Course Description

The data science revolution has produced reams of new data from a wide variety of new sources. These new datasets are being used to answer new questions in ways never before conceived. Visualization remains one of the most powerful ways draw conclusions from data, but the influx of new data types requires the development of new visualization techniques and building blocks. This course provides you with the skills for creating those new visualization building blocks. We focus on the ggplot2 framework and describe how to use and extend the system to suit the specific needs of your organization or team. Upon completing this course, learners will be able to build the tools needed to visualize a wide variety of data types and will have the fundamentals needed to address new data types as they come about.

Course Learning Objectives

After completing this course, you will be able to:

- Create data graphics using the ggplot2 package
- Build graphics by combining multiple geoms
- Recognize the differences between different geom_* functions
- Recall the difference between mapping an aesthetic to a constant and mapping an aesthetic to a variable
- Create a scatterplot or a histogram using ggplot2
- Build plots using the six guidelines for good graphics
- Create simple maps using ggplot2
- Create dynamic maps using the ggmap package
- Build maps that have external data overlaid
- Create chloropleth maps of US counties
- Recognize the different graphical objects presented by the grid package
- Build build simple graphics using the grid package
- Create a ggplot2 theme by modifying an existing theme
- Build a new geom function to implement a new feature or simplify a workflow

Data

The datasets for this course can be obtained by downloading the following zip file:

msdr_data.zip

License

The materials in this course are licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license.

The ggplot2 package allows you to quickly plot attractive graphics, to visualize and explore data. Objects created with ggplot2 can also be extensively customized (more on that in the next subsection). While the structure of ggplot2 code differs substantially from that of base R graphics, it offers a lot of power for the required effort. This subsection focuses on useful, rather than attractive graphs, since this subsection focuses on exploring rather than presenting data. Later sections will give more information about making more attractive or customized plots, as you'd want to do for final reports, papers, etc.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

The first step in creating a plot using ggplot2 is to create a ggplot object. This object will not, by itself, create a plot with anything in it. Instead, it typically specifies the data frame you want to use and which aesthetics will be mapped to certain columns of that data frame (aesthetics are explained more in the next subsection).

Continue reading at the Mastering Software Development with R web site.

https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#initializing-a-ggplot-object

Mark as completed

Aesthetics are properties of the plot that can show certain elements of the data. For example, in Figure 4.1, color shows (i.e., is mapped to) gender, x-position shows height, and y-position shows weight in a sample data set of measurements of children in Nepal.

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#plot-aesthetics

✓ Complete

The system of creating a ggplot object, mapping aesthetics to columns of the data, and adding geoms makes more sense once you try a few plots. For example, say you'd like to create a histogram showing the fares shown by passengers in the example Titantic data set. To plot the histogram, you'll first need to create a ggplot object using the dataframe with the column you want to print. In creating this ggplot object, you only need one aesthetic (x, the variable for which you want to create the histogram), and then you'll need to add a histogram geom.

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#creating-a-basic-ggplot-plot

✓ Complete

Geom functions actually plot elements of the plot; if you do not include at least one geom, you'll get a blank plot space. Geom functions have their own arguments to adjust how the graph is created. For example, you can use the bins argument to change the number of bins used to create a histogram— try:

```
ggplot(titanic, aes(x = Fare)) +
  geom_histogram(bins = 15)
```

As with any R functions, you can find out more about which arguments are available for geom functions by pulling up the function's help file (e.g., ?geom_histogram).

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#geoms

✓ Complete

Several geoms can be added to the same ggplot object, which allows you to build up layers to create interesting graphs. For example, you could make the World Cup scatterplot of time versus shots more interesting by adding label points for noteworthy players with the player's team name and position:

```
library(dplyr)

noteworthy_players <- worldcup %>% filter(Shots == max(Shots) |
                                          Passes == max(Passes)) %>%
  mutate(point_label = paste(Team, Position, sep = "-"))

ggplot(worldcup, aes(x = Passes, y = Shots)) +
  geom_point() +
  geom_text(data = noteworthy_players, aes(label = point_label),
            vjust = "inward", hjust = "inward", color = "red") +
  geom_point(data = noteworthy_players, color = "red")
```

