Introduction to R: ggplot2 **Graphics**

Day 3, Part A

IHME | UNIVERSITY of WASHINGTON

Institute for Health Metrics and Evaluation

In this lecture

1. Understanding the `ggplot` approach
2. Aesthetics
3. Geoms
4. Facets
5. Options and customization
6. Reshaping
7. Saving plots
8. Additional packages

What is ggplot2?

ggplot2 is an R package for making sophisticated and great-looking graphs

It's based on the book "Grammar of Graphics", which defined a fundamental theory of data visualization

ggplot2 contains functions that allow you to build complex graphics using a relatively small set of building blocks

NOTE: the online documentation for ggplot2 is fantastic, and lays all the functions out in terms of these building blocks:

http://ggplot2.tidyverse.org/reference/

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

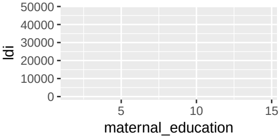
Load libraries & data

```
> library(ggplot2)
> library(reshape2)
> library(RColorBrewer)
>
> rm(list = ls())
> main_dir <- "C:/Users/ngraetz/Documents/repos/r_training_penn/"
> mmr_data <- read.csv(paste0(main_dir, "data/mmr_data.csv"))
> head(mmr_data)
  year_id      mmr maternal_education       ldi  location_name
1    2015 52.57428           9.900764 10593.983          China
2    2015 50.41785           5.943825  2773.896       Cambodia
3    2015 25.58855          11.535423 20782.643       Malaysia
4    2015 61.25871          14.697507 33327.094          Japan
5    2015 25.15193          14.635294 40454.078      Australia
6    2015 33.02467          14.244808 35569.391 United Kingdom
                       super_region_name              region_name
1 Southeast Asia, East Asia, and Oceania                East Asia
2 Southeast Asia, East Asia, and Oceania           Southeast Asia
3 Southeast Asia, East Asia, and Oceania           Southeast Asia
4                            High-income High-income Asia Pacific
5                            High-income              Australasia
6                            High-income           Western Europe
```

How does ggplot2 work?

First, you set up the graph:

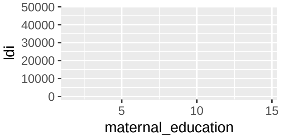
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi))
```



How does ggplot2 work?

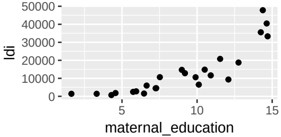
First, you set up the graph:

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi))
```



Then, you add to it. Basically telling ggplot what type of graph to make:

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point()
```



What are the building blocks of a ggplot?

- **Aesthetics**
- **Geoms**
- **Scales**
- **Facets**
- Positions
- Scales
- Labels
- Themes

Aesthetics

The aes in the initial `ggplot()` call

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point()
```

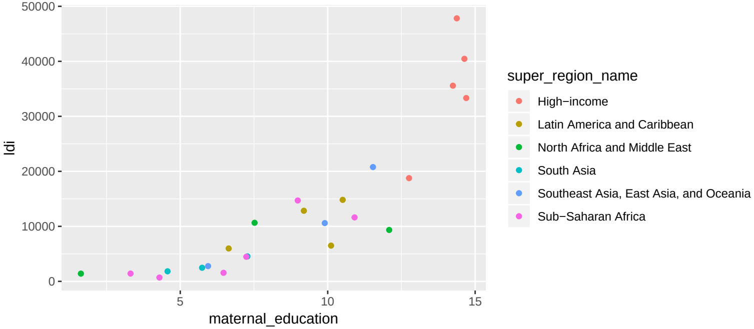
“Aesthetic mapping” is how you tell `ggplot` which variable is `x`, which is `y`

But, you can use them for more than just the axes:

- color (border color)
- fill (fill color)
- shape
- linetype (solid, dashed, dotted etc.)
- size
- alpha (transparency)
- labels

