UPDATED OCT. 21, 2019
(SEE POST-IT NOTES)

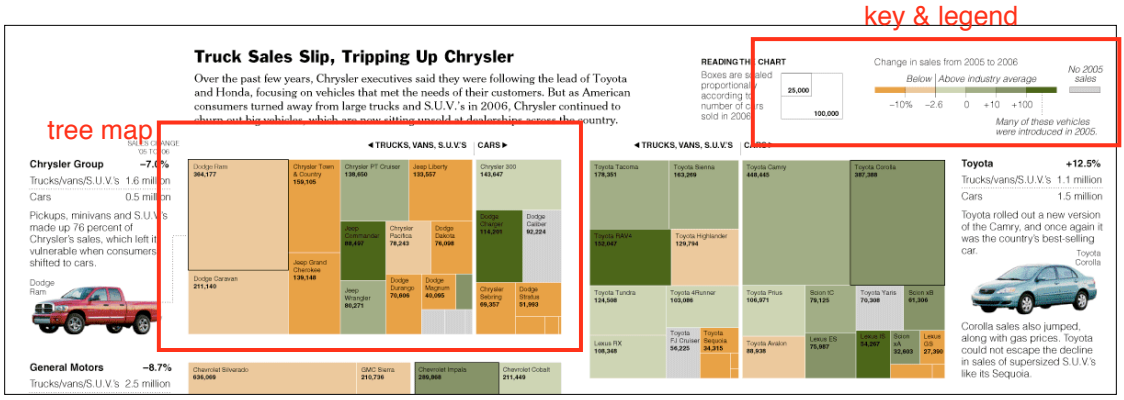
TREE MAP with LEGEND and KEY

location of this file in PDF format:
http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/treemap_legend_key_step-by-step.pdf

This demo shows how to create a tree map using a sample data set reconstructed from a car sales tree map published by The New York Times on Feb. 25, 2007, that was part of an article on the auto industry, just months before Detroit's car industry went into a major crisis. Article in PDF format here.

This exercise is based on the treemap plot described on pp 157-161 in Chapter 5: Visualizing Proportions (pp. 135-178) of the book *Visualize This*. However it uses the "treemap" package instead of the "portfolio" package described in the chapter, which is kind of ugly.

The final basic code can be found here and the commented version can be found on page 9 of this document.
Another version of the code uses custom colors. It can also be found the end of this document. It uses "bucket" to specify exact colors.



In the exercise, we'll reconstruct the tree map section shown above, as well as the key and legend shown in the red box at top right.

Download the data set here: car.sales.txt:

It's a text file (.txt) but R will recognize it as a comma delimited file (.csv). It looks like the pic at left. To see it in a more legible view, try opening it in Excel (pic on right), however remember that saving the file in Excel could create problems like invisible characters etc., therefore always save the last version as a text file by passing it through a simple text editor such as Text Wrangler or NotePad++.

```
id, Model, Change, unitsSold, Category
1, Dodge Ram, -3.5, 364177, Trucks
2, Dodge Caravan, -1.5, 211140, Trucks
3, Chrysler Town & Country, -13, 159105, Trucks
4, Jeep Grand Cherokee, -11, 139148, Trucks
5, Chrysler PT Cruiser, 35, 138650, Trucks
6, Jeep Commander, 85, 88497, Trucks
7, Jeep Wrangler, 9, 80271, Trucks
8, Jeep Liberty, -7, 133557, Trucks
9, Chrysler Pacifica, -4, 78243, Trucks
10, Dodge Dakota, -15, 76098, Trucks
11, Dodge Durango, -12, 70606, Trucks
12, Dodge Magnum, -10, 40095, Trucks
13, Chrysler 300, -5, 143647, Cars
14, Dodge Charger, 95, 114201, Cars
15, Dodge Caliber, -14, 92224, Cars
16, Chrysler Sebring, -12, 69357, Cars
17, Dodge Stratus, -10, 51993, Cars
```

