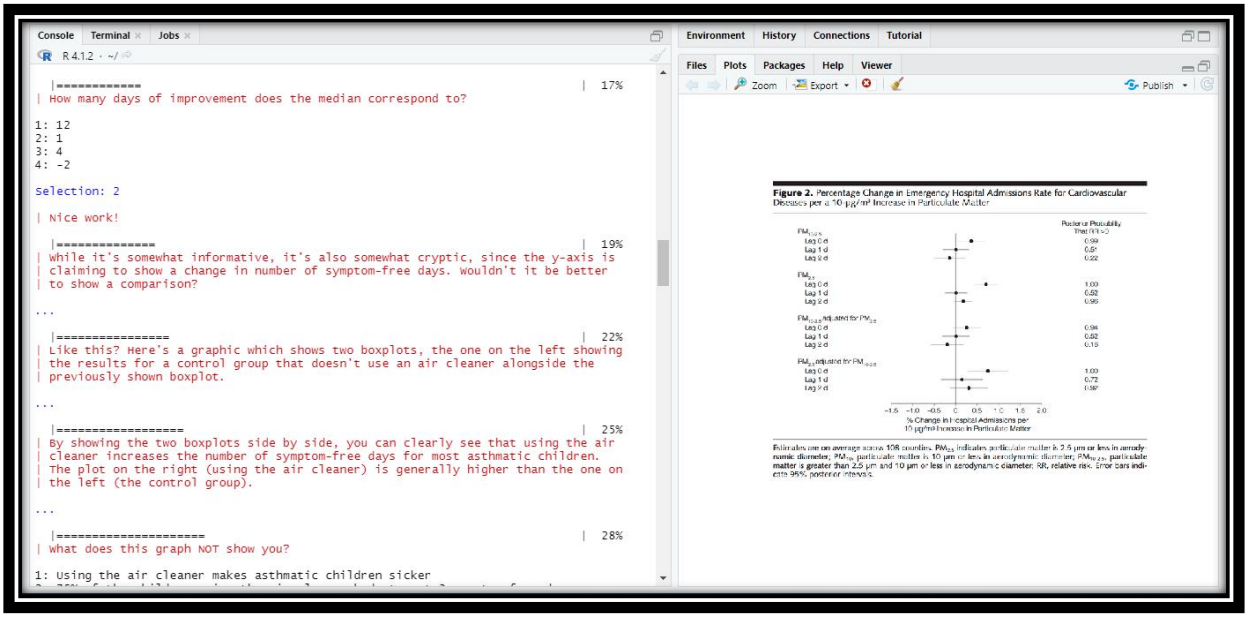
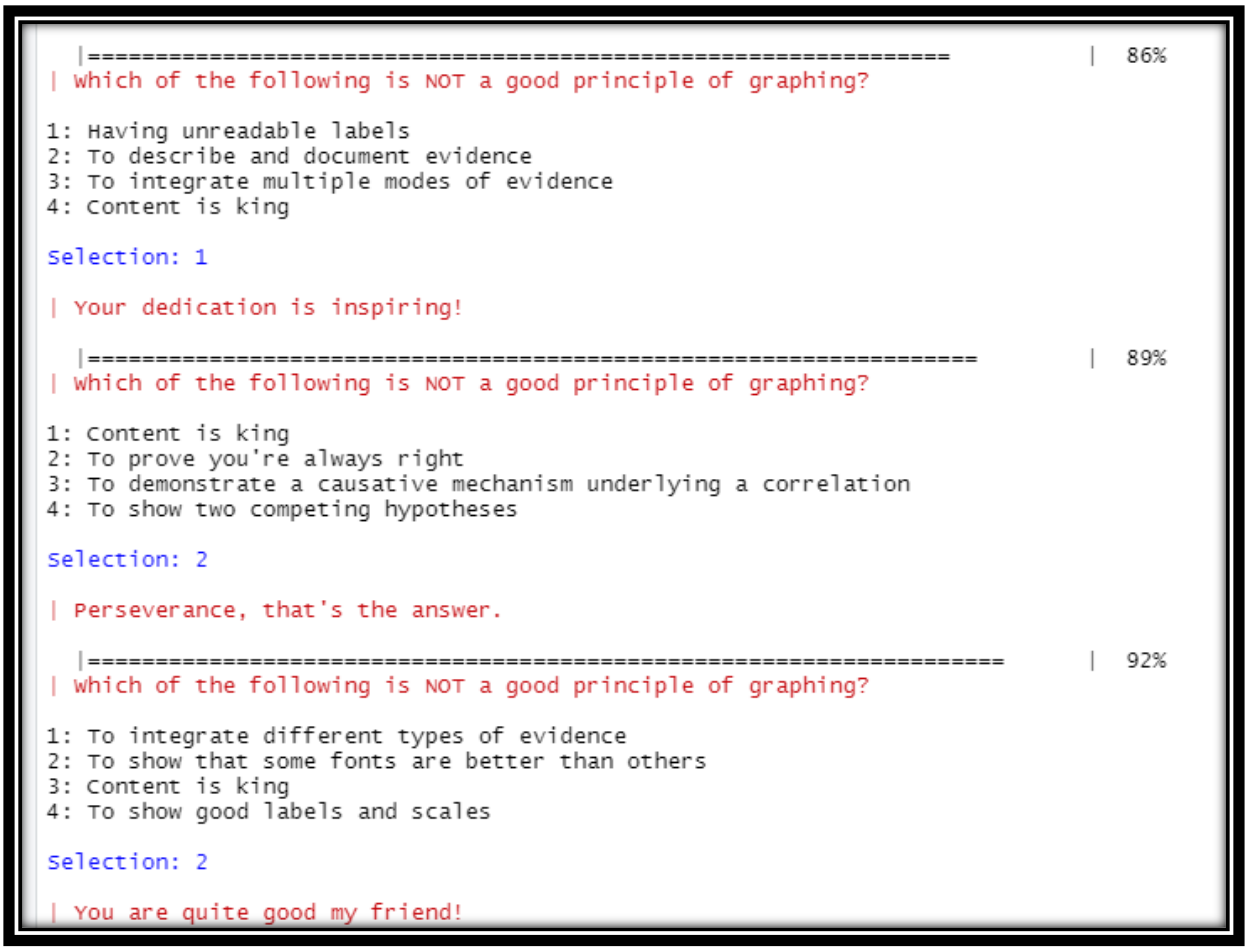
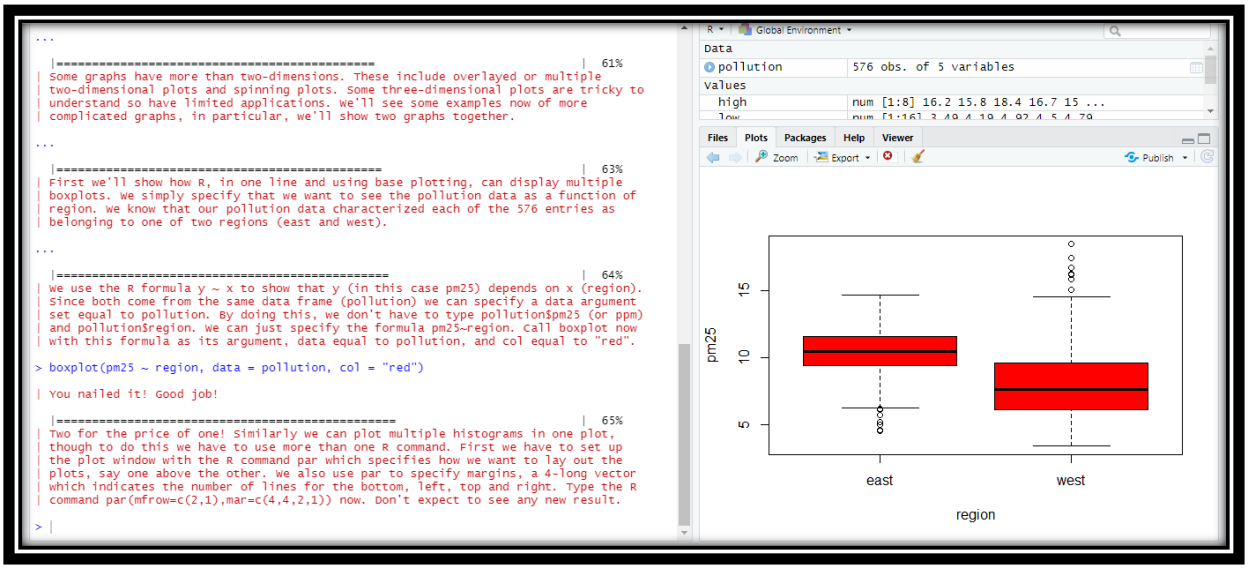
Exploratory Data Analysis