Example of aesthetic mapping

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = super_region_name)) +
+    geom_point()
```

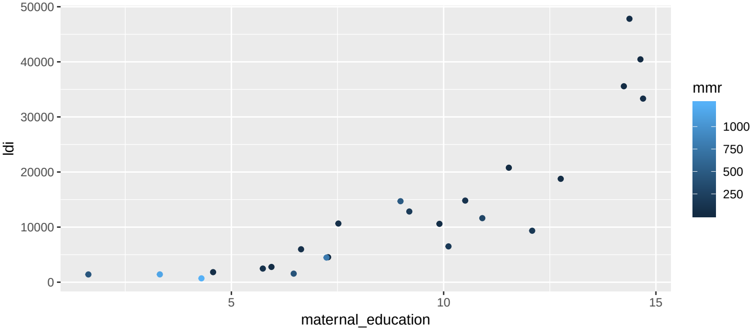


Note that ggplot conveniently makes a legend for you! In ggplot lingo, legends are called "scales"

Example of aesthetic mapping

In many cases, aesthetic mapping works for both continuous and categorical data

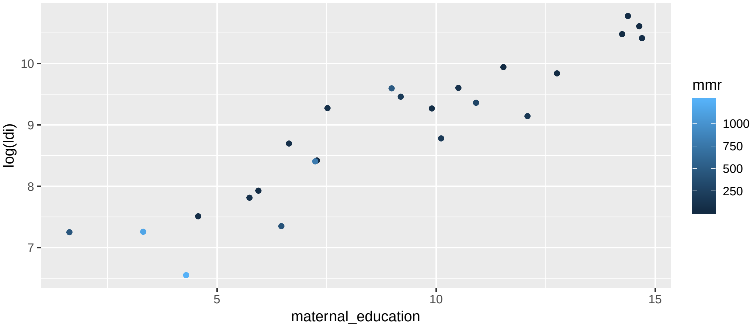
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = mmr)) +
+   geom_point()
```



Example of aesthetic mapping

ggplot allows you to manipulate variables "on the fly":

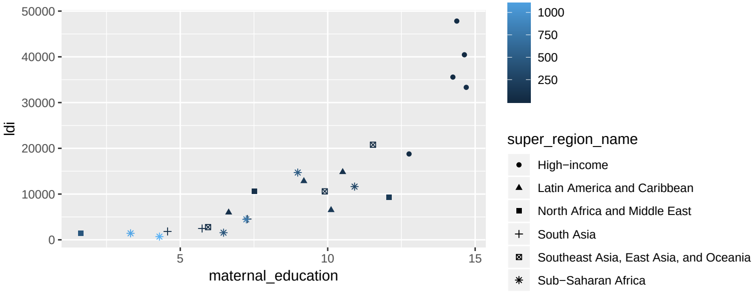
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = log(ldi),
+      color = mmr)) +
+   geom_point()
```



Example of aesthetic mapping

You can keep adding more aesthetics to add more information to your graph:

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = mmr, shape=super_region_name)) +
+   geom_point()
```



Note that not all aesthetics are meaningful for all geoms (e.g., linetype doesn't make sense if there are no lines in your graph)

What are the building blocks of a ggplot?

- **Aesthetics**
- **Geoms**
- **Scales**
- **Facets**
- Positions
- Scales
- Labels
- Themes

Geoms

```r
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point()
```

ggplot "geoms" (geometries) are the different types of graphs you can make:

- `geom_point()` for scatter plots
- `geom_line()` for line graphs
- `geom_bar()` for bar graphs
- And more: `geom_histogram()`, `geom_violin()`, `geomboxplot()`, `geom_errorbar()`, `geom_ribbon()`, `geom_segment()`, `geom_path()`, `geom_tile()`, `geom_polygon()`, etc.

