**Introduction to ggplot in R-Instat**

R-instat uses the ggplot2 graphics framework, which creates plots by adding layers. This introduction takes you through the set-up of ggplot2, several layers in ggplot2 package, the R code to produce subsequent plots, and how to do this in R-Instat.

1. **The Set Up**

This section describes how to tell ggplot what dataset to use and what variables to plot. This information is declared in the ggplot function.

**ggplot function**

The ggplot function initialises a ggplot object. It is used to define the input data frame for a plot and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

ggplot(data, mapping = aes(), …)

**Arguments**

data specify what dataset to use

Mapping aesthetic mappings are set with the aes function

**Aesthetic Mapping**

To produce a plot, we map variables in the data frame to aesthetic properties of the geom. Aesthetic mappings describe how variables in the data are mapped to visual properties. The visual properties of a plot, or aesthetics, are include as the following arguments.

aes(x, y, colour, fill, …)

**Arguments**

position x and y axes

colour outside colour

fill inside colour

shape of points

linetype type of line. Options include: dashed, dot-dash, etc.

size of points and lines

Aesthetic mappings can be set in the:

* ggplot() function, which applies the aesthetics across all layers added
* geom function (layers), which applies the aesthetics to only that specified geom.

**Example 1**

**R Code**

library(ggplot2)

# define data frame

ggplot(data = diamonds)

# map carat variable to the x-axis

ggplot(data = diamonds, mapping = aes(x = carat))

# map carat variable to the x-axis and price variable to y-axis

ggplot(data = diamonds, mapping = aes(x = carat, y = price)) # x and y axes are fixed for all layers

# map cut to colour argument

ggplot(data = diamonds, mapping = aes(x = carat, colour = cut)) # each category of the 'cut' variable will now have a distinct colour, once a geom is added

**2. Layers**

The layers in ggplot2 are also called geoms. Once you have defined your data frame and which variables to map to the aesthetics properties, using ggplot() function, you can add the geoms one on top of the other.

Geometric (geom) objects are the visual representations of observations. For example, to determine whether two continuous variables are correlated with one another, one could create a scatter plot by adding the geom\_point function.

Here you can find documentation on a comprehensive list of all available geoms (provide link when ready) and the list of all available geoms in R-Instat.

R-Instat has a menu of the dialogs for the most commonly used plots, these include scatter, line, boxplot, histograms, bar charts and many more. You can also use the General Graphics dialog to create a plot from scratch.

**Example 2**

Will now create a scatter plot by adding the geom\_point() layer, where carat is mapped on to the x axis and price in mapped on to the y axis. We also demonstrate setting the aesthetic mappings in ggplot and geom function. Where you set the aesthetics becomes important when adding more than one layer, shown in Example 4.

**R Code**

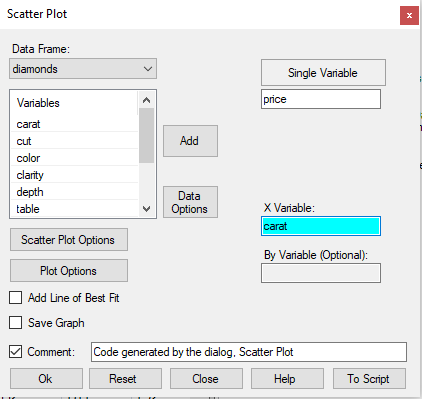
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) + geom\_point() # added geom\_point layer to produce a scatter plot

# specify aesthetics inside geoms

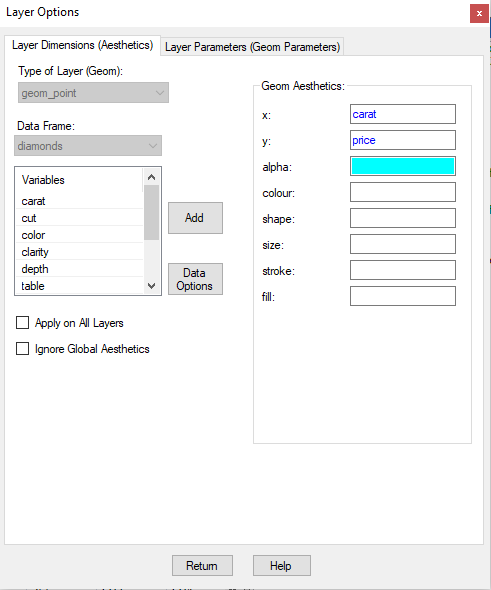
ggplot(data = diamonds) + geom\_point(aes(x = carat, y = price))

**R-Instat**

To create this scatter plot in R-Instat, Describe Menu -> Specific.. -> Scatter plot.. Select the variables you wish to position on the y-axis in the first single receiver.