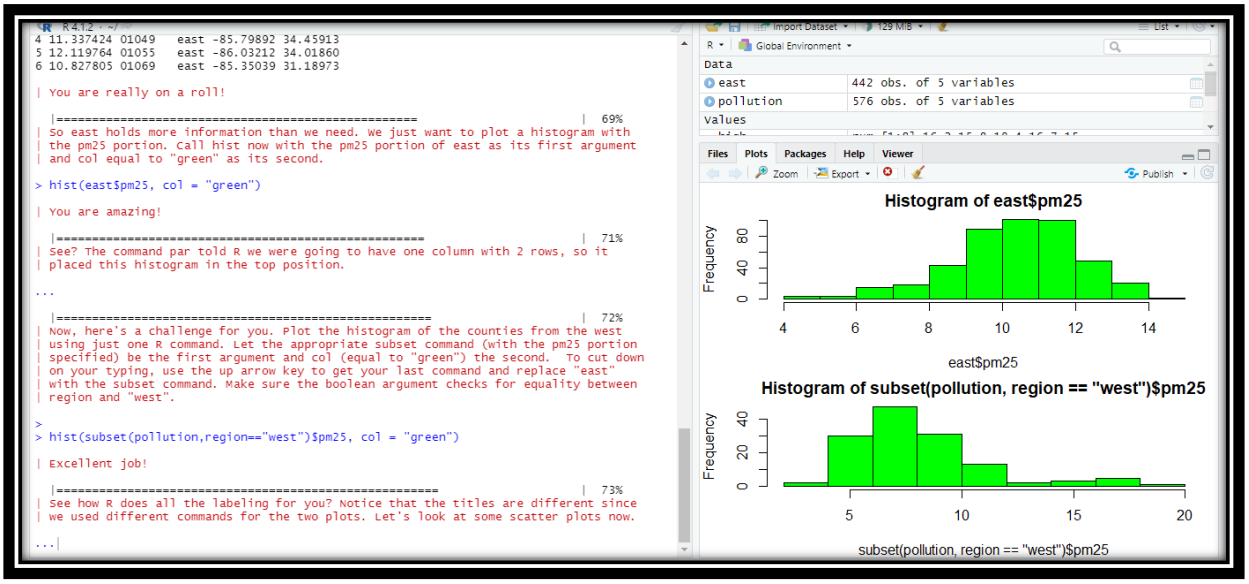
1. Principles of Analytic Graphs

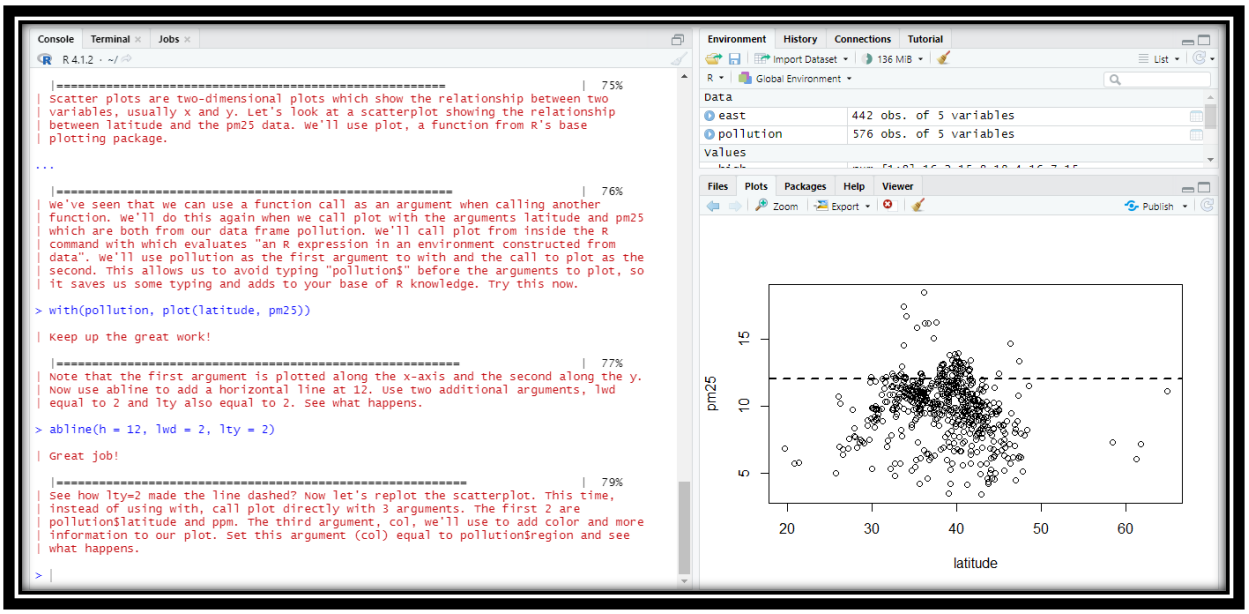


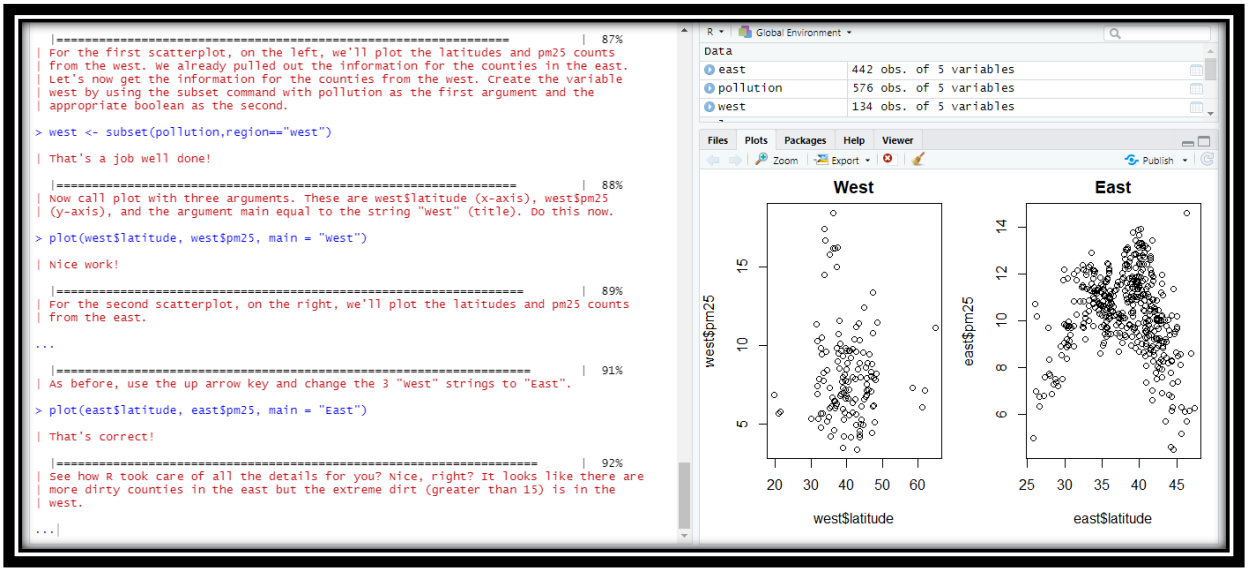


2. Exploratory Graphs

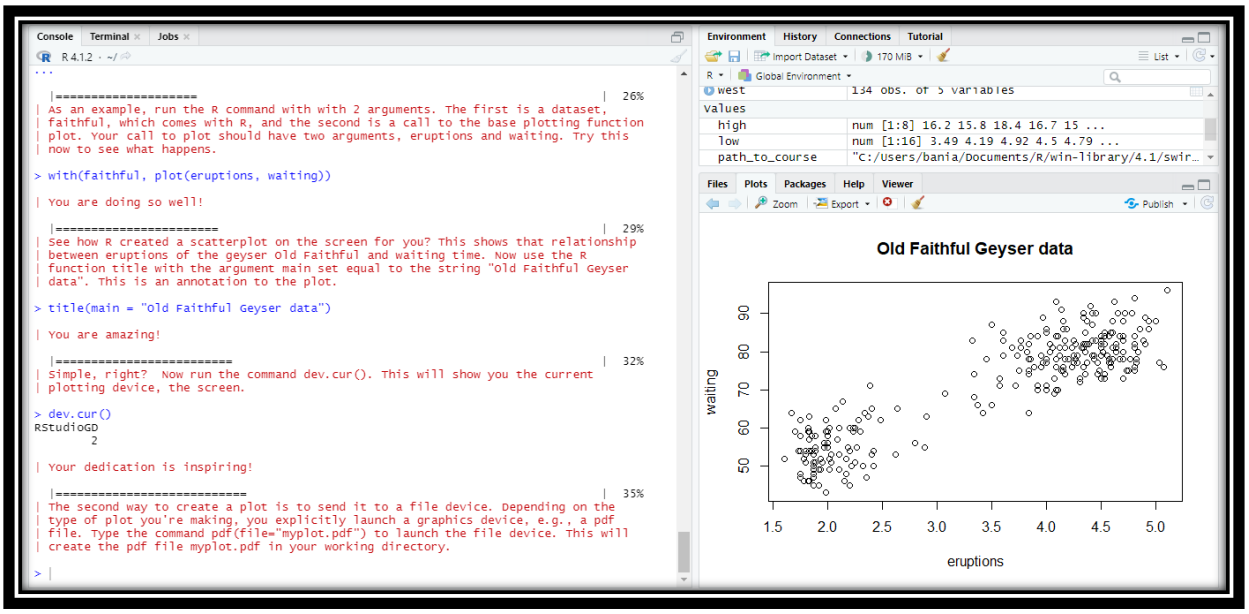


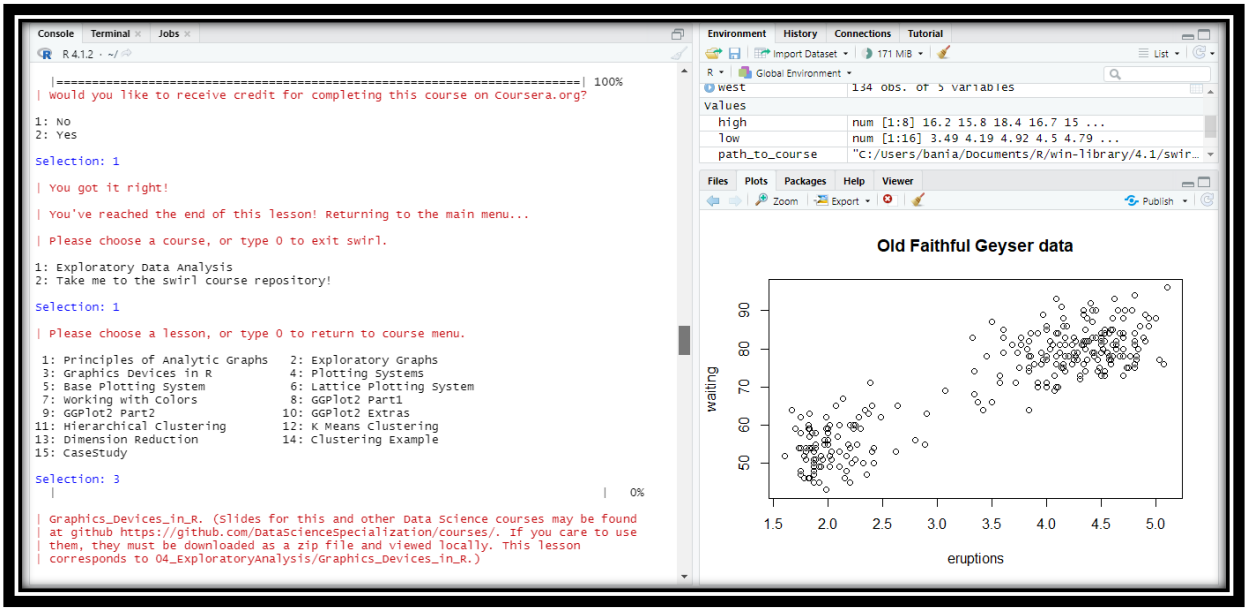






3. Graphics Devices in R

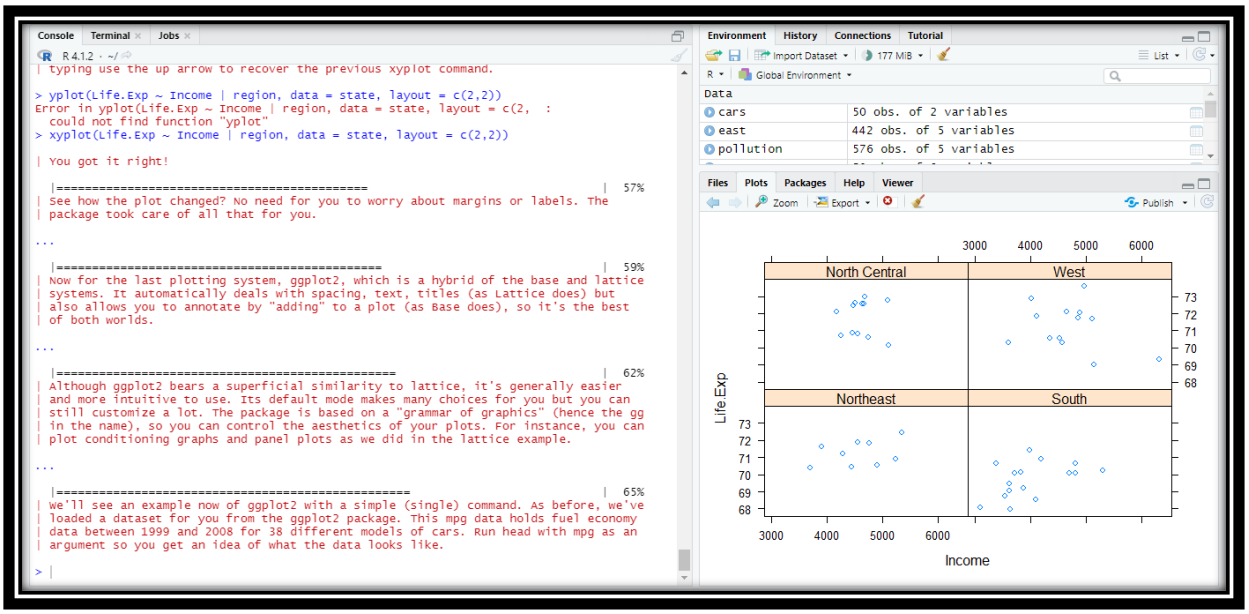


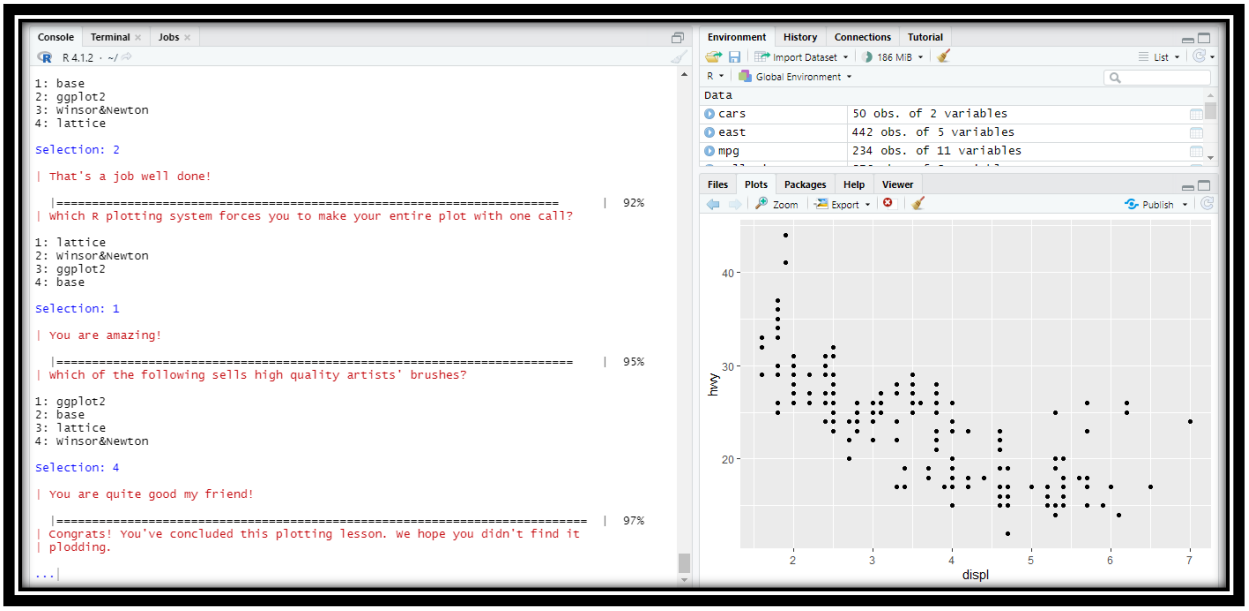


4. Plotting Systems

```
3     7    4
4     7   22
5     8   16
6     9   10

| You got it!

  |===============                                                  |  19%
| So the dataset collates the speeds and distances needed to stop for 50 cars. This
| data was recorded in the 1920's.

...

  |================                                                 |  22%
| We'll use the R command with which takes two arguments. The first specifies a
| dataset or environment in which to run the second argument, an R expression. This
| will save us a bit of typing. Try running the command with now using cars as the
| first argument and a call to plot as the second. The call to plot will take two
| arguments, speed and dist. Please specify them in that order.

> with(cars, plot(speed, dist))

| Excellent job!

  |=================                                                |  24%
| Simple, right? You can see the relationship between the two variables, speed and
| distance. The first variable is plotted along the x-axis and the second along the
| y-axis.

...

  |===================                                              |  27%
| Now we'll show you what the function text does. Run the command text with three
| arguments. The first two, x and y coordinates, specify the placement of the third
| argument, the text to be added to the plot. Let the first argument be
| mean(cars$speed), the second max(cars$dist), and the third the string "SWIRL
| rules!". Try it now.

>
```

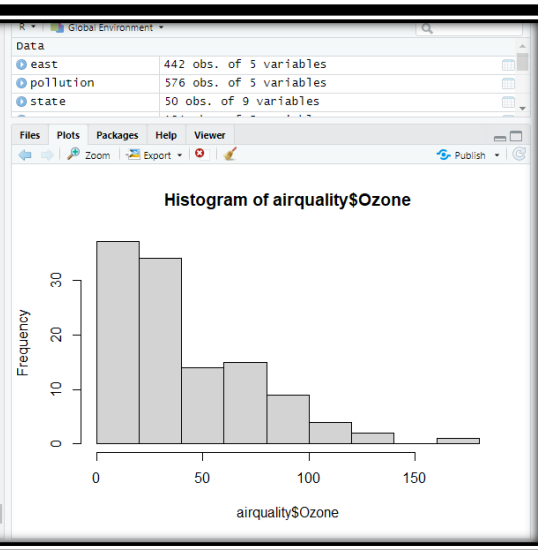
```
  |=================================                                |  46%
| So there are 4 categories and the 50 states are sorted into them appropriately.
| Let's use the lattice command xyplot to see how life expectancy varies with income
| in each of the four regions.

...

  |==================================                               |  49%
| To do this we'll give xyplot 3 arguments. The first is the most complicated. It is
| this R formula, Life.Exp ~ Income | region, which indicates we're plotting life
| expectancy as it depends on income for each region. The second argument, data, is
| set equal to state. This allows us to use "Life.Exp" and "Income" in the formula
| instead of specifying the dataset state for each term (as in state$Income). The
| third argument, layout, is set equal to the two-long vector c(4,1). Run xyplot now
| with these three arguments.

> xyplot(Life.Exp ~ Income | region, data = state, layout = c(4, 1))

| Excellent job!

  |====================================                             |  51%
| We see the data for each of the 4 regions plotted in one row. Based on this plot,
| which region of the U.S. seems to have the shortest life expectancy?

1: South
2: West
3: Northeast
4: North Central

Selection: 1

| Great job!

  |=====================================                            |  54%
| Just for fun rerun the xyplot and this time set layout to the vector c(2,2). To save
| typing use the up arrow to recover the previous xyplot command.

>
```



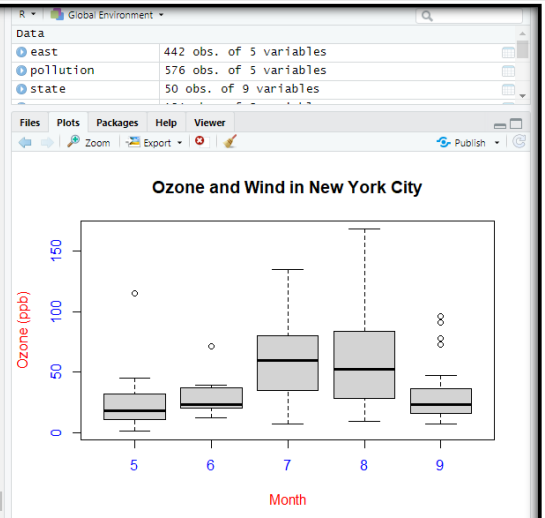


5. Base Plotting Systems

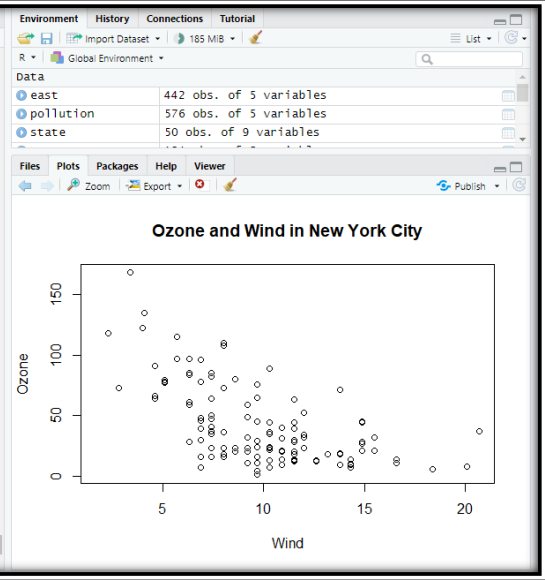
```
  |=============                                                  |  17%
| We see the dataset contains 6 columns of data. Run the command range with two
| arguments. The first is the ozone column of airquality, specified by
| airquality$Ozone, and the second is the boolean na.rm set equal to TRUE. If you
| don't specify this second argument, you won't get a meaningful result.

> range(airquality$Ozone,na.rm=TRUE)
[1]   1 168

| You got it right!

  |==============                                                 |  18%
| So the measurements range from 1 to 168. First we'll do a simple histogram of this
| ozone column to show the distribution of measurements. Use the R command hist with
| the argument airquality$Ozone.

> hist(airquality$Ozone)

| Your dedication is inspiring!

  |===============                                                |  20%
| Simple, right? R put a title on the histogram and labeled both axes for you. What is
| the most frequent count?

1: Between 60 and 75
2: Over 150
3: Over 100
4: Under 25

Selection: 4

| You are doing so well!

  |================                                               |  21%
| Next we'll do a boxplot. First, though, run the R command table with the argument
| airquality$Month.

>
```



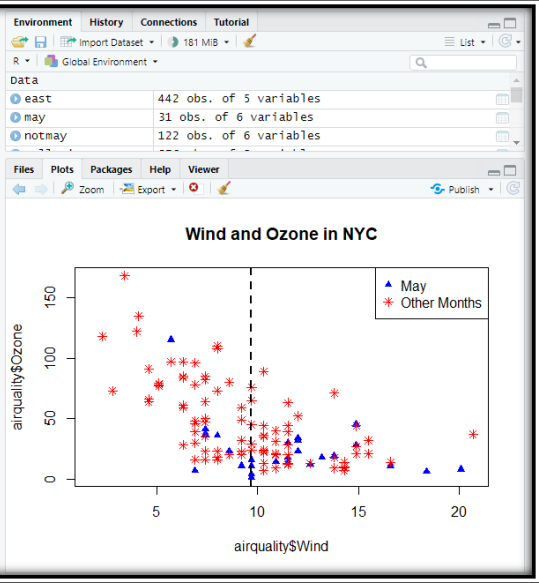

```
| You're the best!

  |==================                                             |  24%
| Note that boxplot, unlike hist, did NOT specify a title and axis labels for you
| automatically.

...

  |====================                                           |  26%
| Let's call boxplot again to specify labels. (Use the up arrow to recover the
| previous command and save yourself some typing.) We'll add more arguments to the
| call to specify labels for the 2 axes. Set xlab equal to "Month" and ylab equal to
| "Ozone (ppb)". Specify col.axis equal to "blue" and col.lab equal to "red". Try this
| now.

> boxplot(Ozone~Month, airquality, xlab="Month", ylab="Ozone (ppb)",col.axis="blue",col.la
b="red")

| You are quite good my friend!

  |====================                                           |  27%
| Nice colors, but still no title. Let's add one with the R command title. Use the
| argument main set equal to the string "Ozone and Wind in New York City".

> title(main="Ozone and Wind in New York City")

| That's the answer I was looking for.

  |======================                                         |  29%
| Now we'll show you how to plot a simple two-dimensional scatterplot using the R
| function plot. We'll show the relationship between Wind (x-axis) and Ozone (y-axis).
| We'll use the function plot with those two arguments (Wind and Ozone, in that
| order). To save some typing, though, we'll call the R command with using 2
| arguments. The first argument of with will be airquality, the dataset containing
| Wind and Ozone; the second argument will be the call to plot. Doing this allows us
| to avoid using the longer notation, e.g., airquality$Wind. Try this now.

>
```



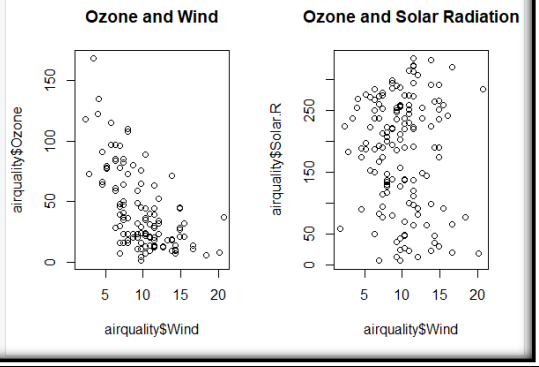

```
Console  Terminal  Jobs
R 4.1.2 · ~/
| Run the command par("lty") to see the default line type.

> par("lty")
[1] "solid"

| You are amazing!

  |=========================================                      |  50%
| So the default line type is solid, but it can be dashed, dotted, etc. Once again,
| R's ?par documentation will tell you what other line types are available. The line
| width is a positive integer; the default value is 1.

...

  |==========================================                     |  52%
| We've seen a lot of examples of col, the plotting color, specified as a number,
| string, or hex code; the colors() function gives you a vector of colors by name.

...

  |==========================================                     |  53%
| What do you think the graphical parameters xlab and ylab control respectively?

1: labels for the x- and y- axes
2: labels for the y- and x- axes

Selection: 1

| You're the best!

  |==========================================                     |  55%
| The par() function is used to specify global graphics parameters that affect all
| plots in an R session. (Use dev.off or plot.new to reset to the defaults.) These
| parameters can be overridden when specified as arguments to specific plotting
| functions. These include las (the orientation of the axis labels on the plot), bg
| (background color), mar (margin size), oma (outer margin size), mfrow and mfcol
| (number of plots per row, column).

...|
```

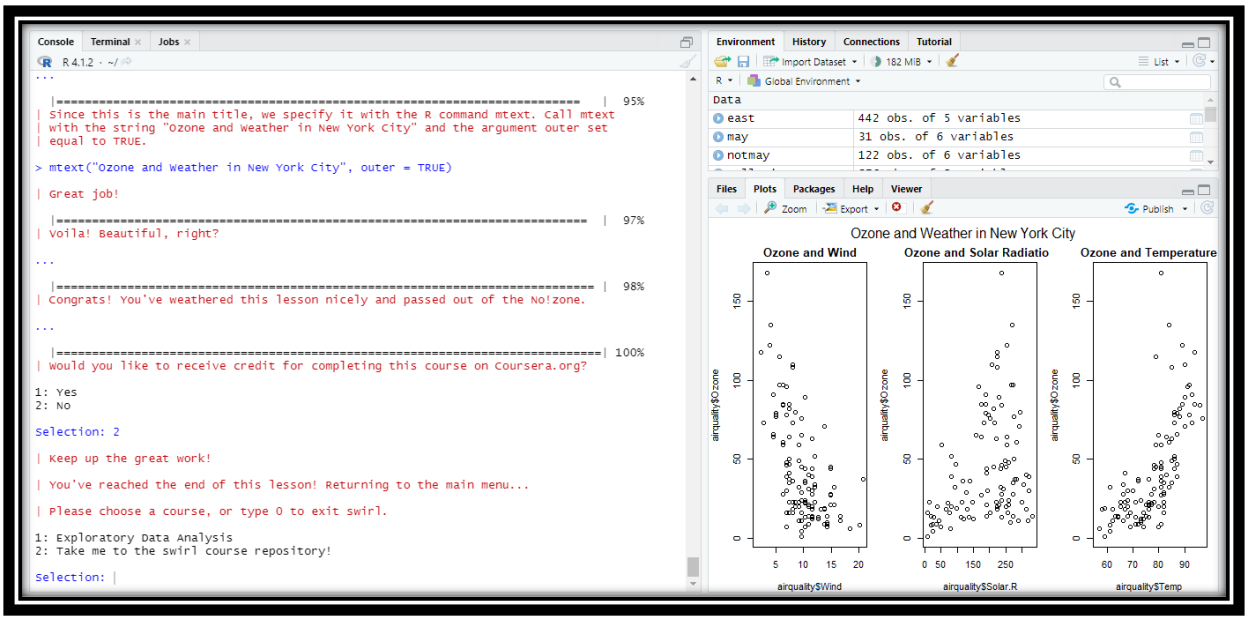


```
| as red snowflakes.

> points(notmay$Wind,notmay$Ozone,col="red",pch=8)

| Your dedication is inspiring!

  |======================================================================| 73%
| Now we'll use the R command legend to clarify the plot and explain what it means.
| The function has a lot of arguments, but we'll only use 4. The first will be the
| string "topright" to tell R where to put the legend. The remaining 3 arguments will
| each be 2-long vectors created by R's concatenate function, e.g., c(). These
| arguments are pch, col, and legend. The first is the vector (17,8), the second
| ("blue","red"), and the third ("May","other Months"). Try it now.

> legend("topright",pch=c(17,8),col=c("blue","red"),legend=c("May","Other Months"))

| Your dedication is inspiring!

  |======================================================================| 74%
| Now add a vertical line at the median of airquality$Wind. Make it dashed (lty=2)
| with a width of 2.

> abline(v=median(airquality$Wind),lty=2,lwd=2)

| You nailed it! Good job!

  |======================================================================| 76%
| Use par with the parameter mfrow set equal to the vector (1,2) to set up the plot
| window for two plots side by side. You won't see a result.

> par(mfrow=c(1,2))

| You are doing so well!

  |======================================================================| 77%
| Now plot airquality$Wind and airquality$Ozone and use main to specify the title
| "Ozone and Wind".

> plot(airquality$Wind, airquality$Ozone, main = "Ozone and Wind")
```

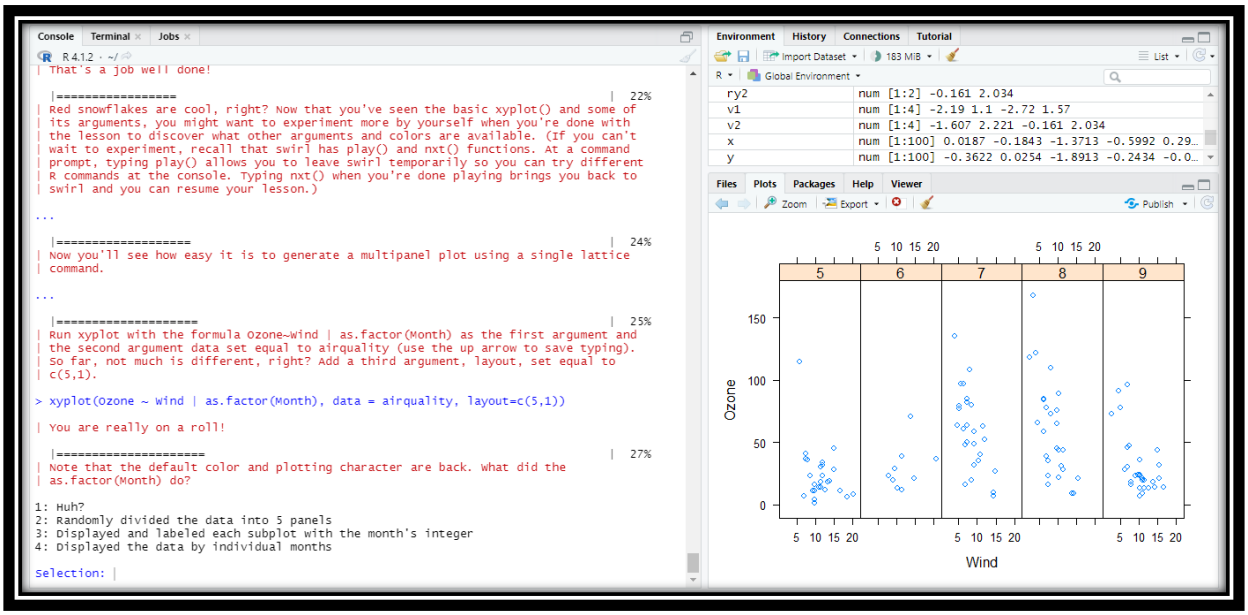



```
| Use par with the parameter mfrow set equal to the vector (1,2) to set up the plot
| window for two plots side by side. You won't see a result.

> par(mfrow=c(1,2))

| You are doing so well!

  |======================================================================| 77%
| Now plot airquality$Wind and airquality$Ozone and use main to specify the title
| "Ozone and Wind".

> plot(airquality$Wind, airquality$Ozone, main = "Ozone and Wind")

| You're the best!

  |======================================================================| 79%
| Now for the second plot.

...

  |======================================================================| 80%
| Plot airquality$Ozone and airquality$Solar.R and use main to specify the title
| "Ozone and Solar Radiation".

> plot(airquality$Ozone, airquality$Solar.R, main = "Ozone and Solar Radiation")

| You are quite good my friend!

  |======================================================================| 82%
| Now for something more challenging.

...

  |======================================================================| 83%
| This one with 3 plots, to illustrate inner and outer margins. First, set up the plot
| window by typing par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))

>
```