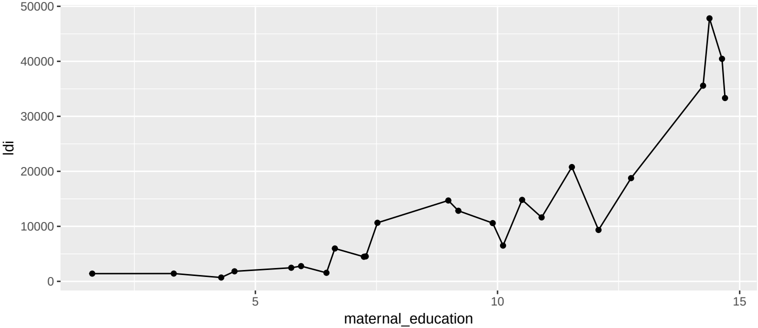
There are dozens of different geometries you can use for ggplot.

See the ggplot cheat sheet for the whole list: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Geoms

If you specify more than one geom, it "layers" them on top of each other

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point() +
+   geom_line()
```

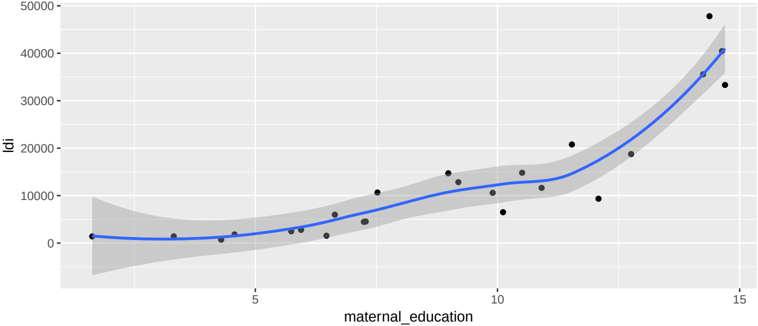


Note: the order matters, it will layer geoms in order that they are written

Geoms

There are some special geoms that do computation for you on the fly, just for convenience

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point() +
+   geom_smooth()
```



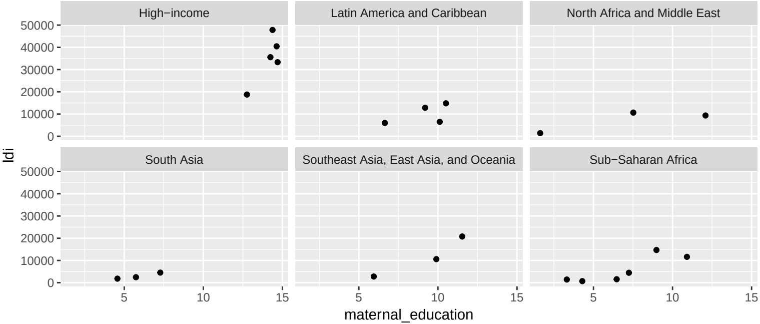
What are the building blocks of a ggplot?

- **Aesthetics**
- **Geoms**
- **Scales**
- **Facets**
- Positions
- Scales
- Labels
- Themes

Facets

Facets allow you to incorporate more complexity into your graphs by adding multiple panels:

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point() +
+   facet_wrap(~super_region_name)
```



What are the building blocks of a ggplot?

- **Aesthetics**
- **Geoms**
- **Scales**
- **Facets**
- Positions
- Scales
- Labels
- Themes

Positions

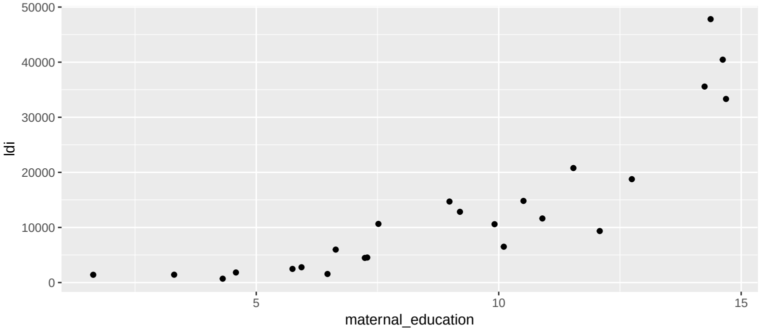
ggplot lets you modify where geoms appear relative to each other, using position functions:

- position_jitter() randomly displaces points (usually just for geom_point)
- position_dodge() automatically (tries to) shift to avoid overlap
- position_stack() stack, or add together geoms (usually just for geom_bar)
- position_fill() rescale the y-axis so the geoms sum to 100% (usually just for geom_bar)

Positions

position_jitter randomly displaces points (usually just for geom_point)

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point(position='jitter')
```