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#using-multiple-geoms

✓ Complete

Instead of mapping an aesthetic to an element of your data, you can use a constant value for it. For example, you may want to make all the points green in the World Cup scatterplot:

```
1  ggplot(worldcup, aes(x = Time, y = Passes)) +
2    geom_point(color = "darkgreen")
```

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#constant-aesthetics

✓ Complete

In this subsection, I'll show some examples of basic plots created with ggplot2. For the example plots in this subsection, I'll use a dataset in the faraway package called nepali. This gives data from a study of the health of a group of Nepalese children.

```
1 library(faraway)
2 data(nepali)
```

Each observation is a single measurement for a child; there can be multiple observations per child. I used the following code to select only the columns for child id, sex, weight, height, and age. I also used distinct to limit the dataset to only include one measurement for each chile, the child's first measurement in the dataset.

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#example-plots

✓ Complete

There are lots of R extensions for creating other interesting plots. For example, you can use the ggpairs function from the GGally package to plot all pairs of scatterplots for several variables (Figure 4.11).

Continue reading: https://bookdown.org/rdpeng/RProgDA/basic-plotting-with-ggplot2.html#extensions-of-ggplot2

✓ Complete

With slightly more complex code, you can create very interesting and customized plots using ggplot2. This functionality is useful for creating plots that follow some of the guidelines for effective visualization, based for example on ideas from Edward Tufte and others. In this section, we'll provide an overview of some of the guidelines for creating good plots and show how you can customize ggplot objects to adhere to some of these guidelines. This will also give us the chance to go over how to customize ggplot objects, and we'll end by going over scales, themes, faceting, and color specifically.

✓ Complete

There are a number of very thoughtful books and articles about creating graphics that effectively communicate information. Some of the authors I highly recommend (and from whose work I've pulled and aggregated the guidelines for good graphics we'll go over) are:

- Edward Tufte (The Visual Display of Quantitative Information is a classic)
- Howard Wainer
- Stephen Few
- Nathan Yau

Continue reading: https://bookdown.org/rdpeng/RProgDA/customizing-ggplot2-plots.html#guidelines-for-good-plots

✓ Complete

We'll finish this section by going into a bit more details about how to customize the scales and colors for ggplot objects, including more on scales and themes.

There are a number of different scale functions that allow you to customize the scales of ggplot objects. Because color is often mapped to an aesthetic, you can adjust colors in many ggplot objects using scales, as well (the exception is if you are using a constant color for an element). These functions follow the following convention:

```
## Generic code
scale_[aesthetic]_[vector type]
```

For example, to adjust the x-axis scale for a continuous variable, you'd use scale_x_continuous. You can use a scale function for an axis to change things like the axis label (which you could also change with xlab or ylab) as well as position and labeling of breaks.

Continue reading: https://bookdown.org/rdpeng/RProgDA/customizing-ggplot2-plots.html#scales-and-color

✓ Complete

There are some excellent resources available for finding out more about creating plots using the gpplot2 package.

If you want to get more practical tips on how to plot with ggplot2, these are good resources

- *R Graphics Cookbook* by Winston Chang: This "cookbook" style book is a useful reference to have to flip through when you have a specific task you want to figure out how to do with ggplot2 (e.g., flip the coordinate axes, remove the figure legend).
- http://www.cookbook-r.com/Graphs/: Also created by Winston Chang, this website goes with the *R Graphics Cookbook* and is an excellent reference for quickly finding out how to do something specific in ggplot2.
- Google images: If you want to find example code for how to create a specific type of plot in R, try googling the name of the plot and "R", and then search through the "Images" results. For example, if you wanted to plot a wind rose in R, google "wind rose r" and click on the "Images" tab. Often, the images that are returned will link back to a page that includes the example code to create the image (a blog post, for example).

For more technical details about plotting in R, check out the following resources:

- *ggplot2: Elegant Graphics for Data Analysis* by Hadley Wickham: Now in its second edition, this book was written by the creator of grid graphics and goes deeply into the details of why ggplot2 was created and how to use it.
- *R Graphics* by Paul Murrell: Also in its second edition, this book explains grid graphics, the graphics system that ggplot2 is built on. This course covers the basics of grid graphics in a later section to give you the tools to create your own ggplot2 extensions. However, if you want the full details on grid graphics, this book is where to find them.