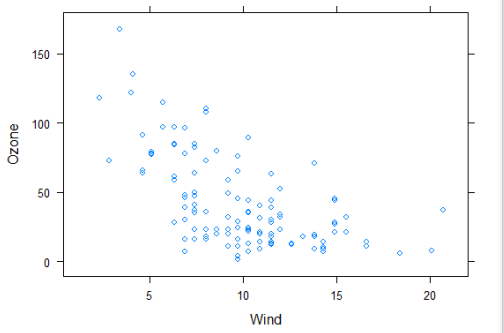


6. Lattice Plotting System



```
  |=========================                                |  33%
| The print methods for lattice functions actually do the work of plotting the data on
| the graphics device. They return "plot objects" that can be stored (but it's usually
| better to just save the code and data). On the command line, trellis objects are
| auto-printed so that it appears the function is plotting the data.

...

  |==========================                               |  34%
| To see this, create a variable p which is assigned the output of this simple call to
| xyplot, xyplot(Ozone~Wind,data=airquality).

> xyplot(Ozone ~ Wind | Month, data = airquality, layout=c(5,1))

| That's not the answer I was looking for, but try again. Or, type info() for more
| options.

| Type p <- xyplot(Ozone~Wind,data=airquality) at the command prompt.

> p <- xyplot(Ozone~Wind,data=airquality)

| You are amazing!

  |===========================                              |  36%
| Nothing plotted, right? But the object p is around.

...

  |============================                             |  37%
| Type p or print(p) now to see it.

> p

| Excellent job!

  |=============================                            |  39%
| Like magic, it appears. Now run the R command names with p as its argument.

>
```

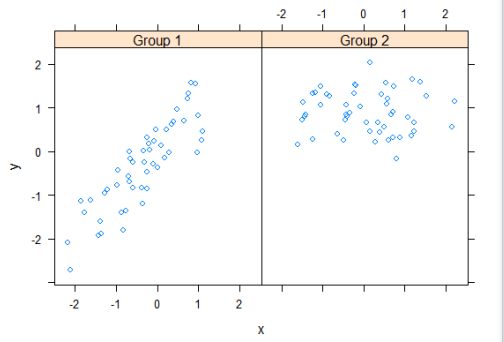
Environment	
p	List of 45
pollution	576 obs. of 5 variables
state	50 obs. of 9 variables
west	134 obs. of 5 variables

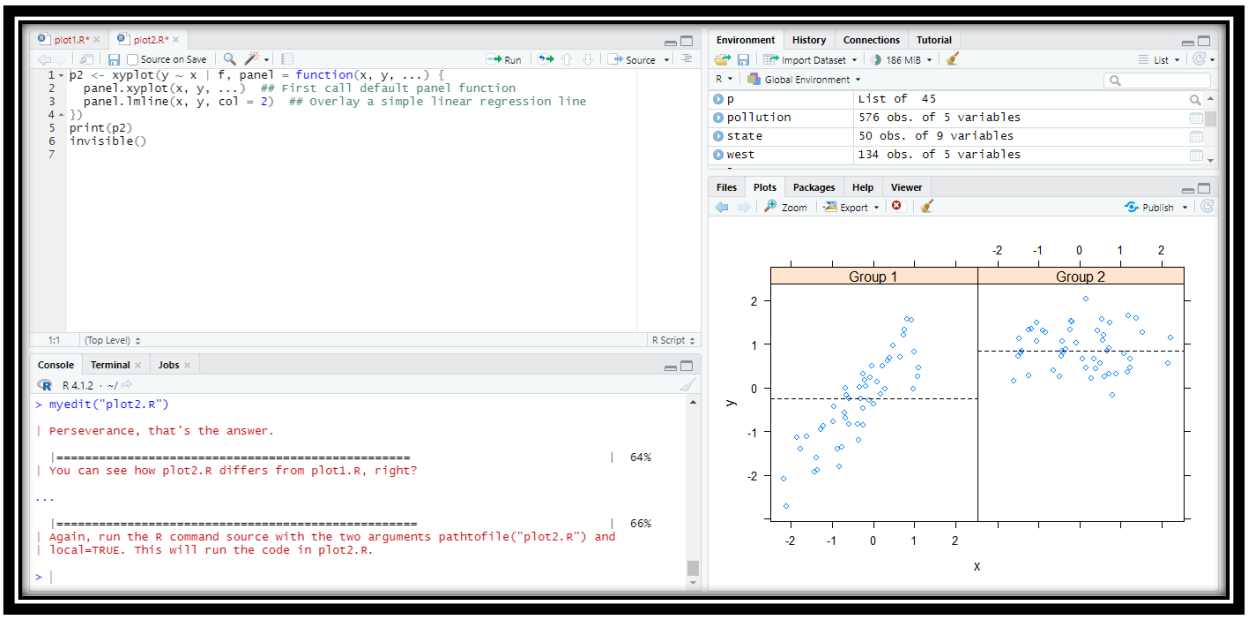


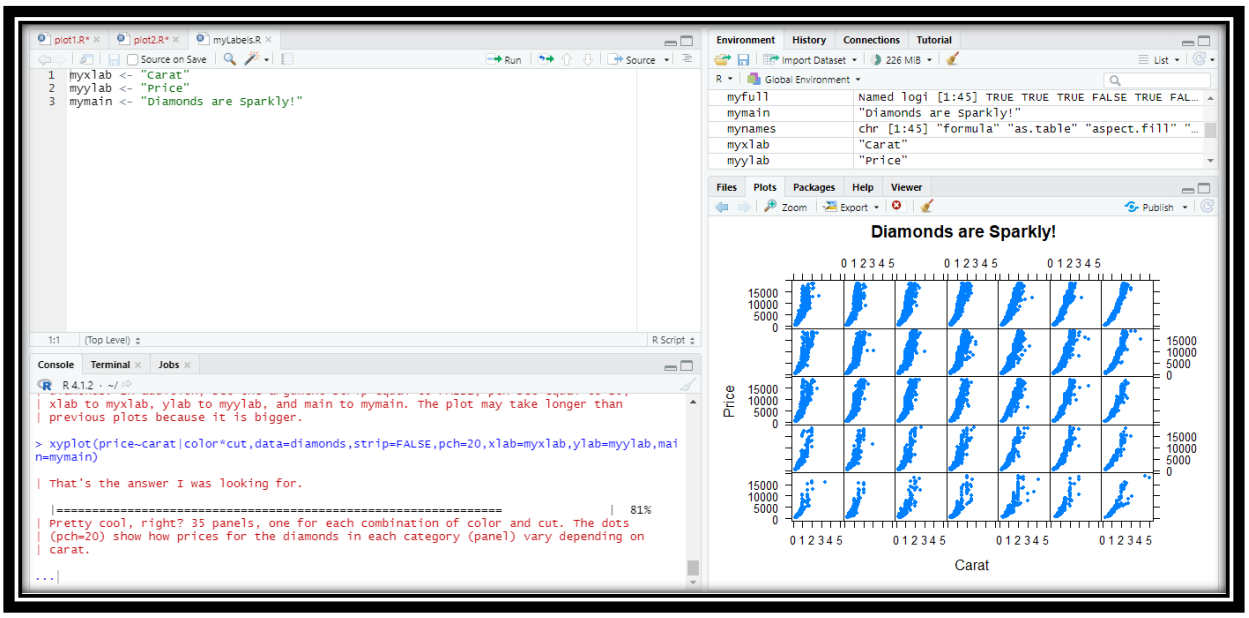

```
1 p <- xyplot(y ~ x | f, panel = function(x, y, ...) {
2   panel.xyplot(x, y, ...)  ## First call the default panel function for 'xyplot'
3   panel.abline(h = median(y), lty = 2)  ## Add a horizontal line at the median
4 })
5 print(p)
6 invisible()
7
```

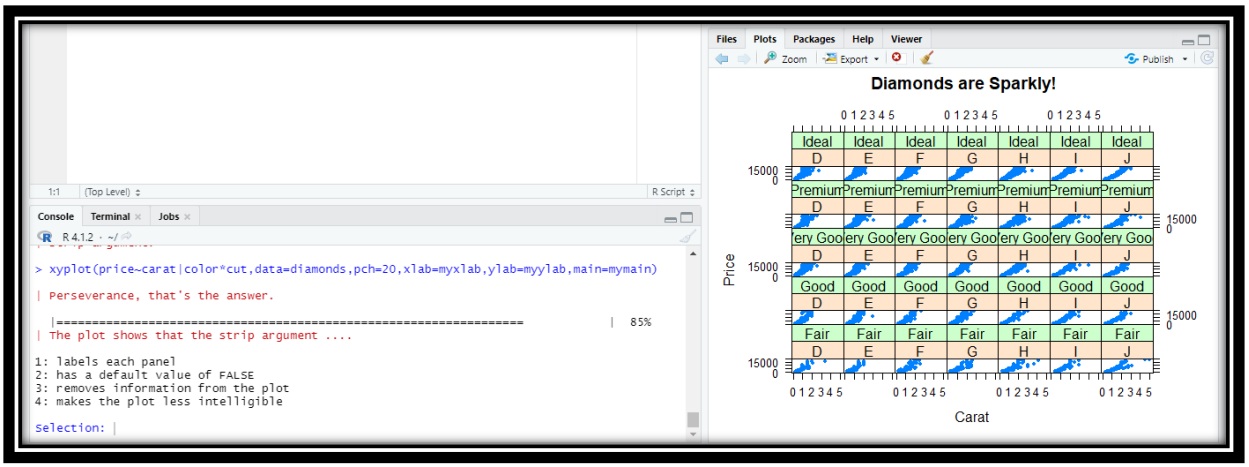
```
| Type v2 at the command prompt.

> v2
[1] -1.6066772  2.2205197 -0.1605085  2.0341048

| You nailed it! Good job!

  |=========================================                |  54%
| Again, the values match the plot. That's reassuring. We've copied some code from the
| slides for you. To see it, type myedit("plot1.R"). This will open your editor and
| display the R code in it.

> myedit("plot1.R")
```

Environment	
p	List of 45
pollution	576 obs. of 5 variables
state	50 obs. of 9 variables
west	134 obs. of 5 variables







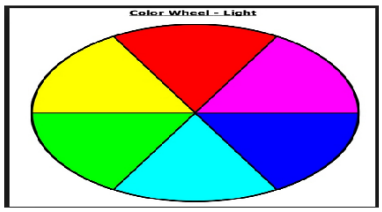


7. Working with Colors

```
1  myxlab <- "Carat"
2  myylab <- "Price"
3  mymain <- "Diamonds are Sparkly!"
```

```
  |=====================================                                |  46%
| Not surprisingly the first color we see is FF0000, which we know represents red. The
| second color returned, FFFF00, must represent yellow, a combination of full intensity
| red and full intensity green. This makes sense, since yellow falls between red and
| green on the color wheel as we see here. (we borrowed this image from
| lucaskrech.com.)

...

  |=====================================                                |  48%
| Let's now call p2 with the argument 10. This will show us how the two extremes, red
| and yellow, are blended together.

>
```

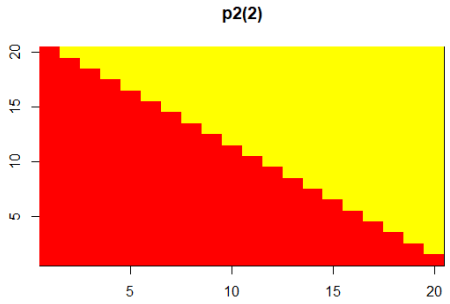
```
plotArea  num [1:4] 0 1 0 1
ppm       num [1:576] 9.77 9.99 10.69 11.34 12.12 ...
reg       'table' int [1:2(1d)] 442 134
rx1       num [1:2] -2.19 1.1
rx2       num [1:2] -1.61 2.22
```

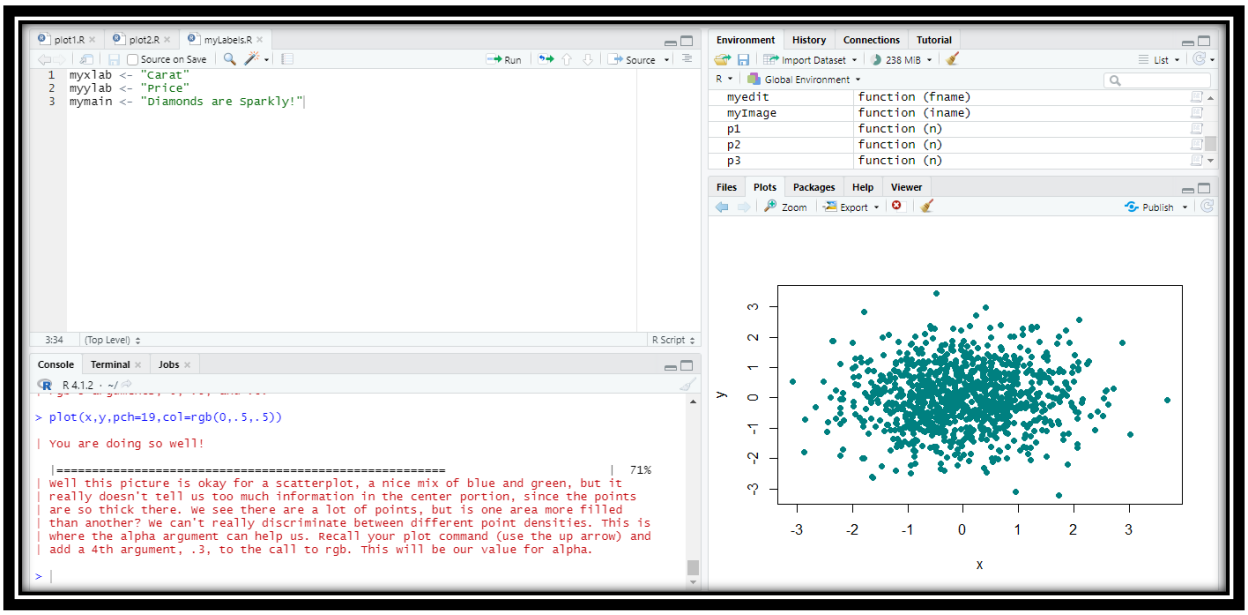



```
1  myxlab <- "Carat"
2  myylab <- "Price"
3  mymain <- "Diamonds are Sparkly!"
```

```
  |================================================                     |  62%
| We see that rgb is a color specification function that can be used to produce any
| color with red, green, blue proportions. We see the maxcolorValue is 1 by default, so
| if we called rgb with values for red, green and blue, we would specify numbers at
| most 1 (assuming we didn't change the default for maxcolorValue). According to the
| documentation, what is the maximum number of arguments rgb can have?

1: 4
2: 5
3: 3
4: 6

Selection:
```

```
plotArea  num [1:4] 0 1 0 1
ppm       num [1:576] 9.77 9.99 10.69 11.34 12.12 ...
reg       'table' int [1:2(1d)] 442 134
rx1       num [1:2] -2.19 1.1
rx2       num [1:2] -1.61 2.22
```