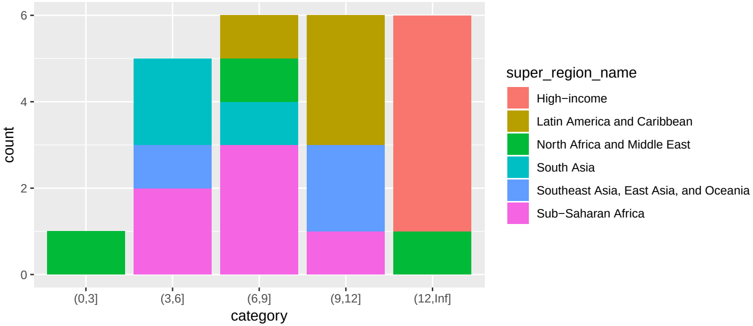


It's built right into the geom_point() function for convenience

Positions

position_stack is the default for geom_bar for factor variables

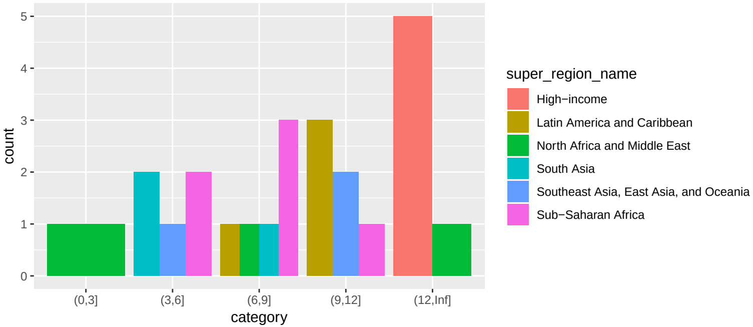
```
> mmr_data$category <- cut(mmr_data$maternal_education, breaks=c(0,3,6,9,12,Inf
> ggplot(data = mmr_data, aes(x = category, fill=super_region_name)) +
+   geom_bar()
```



Positions

`position_dodge` would put the bars side-by-side

```
> ggplot(data = mmr_data, aes(x = category, fill=super_region_name)) +
+   geom_bar(position='dodge')
```

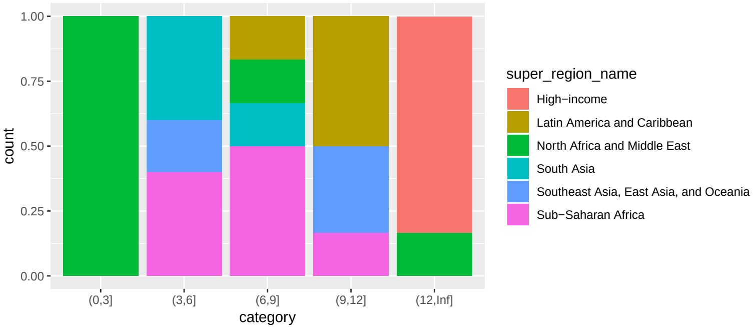


It's built right into the `geom_bar()` function for convenience

Positions

position_fill makes the bars sum to 100%

```
> ggplot(data = mmr_data, aes(x = category, fill=super_region_name)) +
+   geom_bar(position='fill')
```