Mark as completed

Mapping

htmlWidgets

Mapping Quiz

Often, data will include a spatial component, and you will want to map the data either for exploratory data analysis or to present interesting aspects of the data to others. R has a range of capabilities for mapping data. The simplest techniques involve using data that includes latitude and longitude values, and use these location values as the x and y aesthetics in a regular plot. R also has the ability to work with more complex spatial data objects and import shapefiles through extensions like the sp package.

In this section, we will cover the basics of mapping in R and touch on some of the more advanced possibilities. We will also present some useful packages for making quick but attractive maps in R. R also now has the capability to make interactive maps using the plotly and leaflet packages; in the end of this section, we'll present these packages and explain a bit more about htmlWidgets in general.

✓ Complete

It is very easy now to create point maps in R based on longitude and latitude values of specific locations. You can use the map_data function from the ggplot2 package to pull data for maps at different levels ("usa", "state", "world", "county").

The maps you pull using map_data are just data to use to plot polygon shapes for areas like states and counties.

Continue reading: https://bookdown.org/rdpeng/RProgDA/mapping.html#basics-of-mapping

✓ Complete

The ggmap package allows you to use tools from Google Maps directly from R.

```
## install.packages("ggmap")
library(ggmap)
```

This package uses the Google Maps API, so you should read their terms of service and make sure you follow them. In particular, you are limited to just a certain number of queries per time.

You can use the get_map function to get maps for different locations. You can either use the longitude and latitude of the center point of the map, along with the zoom option to say how much to zoom in (3: continent to 20: building) or you can use a character string to specify a location.

Continue reading: https://bookdown.org/rdpeng/RProgDA/mapping.html#ggmap-google-maps-api

✓ Complete

If you need to map US states and counties, the choroplethr and choroplethrMaps packages offer functions for fast and straightforward mapping.

```
library(choroplethr)
library(choroplethrMaps)
```

As an example, I'll use data on county-level population in 2012 that comes as the dataset df_pop_county with the choroplethr package. This dataset gives the population of each county (value) and the county FIPS number (region), which is a unique identification number for each US county.

```
data(df_pop_county)
df_pop_county %>% slice(1:3)
```

```
##   region  value
## 1   1001  54590
## 2   1003 183226
## 3   1005  27469
```

To map population by county, you can use the countyChoropleth function.

```
county_choropleth(df_pop_county)
```

Continue reading: https://bookdown.org/rdpeng/RProgDA/mapping.html#mapping-us-counties-and-states

✓ Complete

R has a series of special object types for spatial data. For many mapping / GIS tasks, you will need your data to be in one of these objects.

Spatial objects:

- SpatialPolygons
- SpatialPoints
- SpatialLines

Spatial objects + dataframes:

- SpatialPolygonsDataFrame
- SpatialPointsDataFrame
- SpatialLinesDataFrame

The tigris package lets you pull spatial data directly from the US Census. This data comes in directly as a spatial object.

Continue reading: https://bookdown.org/rdpeng/RProgDA/mapping.html#more-advanced-mapping-spatial-objects

Mark as completed

- *Applied Spatial Data Analysis with R* by Roger Bivand (available online through CSU library)
- *An Introduction to R for Spatial Analysis and Mapping* by Chris Brunsdon and Lex Comber
- CRAN Spatial Data Task View
- R Spatial Cheatsheet
- Great blog post (among many) by Zev Ross

https://bookdown.org/rdpeng/RProgDA/mapping.html#where-to-find-more-on-mapping-with-r

✓ Complete

Mapping

htmlWidgets

- **Reading:** Overview of htmlWidgets
  10 min
- **Reading:** plotly package
  10 min
- **Reading:** Creating your own widget
  10 min

Mapping Quiz

Very smart people have been working on creating interactive graphics in R for a long time. So far, nothing coded in R has taken off in a big way.

JavaScript has developed a number of interactive graphics libraries that can be for documents viewed in a web browser. There is now a series of R packages that allow you to create plots from these JavaScript libraries from within R.

Continue reading: https://bookdown.org/rdpeng/RProgDA/htmlwidgets.html

✓ Complete

From the plotly package documentation, the purpose of the plotly package is to allow a user to

"Easily translate ggplot2 graphs to an interactive web-based version and / or create custom web-based visualizations directly from R."

- Like many of the packages today, it draws on functionality external to R, but within a package that allows you to work exclusively within R.
- Allows you to create interactive graphs from R. Some of the functions extend the ggplot2 code you've learned.
- Interactivity will only work within RStudio or on documents rendered to HTML.