When we create a plot using specific plot dialogs, R-Instat by default applies the aesthetics across all layers. To specify the aesthetics inside the geom, so that these aesthetics are applied to geom\_point(), select “Scatter Plot Options”. This takes you to the Layer Options subdialog, where we must uncheck the “Apply on All Layers” checkbox, it is checked by default.



**Example 3**

We can assign variables to many aesthetic properties, such as mapping cut of diamond variable to colour of the points. Then, the colour of the points depends on the level of variable cut.

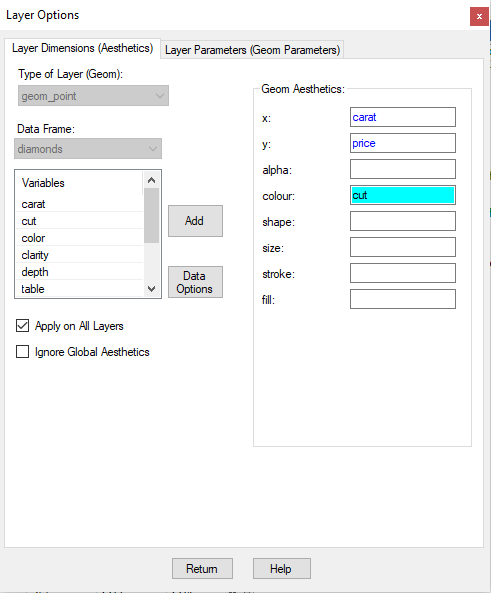
**R Code**

gg <- ggplot(data = diamonds, mapping = aes(x = carat, y = price, colour = cut)) # mapping cut variable to colour

gg + geom\_point()

**R-Instat**

To map variables to various aesthetics such as colour, size, shape etc. select “Scatter Plot Options” and specify variables for particular aesthetics on the Layer Options sub-dialog.



The following graph is produced:



**Multiple groups**

We can also assign other variables in the data frame to other aesthetics in ggplot() and/or geom function, to produce plots with multiple groups.

**Example 4**

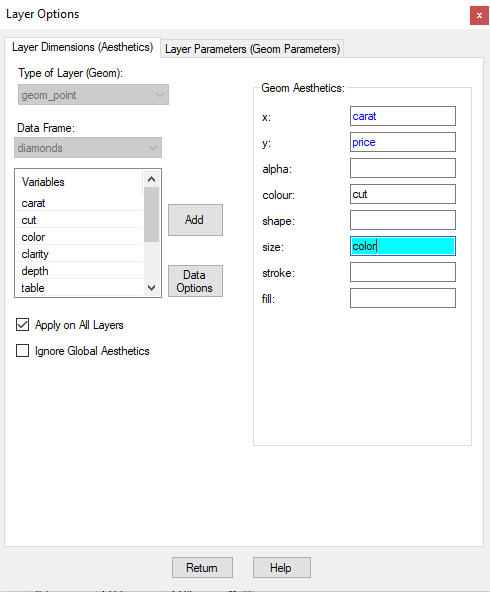
Previously we mapped cut variable to colour, so that colour is controlled by the levels of the factor variable cut. Now we will also map colour of diamond to shape, so that the shape of the points in controlled by levels of the factor variable colour.

**R Code**

gg <- ggplot(data = diamonds, mapping = aes(x = carat, y = price, colour = cut, shape = colour)) # mapping cut variable to colour and colour of diamond to shape

gg + geom\_point()

**R-Instat**



**3. Multiple Layers**

We can multiple layers to a plot. This is done by adding another geom function,which produces the visuals you wish to add to your plot.

**Applying aesthetics across all layers**

When we specify the aesthetics in the ggplot() function, all the layers (geoms) added will inherit these aesthetics**.**

**Applying aesthetics to a specific layer**

We can alternatively specify aesthetics in the geom function. The aesthetics specified in the geom will only be used by that geom, and not inherited by any other layers.

Where we specify certain aesthetics is important in producing the plot you have in mind. This is demonstrated in the next example.

**Example 4**

We can add a smoothing line to the scatter plot in example 3 by adding a second layer, geom\_smooth().