It's built right into the geom_bar() function for convenience

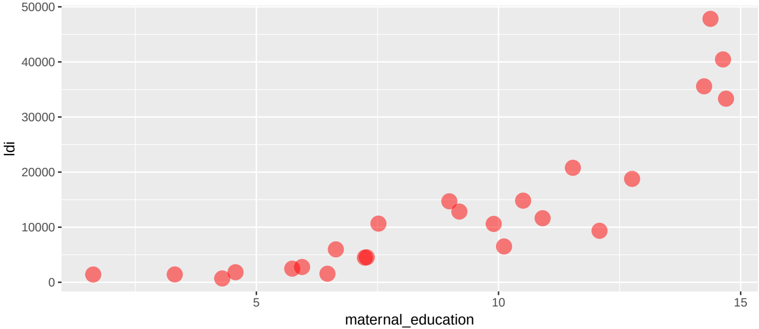
What are the building blocks of a ggplot?

- **Aesthetics**
- **Geoms**
- **Scales**
- **Facets**
- Positions
- Scales
- Labels
- Themes

Options and customization

Aesthetic arguments can also be provided directly to a geom in cases where you don't want them to map to some variable

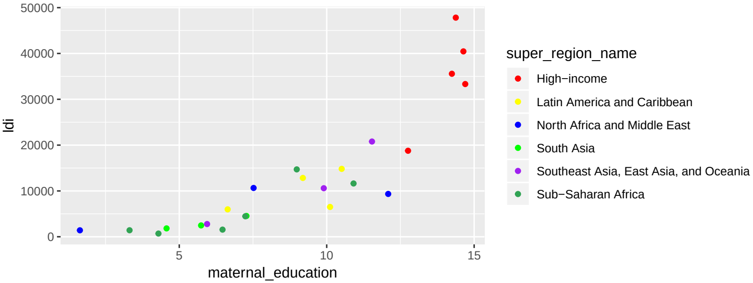
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point(color='red', size=2, alpha=.5)
```



Options and customization

You can also modify the "scales" (i.e., legends) to customize aesthetic mapping

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+     color = super_region_name)) +
+   geom_point() +
+   scale_color_manual(values=c('red','yellow','blue','green','purple','#31a354
```

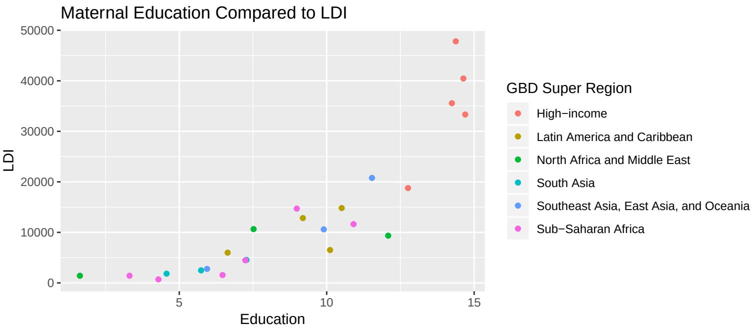


Every aesthetic (fill, color, shape, linetype) has corresponding `scale_*` function (`scale_fill_manual`, `scale_color_manual` etc.)

Options and customization

Titles for everything can be added with the `labs()` function:

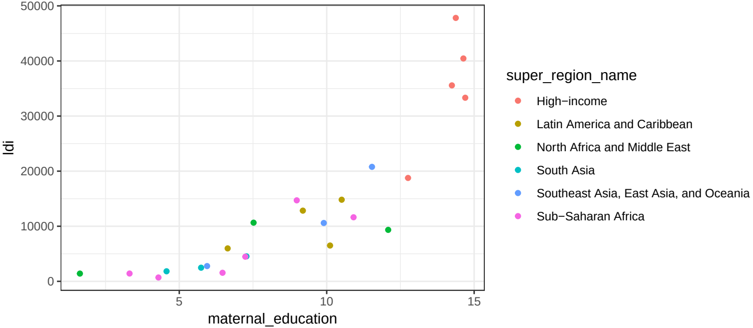
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = super_region_name)) +
+    geom_point() +
+    labs(title='Maternal Education Compared to LDI', y='LDI',
+      x='Education', color='GBD Super Region')
```



Options and customization

ggplot also comes with handy "themes", or preset options:

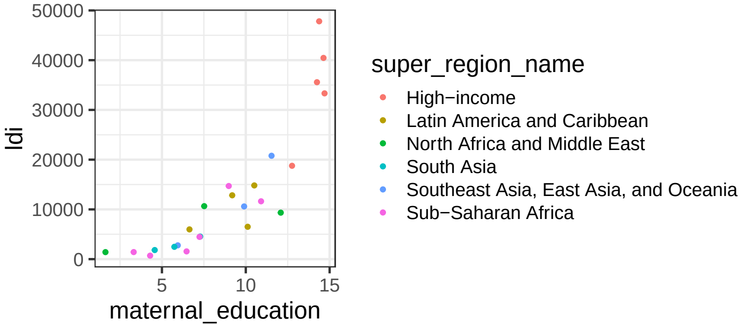
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = super_region_name)) +
+    geom_point() +
+    theme_bw()
```



Options and customization

Themes also allow you to rescale all text at the same time

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+      color = super_region_name)) +
+   geom_point() +
+   theme_bw(base_size=18)
```



Reshaping

ggplot2 is designed to work with data shaped such that each desired aesthetic is mapped to **one** variable. If your data is not shaped this way, it's almost always easier to reshape the data than to try and make ggplot2 work with original data structure.

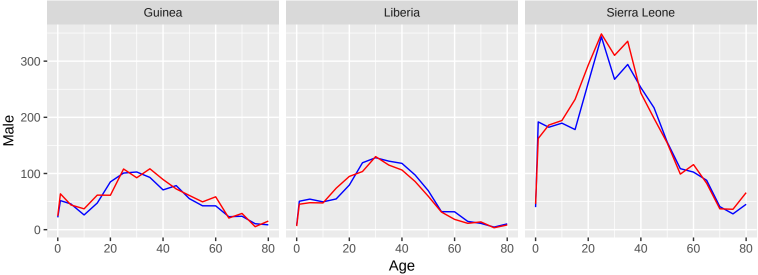
For example, if you want to plot the number of Ebola deaths by age group for both males and females, this is an inconvenient data structure since there are separate columns for deaths among males and females:

```
  Country Age Female Male
1  Guinea   0   24.5 21.9
2  Guinea   1   63.8 51.7
3  Guinea   5   44.0 45.8
4  Guinea  10   37.1 26.2
5  Guinea  15   61.5 47.4
6  Guinea  20   61.3 85.1
```

Reshaping

One option is to just add different geoms for each variable:

```
> ggplot(wide_data, aes(x = Age, y = Male)) +
+   facet_wrap(~ Country) +
+   geom_line(color='blue') +
+   geom_line(data = wide_data, aes(y = Female))
```



But that could get tedious because you have to manually map the aesthetic. It also doesn't make a legend for you.

Reshaping

A better option is to reshape long before attempting to plot these data:

```
> long_data <- melt(wide_data, id.vars=c("Country", "Age"),
+              value.name = "Deaths", variable.name = "Gender")
> head(long_data, 3)
  Country Age Gender Deaths
1  Guinea   0 Female   24.5
2  Guinea   1 Female   63.8
3  Guinea   5 Female   44.0
>
> ggplot(long_data, aes(x = Age, y = Deaths, color = Gender)) +
+   facet_wrap(~ Country) +
+   geom_line()
```

Saving plots

You can save your plot directly into a pdf or image file.

First store the plot as an R object (rather than just letting it print to RStudio's viewer)