The plotly package allows an interface to let you work with plotly.js code directly using R code.

Continue reading: https://bookdown.org/rdpeng/RProgDA/htmlwidgets.html#plotly-package

✓ Complete

If you find a JavaScript visualization library and would like to create bindings to R, you can create your own package for a new htmlWidget.

There is advice on creating your own widget for R available at http://www.htmlwidgets.org/develop_intro.html.

✓ Complete

The grid package

Grid Package quiz

The grid package in R implements the primitive graphical functions that underly the ggplot2 plotting system. While one typically does not interact directly with the grid package (it is imported by the ggplot2 package), it is necessary to understand some aspects of the grid package in order to build new geoms and graphical elements for ggplot2. In this section we will discuss key elements of the grid package that can be used in extending `ggplot2.

While the grid package can be used to produce graphical output directly, it is seldom used for that purpose. Rather, the grid package provides a set of functions and classes that represent graphical objects or grobs, that can be manipulated like any other R object. With grobs, we can manipulate graphical elements ("edit" them) using standard R functions.

Continue reading: https://bookdown.org/rdpeng/RProgDA/the-grid-package.html

✓ Complete

The grid package

Grid Package quiz

Grid graphics is a system for plotting within R, and should be thought of as a separate system than base R graphics. The ggplot2 package is built on top of grid graphics, so grid graphics functions can be used to customize and manipulate ggplot2 objects. The grid graphics functions interact less well with plots created using the base R graphics system. While we have focused on plotting using ggplot2 in this course, we have covered a few plots created using base R, specifically the maps created by running a plot call on a spatial object, like a SpatialPoints object.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

The grid package

Grid Package quiz

The most critical concept of grid graphics to understand for extending ggplot2 it the concept of grobs. Grobs are graphical objects that you can make and change with grid graphics functions. For example, you may create a circle grob or points grobs. Once you have created one or more of these grobs, you can add them to or take them away from larger grid graphics objects, including ggplot objects. These grobs are the actual objects that get printed to a graphics device when you print a grid graphics plot; if you tried to create a grid graphics plot without any grobs, you would get a blank plot.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

The grid package

Grid Package quiz

Much of the power of grid graphics comes from the ability to move in and out of working spaces around the full graph area. As an example, say you would like to create a map of the states of the US with a small pie chart added at the centroid of each state showing the distribution of population in that state by education level. This kind of plot is where grid graphics shines (although it appears that you now can create such a plot directly in ggplot2). In this case, you want to zoom in at the coordinates of a state centroid, have your own smaller working space at that location, add a pie chart showing data specific to that state, then zoom out and do the process again for a different state centroid.

In grid graphics, these smaller working spaces within the larger plot are called *viewports*. You can navigate to one of the viewports, make some changes, and then pop back up and navigate into another viewport somewhere else in the plotting region.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

The grid package

Grid Package quiz

Once you have created a grob and moved into the viewport in which you want to plot it, you need a way to specify where in the viewport to write the grob. The numbers you use to specify x- and y-placements for a grob will depend on the coordinate system you use. In grid graphics, you have a variety of options for the units to use in this coordinate system, and picking the right units for this coordinate system will make it much easier to create the plot you want.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

The grid package

- **Reading:** Introduction
  10 min
- **Reading:** Overview of grid graphics
  10 min
- **Reading:** Grobs
  10 min
- **Reading:** Viewports
  10 min
- **Reading:** Grid graphics coordinate systems
  10 min
- **Reading:** The gridExtra package
  10 min
- **Reading:** Where to find more about grid graphics
  10 min

Grid Package quiz

The gridExtra package provides useful extensions to the grid system, with an emphasis on higher-level functions to work with grid graphic objects, rather than the lower-level utilities in the grid package that are used to create and edit specific lower-level elements of a plot. This package has particularly useful functions that allow you to arrange and write multiple grobs to a graphics device and to include tables in grid graphics objects.

Continue reading at the Mastering Software Development with R web site.

✓ Complete

Grid graphics provides an extensive graphics system that can allow you to create almost any plot you can imagine in R. It takes quite a bit of work to fully understand all elements of the grid graphics system, but you might find it worthwhile to study grid graphics in greater depth if you often need to create very tailored, unusual graphs.