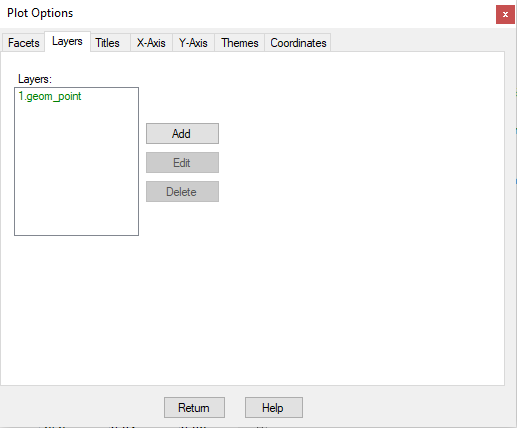
**R Code**

ggplot(diamonds, aes(x = carat, y = price, colour = cut))

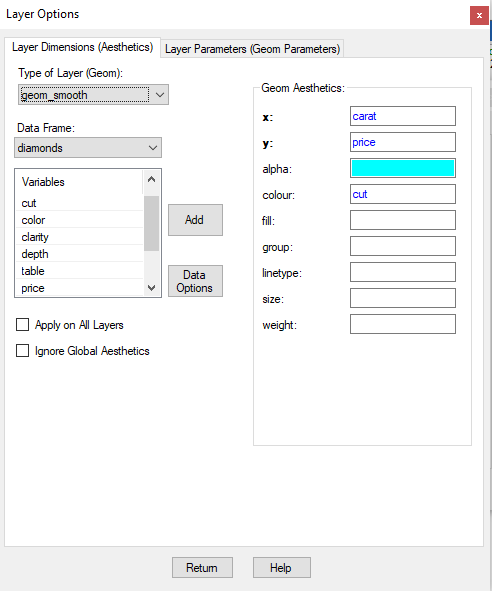
gg + geom\_point() + geom\_smooth()

**R-Instat**

Click “Plot Options” on the Scatter Plot dialog, which takes you to the Plot Options sub-dialog. Select the Layers tab, and click “Add” to add a new layer.

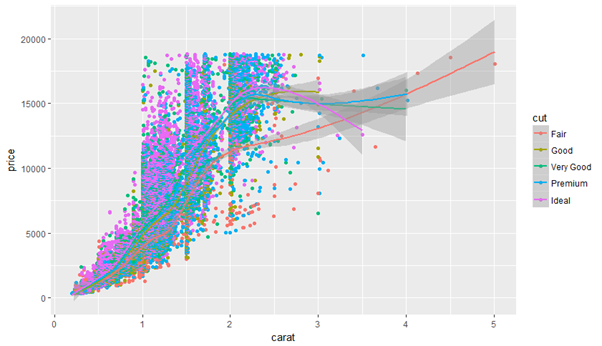


Clicking on the “Add” button takes you to the Layers Options sub-dialog. Notice that the geom drop down box is no longer greyed out. This is because we are adding a new layer, so we wish to select which layer (geom) to add to the existing plot not to add aesthetics to geom\_point().



Notice that x, y and colour are filled in with the variables from the scatter plot. This is because we had “Apply on All Layers” checked when we specified them on the scatter plot dialog. So geom\_smooth() inherits these aesthetics and as a result will produce a smoothing line for each level of the factor variable cut.

The Plot Produced:



We have added two layers (geoms) to this plot – geom\_point() and geom\_smooth(). Since the X and Y axis and the colour were defined in ggplot() set-up, these two layers inherited those aesthetics. Remember, R-Instat does this by default (- it applies the aesthetics specified on the dialogs across all layers).

Alternatively, you can specify those aesthetics inside the geom layer. When we define the colour aesthetic in the geom\_point() function, it will not be inherited by the geom\_smooth layer. This means that geom\_smooth() will produce one smoothing line.

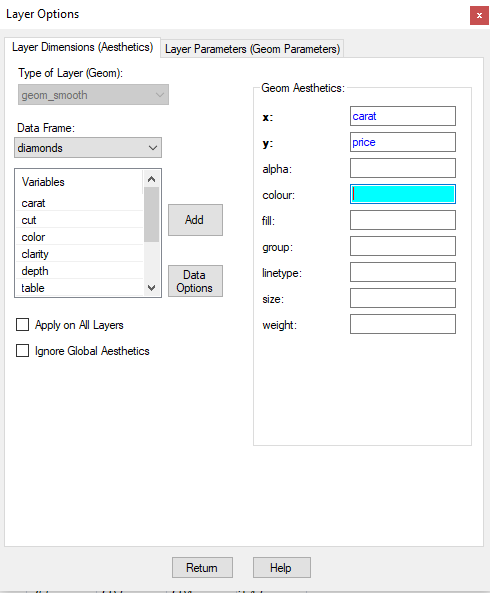
**R Code**

ggm <- ggplot(diamonds, aes(x = carat, y = price)) # removed colour from ggplot()