id	Model	Change	unitsSold	Category
1	Dodge Ram	-3.5	364177	Trucks
2	Dodge Caravan	-1.5	211140	Trucks
3	Chrysler Town & Country	-13	159105	Trucks
4	Jeep Grand Cherokee	-11	139148	Trucks
5	Chrysler PT Cruiser	35	138650	Trucks
6	Jeep Commander	85	88497	Trucks
7	Jeep Wrangler	9	80271	Trucks
8	Jeep Liberty	-7	133557	Trucks
9	Chrysler Pacifica	-4	78243	Trucks
10	Dodge Dakota	-15	76098	Trucks
11	Dodge Durango	-12	70606	Trucks
12	Dodge Magnum	-10	40095	Trucks
13	Chrysler 300	-5	143647	Cars
14	Dodge Charger	95	114201	Cars
15	Dodge Caliber	-14	92224	Cars
16	Chrysler Sebring	-12	69357	Cars
17	Dodge Stratus	-10	51993	Cars

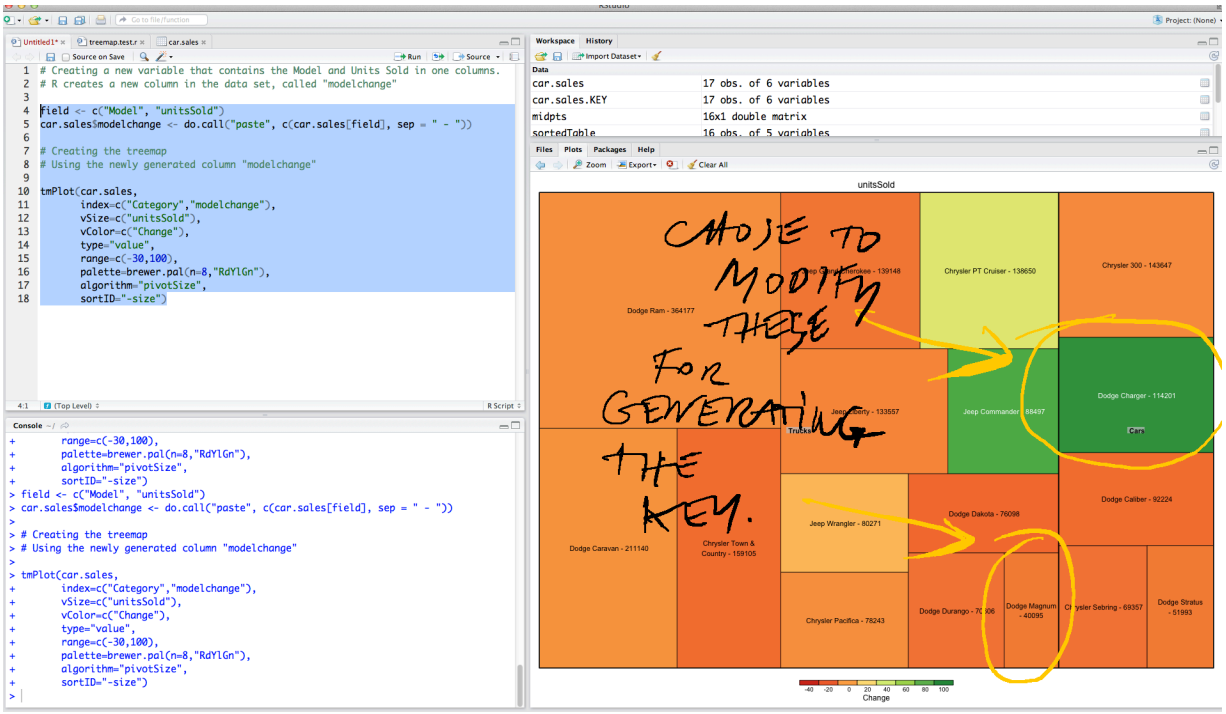
Import the data set into R (from text file):
car.sales.txt
From R, you could also import it from the URL:
http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/data_set/car.sales.txt

Download the basic R code to create the tree map:
http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/R_code/treemap.test.r