There are a number of resources you can use to learn more about grid graphics. The most comprehensize is the *R Graphics* book by Paul Murrell, the creator of grid graphics. This book is now in its second edition, and its first edition was written before ggplot2 became so popular. It is worth try to get the second edition, which includes some content specifically on ggplot2 and how that package relates to grid graphics. The vignettes that go along with the grid package are also by Paul Murrell and give a useful introduction to grid graphics, and the vignettes for the gridExtra package are also a useful next step for finding out more.

- Links to pdfs of vignettes for the grid graphics package are available at https://stat.ethz.ch/R-manual/R-devel/library/grid/doc/index.html
- Links to pdfs of vignettes for the gridGraphics package are available on the package's CRAN page: https://cran.r-project.org/web/packages/gridExtra/index.html

✓ Complete

Building and modifying a theme in ggplot2 is a key feature of the ggplot2 package and system for building data graphics. The original base graphics system in R did not have a notion of a "theme" for how graphical elements are presented—users were left to individually customize each graphic without any clear way to programmatically implement shared elements across plots.

The ggplot2 package implements the notion of a theme for its plots by allowing you to modify many different elements of a plot and to store all those modifications as a special "theme" object. Those elements that can be modified are documented in the help page ?theme, which documents the theme() function.

The default theme for ggplot2 is encapsulated by the theme_gray() function. Like other elements in the ggplot2 universe, themes can be "added" using the + operator to plot commands in order to change the look and feel of a plot. Adding a theme (either existing or custom built by you) will override elements of any default theme.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

Why would one want to build a new theme? For many people, it is a matter of personal preference with respect to colors, shapes, fonts, positioning of labels, etc. Because plots, much like writing, are an expression of your ideas, it is often desirable to customize those plots so that they accurately represent your vision.

In corporate or institutional settings, developing themes can be a powerful branding tool. Plots that are distributed on the web or through marketing materials that have a common theme can be useful for reinforcing a brand. For example, plots made by the FiveThirtyEight.com web site have a distinct look and feel (see this article by Walt Hickey for one of many examples). When you see one of those plots you instinctively know that it is a "FiveThirtyEight" plot. Developing a theme for your organization can help to get others to better understand what your organization is about when it produces data graphics.

Another advantage of having a pre-programmed theme is that it removes the need for you to think about it later! One key reason why news organizations like FiveThirtyEight or the New York Times have common themes for their data graphics is because they are constantly producing those graphics on a daily basis. If every plot required a custom look and feel with a separate palette of colors, the entire process would grind to a halt. If you are in an environment where there is a need for reproducible graphics with a consistent feel, then developing a custom theme is probably a good idea. While using the default ggplot2 theme is perfectly fine from a data presentation standpoint, why not try to stand out from the crowd?

✓ Complete

As noted above, ggplot2 has a default theme, which is theme_gray(). This theme produces the familiar gray-background-white-grid-lines plot. You can obtain the default theme using the theme_get() function.

```
1  x <- theme_get()
2  class(x)
3
4  [1] "theme" "gg"
```

The object returned by theme_get() is rather large so it's not recommeneded to print it to the console. Notice that the object returned by theme_get() is an S3 object of class "theme" and "gg". This is the kind of object you will need to create or modify in order to customize your theme.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

Perhaps the easiest thing to start with when customizing your own theme is to modify an existing theme (i.e. one that comes built-in to ggplot2). In case your are interested in thoroughly exploring this area and learning from others, there is also the ggthemes package on CRAN which provides a number of additional themes for ggplot2.

Looking at the help page ?theme you'll see that there are many things to modify. We will start simple here by illustrating the general approach to making theme modifications. We will begin with the theme_bw() theme. This theme is a simple black and white theme that has little ornamentation and few features.

Continue reading at the Mastering Software Development web site.

✓ Complete

Building a new theme allows you to customize the look and feel of a plot to match your personal preferences. It also allows you to define a consistent “branded” presentation of your data graphics that can be clearly identified with your organization or company.

✓ Complete

Some of the key elements of a data graphic made with ggplot2 are geoms and stats. The fact is, the ggplot2 package comes with tremendous capabilities that allow users to make a wide range of interesting and rich data graphics. These graphics can be made through a combination of calls to various geom_* and stat_* functions (as well as other classes of functions).