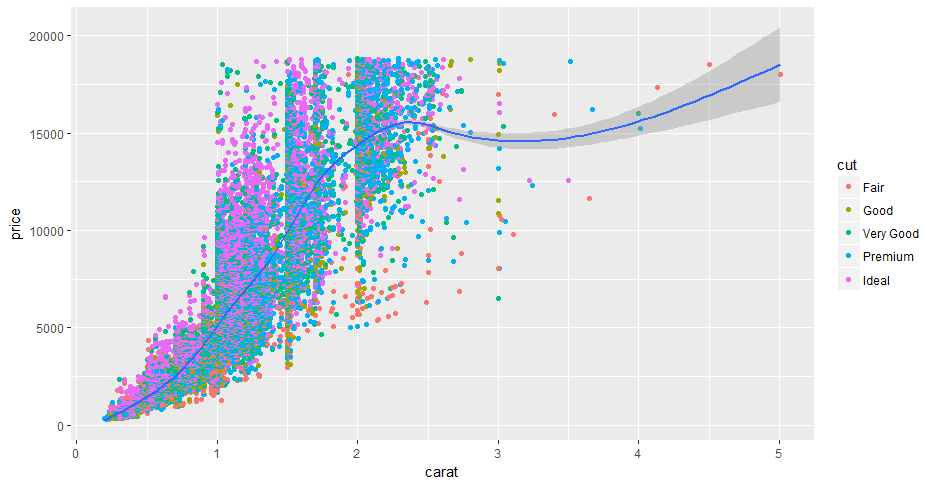
ggm + geom\_point(aes(colour = cut))+ geom\_smooth() # subsequently removed colour from geom\_smooth()

**R-Instat**

Here we can remove cut from the colour aesthetic receiver, so that geom\_smooth() does not inherit this property and produces one smoothing line.



Alternatively we could uncheck the “Apply on All layers” checkbox on the “Layer Options” sub-dialog for the Scatter Plot - accessed through “Scatter Plot Options” button on Scatter Plot dialog.



**Geom parameters**

Each geom has its own arguments, or parameters, based upon the required functioning of a plot. To demonstrate we shall show the possible arguments in geom\_point() and geom\_smooth()

**Arguments for geom\_point()**

data define the data frame here, or define a second data frame

mapping aes

size define the size of the points

colour define the colour of the points

shape define the shape of the points

**Arguments for geom\_smooth()**

method default is “loess”. Following parameter options are: “lm” to fit a regression line, “gam”

se default is FALSE. If TRUE, confidence interval is displayed around the smoothing line

level the level of the confidence interval

**Example 5**

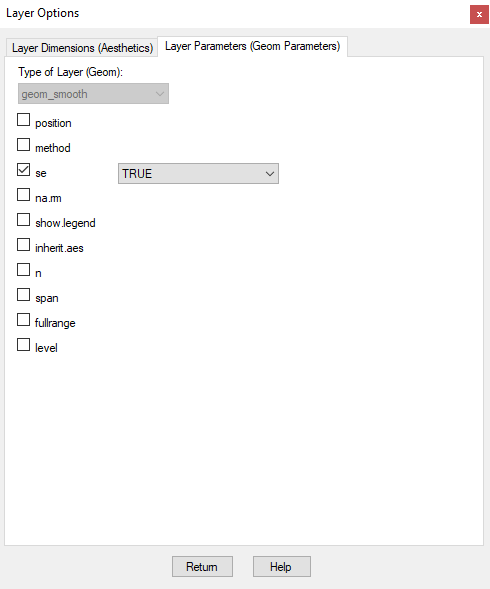
We are going to add a confidence interval around the smoothing line on the scatter plot.

**R Code**

gg + geom\_point(aes(colour = cut)) + geom\_smooth(se = TRUE) # se if true, confidence interval is displayed around the line, or smooth

**R-Instat**

To display the confidence interval of the smoothing line, click “Plot Options” on the scatter plot dialog, select the Layers tab and the geom\_smooth layer in the list of layers, then click on “Edit”. This takes you to the Layer options Sub dialog corresponding to the geom\_smooth() layer. Select the “Geom parameters” tab, check the “se” checkbox and select TRUE.

**Change the appearance of points and lines**

In previous examples we have mapped variables to arguments such as colour, shape and size, which change depending on the value of the specified variable. We can also set these arguments outside of the aes() function to fixed aesthetics.