```
> p <- ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi)) +
+   geom_point()
```

Then open a "graphics device" and print the plot into it:

```
> pdf(file = paste0(main_dir, "output/my_plot.pdf"), height = 5,
+     width = 9)
> p
> dev.off()
pdf
  2
```

The `dev.off()` part closes the device, i.e., saves your pdf.

Note: most file formats you'd expect are possible: `pdf()`, `png()`, `jpeg()` etc.

Saving plots

In a pdf, R will save each subsequent plot on a new page:

```
> p1 <- ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi))
+   geom_point()
>
> p2 <- ggplot(data = mmr_data, aes(x = maternal_education,  y = mmr))
+   geom_point()
```

```
> pdf(file = paste0(main_dir, "output/my_plot.pdf"), height = 5,
+     width = 9)
> p1
> p2
> dev.off()
pdf
  2
```

Saving plots

Common pitfall: when you open a device (using `pdf()`, `jpeg()`, etc.) it's easy to forget it's open, and then fail to close it. R will not actually write the file until the device is closed, so you can end up with multiple devices open and no actual files.

If this happens, keep typing `dev.off()` into you get the readout `null device`:

```
> dev.off()
pdf
  3
> dev.off()
pdf
  4
> dev.off()
null device
          1
```

Saving plots

Another common irritation when developing graphics code is that some programs put a lock on a file when you open it, which means that R can't overwrite it.

This will cause an error where R says it cannot open the file.

The solution is to go and close the program that currently has that file open.

(an alternative solution is to use a viewer that doesn't lock the files, e.g., view PDF files in Chrome rather than Acrobat)

Additional packages

ggplot2 has become so popular that other users have started writing add-ons to it:

- **gridExtra** - plot tables and arrange multiple plots together
- **ggrepel** - label points nicely
- **RColorBrewer** - easy-to-use color schemes of various types (colorbrewer2.org)
- GGally - various extensions to ggplot2 like a matrix of graphs
- cowplot - combine images with ggplots, highly-flexible multi-figure graphs
- ggthemes - more themes, preset colors

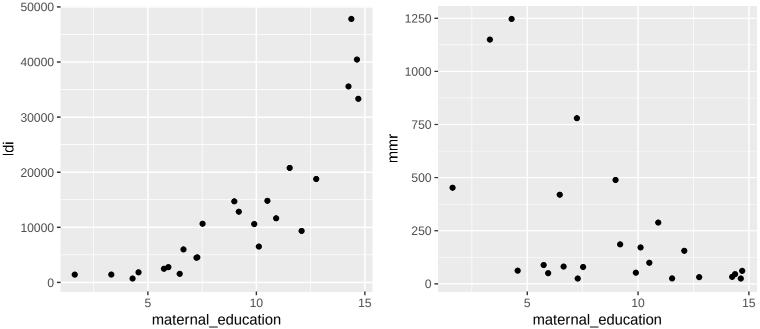
Additional package: gridExtra

The most important thing the gridExtra package can do is more flexibly combine graphs

It's often a useful alternative to facet_wrap when you don't want to reshape your data