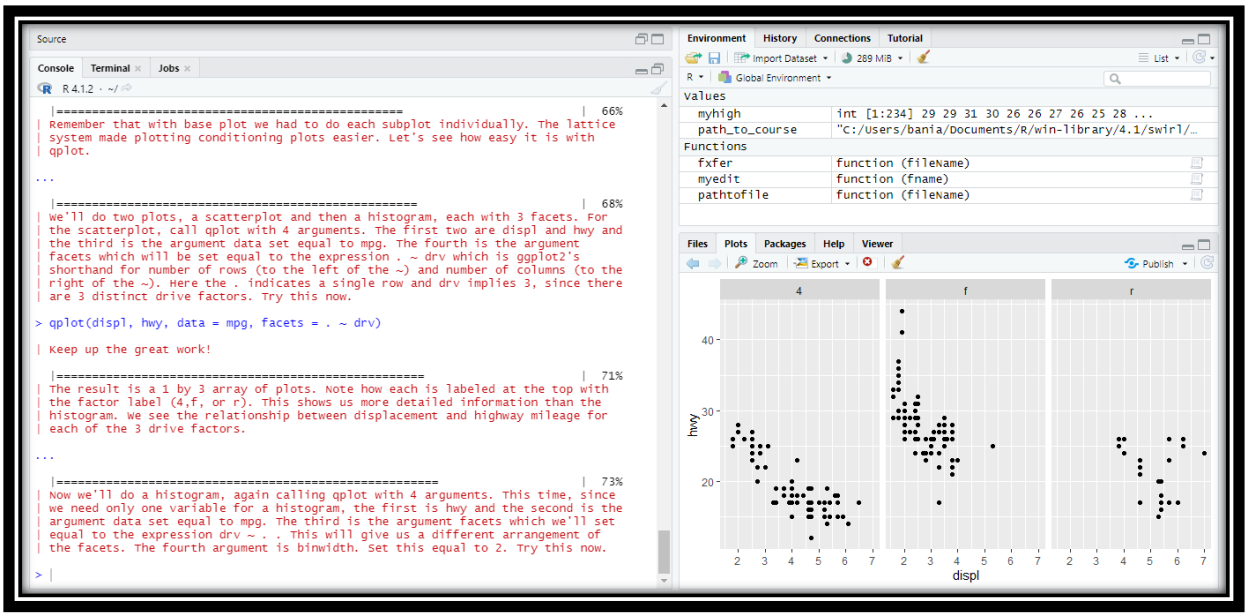




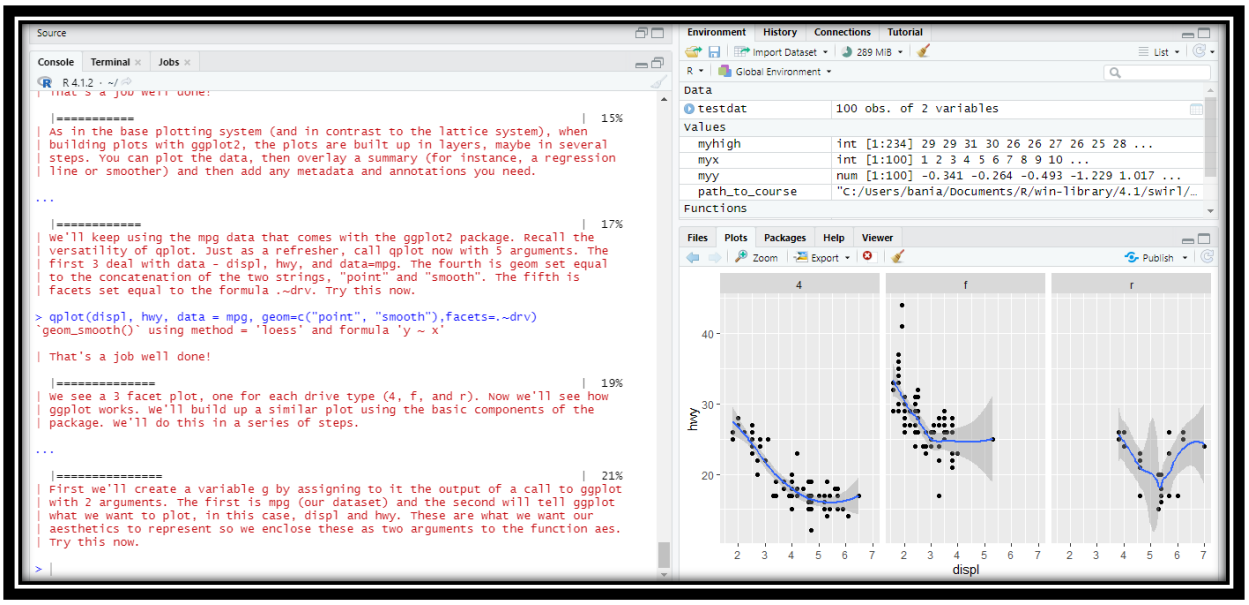
8. GGPLOT2 Part 1

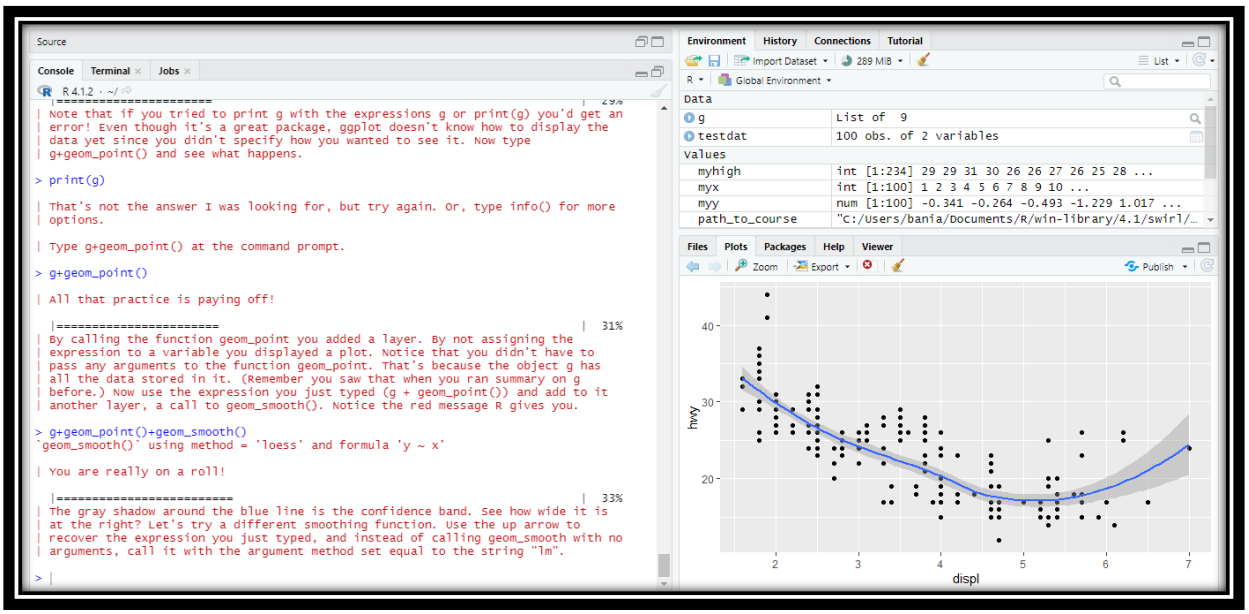
```
| the data (front-wheel, rear-wheel, and 4-wheel). Again, qplot makes this very
| easy. We'll just add what ggplot2 calls an aesthetic, a fourth argument, color,
| and set it equal to drv. Try this now. (Use the up arrow key to save some
| typing.)

> qplot(displ, hwy, data = mpg, color = drv)

| That's the answer I was looking for.

  |==========================                                            |  37%
| Pretty cool, right? See the legend to the right which qplot helpfully supplied?
| The colors were automatically assigned by qplot so the legend decodes the colors
| for you. Notice that qplot automatically used dots or points to indicate the
| data. These points are geoms (geometric objects). We could have used a different
| aesthetic, for instance shape instead of color, to distinguish between the drive
| types.

...

  |===========================                                           |  39%
| Now let's add a second geom to the default points. How about some smoothing
| function to produce trend lines, one for each color? Just add a fifth argument,
| geom, and using the R function c(), set it equal to the concatenation of the two
| strings "point" and "smooth". The first refers to the data points and second to
| the trend lines we want plotted. Try this now.

> qplot(displ, hwy, data = mpg, color=drv, geom = c("point", "smooth"))
`geom_smooth()` using method = 'loess' and formula 'y ~ x'

| That's correct!

  |=============================                                         |  41%
| Notice the gray areas surrounding each trend lines. These indicate the 95%
| confidence intervals for the lines.

...|
```

```
| Comparing the values of myhigh with the plot, we see the first entries in the
| vector (29, 29, 31, 30,...) correspond to the leftmost points in the the plot (in
| order), and the last entries in myhigh (28, 29, 26, 26, 26) correspond to the
| rightmost plotted points. So, specifying the y parameter only, without an x
| argument, plots the values of the y argument in the order in which they occur in
| the data.

...

  |=======================================                               |  51%
| The all-purpose qplot can also create box and whisker plots.  Call qplot now with
| 4 arguments. First specify the variable by which you'll split the data, in this
| case drv, then specify the variable which you want to examine, in this case hwy.
| The third argument is data (set equal to mpg), and the fourth, the geom, set
| equal to the string "boxplot"

> qplot(drv,hwy,data=mpg,geom="boxplot")

| Keep up the great work!

  |=========================================                             |  54%
| We see 3 boxes, one for each drive. Now to impress you, call qplot with 5
| arguments. The first 4 are just as you used previously, (drv, hwy, data set equal
| to mpg, and geom set equal to the string "boxplot"). Now add a fifth argument,
| color, equal to manufacturer.

> qplot(drv,hwy,data=mpg,geom="boxplot",color=manufacturer)

| You got it right!

  |=============================================                         |  56%
| It's a little squished but we just wanted to illustrate qplot's capabilities.
| Notice that there are still 3 regions of the plot (determined by the factor drv).
| Each is subdivided into several boxes depicting different manufacturers.

...|
```



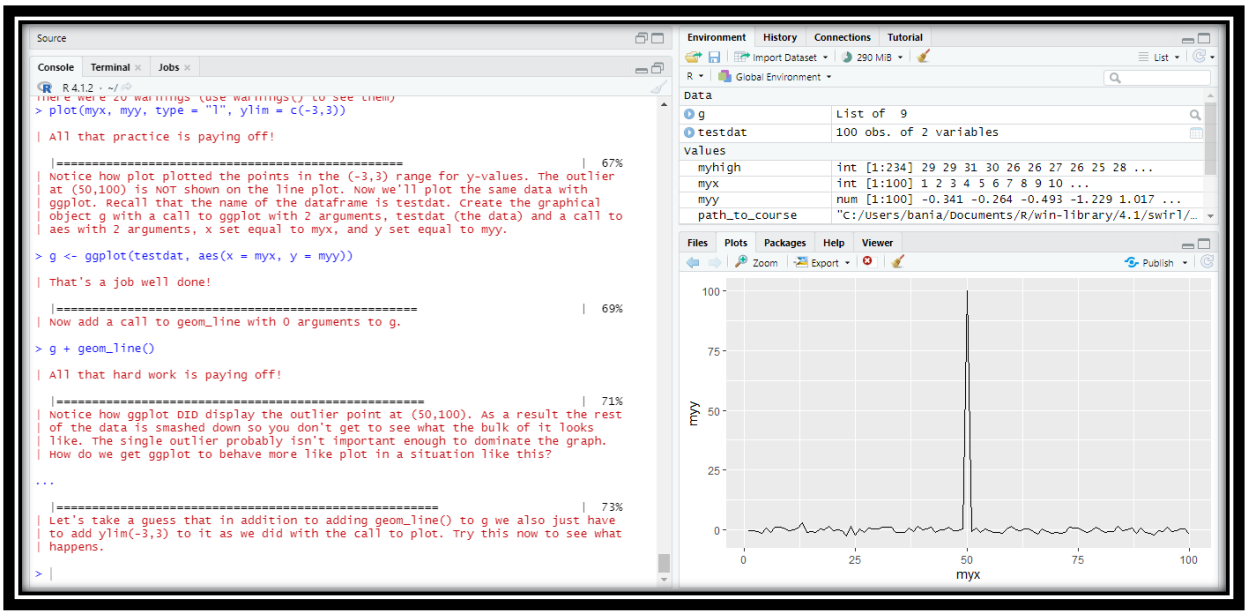
9. GGPLOT2 Part 2

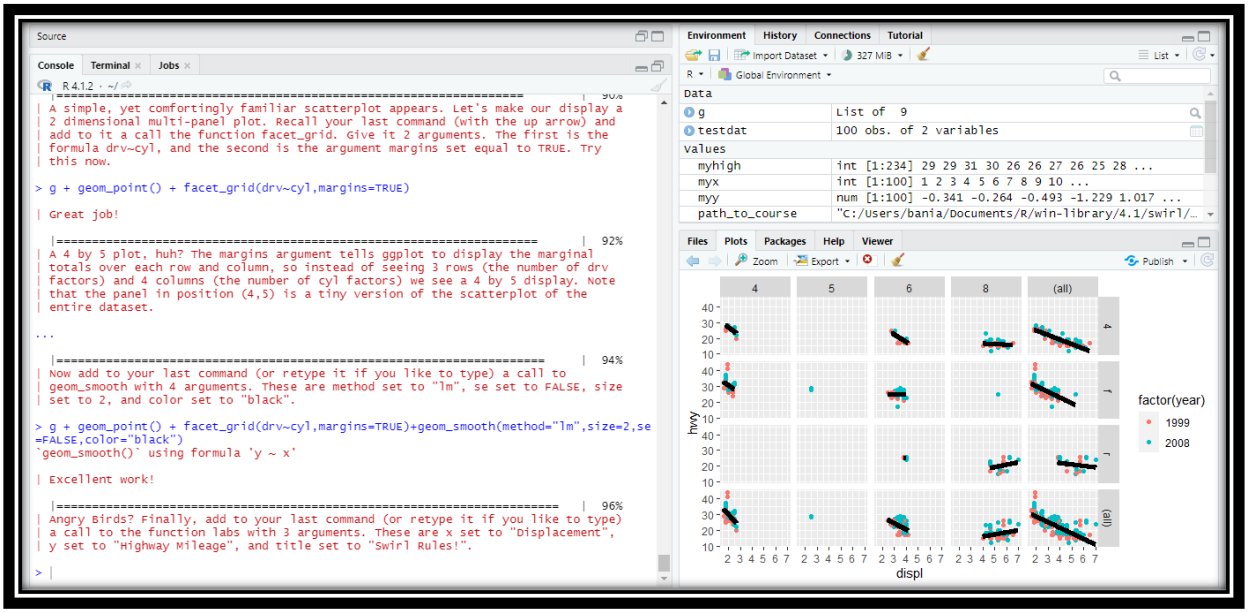

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
That's a job well done!
|========== | 15%
| As in the base plotting system (and in contrast to the lattice system), when
| building plots with ggplot2, the plots are built up in layers, maybe in several
| steps. You can plot the data, then overlay a summary (for instance, a regression
| line or smoother) and then add any metadata and annotations you need.
...
|========== | 17%
| We'll keep using the mpg data that comes with the ggplot2 package. Recall the
| versatility of qplot. Just as a refresher, call qplot now with 5 arguments. The
| first 3 deal with data - displ, hwy, and data=mpg. The fourth is geom set equal
| to the concatenation of the two strings, "point" and "smooth". The fifth is
| facets set equal to the formula .~drv. Try this now.
> qplot(displ, hwy, data = mpg, geom=c("point", "smooth"),facets=.~drv)
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
| That's a job well done!
|========== | 19%
| We see a 3 facet plot, one for each drive type (4, f, and r). Now we'll see how
| ggplot works. We'll build up a similar plot using the basic components of the
| package. We'll do this in a series of steps.
...
|========== | 21%
| First we'll create a variable g by assigning to it the output of a call to ggplot
| with 2 arguments. The first is mpg (our dataset) and the second will tell ggplot
| what we want to plot, in this case, displ and hwy. These are what we want our
| aesthetics to represent so we enclose these as two arguments to the function aes.
| Try this now.
>
Environment
History
Connections
Tutorial
Import Dataset
289 MiB
List
R
Global Environment
Data
testdat
100 obs. of 2 variables
Values
myhigh
int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
myx
int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy
num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
path_to_course
"C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Functions
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
4
f
r
hwy
displ


Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| Note that if you tried to print g with the expressions g or print(g) you'd get an
| error! Even though it's a great package, ggplot doesn't know how to display the
| data yet since you didn't specify how you wanted to see it. Now type
| g+geom_point() and see what happens.
> print(g)
| That's not the answer I was looking for, but try again. Or, type info() for more
| options.
| Type g+geom_point() at the command prompt.
> g+geom_point()
| All that practice is paying off!
|========== | 31%
| By calling the function geom_point you added a layer. By not assigning the
| expression to a variable you displayed a plot. Notice that you didn't have to
| pass any arguments to the function geom_point. That's because the object g has
| all the data stored in it. (Remember you saw that when you ran summary on g
| before.) Now use the expression you just typed (g + geom_point()) and add to it
| another layer, a call to geom_smooth(). Notice the red message R gives you.
> g+geom_point()+geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
| You are really on a roll!
|========== | 33%
| The gray shadow around the blue line is the confidence band. See how wide it is
| at the right? Let's try a different smoothing function. Use the up arrow to
| recover the expression you just typed, and instead of calling geom_smooth with no
| arguments, call it with the argument method set equal to the string "lm".
>
Environment
History
Connections
Tutorial
Import Dataset
289 MiB
List
R
Global Environment
Data
g
List of 9
testdat
100 obs. of 2 variables
Values
myhigh
int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
myx
int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy
num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
path_to_course
"C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
hwy
displ

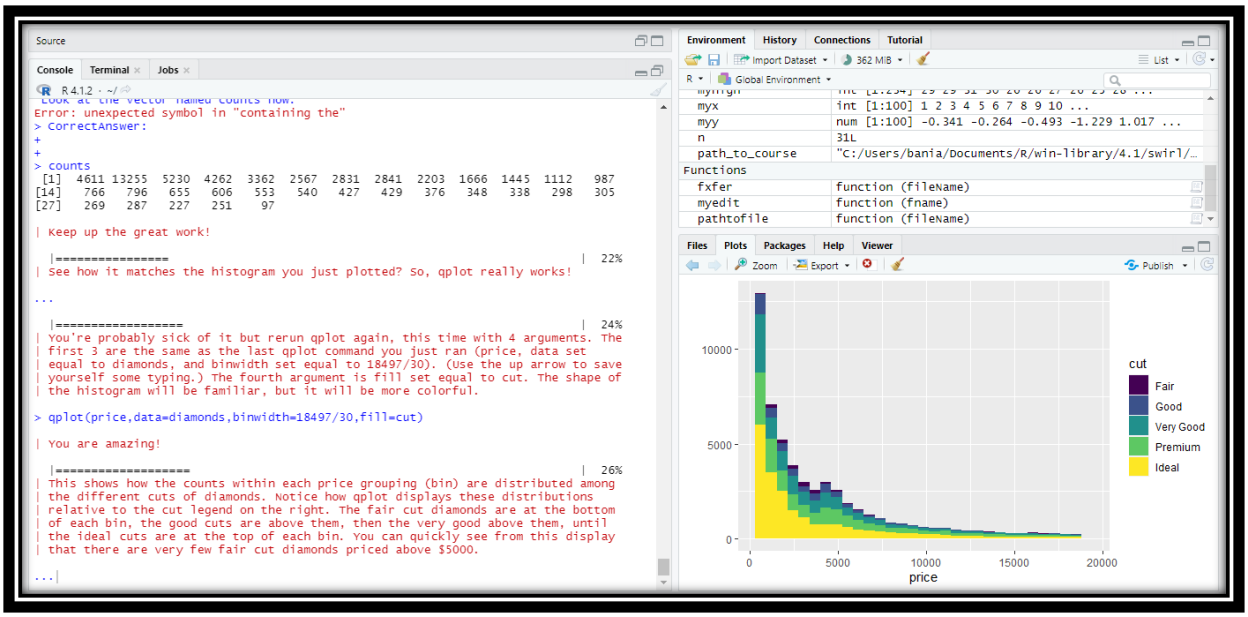
```
| arguments, call it with the argument method set equal to the string "lm".

> g+geom_point()+geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'

| You are amazing!

  |=========================                                             |  35%
| By changing the smoothing function to "lm" (linear model) ggplot2 generated a
| regression line through the data. Now recall the expression you just used and add
| to it another call, this time to the function facet_grid. Use the formula . ~ drv
| as it argument. Note that this is the same type of formula used in the calls to
| qplot.

> g+geom_point()+geom_smooth(method="lm") + facet_grid(.~drv)
`geom_smooth()` using formula 'y ~ x'

| Keep up the great work!

  |=============================                                         |  38%
| Notice how each panel is labeled with the appropriate factor. All the data
| associated with 4-wheel drive cars is in the leftmost panel, front-wheel drive
| data is shown in the middle panel, and rear-wheel drive data in the rightmost.
| Notice that this is similar to the plot you created at the start of the lesson
| using qplot. (We used a different smoothing function than previously.)

...

  |=============================                                         |  40%
| So far you've just used the default labels that ggplot provides. You can add your
| own annotation using functions such as xlab(), ylab(), and ggtitle(). In
| addition, the function labs() is more general and can be used to label either or
| both axes as well as provide a title. Now recall the expression you just typed
| and add a call to the function ggtitle with the argument "Swirl Rules!".

>
```

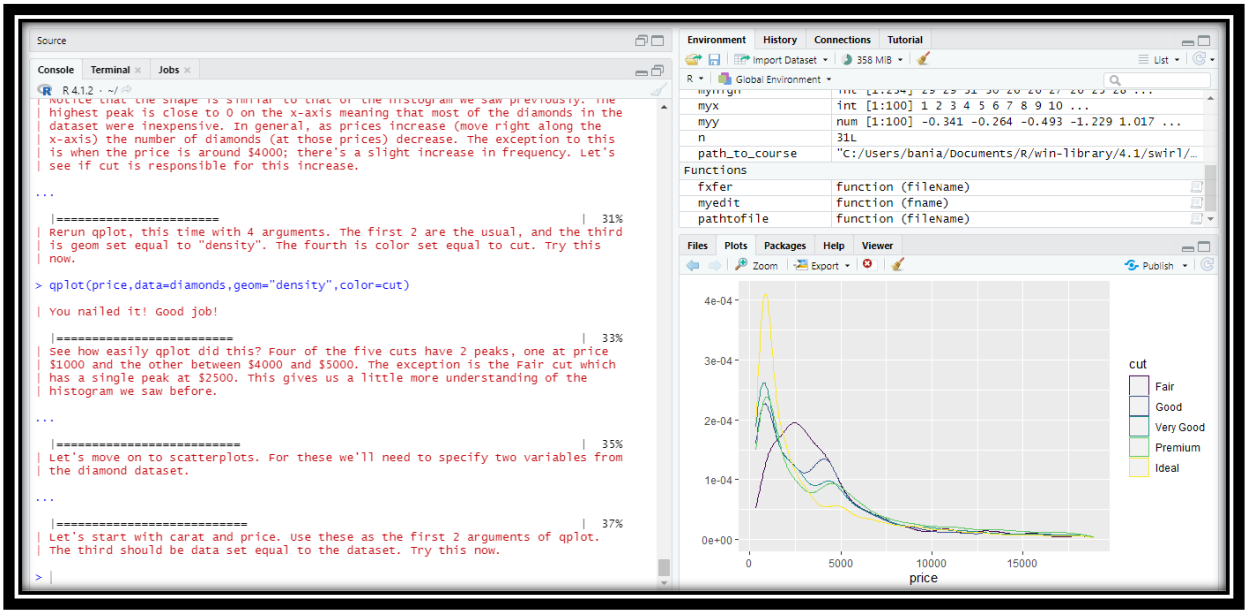
```
| "displacement" and the other setting y equal to "Hwy Mileage".

> g + geom_point(aes(color = drv)) + labs(title="Swirl Rules!") + labs(x="Displacemen
t", y="Hwy Mileage")

| That's the answer I was looking for.

  |=========================================                             |  54%
| Note that you could have combined the two calls to the function labs in the
| previous example. Now we'll practice customizing the geom_smooth calls. Use g and
| add to it a call to geom_point setting the color to drv type (remember to use the
| call to the aes function), size set to 2 and alpha to 1/2. Then add a call to
| geom_smooth with 4 arguments. Set size equal to 4, linetype to 3, method to "lm",
| and se to FALSE.

> g + geom_point(aes(color = drv),size=2,alpha=1/2) + geom_smooth(size=4,linetype=3,met
hod="lm",se=FALSE)
`geom_smooth()` using formula 'y ~ x'

| That's a job well done!

  |=============================================                         |  56%
| What did these arguments do? The method specified a linear regression (note the
| negative slope indicating that the bigger the displacement the lower the gas
| mileage), the linetype specified that it should be dashed (not continuous), the
| size made the dashes big, and the se flag told ggplot to turn off the gray
| shadows indicating standard errors (confidence intervals).

...

  |=============================================                         |  58%
| Finally, let's do a simple plot using the black and white theme, theme_bw.
| Specify g and add a call to the function geom_point with the argument setting the
| color to the drv type. Then add a call to the function theme_bw with the argument
| base_family set equal to "Times". See if you notice the difference.

>
```