The code looks like this:

```
# Creating a new variable that contains the Model and Units Sold in one columns.
# R creates a new column in the data set, called "modelchange"

field <- c("Model", "unitsSold")
car.sales$modelchange <- do.call("paste", c(car.sales[field], sep = " - "))

# Creating the treemap
# Using the newly generated column "modelchange"

tmPlot(car.sales,
       index=c("Category","modelchange"),
       vSize=c("unitsSold"),
       vColor=c("Change"),
       type="value",
       range=c(-30,100),
       palette=brewer.pal(n=8,"RdYlGn"),
       algorithm="pivotSize",
       sortID="-size")
```

Run the code in R.
Go here for a fully commented version of the same code, or go to the commented version on page 9 of this document.
The result looks like this:



HOW TO GENERATE THE KEY FOR THE MAP

In the previous picture, I circled the rectangles for which I will modify the data in a “duplicate scrap data set file” whose sole purpose will be to generate the key for the map. I picked those rectangles because the quantities are close to 25K and 100K.

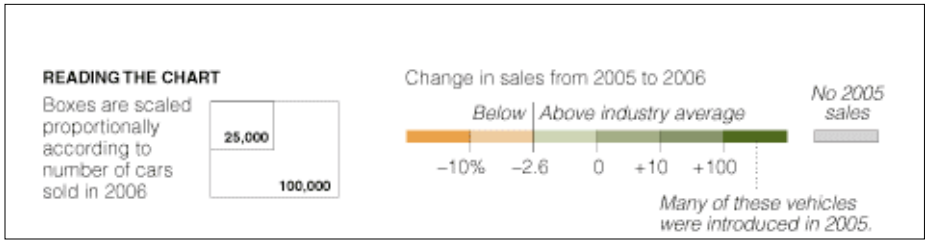
In contrast to making the key for a bubble graph (scatterplot), here we cannot simply add fake rows to the data set. Doing so would change the total for the units sold column, thereby creating a map that would be different from the original, and the key would be correct for the new data set but not for the original.

Therefore we will modify a few rows (a car and a truck) to make them even numbers (25K and 100K). In this case, 25K and 100K will be smaller than the true values, so we will need to make a note of the difference and add that to two other rectangles (trucks), simply to have the total come out identical to the original.