**Example 5**

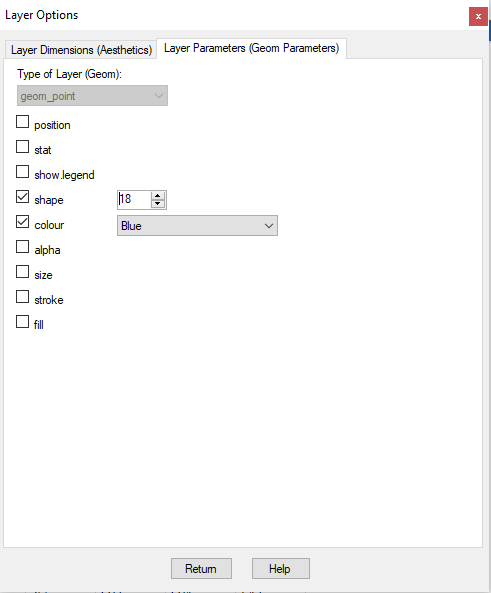
We can change the colour and shape of the points in the scatter plot to blue and diamonds.

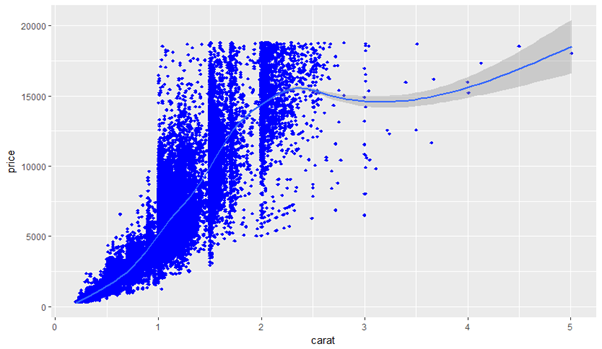
**R Code**

gg + geom\_point(colour = “blue”, shape = 18) + geom\_smooth() # changed the colour of point to blue and the shape to diamonds

**R-Instat**

On the Scatter plot dialog click on “Plot Options” -> Scatter Plot Options sub-dialog -> Geom parameters tab. Here we can check the box for Colour and change it to blue, and similarly change the shape to 18 for diamond shape.





**4. Labels**

This section will cover how to add a plot title and change the X and Y axis titles.

There are a few ways to alter and add labels to a ggplot, but we will only cover the R code that R-Instat uses.

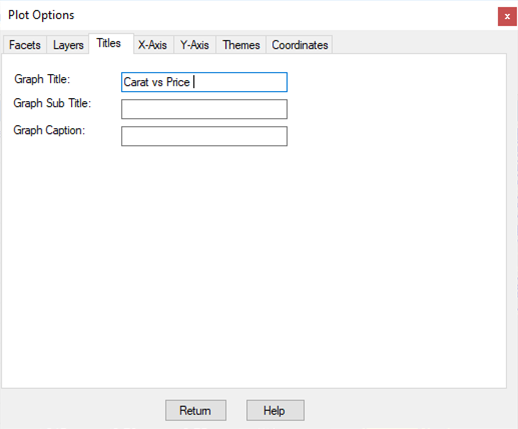
**Example 5**: Changing the x and y-axis title and adding a plot title

**R Code**

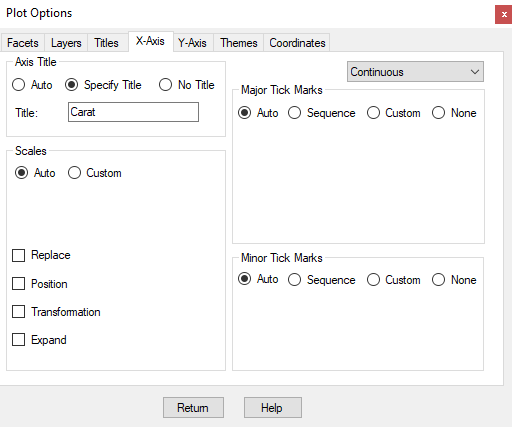
ggplot(diamonds, aes(carat, price)) + geom\_point(aes(colour = cut)) + geom\_smooth() + labs(title = “Carat vs Price”) + xlab(label = “Carat”) + ylab(label = “Price”)

**R-Instat**

To add a **plot title** to the scatter plot, select “Plot Options” on the Scatter Plot Dialog. Then select the Title tab on the Plot Options sub-dialog



To change the **X axis title**, select the X-Axis tab, click the Specify Title radio button and simply type the title you want into the Title textbox.



**5. Theme**

Many aspects of the plot can be changed using the theme() function in the ggplot2 package. Here we will look the many components which allow you to control the appearance of a plot, and example of how to change the size of the axis title and tick marker labels.