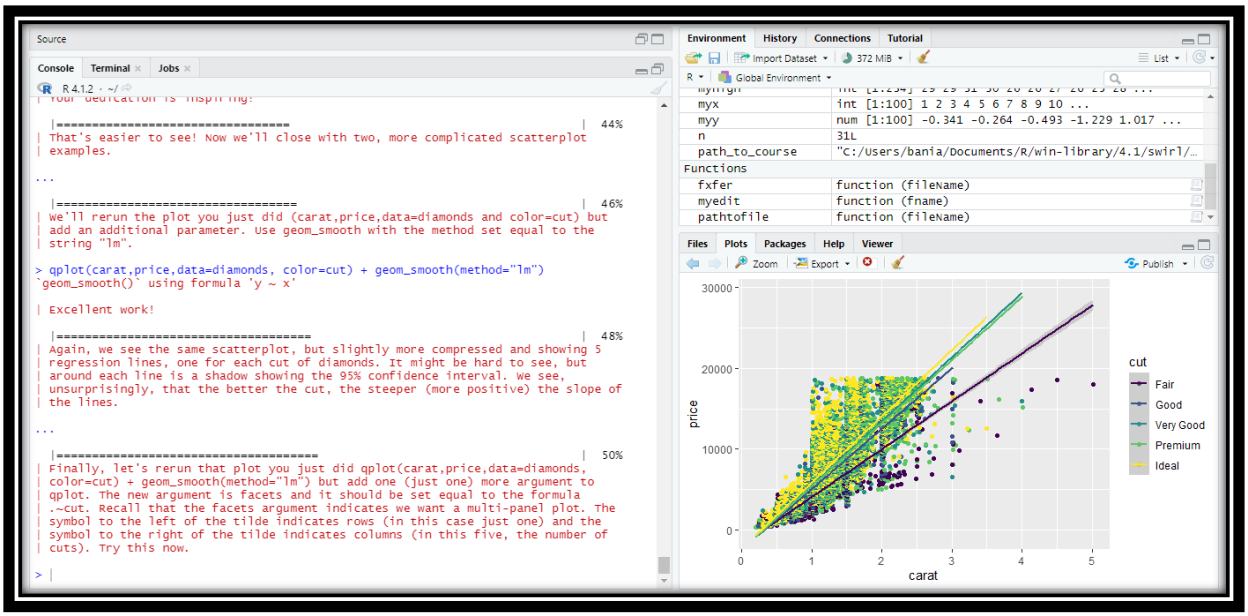
10. GGPLOT2 Extra

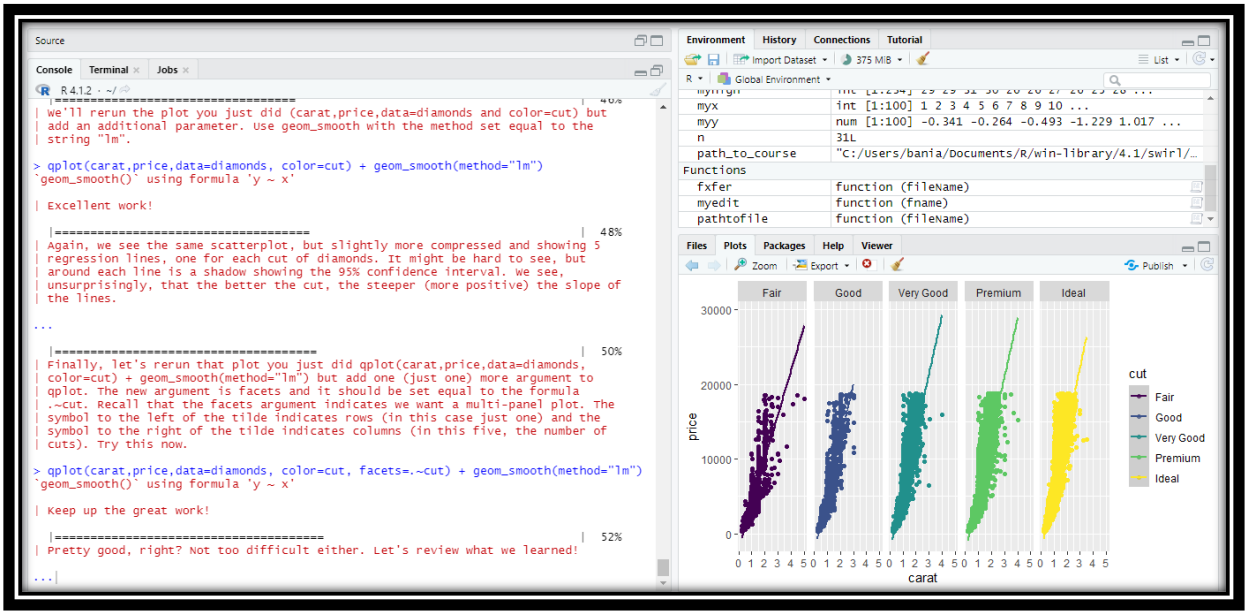

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
Look at the vector named counts now.
Error: unexpected symbol in "containing the"
> CorrectAnswer:
+
+
> counts
[1] 4611 13255 5230 4262 3362 2567 2831 2841 2203 1666 1445 1112 987
[14] 766 796 655 606 553 540 427 429 376 348 338 298 305
[27] 269 287 227 251 97
| Keep up the great work!
| 22%
| See how it matches the histogram you just plotted? So, qplot really works!
...
| 24%
| You're probably sick of it but rerun qplot again, this time with 4 arguments. The
| first 3 are the same as the last qplot command you just ran (price, data set
| equal to diamonds, and binwidth set equal to 18497/30). (Use the up arrow to save
| yourself some typing.) The fourth argument is fill set equal to cut. The shape of
| the histogram will be familiar, but it will be more colorful.
> qplot(price,data=diamonds,binwidth=18497/30,fill=cut)
| You are amazing!
| 26%
| This shows how the counts within each price grouping (bin) are distributed among
| the different cuts of diamonds. Notice how qplot displays these distributions
| relative to the cut legend on the right. The fair cut diamonds are at the bottom
| of each bin, the good cuts are above them, then the very good above them, until
| the ideal cuts are at the top of each bin. You can quickly see from this display
| that there are very few fair cut diamonds priced above $5000.
...
Environment
History
Connections
Tutorial
Import Dataset
362 MiB
List
Global Environment
myx int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
n 31L
path_to_course "C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Functions
fxfer function (fileName)
myedit function (fname)
pathtofile function (fileName)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
10000
5000
0
0
5000
10000
15000
20000
price
cut
Fair
Good
Very Good
Premium
Ideal

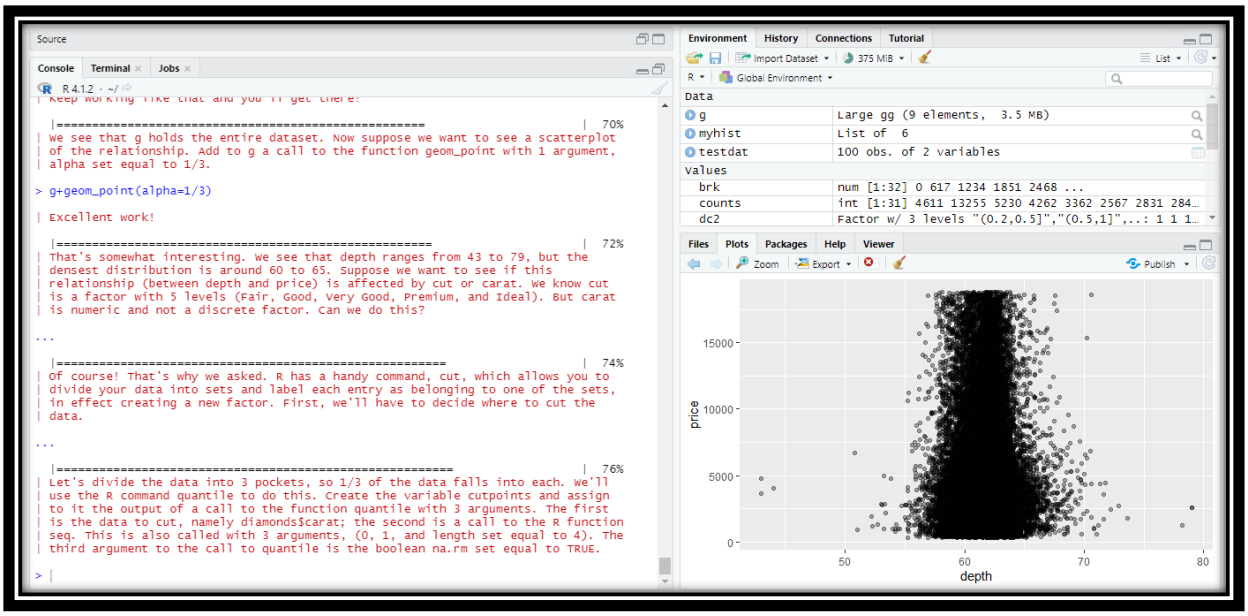

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| highest peak is close to 0 on the x-axis meaning that most of the diamonds in the
| dataset were inexpensive. In general, as prices increase (move right along the
| x-axis) the number of diamonds (at those prices) decrease. The exception to this
| is when the price is around $4000; there's a slight increase in frequency. Let's
| see if cut is responsible for this increase.
...
| 31%
| Rerun qplot, this time with 4 arguments. The first 2 are the usual, and the third
| is geom set equal to "density". The fourth is color set equal to cut. Try this
| now.
> qplot(price,data=diamonds,geom="density",color=cut)
| You nailed it! Good job!
| 33%
| See how easily qplot did this? Four of the five cuts have 2 peaks, one at price
| $1000 and the other between $4000 and $5000. The exception is the Fair cut which
| has a single peak at $2500. This gives us a little more understanding of the
| histogram we saw before.
...
| 35%
| Let's move on to scatterplots. For these we'll need to specify two variables from
| the diamond dataset.
...
| 37%
| Let's start with carat and price. Use these as the first 2 arguments of qplot.
| The third should be data set equal to the dataset. Try this now.
>
Environment
History
Connections
Tutorial
Import Dataset
358 MiB
List
Global Environment
myx int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
n 31L
path_to_course "C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Functions
fxfer function (fileName)
myedit function (fname)
pathtofile function (fileName)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
4e-04
3e-04
2e-04
1e-04
0e+00
0
5000
10000
15000
price
cut
Fair
Good
Very Good
Premium
Ideal

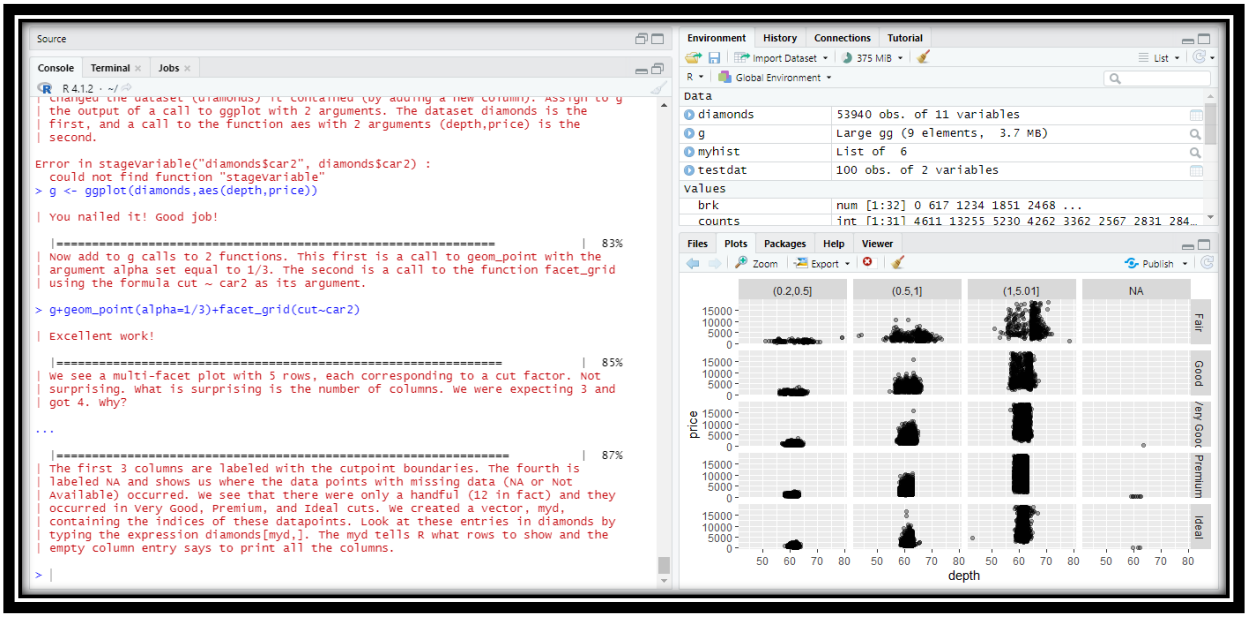
```
  |=======================                                  |  31%
| Rerun qplot, this time with 4 arguments. The first 2 are the usual, and the third
| is geom set equal to "density". The fourth is color set equal to cut. Try this
| now.

> qplot(price,data=diamonds,geom="density",color=cut)

| You nailed it! Good job!

  |========================                                 |  33%
| See how easily qplot did this? Four of the five cuts have 2 peaks, one at price
| $1000 and the other between $4000 and $5000. The exception is the Fair cut which
| has a single peak at $2500. This gives us a little more understanding of the
| histogram we saw before.

...

  |=========================                                |  35%
| Let's move on to scatterplots. For these we'll need to specify two variables from
| the diamond dataset.

...

  |=========================                                |  37%
| Let's start with carat and price. Use these as the first 2 arguments of qplot.
| The third should be data set equal to the dataset. Try this now.

> qplot(carat,price,data=diamonds)

| You got it!

  |==========================                               |  39%
| We see the positive trend here, as the number of carats increases the price also
| goes up.

...|
```

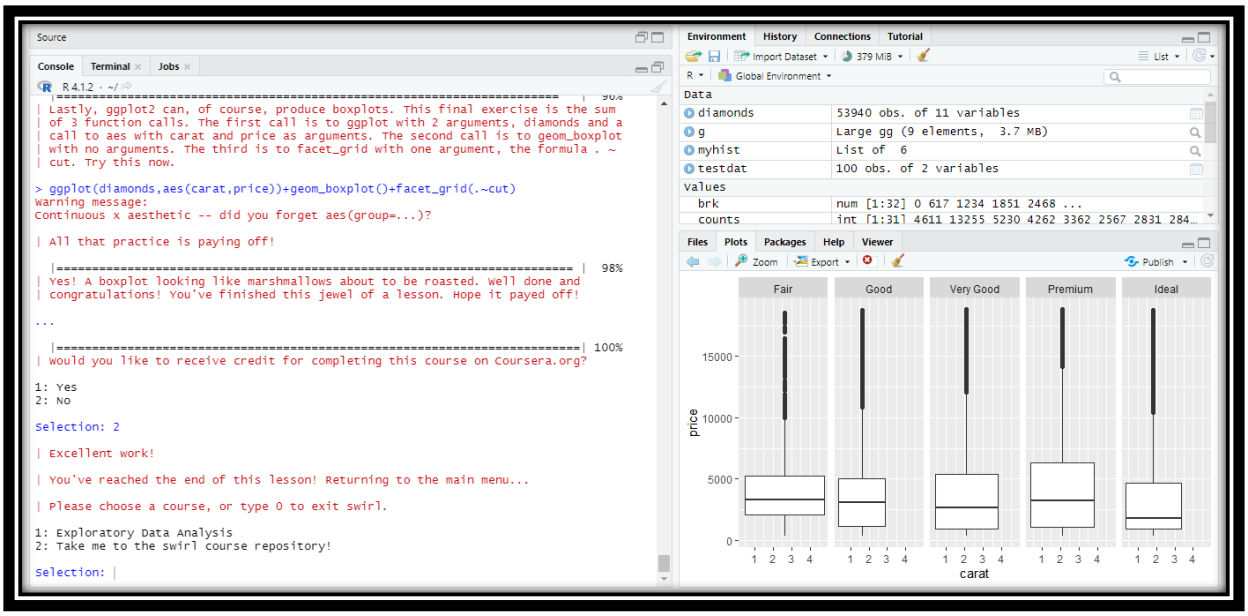
```
| goes up.

...

  |==============================                           |  41%
| Now rerun the same command, except add a fourth parameter, shape, set equal to
| cut.

> qplot(carat,price,data=diamonds, shape=cut)
warning message:
Using shapes for an ordinal variable is not advised

| That's the answer I was looking for.

  |===============================                          |  43%
| The same scatterplot appears, except the cuts of the diamonds are distinguished
| by different symbols. The legend at the right tells you which symbol is
| associated with each cut. These are small and hard to read, so rerun the same
| command, except this time instead of setting the argument shape equal to cut, set
| the argument color equal to cut.

> qplot(carat,price,data=diamonds, color=cut)

| Your dedication is inspiring!

  |================================                         |  44%
| That's easier to see! Now we'll close with two, more complicated scatterplot
| examples.

...

  |=================================                        |  46%
| We'll rerun the plot you just did (carat,price,data=diamonds and color=cut) but
| add an additional parameter. Use geom_smooth with the method set equal to the
| string "lm".

>
```


Source
Console
Terminal
Jobs
R 4.1.2 · ~/
|=================================== | 44%
| That's easier to see! Now we'll close with two, more complicated scatterplot
| examples.
...
|=================================== | 46%
| We'll rerun the plot you just did (carat,price,data=diamonds and color=cut) but
| add an additional parameter. Use geom_smooth with the method set equal to the
| string "lm".
> qplot(carat,price,data=diamonds, color=cut) + geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'
| Excellent work!
|==================================== | 48%
| Again, we see the same scatterplot, but slightly more compressed and showing 5
| regression lines, one for each cut of diamonds. It might be hard to see, but
| around each line is a shadow showing the 95% confidence interval. We see,
| unsurprisingly, that the better the cut, the steeper (more positive) the slope of
| the lines.
...
|==================================== | 50%
| Finally, let's rerun that plot you just did qplot(carat,price,data=diamonds,
| color=cut) + geom_smooth(method="lm") but add one (just one) more argument to
| qplot. The new argument is facets and it should be set equal to the formula
| .~cut. Recall that the facets argument indicates we want a multi-panel plot. The
| symbol to the left of the tilde indicates rows (in this case just one) and the
| symbol to the right of the tilde indicates columns (in this five, the number of
| cuts). Try this now.
>
Environment
History
Connections
Tutorial
Import Dataset
372 MiB
List
Global Environment
myx int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
n 31L
path_to_course "C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Functions
fxfer function (fileName)
myedit function (fname)
pathtofile function (fileName)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
price
carat
cut
Fair
Good
Very Good
Premium
Ideal

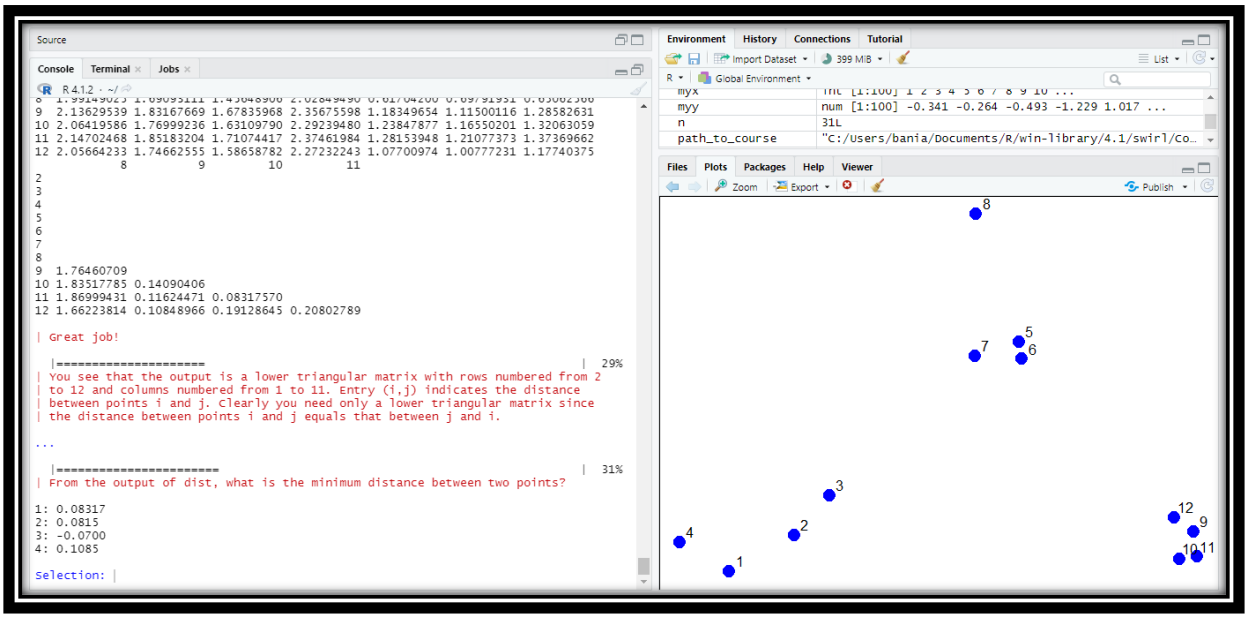

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| We'll rerun the plot you just did (carat,price,data=diamonds and color=cut) but
| add an additional parameter. Use geom_smooth with the method set equal to the
| string "lm".
> qplot(carat,price,data=diamonds, color=cut) + geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'
| Excellent work!
|==================================== | 48%
| Again, we see the same scatterplot, but slightly more compressed and showing 5
| regression lines, one for each cut of diamonds. It might be hard to see, but
| around each line is a shadow showing the 95% confidence interval. We see,
| unsurprisingly, that the better the cut, the steeper (more positive) the slope of
| the lines.
...
|==================================== | 50%
| Finally, let's rerun that plot you just did qplot(carat,price,data=diamonds,
| color=cut) + geom_smooth(method="lm") but add one (just one) more argument to
| qplot. The new argument is facets and it should be set equal to the formula
| .~cut. Recall that the facets argument indicates we want a multi-panel plot. The
| symbol to the left of the tilde indicates rows (in this case just one) and the
| symbol to the right of the tilde indicates columns (in this five, the number of
| cuts). Try this now.
> qplot(carat,price,data=diamonds, color=cut, facets=.~cut) + geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'
| Keep up the great work!
|==================================== | 52%
| Pretty good, right? Not too difficult either. Let's review what we learned!
...
Environment
History
Connections
Tutorial
Import Dataset
375 MiB
List
Global Environment
myx int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
myy num [1:100] -0.341 -0.264 -0.493 -1.229 1.017 ...
n 31L
path_to_course "C:/Users/bania/Documents/R/win-library/4.1/swirl/...
Functions
fxfer function (fileName)
myedit function (fname)
pathtofile function (fileName)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Fair
Good
Very Good
Premium
Ideal
price
carat
cut
Fair
Good
Very Good
Premium
Ideal

