, using the `dev.off()` function. There are also `png`, `tiff`, and `jpeg` functions, through the same functions name as the file extension.

```
dev.off()
```

Note 1: We will not be able to open and look at your file using standard methods (Adobe Acrobat or Preview etc.) until you execute the `dev.off()` function.

Note 2: If you had made any additional plots before closing the device, they will all be stored in the same file, each plot should start on new page, unless you specify otherwise.

Boxplot

Now that we have all the required information for plotting with `ggplot` let's try plotting a boxplot. A boxplot provides a graphical view of the distribution of data based on a five number summary. The top and bottom of the box represent the Q1 (1st and 3rd) Quartiles (25th and 75th percentiles, respectively). The line inside the box represents the Q2 (median). The whiskers represent the minimum and maximum values, which are calculated as the 1.5 times the interquartile range (IQR). The distance of the plot beyond the minimum and maximum values that are not outliers.

Outliers are determined using the interquartile range (IQR), which is defined as Q3 - Q1. Any values that exceeds 1.5 x IQR below Q1 or above Q3 are considered outliers and are represented as points above or below the whiskers. These outliers are useful to identify any unexpected observations.

1. Add in the `ggplot()` layer to pass the differential expression data frame into `ggplot2`
2. Add in the `aesthetics`
3. Add `Genotype` as your x-axis label and `fitness expression` as your y-axis labels
4. Add in a `geom_boxplot()` function to add the boxplot to the plot
5. Change the axis of your data text the labels on the tick marks to 1.5x larger than the default
6. Change the size of your plot title in the same way that you change the size of the axes text but use `plot_title`

BONUS: Use the `ggsci` aesthetic to set a differential expression palette to represent differences between subtypes within each genotype.

Do that figure should look something like that provided below:

Volcano Plots

Volcano plots represent a useful way to visualize the results of differential expression analyses. A volcano plot is a type of scatterplot that shows statistical significance (adjusted p-value) versus magnitude of gene expression (log2 fold change) and allows for a quick assess in the plot across as possible, thus assisting. Usually, on the plot with labels that could not otherwise have been read. Often functionality allows the user to identify up to 3 different types of attributes (i.e. differential expression, p-value, and gene ontology) simultaneously.

Unlikely all aspects of all bioinformatics plot can be contributed for the purposes of accommodating all types of data, there are many options available to customize the plot to your needs. Although, there are many options, I'll give the therefore that you will be statistical significance, i.e. `pvalue` and `FoldChange`. In addition, it will only take as many of these that you necessarily fit in the plot space. The user can optionally supply a vector of labels (as `selectLab` that index along to input in the plot.

```
# install EnhancedVolcano
BiocManager::install("EnhancedVolcano", ncu.ask=F)

# load the library
library(EnhancedVolcano)

# load in the data
rnaseqdata.df = df.tab

# make your plot as easily customizable for legend and # value
# cutoffs as possible, and save to a variable
volcano_plot <- EnhancedVolcano(df,
                                lab = rownames(df),
                                x = 'log2FC',
                                y = 'pvalue',
                                pCutoff = 0.05,
                                FCcutoff = 1.5,
                                pointSize = 3.0,
                                labSize = 6.0,
                                legendPosition = 'right',
                                legendLabSize = 14,
                                legendIconSize = 5.0,
                                gene = c(TP53, TP73, TP63, CDKN1A, MDM2, MDM4, BAX, BCL2, PUMA, NOXA, IGF1R, IGF2R, CCNG1, CCNG2, CCND1, CCND2, CCND3, CCNE1, CCNE2, CCNA1, CCNA2, CCNB1, CCNB2, CDK1, CDK2, CDK4, CDK6, CDKN1B, CDKN2A, CDKN2B, CDKN2C, CDKN2D, CDKN3, CDC25A, CDC25B, CDC25C, CHEK1, CHEK2, SFN, BTG2, GTSE1, DDB2, TP53I3, EI24, PIDD, CASP1, CASP3, CASP4, CASP6, CASP7, CASP8, CASP9, CASP10, BIRC5, TNFRSF6, TNFRSF10B, TNFRSF10C, TNFRSF10D, TNFRSF10A, CD82, SERPINB5, SERPINE1, SERPINE2, IGFBP3, PRKDC, RPA1, RPA2, RPA3, RAD51, RAD51C, RAD23B, DDB2, POLH, CSN1, XPC, GADD45A, GADD45G, HIC1, SESN1, SESN2, DRAM, TSC22D3, ZMAT3, WWOX, FHIT, SAT1, IGF1, IGFBP1, IGFBP2, IGFBP6, PTEN, AKT1, AKT2, PIK3CG, PTK2, ATM, ATR, RB1, E2F1, E2F2, E2F3, E2F4, E2F5, TP73L, SERPINB5, ATF3, PLAGL1, HIPK2, MYOD1, RUNX3, NGFR, TGFB1, TGFB2, TGFB3, TGFBR1, TGFBR2, SMAD2, SMAD3, SMAD4, SMAD5, SMAD6, SMAD7, SKI, SKIL, TGIF1, MYC, CTNNB1, APC, APC2, NOTCH1, NOTCH2, NOTCH3, HES1, HES2, HES3, HES4, HES5, HES6, HES7, HEY1, HEY2, HEYL, JAG1, JAG2, DLL1, DLL3, DLL4, PTCH1, PTCH2, GLI1, GLI2, GLI3, SUFU, FOXM1, FOXO1, FOXO3, FOXO4, FOXO6, EGR1, EGR2, EGR3, EGR4, NR4A1, NR4A2, NR4A3, NR5A1, NR5A2, NR5A5, NR0B1, NR0B2, NR1D1, NR1D2, NR1H2, NR1H3, NR1H4, NR1I2, NR1I3, NR2C1, NR2C2, NR2E1, NR2E3, NR2F1, NR2F2, NR2F6, NR3C1, NR3C2
```