**theme() function**

Use theme() to modify individual components of a theme, allowing you to control the appearance of all non-data components of the plot. theme() only affects a single plot

theme((line, rect, text, title, aspect.ratio, axis.title, axis.title.x, axis.title.y, axis.text, axis.text.x, axis.text.y, legend.background, legend.margin, legend.spacing, legend.spacing.x, legend.spacing.y, legend.key, legend.key.size, legend.key.height, legend.key.width, legend.text, ...)

**Arguments**

line all line elements (element\_line)

rect all rectangular elements (element\_rect)

text all text elements (element\_text)

title all title elements: plot, axes, legends (element\_text; inherits from text) aspect.ratio aspect ratio of the panel

axis.title label of axes (element\_text; inherits from text)

axis.title.x x axis label (element\_text; inherits from axis.title)

axis.title.y y axis label (element\_text; inherits from axis.title)

axis.text axis tick labels (element\_text; inherits from axis.text)

axis.text.x x axis tick labels (element\_text; inherits from axis.text)

axis.text.y y axis tick labels (element\_text; inherits from axis.text)

legend.background background of legend (element\_rect; inherits from rect)

legend.margin the margin around each legend (margin)

legend.spacing the spacing between legends (unit)

Full documentation can be found here.

**Example** **6**

Here we are going to change the size of the x-axis title and labels, and the y-axis title and labels.

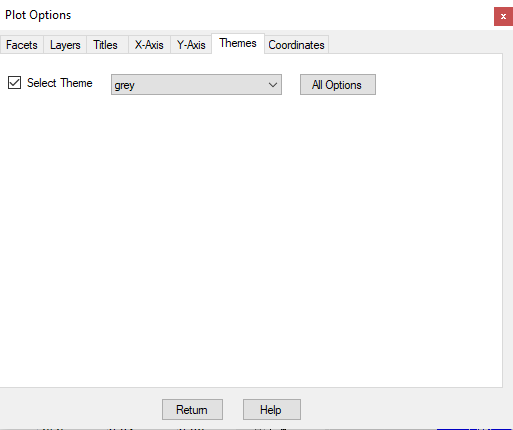
**R Code**

ggplot(diamonds, aes(carat, price)) + geom\_point(aes(colour = cut)) + geom\_smooth() + labs(title = “Carat vs Price”) + xlab(label = “Carat”) + ylab(label = “Price”) +

theme(axis.title.x = element\_text(size=10.0), axis.title.y = element\_text(size=10.0), plot.title = element\_text(size=20.0)), axis.text.x = element\_text(size = 6), axis.text.y = element\_text(size = 6))

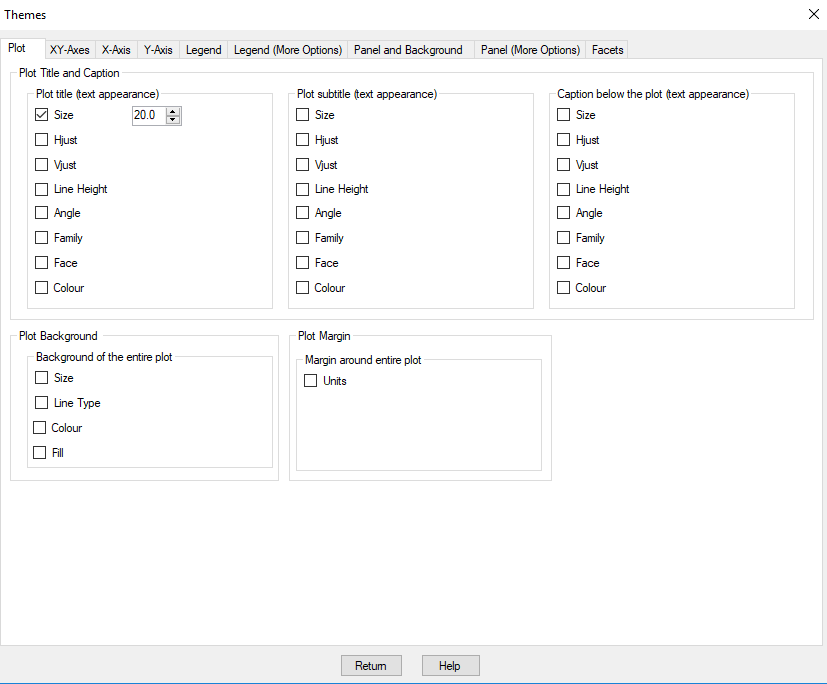
**R-Instat**

Click “Plot Options” -> Theme tab -> All Options -> Theme dialog. Here we can change the appearance of a plot by applying a specified theme from the ggthemes package.