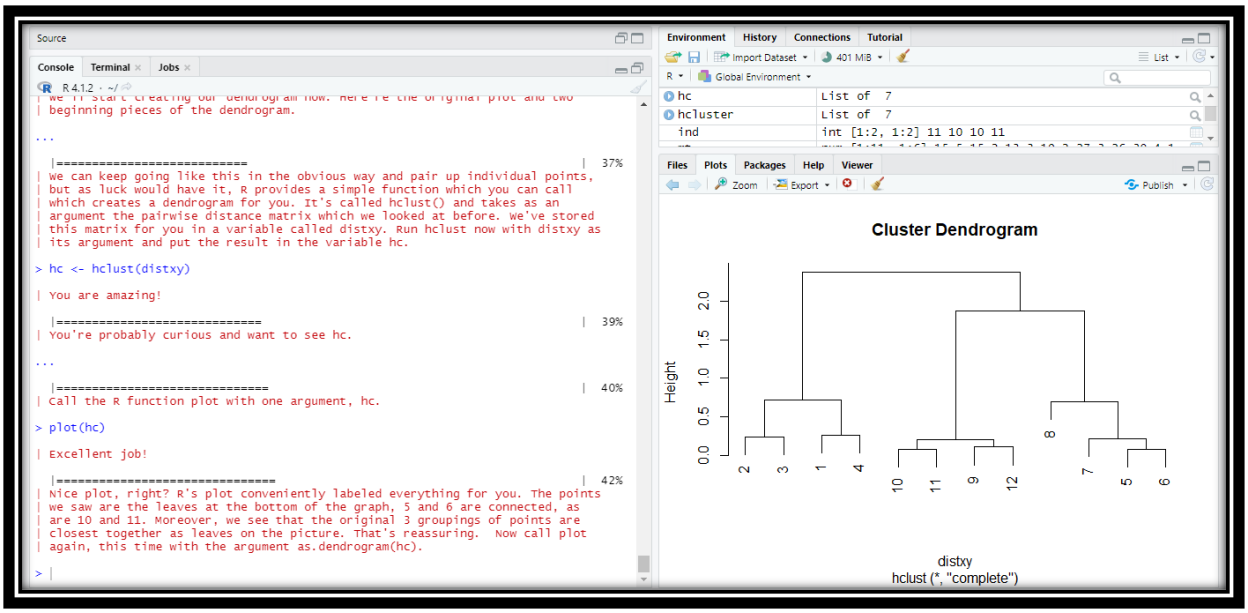

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| Keep working like that and you'll get there!
|== | 70%
| We see that g holds the entire dataset. Now suppose we want to see a scatterplot
| of the relationship. Add to g a call to the function geom_point with 1 argument,
| alpha set equal to 1/3.
> g+geom_point(alpha=1/3)
| Excellent work!
|== | 72%
| That's somewhat interesting. We see that depth ranges from 43 to 79, but the
| densest distribution is around 60 to 65. Suppose we want to see if this
| relationship (between depth and price) is affected by cut or carat. We know cut
| is a factor with 5 levels (Fair, Good, Very Good, Premium, and Ideal). But carat
| is numeric and not a discrete factor. Can we do this?
...
|== | 74%
| Of course! That's why we asked. R has a handy command, cut, which allows you to
| divide your data into sets and label each entry as belonging to one of the sets,
| in effect creating a new factor. First, we'll have to decide where to cut the
| data.
...
|== | 76%
| Let's divide the data into 3 pockets, so 1/3 of the data falls into each. We'll
| use the R command quantile to do this. Create the variable cutpoints and assign
| to it the output of a call to the function quantile with 3 arguments. The first
| is the data to cut, namely diamonds$carat; the second is a call to the R function
| seq. This is also called with 3 arguments, (0, 1, and length set equal to 4). The
| third argument to the call to quantile is the boolean na.rm set equal to TRUE.
>
Environment
History
Connections
Tutorial
Import Dataset
375 MiB
List
R
Global Environment
Data
g
Large gg (9 elements, 3.5 MB)
myhist
List of 6
testdat
100 obs. of 2 variables
Values
brk
num [1:32] 0 617 1234 1851 2468 ...
counts
int [1:31] 4611 13255 5230 4262 3362 2567 2831 284...
dc2
Factor w/ 3 levels "(0.2,0.5]","(0.5,1]",..: 1 1 1...
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
price
15000
10000
5000
0
50
60
70
80
depth

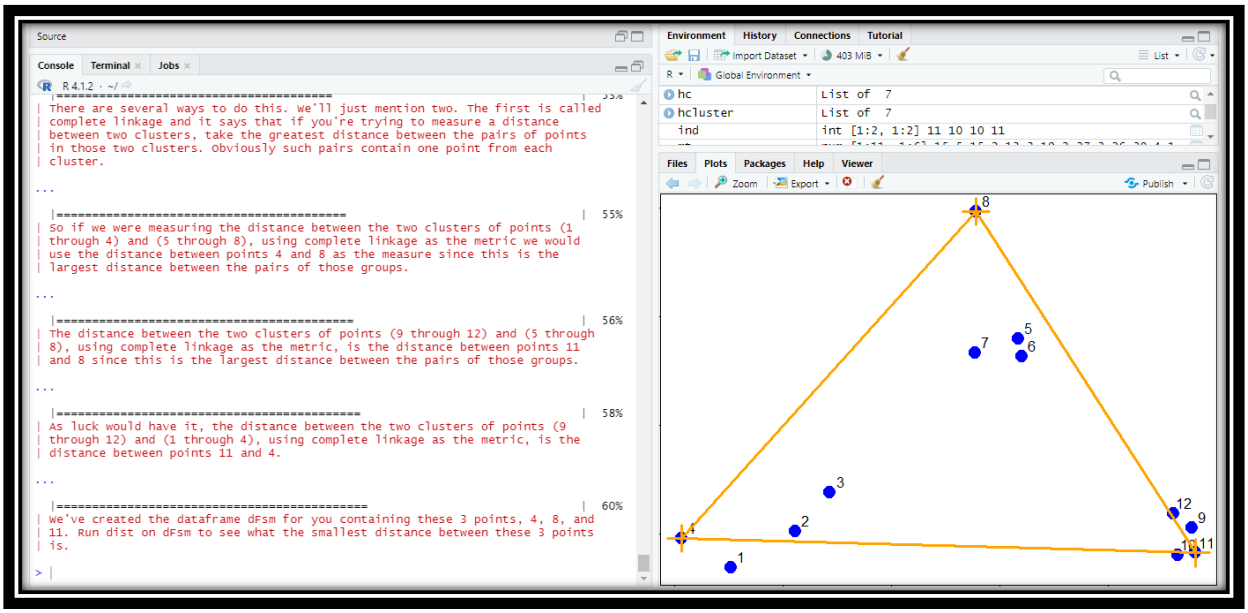

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| Changed the dataset (diamonds) it contained (by adding a new column). Assign to g
| the output of a call to ggplot with 2 arguments. The dataset diamonds is the
| first, and a call to the function aes with 2 arguments (depth,price) is the
| second.
Error in stageVariable("diamonds$car2", diamonds$car2) :
could not find function "stageVariable"
> g <- ggplot(diamonds,aes(depth,price))
| You nailed it! Good job!
|== | 83%
| Now add to g calls to 2 functions. This first is a call to geom_point with the
| argument alpha set equal to 1/3. The second is a call to the function facet_grid
| using the formula cut ~ car2 as its argument.
> g+geom_point(alpha=1/3)+facet_grid(cut~car2)
| Excellent work!
|== | 85%
| We see a multi-facet plot with 5 rows, each corresponding to a cut factor. Not
| surprising. What is surprising is the number of columns. We were expecting 3 and
| got 4. Why?
...
|== | 87%
| The first 3 columns are labeled with the cutpoint boundaries. The fourth is
| labeled NA and shows us where the data points with missing data (NA or Not
| Available) occurred. We see that there were only a handful (12 in fact) and they
| occurred in Very Good, Premium, and Ideal cuts. We created a vector, myd,
| containing the indices of these datapoints. Look at these entries in diamonds by
| typing the expression diamonds[myd,]. The myd tells R what rows to show and the
| empty column entry says to print all the columns.
>
Environment
History
Connections
Tutorial
Import Dataset
375 MiB
List
R
Global Environment
Data
diamonds
53940 obs. of 11 variables
g
Large gg (9 elements, 3.7 MB)
myhist
List of 6
testdat
100 obs. of 2 variables
Values
brk
num [1:32] 0 617 1234 1851 2468 ...
counts
int [1:31] 4611 13255 5230 4262 3362 2567 2831 284...
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
(0.2,0.5]
(0.5,1]
(1,5.01]
NA
Fair
Good
Very Good
Premium
Ideal
price
15000
10000
5000
0
50
60
70
80
depth

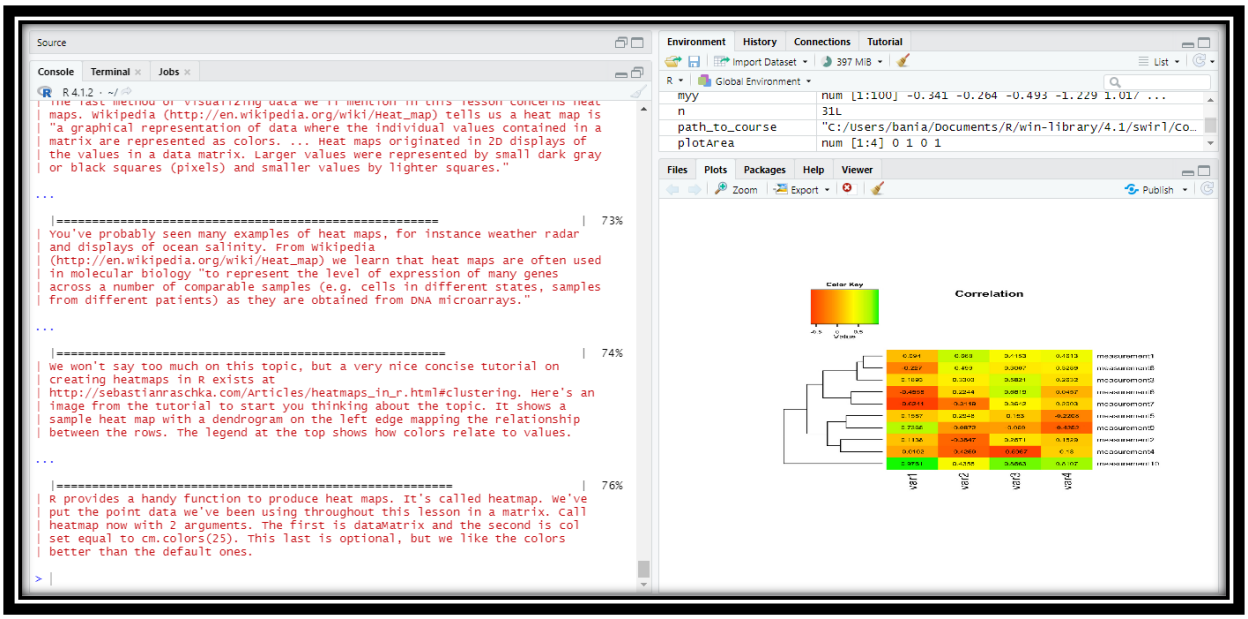


11. Hierarchical Clustering

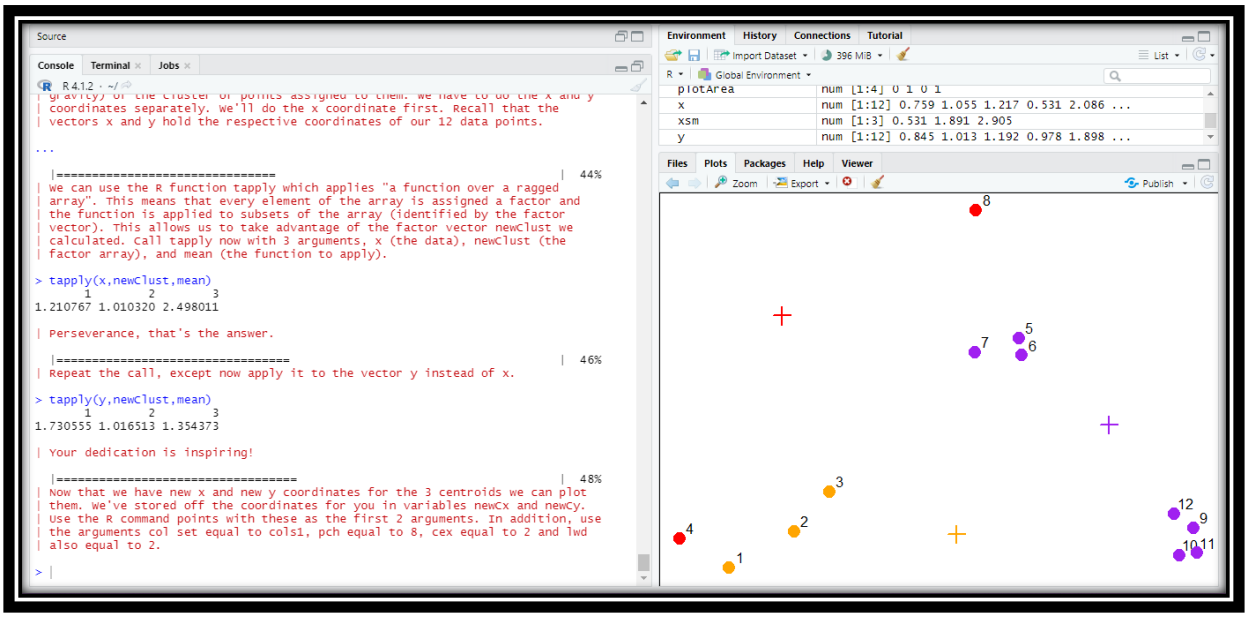


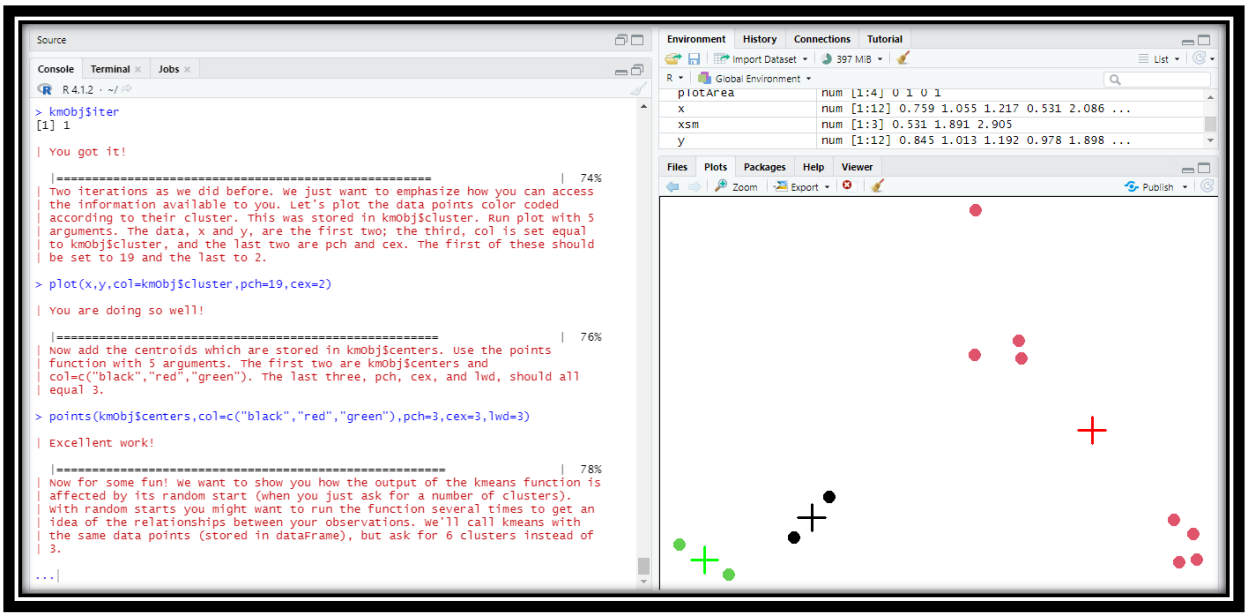

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| We'll start creating our dendrogram now. Here're the original plot and two
| beginning pieces of the dendrogram.
...
|========== | 37%
| We can keep going like this in the obvious way and pair up individual points,
| but as luck would have it, R provides a simple function which you can call
| which creates a dendrogram for you. It's called hclust() and takes as an
| argument the pairwise distance matrix which we looked at before. We've stored
| this matrix for you in a variable called distxy. Run hclust now with distxy as
| its argument and put the result in the variable hc.
> hc <- hclust(distxy)
| You are amazing!
|========== | 39%
| You're probably curious and want to see hc.
...
|========== | 40%
| Call the R function plot with one argument, hc.
> plot(hc)
| Excellent job!
|========== | 42%
| Nice plot, right? R's plot conveniently labeled everything for you. The points
| we saw are the leaves at the bottom of the graph, 5 and 6 are connected, as
| are 10 and 11. Moreover, we see that the original 3 groupings of points are
| closest together as leaves on the picture. That's reassuring. Now call plot
| again, this time with the argument as.dendrogram(hc).
>
Environment
History
Connections
Tutorial
Import Dataset
401 MiB
List
R
Global Environment
hc
List of 7
hcluster
List of 7
ind
int [1:2, 1:2] 11 10 10 11
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Cluster Dendrogram
Height
0.0
0.5
1.0
1.5
2.0
2
3
1
4
10
11
9
12
8
7
5
6
distxy
hclust (*, "complete")

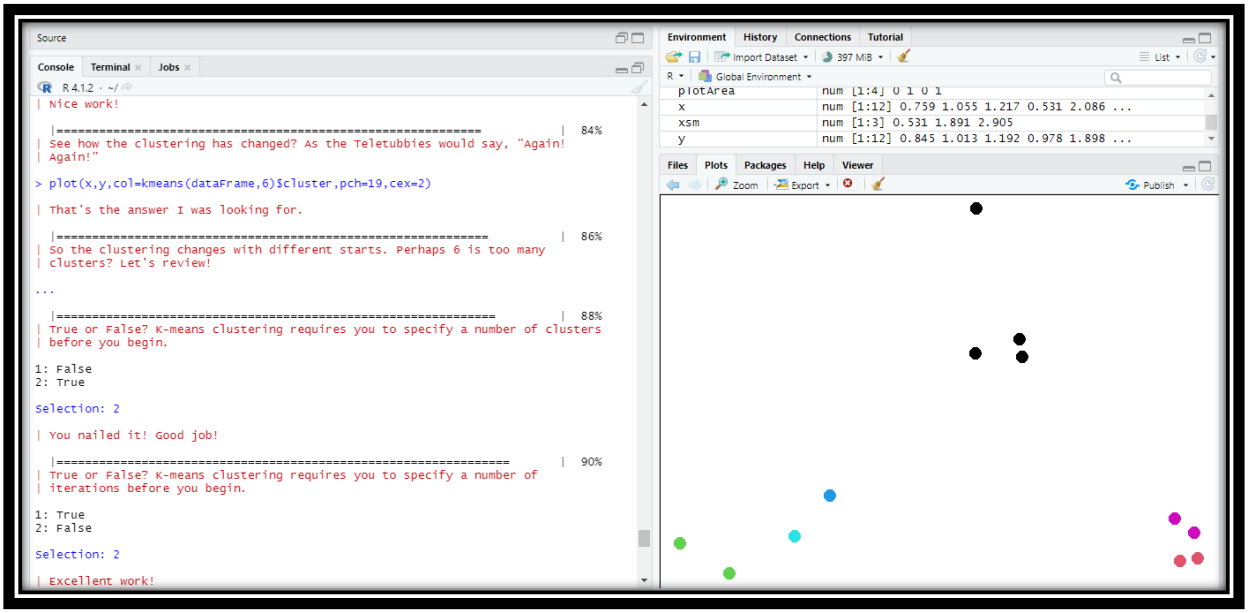

Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| There are several ways to do this. We'll just mention two. The first is called
| complete linkage and it says that if you're trying to measure a distance
| between two clusters, take the greatest distance between the pairs of points
| in those two clusters. Obviously such pairs contain one point from each
| cluster.
...
|========== | 55%
| So if we were measuring the distance between the two clusters of points (1
| through 4) and (5 through 8), using complete linkage as the metric we would
| use the distance between points 4 and 8 as the measure since this is the
| largest distance between the pairs of those groups.
...
|========== | 56%
| The distance between the two clusters of points (9 through 12) and (5 through
| 8), using complete linkage as the metric, is the distance between points 11
| and 8 since this is the largest distance between the pairs of those groups.
...
|========== | 58%
| As luck would have it, the distance between the two clusters of points (9
| through 12) and (1 through 4), using complete linkage as the metric, is the
| distance between points 11 and 4.
...
|========== | 60%
| We've created the dataframe dFsm for you containing these 3 points, 4, 8, and
| 11. Run dist on dFsm to see what the smallest distance between these 3 points
| is.
>
Environment
History
Connections
Tutorial
Import Dataset
403 MiB
List
R
Global Environment
hc
List of 7
hcluster
List of 7
ind
int [1:2, 1:2] 11 10 10 11
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
8
5
6
7
3
2
4
1
12
9
11
10