```
> library(gridExtra)
> p1 <- ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi))
+   geom_point()
>
> p2 <- ggplot(data = mmr_data, aes(x = maternal_education,  y = mmr))
+   geom_point()
>
> grid.arrange(p1, p2, ncol=2)
```

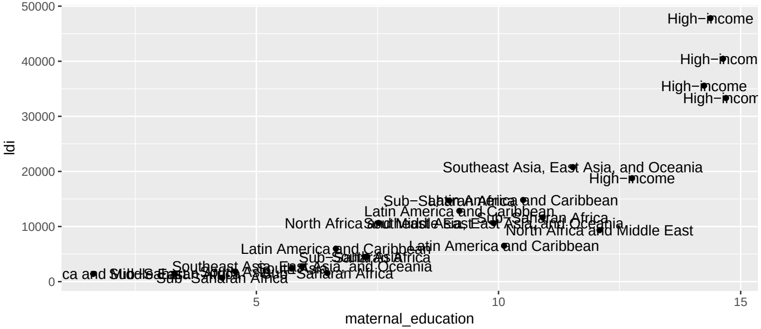
Additional package: `gridExtra`



Additional package: ggrepel

ggrepel helps you label points in a cleaner way than geom_text(), by adding geom_text_repel()

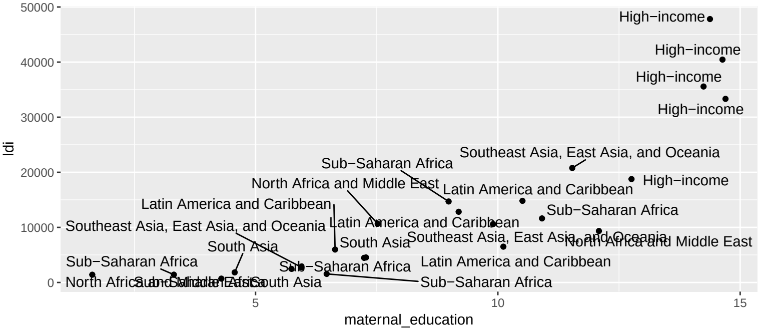
```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+     label = super_region_name)) +
+   geom_point() +
+   geom_text()
```



Additional package: ggrepel

ggrepel helps you label points in a cleaner way than geom_text(), by adding geom_text_repel()

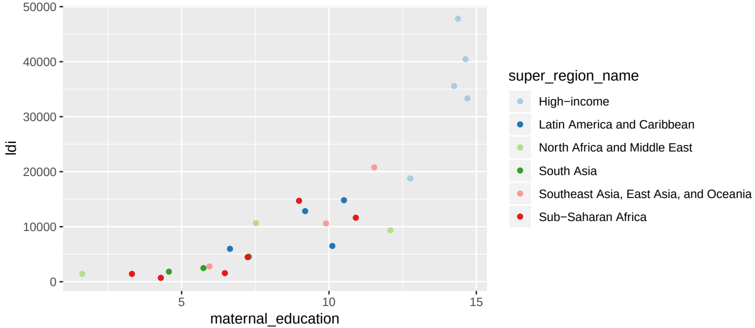
```
> library(ggrepel)
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+     label = super_region_name)) +
+   geom_point() +
+   geom_text_repel()
```



Additional package: RColorBrewer

RColorBrewer helps you choose nicer-looking colors

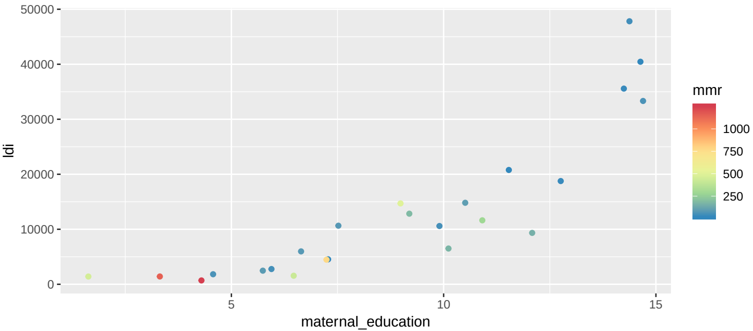
```
> library(RColorBrewer)
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+     color = super_region_name)) +
+   geom_point() +
+   scale_color_manual(values=brewer.pal(6, 'Paired'))
```



Additional package: RColorBrewer

It comes with sequential, diverging and qualitative color palettes that “match” each other

```
> ggplot(data = mmr_data, aes(x = maternal_education,  y = ldi,
+     color = mmr)) +
+   geom_point() +
+   scale_color_gradientn(colors=rev(brewer.pal(6, 'Spectral')))
```


ldi
maternal_education
mmr