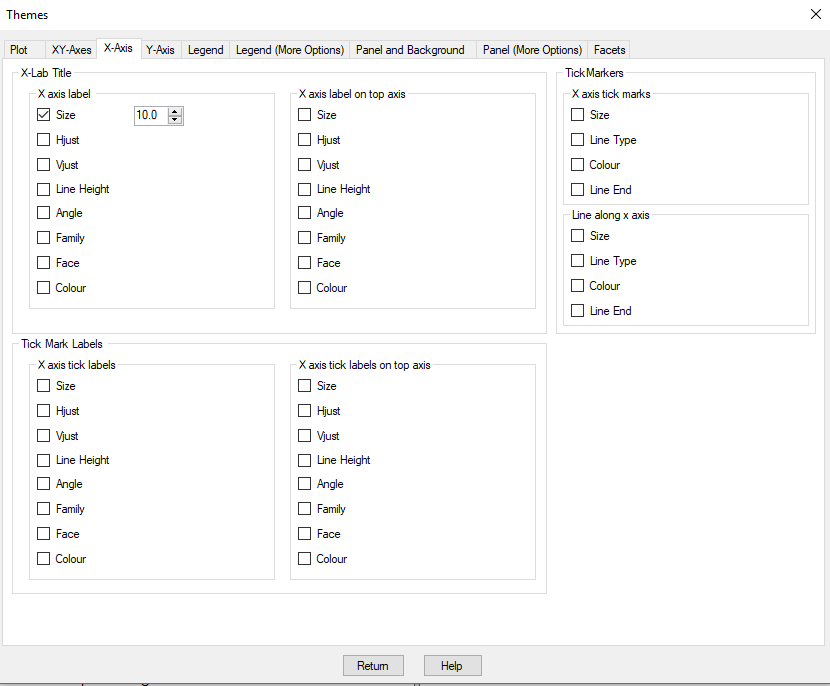


To change the size of the title, x and y-axis text, click on “All Options”, which takes you to the Theme subdialog. These changes to the plot appearance are applied to only a single plot - in the future the user will be able to save a theme and apply that theme to any plot created.

On the Plot tab, we can change the size of the plot title. Check the box for Size within Plot Title and select the size you want.



On the XY-Axis tab, we can change the size and other aspects of the axes to both the x and y axis. On the X-Axis tab we can change the aspects of the x-axis only, similarly for the y axis properties on the Y-Axis tab.



Changing the plot legend will come later in R-Instat

**6. Facets**

In the previous chart, we had the scatterplot for all different values of cut plotted in the same chart. The colour of points was controlled by the level of the variable cut. However, we can create a plot for each levels of cut. This can be done using the facet command. There are two options, facet\_wrap() or facet\_grid() function. R-Instat used facet\_wrap for and facet\_grid when.

**facet\_grid() function**

facet\_grid forms a matrix of panels defined by row and column facetting variables. It is most useful when you have two discrete variables, and all combinations of the variables exist in the data.

facet\_grid(facets, margins = FALSE, scales = "fixed", space = "fixed", shrink = TRUE, labeller = "label\_value", as.table = TRUE, switch = NULL, drop = TRUE) 32 facet\_grid

**Arguments**

**facets**  a formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column). The formula can also be provided as a string instead of a classical formula object margins either a logical value or a character vector.

**nrow, ncol**  Number of rows and columns.

**scales** Should Scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free\_x", "free\_y").

**drop** If TRUE, the default, all factor levels not used in the data will automatically be dropped. If FALSE, all factor levels will be shown, regardless of whether or not they appear in the data.

**dir** Direction: either "h" for horizontal, the default, or "v", for vertical.

**strip.position** By default, the labels are displayed on the top of the plot. Using strip.position it is possible to place the labels on either of the four sides by setting strip.position = c("top","botto

**facet\_wrap() function**

facet\_wrap wraps a 1d sequence of panels into 2d.

facet\_wrap(facets, nrow, ncol, scales, drop = TRUE, dir = "h", strip.position = "top")

**Arguments**

**facets**  Either a formula or character vector. Use either a one sided formula, ~a + b, or a character vector, c("a", "b").

**nrow, ncol**  Number of rows and columns.

**scales** Should Scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free\_x", "free\_y").

**drop** If TRUE, the default, all factor levels not used in the data will automatically be dropped. If FALSE, all factor levels will be shown, regardless of whether or not they appear in the data.