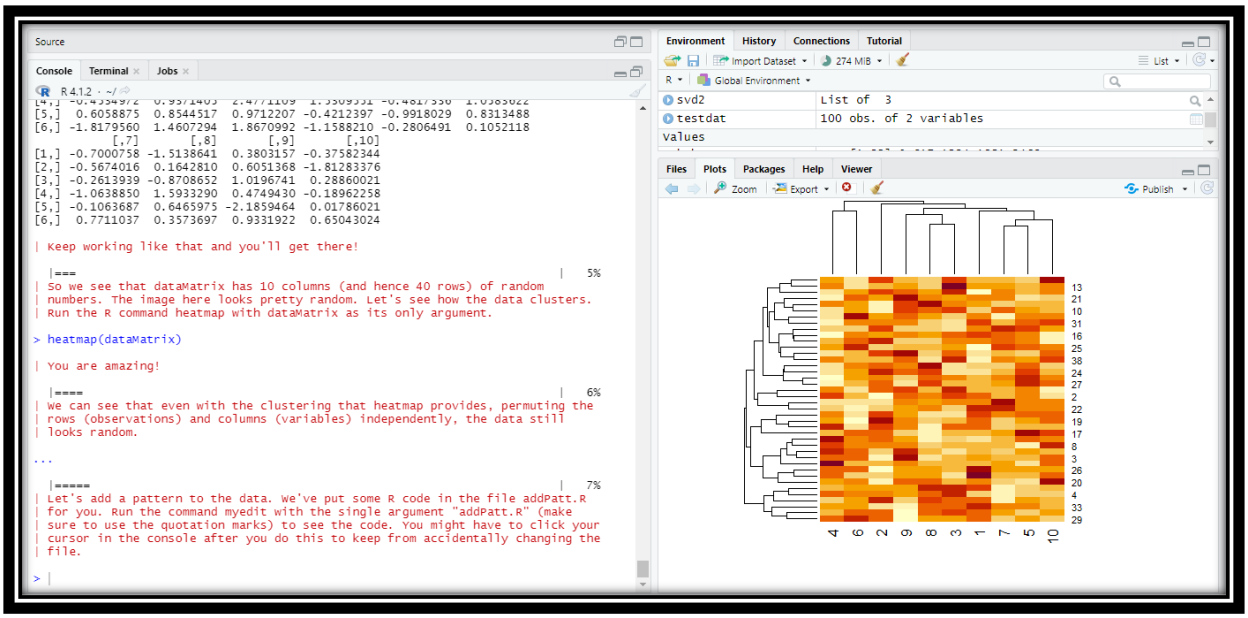
12. K Means Clustering

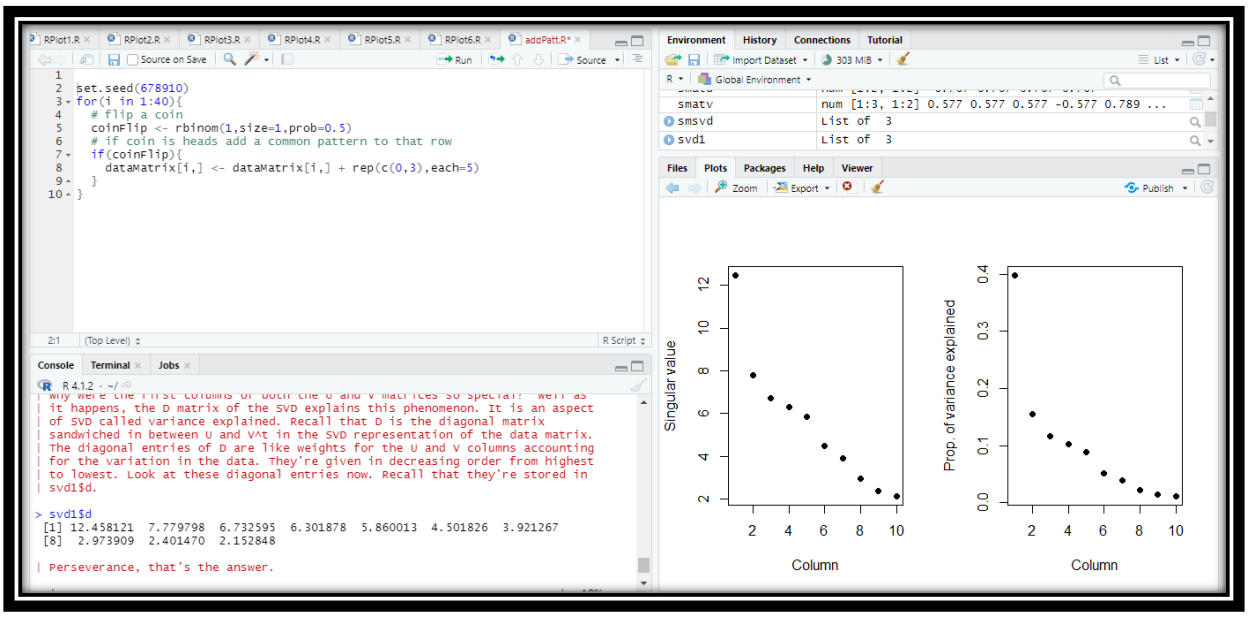






13. Dimension Reduction

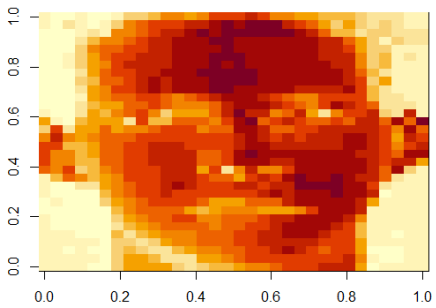

[5,] 0.6058875 0.8544517 0.9712207 -0.4212397 -0.9918029 0.8313488
[6,] -1.8179560 1.4607294 1.8670992 -1.1588210 -0.2806491 0.1052118
[,7] [,8] [,9] [,10]
[1,] -0.7000758 -1.5138641 0.3803157 -0.37582344
[2,] -0.5674016 0.1642810 0.6051368 -1.81283376
[3,] -0.2613939 -0.8708652 1.0196741 0.28860021
[4,] -1.0638850 1.5933290 0.4749430 -0.18962258
[5,] -0.1063687 0.6465975 -2.1859464 0.01786021
[6,] 0.7711037 0.3573697 0.9331922 0.65043024
| Keep working like that and you'll get there!
| 5%
| So we see that dataMatrix has 10 columns (and hence 40 rows) of random
| numbers. The image here looks pretty random. Let's see how the data clusters.
| Run the R command heatmap with dataMatrix as its only argument.
> heatmap(dataMatrix)
| You are amazing!
| 6%
| We can see that even with the clustering that heatmap provides, permuting the
| rows (observations) and columns (variables) independently, the data still
| looks random.
| 7%
| Let's add a pattern to the data. We've put some R code in the file addPatt.R
| for you. Run the command myedit with the single argument "addPatt.R" (make
| sure to use the quotation marks) to see the code. You might have to click your
| cursor in the console after you do this to keep from accidentally changing the
| file.
svd2 List of 3
testdat 100 obs. of 2 variables
values
13 21 10 31 16 25 38 24 27 2 22 19 17 8 3 26 20 4 33 29
4 6 2 9 8 3 1 7 5 10


set.seed(678910)
for(i in 1:40){
# flip a coin
coinFlip <- rbinom(1,size=1,prob=0.5)
# if coin is heads add a common pattern to that row
if(coinFlip){
dataMatrix[i,] <- dataMatrix[i,] + rep(c(0,3),each=5)
}
}
smatv num [1:3, 1:2] 0.577 0.577 0.577 -0.577 0.789 ...
smsvd List of 3
svd1 List of 3
| it happens, the D matrix of the SVD explains this phenomenon. It is an aspect
| of SVD called variance explained. Recall that D is the diagonal matrix
| sandwiched in between U and VAt in the SVD representation of the data matrix.
| The diagonal entries of D are like weights for the U and V columns accounting
| for the variation in the data. They're given in decreasing order from highest
| to lowest. Look at these diagonal entries now. Recall that they're stored in
| svd1$d.
> svd1$d
[1] 12.458121 7.779798 6.732595 6.301878 5.860013 4.501826 3.921267
[8] 2.973909 2.401470 2.152848
| Perseverance, that's the answer.
Singular value
Column
Prop. of variance explained
Column


set.seed(678910)
for(i in 1:40){
# flip a coin
coinFlip <- rbinom(1,size=1,prob=0.5)
# if coin is heads add a common pattern to that row
if(coinFlip){
dataMatrix[i,] <- dataMatrix[i,] + rep(c(0,3),each=5)
}
}
svd2 List of 3
testdat 100 obs. of 2 variables
values
| than the first 5. This picks up, or at least alludes to, the first pattern we
| added in which affected the last 5 columns of the matrix. The rightmost plot,
| showing the second column of v, looks more random. However, closer inspection
| shows that the entries alternate or bounce up and down as you move from left
| to right. This hints at the second pattern we added in which affected only
| even columns of selected rows.
| 61%
| To see this more closely, look at the first 2 columns of the v component. We
| stored the SVD output in the svd object svd2.
v and variation in columns
First right singular vector
Index
Second right singular vector
Index

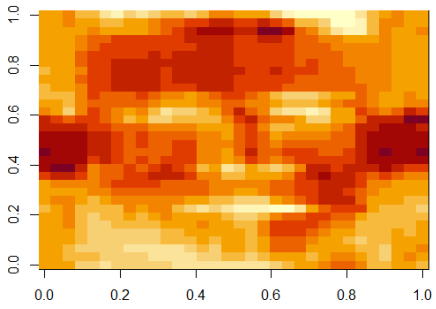
```
set.seed(678910)
for(i in 1:40){
  # flip a coin
  coinFlip <- rbinom(1,size=1,prob=0.5)
  # if coin is heads add a common pattern to that row
  if(coinFlip){
    dataMatrix[i,] <- dataMatrix[i,] + rep(c(0,3),each=5)
  }
}
```

```
...

  |=======================================================              |  72%
| Consider this low resolution image file showing a face. We'll use SVD and see
| how the first several components contain most of the information in the file
| so that storing a huge matrix might not be necessary.

...

  |=========================================================            |  73%
| The image data is stored in the matrix faceData. Run the R command dim on
| faceData to see how big it is.

>
```

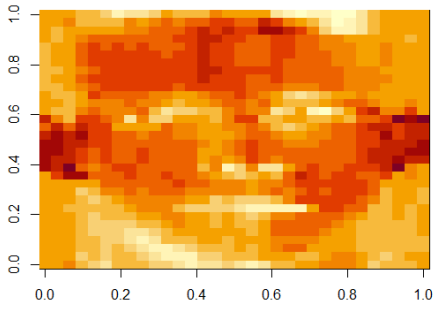



```
set.seed(678910)
for(i in 1:40){
  # flip a coin
  coinFlip <- rbinom(1,size=1,prob=0.5)
  # if coin is heads add a common pattern to that row
  if(coinFlip){
    dataMatrix[i,] <- dataMatrix[i,] + rep(c(0,3),each=5)
  }
}
```

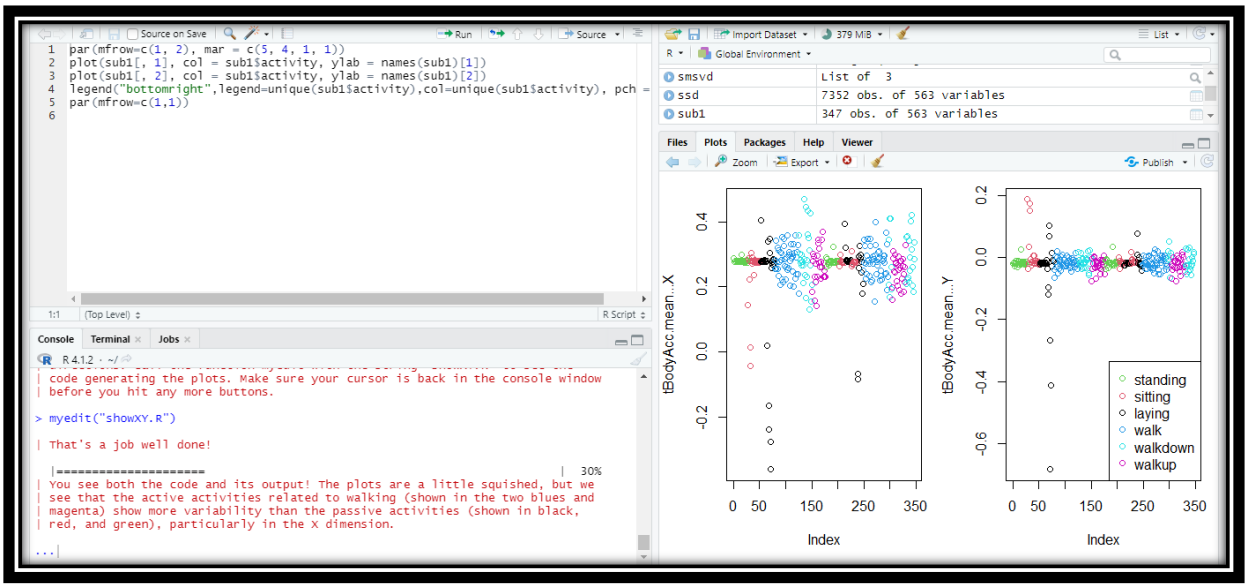
```
> a2 <- svd1$u[,1:2] %*%  diag(svd1$d[1:2])  %*% t(svd1$v[,1:2])

| Great job!

  |=============================================================        |  87%
| Use myImage again to see how a2 displays.

> myImage(svd1$u[,1:5] %*%  diag(svd1$d[1:5])  %*% t(svd1$v[,1:5]))

| One more time. You can do it! Or, type info() for more options.

| Type myImage(a2) at the command prompt.

> myImage(svd1$u[,1:10] %*%  diag(svd1$d[1:10])  %*% t(svd1$v[,1:10]))
```

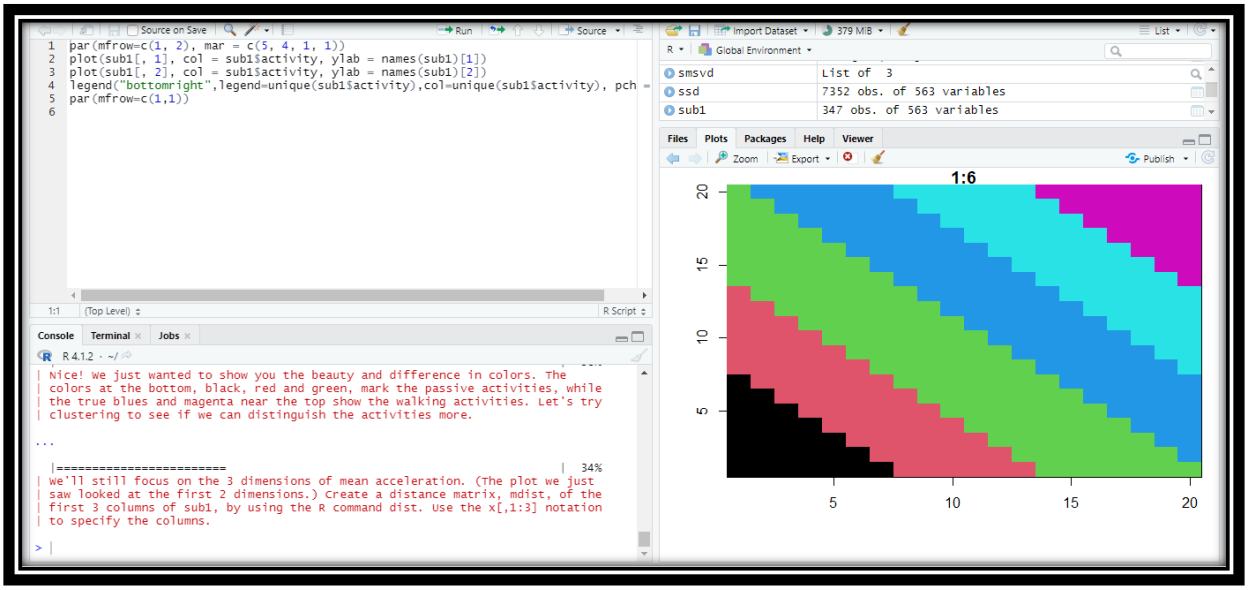


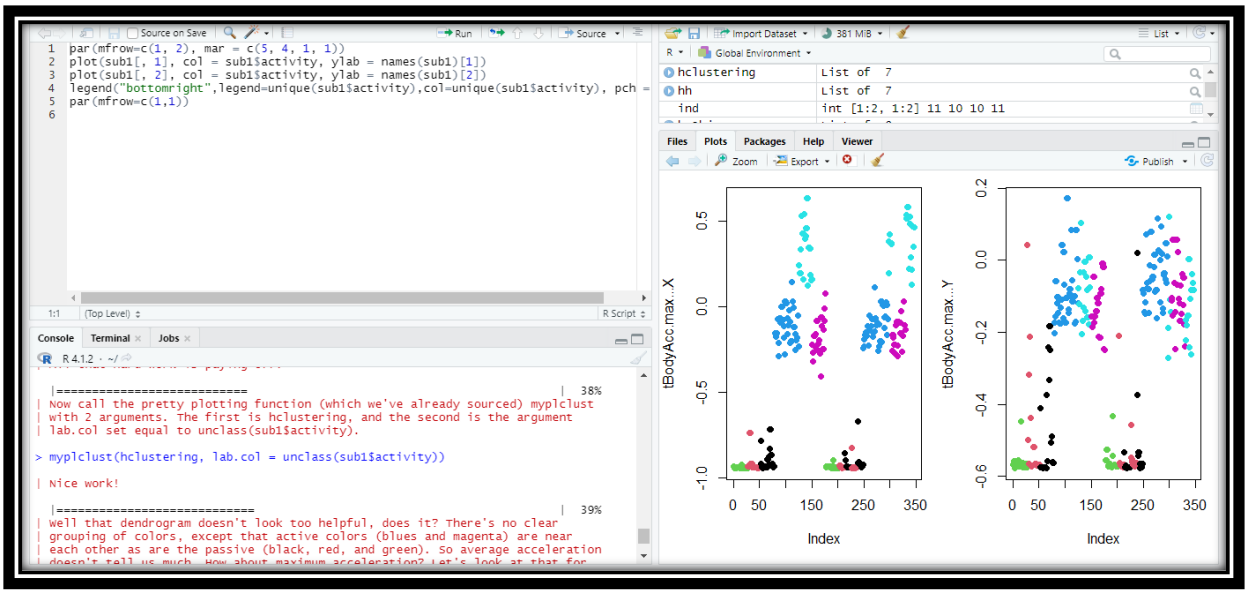

```
...

  |=============================================================        |  92%
| We'll close now with a few comments. First, when reducing dimensions you have
| to pay attention to the scales on which different variables are measured and
| make sure that all your data is in consistent units. In other words, scales of
| your data matter. Second, principal components and singular values may mix
| real patterns, as we saw in our simple 2-pattern example, so finding and
| separating out the real patterns require some detective work. Let's do a quick
| review now.

...

  |=============================================================        |  93%
| Which of the following cliches LEAST captures the essence of dimension
| reduction?

1: see the forest through the trees
2: a face that could launch a 1000 ships
3: find the needle in the haystack
4: separate the wheat from the chaff

Selection: 2

| You nailed it! Good job!

  |=============================================================        |  94%
| A matrix X has the singular value decomposition UDV^t. The principal
| components of X are ?

1: the rows of U
2: the rows of V
3: the columns of V
4: the columns of U

Selection:
```