So why would one want to build a new geom or stat on top of all that ggplot2 already provides?

1. There are two key reasons for building new geoms and stats for ggplot2:
2. Implement a new feature. There may be something very specific to your application that is not yet implemented—a new statistical modeling approach or a novel plotting symbol. In this case you don't have much choice and need to extend the functionality of ggplot2.

Simplify a complex workflow. With certain types of analyses you may find yourself producing the same kind of plot elements repeatedly. These elements may involve a combination of points, lines, facets, or text and essentially encapsulate a single idea. In that case it may make sense to develop a new geom to literally encapsulate the collection of plot elements and to make it simpler to include these things in your future plots.

Building new stats and geoms is the plotting equivalent of writing functions (that may sound a little weird because stats and geoms are functions, but they are thought of a little differently from generic functions). While the action taken by a function can typically be executed using separate expressions outside of a function context, it is often convenient for the user to encapsulate those actions into a clean function. In addition, writing a function allows you to easily parameterize certain elements of that code. Creating new geoms and stats similarly allows for a simplification of code and for allowing users to easily tweak certain elements of a plot without having to wade through an entire mess of code every time.

✓ Complete

New geoms in ggplot2 inherit from a top level class called Geom and are constructed using a two step process.

1. The ggproto() function is used to construct a new class corresponding to your new geom. This new class specifies a number of attributes and functions that describe how data should be drawn on a plot.
2. The geom_* function is constructed as a regular function. This functdion returns a layer to that can be added to a plot created with the ggplot() function.

Continue reading at the Mastering Software Development web site.

✓ Complete

One problem when making scatterplots of large amounts of data is overplotting. In particular, with ggplot2's default solid circle as the plotting shape, if there are many overlapping points all you will see is a solid mass of black.

One solution to this problem of overplotting is to make the individual points transparent by setting the alpha channel. The alpha channel is a number between 0 and 1 where 0 is totally transparent and 1 is completely opaque. With transparency, if two points overlap each other, they will be darker than a single point sitting by itself. Therefore, you can see more of the "density" of the data when the points are transparent.

The one requirement for using transparency in scatterplots is computing the amount of transparency, or the the alpha channel. Often this will depend on the number of points in the plot. For a simple plot with a few points, no transparency is needed. For a plot with hundreds or thousands of points, transparency is required. Computing the exact amount of transparency may require some experimentation.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

In addition to geoms, we can also build a new *stat* in ggplot2, which can be used to abstract out any computation that may be needed in the creation/drawing of a geom on a plot. Separating out any complex computation that may be needed by a geom can simplify the writing of the geom down the road.

Building a stat looks a bit like building a geom but there are different functions and classes that need to be specified. Analogous to creating a geom, we need to use the ggproto() function to create a new class that will usually inhert from the Stat class. Then we will need to specify a stat_* function that will create the layer that will be used by ggplot2 and related geom_* functions.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

One task that is common in the course of data analysis or statistical modeling is plotting a set of parameter estimates along with a 95% confidence interval around those points. Given an estimate and a standard error, basic statistical theory says that we can approximate a 95% confidence interval for the parameter by taking the estimate and adding/subtracting 1.96 times the standard error. We can build a simple stat that takes an estimate and standard error and constructs the data that would be needed by geom_segment() in order to draw the approximate 95% confidence intervals.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

Combining geoms and stats gives you a way of creating new graphical elements that make use of special computations that you define. In addition, if you require some custom drawing that is not immediately handled by an existing geom, then you may consider writing a separate geom to handle the data computed by your stat. In this section we show how to combine stats with geoms to create a custom plot.

Continue reading at the Mastering Software Development in R web site.

✓ Complete

Building new geoms can be a useful way to implement a completely new graphical procedure or to simplify a complex graphical task that must be used repeatedly in many plots. Building a new geom requires defining a new Geom class via ggproto() and defining a new geom_* function that builds a layer based on the new Geom class.

Some further resources that are worth investigating if you are interested in building new graphical elements are

- R Graphics by Paul Murrell, describes the grid graphical system on which ggplot2 is based.
- Extending ggplot2 vignette, provides further details about how to build new geoms and stats.
- ggplot2 Extensions web site, provides numerous examples of ggplot2 extensions that members of the community have developed.

✓ Complete