**dir** Direction: either "h" for horizontal, the default, or "v", for vertical.

**strip.position** By default, the labels are displayed on the top of the plot. Using strip.position it is possible to place the labels on either of the four sides by setting strip.position = c("top","botto

**The difference between facet\_grid() and facet\_wrap()**

For comparison purposes, you can put all the plots in a grid as well using facet\_grid

In R-Instat, facet\_wrap is used when facetting by a single variable. Facet\_grid is used when facetting by more than one variable, since it is most useful when you have two discrete variables.

**Example 7: Facetting by one variable**

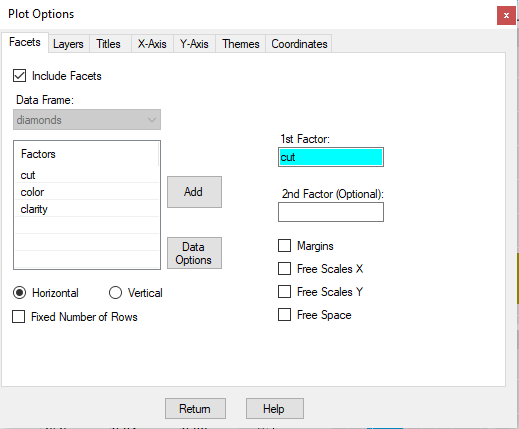
We now create a plots of Carat vs Price for each level of the factor variable Cut. There are 5 levels, so 5 graphs will be produced; fair, good, very good, premium, ideal.

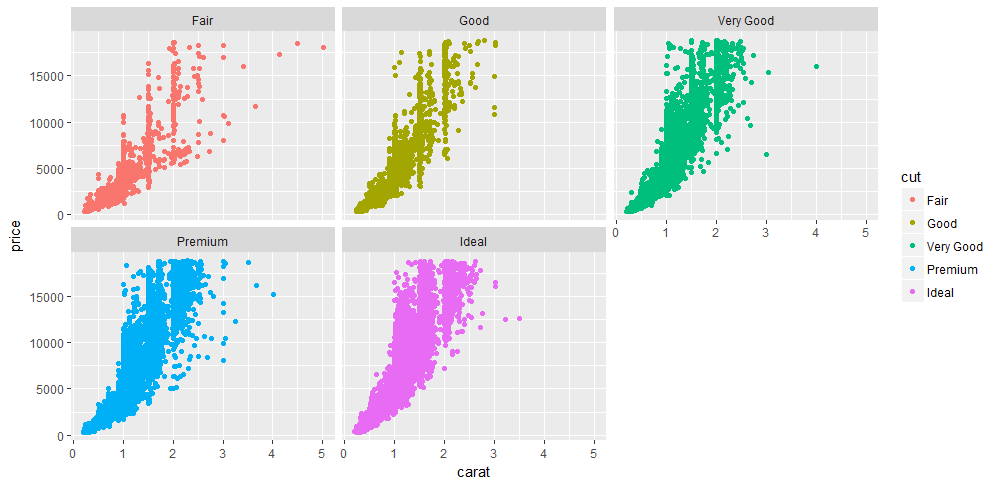
**R Code**

ggplot(diamonds, aes(x = carat, y = price)) + geom\_point() + facet\_wrap(facets= ~ cut)

**R-Instat**

Scatter Plot dialog -> Click on “Plot Options” -> Plot Options sub-dialog -> Facets tab. Select cut in the 1st Factor receiver, to produce facet plot by cut of diamond.

****

****

**Example 8: Facetting by two variables**

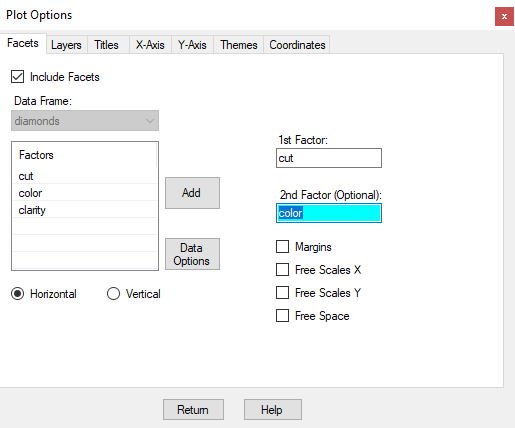
To plot carat vs price by cut and colour of diamonds, we can use facet\_grid().

**R Code**

ggplot(diamonds, aes(x = carat, y = price)) + geom\_point() + facet\_grid(facets= cut ~ colour)

**R-Instat**

Scatter Plot dialog -> Click on “Plot Options” -> Plot Options sub-dialog -> Facets tab. Select cut in the 1st Factor receiver and colour in the 2nd Factor receiver, to produce facet plot by cut and colour of diamond.

****