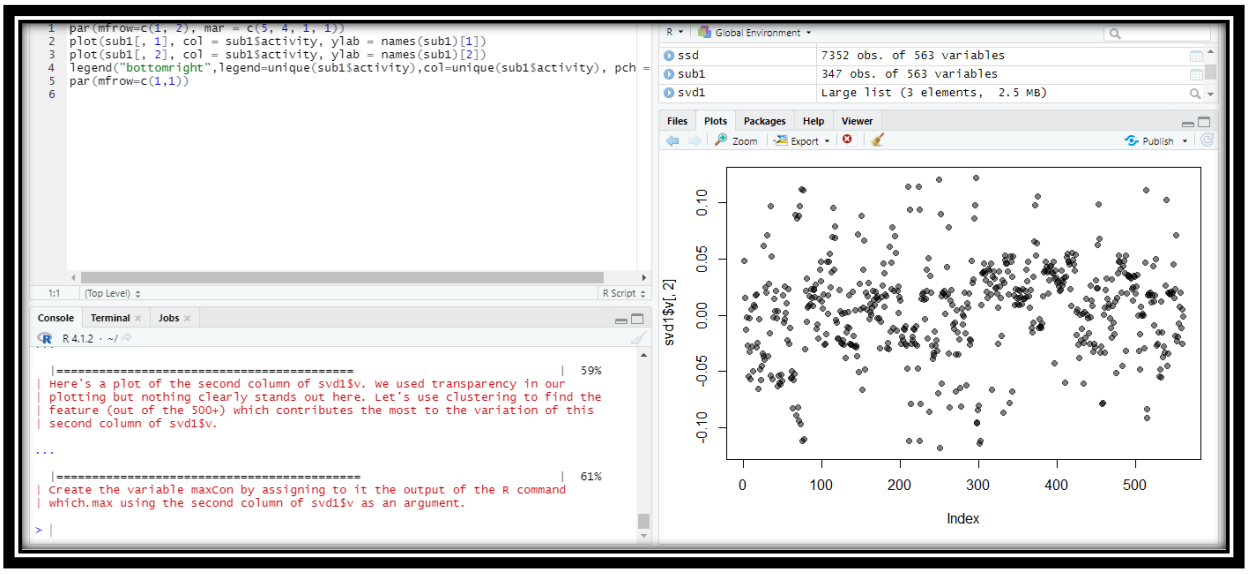


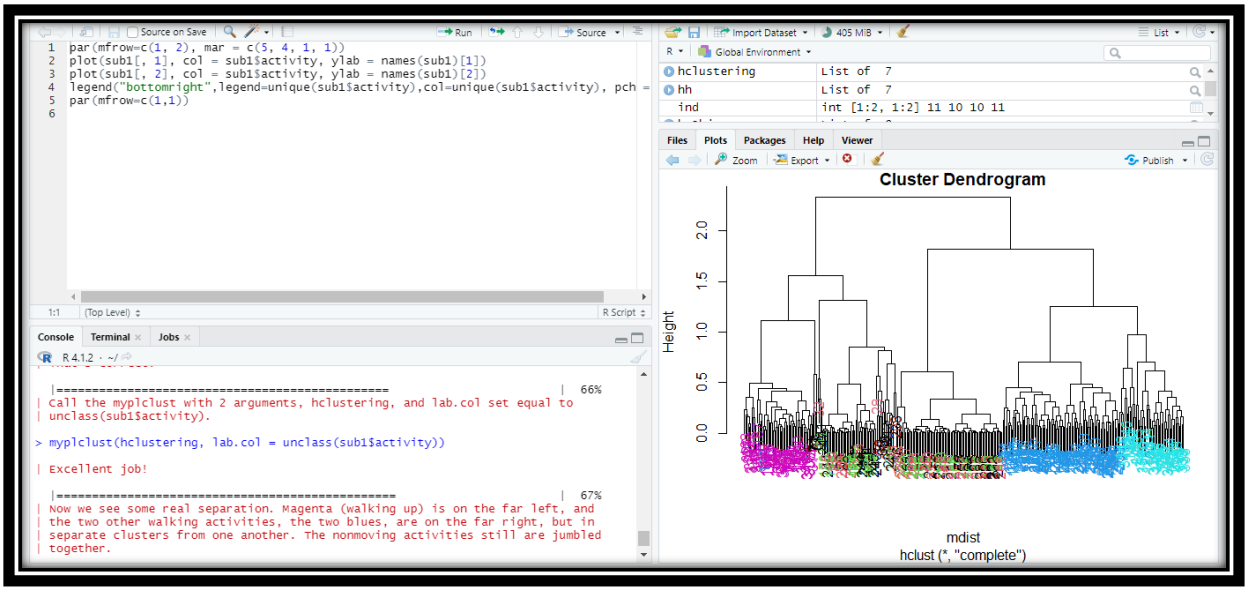
14. Clustering Example

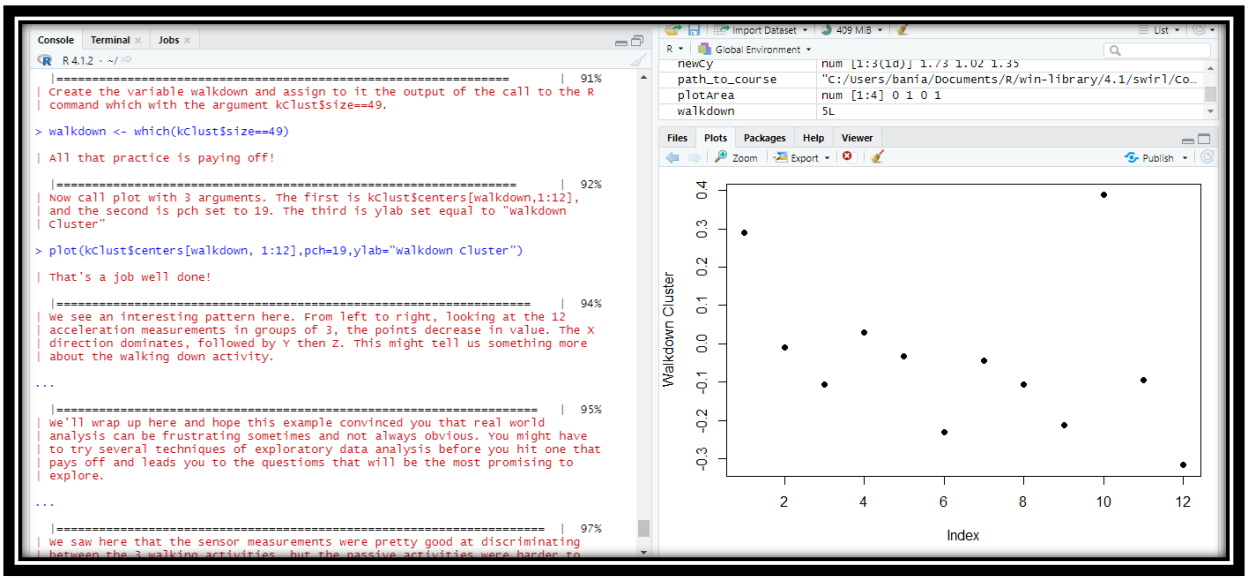




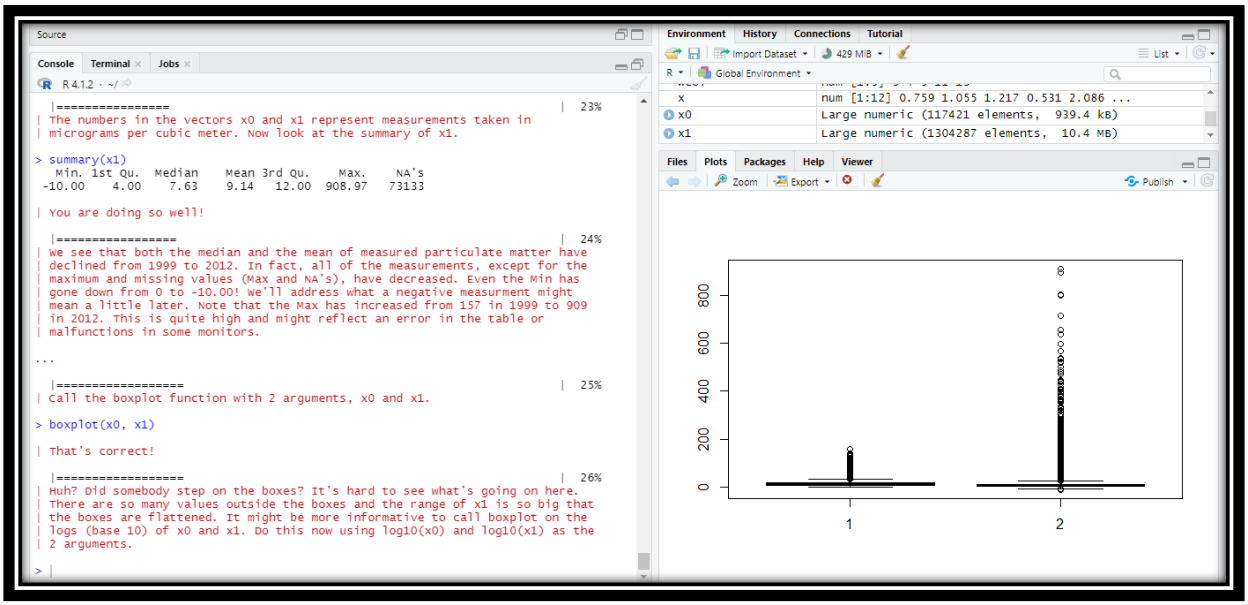

par(mfrow=c(1, 2), mar = c(5, 4, 1, 1))
plot(sub1[, 1], col = sub1$activity, ylab = names(sub1)[1])
plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])
legend("bottomright",legend=unique(sub1$activity),col=unique(sub1$activity), pch =
par(mfrow=c(1,1))
Console
Terminal
Jobs
R 4.1.2 · ~/
| 38%
| Now call the pretty plotting function (which we've already sourced) myplclust
| with 2 arguments. The first is hclustering, and the second is the argument
| lab.col set equal to unclass(sub1$activity).
> myplclust(hclustering, lab.col = unclass(sub1$activity))
| Nice work!
| 39%
| Well that dendrogram doesn't look too helpful, does it? There's no clear
| grouping of colors, except that active colors (blues and magenta) are near
| each other as are the passive (black, red, and green). So average acceleration
hclustering List of 7
hh List of 7
ind int [1:2, 1:2] 11 10 10 11
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
tBodyAcc.max...X
tBodyAcc.max...Y
Index
Index


par(mfrow=c(1, 2), mar = c(5, 4, 1, 1))
plot(sub1[, 1], col = sub1$activity, ylab = names(sub1)[1])
plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])
legend("bottomright",legend=unique(sub1$activity),col=unique(sub1$activity), pch =
par(mfrow=c(1,1))
Console
Terminal
Jobs
R 4.1.2 · ~/
| 59%
| Here's a plot of the second column of svd1$v. We used transparency in our
| plotting but nothing clearly stands out here. Let's use clustering to find the
| feature (out of the 500+) which contributes the most to the variation of this
| second column of svd1$v.
...
| 61%
| Create the variable maxCon by assigning to it the output of the R command
| which.max using the second column of svd1$v as an argument.
>
ssd 7352 obs. of 563 variables
sub1 347 obs. of 563 variables
svd1 Large list (3 elements, 2.5 MB)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
svd1$v[, 2]
Index

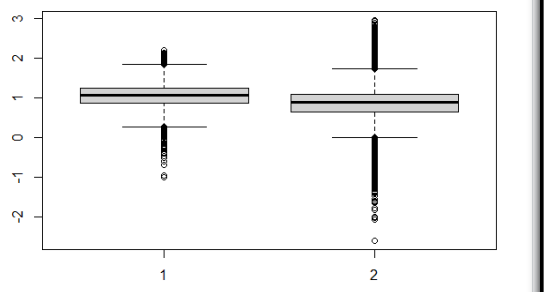

par(mfrow=c(1, 2), mar = c(5, 4, 1, 1))
plot(sub1[, 1], col = sub1$activity, ylab = names(sub1)[1])
plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])
legend("bottomright",legend=unique(sub1$activity),col=unique(sub1$activity), pch =
par(mfrow=c(1,1))
Console
Terminal
Jobs
R 4.1.2 · ~/
| 66%
| Call the myplclust with 2 arguments, hclustering, and lab.col set equal to
| unclass(sub1$activity).
> myplclust(hclustering, lab.col = unclass(sub1$activity))
| Excellent job!
| 67%
| Now we see some real separation. Magenta (walking up) is on the far left, and
| the two other walking activities, the two blues, are on the far right, but in
| separate clusters from one another. The nonmoving activities still are jumbled
| together.
hclustering List of 7
hh List of 7
ind int [1:2, 1:2] 11 10 10 11
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Cluster Dendrogram
Height
mdist
hclust (*, "complete")



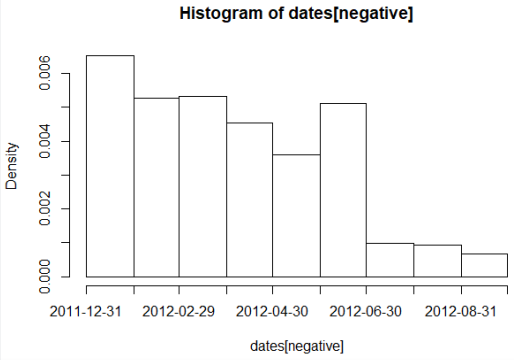
15. CaseStudy



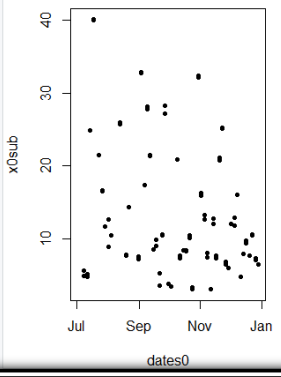
```
| Huh? Did somebody step on the boxes? It's hard to see what's going on here.
| There are so many values outside the boxes and the range of x1 is so big that
| the boxes are flattened. It might be more informative to call boxplot on the
| logs (base 10) of x0 and x1. Do this now using log10(x0) and log10(x1) as the
| 2 arguments.

> boxplot(log10(x0), log10(x1))
Warning messages:
1: In boxplot.default(log10(x0), log10(x1)) : NaNs produced
2: In bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group ==  :
  Outlier (-Inf) in boxplot 1 is not drawn
3: In bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group ==  :
  Outlier (-Inf) in boxplot 2 is not drawn

| You are really on a roll!

  |==================                                                    |  27%
| A bonus! Not only do we get a better looking boxplot we also get some warnings
| from R in Red. These let us know that some values in x0 and x1 were
| "unloggable", no doubt the 0 (Min) we saw in the summary of x0 and the
| negative values we saw in the Min of the summary of x1.

...

  |====================                                                  |  28%
| From the boxplot (x0 on the left and x1 on the right), what can you say about
| the data?

1: The median of x1 is less than the median of x0
2: The range of x0 is greater than the range of x1
3: The mean of x1 is less than the mean of x0
4: The boxes are too small to interpret

Selection: 1

| All that practice is paying off!
```

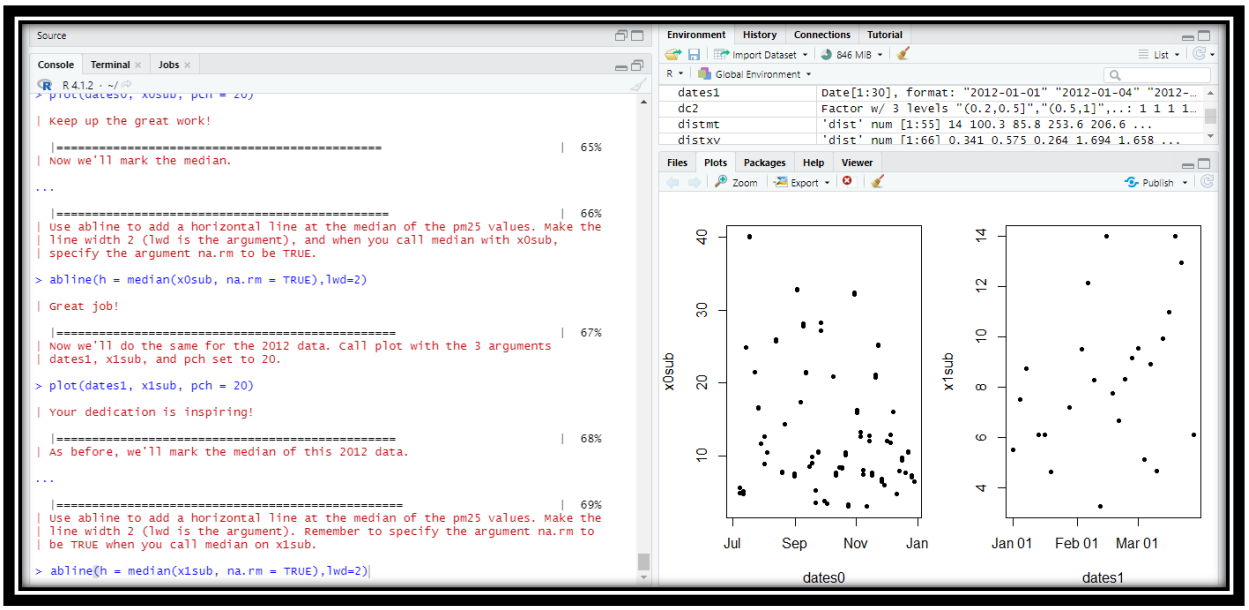


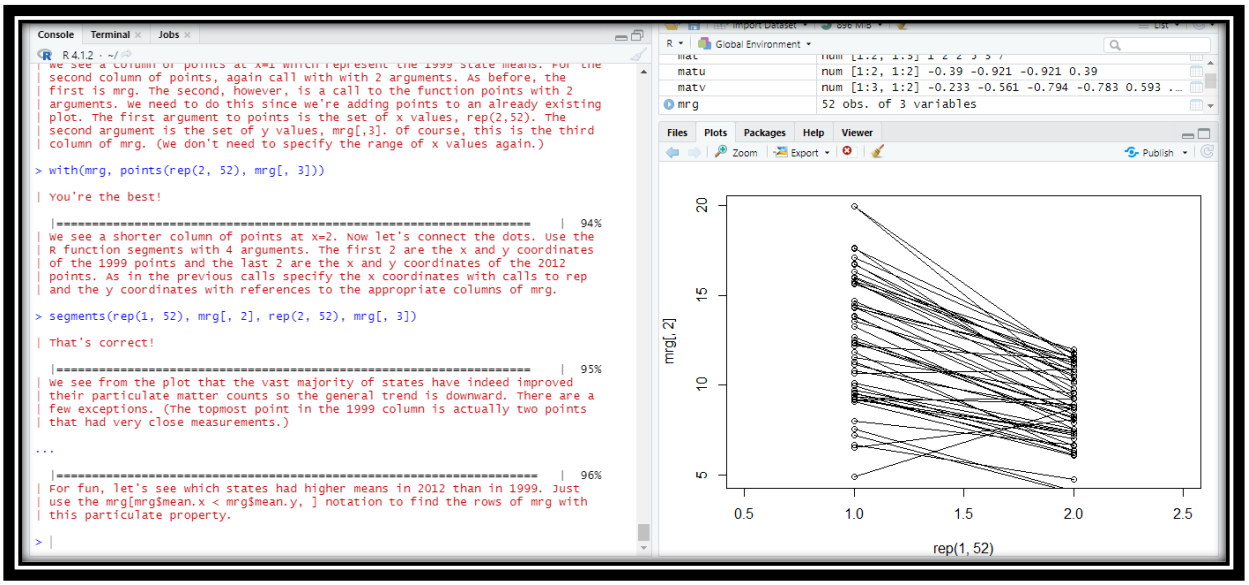

```
| as.character(dates) as the first argument and the string "%Y%m%d" as the
| second.

> dates <- as.Date(as.character(dates), "%Y%m%d")

| That's correct!

  |============================                                          |  37%
| Now when you run head on dates you'll see the dates in a nicer format. Try
| this now.

> head(dates)
[1] "2012-01-01" "2012-01-04" "2012-01-07" "2012-01-10" "2012-01-13" "2012-01-16"

| Nice work!

  |============================                                          |  38%
| Let's plot a histogram of the months when the particulate matter measurements
| are negative. Run hist with 2 arguments. The first is dates[negative] and the
| second is the string "month".

> hist(dates[negative],"month")

| All that practice is paying off!

  |============================                                          |  39%
| We see the bulk of the negative measurements were taken in the winter months,
| with a spike in May. Not many of these negative measurements occurred in
| summer months. We can take a guess that because particulate measures tend to
| be low in winter and high in summer, coupled with the fact that higher
| densities are easier to measure, that measurement errors occurred when the
| values were low. For now we'll attribute these negative measurements to
| errors. Also, since they account for only 2% of the 2012 data, we'll ignore
| them.

...
```




```
| as your input.

> dates1 <- as.Date(as.character(pm1sub$Date),"%Y%m%d")

| You are doing so well!

  |================================================                      |  63%
| Now we'll plot these 2 time series in the same panel using the base plotting
| system. Call par with 2 arguments. The first is mfrow set equal to c(1,2).
| This will tell the system we're plotting 2 graphs in 1 row and 2 columns. The
| second argument will adjust the panel's margins. It is mar set to c(4,4,2,1).

> par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

| Great job!

  |================================================                      |  64%
| Call plot with the 3 arguments dates0, x0sub, and pch set to 20. The first two
| arguments are the x and y coordinates. This will show the pm25 values as
| functions of time.

> plot(dates0, x0sub, pch = 20)

| Keep up the great work!

  |================================================                      |  65%
| Now we'll mark the median.

...

  |================================================                      |  66%
| Use abline to add a horizontal line at the median of the pm25 values. Make the
| line width 2 (lwd is the argument), and when you call median with x0sub,
| specify the argument na.rm to be TRUE.

>
```




Source
Console
Terminal
Jobs
R 4.1.2 · ~/
| Keep up the great work!
| 65%
| Now we'll mark the median.
...
| 66%
| Use abline to add a horizontal line at the median of the pm25 values. Make the
| line width 2 (lwd is the argument), and when you call median with x0sub,
| specify the argument na.rm to be TRUE.
> abline(h = median(x0sub, na.rm = TRUE),lwd=2)
| Great job!
| 67%
| Now we'll do the same for the 2012 data. Call plot with the 3 arguments
| dates1, x1sub, and pch set to 20.
> plot(dates1, x1sub, pch = 20)
| Your dedication is inspiring!
| 68%
| As before, we'll mark the median of this 2012 data.
...
| 69%
| Use abline to add a horizontal line at the median of the pm25 values. Make the
| line width 2 (lwd is the argument). Remember to specify the argument na.rm to
| be TRUE when you call median on x1sub.
> abline(h = median(x1sub, na.rm = TRUE),lwd=2)
Environment
History
Connections
Tutorial
Import Dataset
846 MiB
Global Environment
dates1
Date[1:30], format: "2012-01-01" "2012-01-04" "2012-...
dc2
Factor w/ 3 levels "(0.2,0.5]","(0.5,1]",...: 1 1 1 1...
distmt
'dist' num [1:55] 14 100.3 85.8 253.6 206.6 ...
distxy
'dist' num [1:66] 0.341 0.575 0.264 1.694 1.658 ...
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
x0sub
Jul
Sep
Nov
Jan
dates0
x1sub
Jan 01
Feb 01
Mar 01
dates1


Console
Terminal
Jobs
R 4.1.2 · ~/
| second column of points, again call with with 2 arguments. As before, the
| first is mrg. The second, however, is a call to the function points with 2
| arguments. We need to do this since we're adding points to an already existing
| plot. The first argument to points is the set of x values, rep(2,52). The
| second argument is the set of y values, mrg[,3]. Of course, this is the third
| column of mrg. (We don't need to specify the range of x values again.)
> with(mrg, points(rep(2, 52), mrg[, 3]))
| You're the best!
| 94%
| We see a shorter column of points at x=2. Now let's connect the dots. Use the
| R function segments with 4 arguments. The first 2 are the x and y coordinates
| of the 1999 points and the last 2 are the x and y coordinates of the 2012
| points. As in the previous calls specify the x coordinates with calls to rep
| and the y coordinates with references to the appropriate columns of mrg.
> segments(rep(1, 52), mrg[, 2], rep(2, 52), mrg[, 3])
| That's correct!
| 95%
| We see from the plot that the vast majority of states have indeed improved
| their particulate matter counts so the general trend is downward. There are a
| few exceptions. (The topmost point in the 1999 column is actually two points
| that had very close measurements.)
...
| 96%
| For fun, let's see which states had higher means in 2012 than in 1999. Just
| use the mrg[mrg$mean.x < mrg$mean.y,] notation to find the rows of mrg with
| this particulate property.
>
Import Dataset
Global Environment
matu
num [1:2, 1:2] -0.39 -0.921 -0.921 0.39
matv
num [1:3, 1:2] -0.233 -0.561 -0.794 -0.783 0.593 ...
mrg
52 obs. of 3 variables
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
mrg[, 2]
rep(1, 52)