This is the scratch Excel file that I used to do create the new scrap data set. I have also annotated a PDF version of the scratch Excel file, explaining all the steps. Below is a picture of what the Excel file looks like:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	id	Model	Change	unitsSold	Category		id	Model	Change	unitsSold	Category			id	Model	Change	unitsSold	Category
2	1	Dodge Ram	-3.5	364177	Trucks		1	Dodge Ram	-3.5	364177	Trucks			1	Dodge Ram	-3.5	364177	Trucks
3	2	Dodge Caravan	-1.5	211140	Trucks		2	Dodge Caravan	-1.5	211140	Trucks			2	Dodge Caravan	-1.5	211140	Trucks
4	3	Chrysler Town & Country	-13	159105	Trucks		3	Chrysler Town & Cour	-13	159105	Trucks			3	Chrysler Town & Countr	-13	159105	Trucks
5	4	Jeep Grand Cherokee	-11	139148	Trucks		4	Jeep Grand Cherokee	-11	139148	Trucks			4	Jeep Grand Cherokee	-11	139148	Trucks
6	5	Chrysler PT Cruiser	35	138650	Trucks		5	Chrysler PT Cruiser	35	138650	Trucks			5	Chrysler PT Cruiser	35	138650	Trucks
7	6	Jeep Commander	85	88497	Trucks		6	Jeep Commander	85	88497	Trucks			6	Jeep Commander	85	88497	Trucks
8	7	Jeep Wrangler	9	80271	Trucks		7	Jeep Wrangler	9	80271	Trucks			7	Jeep Wrangler	9	80271	Trucks
9	8	Jeep Liberty	-7	133557	Trucks		8	Jeep Liberty	-7	133557	Trucks			8	Jeep Liberty	-7	147758	Trucks
10	9	Chrysler Pacifica	-4	78243	Trucks		9	Chrysler Pacifica	-4	78243	Trucks			9	Chrysler Pacifica	-4	78243	Trucks
11	10	Dodge Dakota	-15	76098	Trucks		10	Dodge Dakota	-15	76098	Trucks			10	Dodge Dakota	-15	76098	Trucks
12	11	Dodge Durango	-12	70606	Trucks		11	Dodge Durango	-12	70606	Trucks			11	Dodge Durango	-12	85701	Trucks
13	12	Dodge Magnum	-10	40095	Trucks		12	Dodge Magnum	-10	40095	Trucks			12	**25000 box for key**	-10	**25000**	Trucks
14	13	Chrysler 300	-5	143647	Cars		13	Chrysler 300	-5	143647	Cars			13	Chrysler 300	-5	143647	Cars
15	14	Dodge Charger	95	114201	Cars		14	Dodge Charger	95	114201	Cars			14	**100000 box for key**	95	**100000**	Cars
16	15	Dodge Caliber	-14	92224	Cars		15	Dodge Caliber	-14	92224	Cars			15	Dodge Caliber	-14	92224	Cars
17	16	Chrysler Sebring	-12	69357	Cars		16	Chrysler Sebring	-12	69357	Cars			16	Chrysler Sebring	-12	69357	Cars
18	17	Dodge Stratus	-10	51993	Cars		17	Dodge Stratus	-10	51993	Cars			17	Dodge Stratus	-10	51993	Cars
19				2051009													2051009	
20		**ORIGINAL**					**MODIFY CHART ABOVE TO GENERATE KEY (SEE STEPS BELOW)**							**CHART ABOVE INCLUDES NEW VALUES (CHECK TOTAL)**				
21																		
22				2051009														
23								FOR 25,000 KEY			FOR 100,000 KEY							
24														IF THE TOTAL MATCHES THE ORIGINAL, USE THIS NEW DATA SET AS A SCRAP FILE TO GENERATE THE KEY WITH THE TWO 25K AND 100K BOXES.				
25							40095			114201								
26							**25000**	NEW DODGE MAGNUM		**100000**	NEW DODGE CHARGER							
27							15095			14201								
28																		
29							70606			133557					TH OCT 10, 2013			
30							15095			14201								
31							85701	NEW DODGE DURANGO		147758	NEW JEEP LIBERTY							
32																		
33																		
34		**ORIGINAL SET**						**IDENTIFY VALUES TO CHANGE FOR KEY: ROWS 12 and 14**							**CHANGE VALUES IN ROWS 12 and 14 BUT MAKE SURE TO "FIX" VALUES IN ROWS 8 and 11 SO THAT TOTAL IS UNCHANGED.**				
35																		
36																		

The new dataset will be used in R to create a similar key as the one in the original graphic (nested rectangles showing 25K and 100K units):



Here is the new cleaned up scrap data set in Excel:
http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/excel/car.sales.KEY.xls

Or simply download the text version and import it into R:
http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/data_set/car.sales.KEY.txt

```
id, Model, Change, unitsSold, Category
1, Dodge Ram,-3.5,364177, Trucks
2, Dodge Caravan,-1.5,211140, Trucks
3, Chrysler Town & Country,-13,159105, Trucks
4, Jeep Grand Cherokee,-11,139148, Trucks
5, Chrysler PT Cruiser,35,138650, Trucks
6, Jeep Commander,85,88497, Trucks
7, Jeep Wrangler,9,80271, Trucks
8, Jeep Liberty,-7,147758, Trucks
9, Chrysler Pacifica,-4,78243, Trucks
10, Dodge Dakota,-15,76098, Trucks
11, Dodge Durango,-12,85701, Trucks
12,25000 box for key,-10,25000, Trucks
13, Chrysler 300,-5,143647, Cars
14,100000 box for key,95,100000, Cars
15, Dodge Caliber,-14,92224, Cars
16, Chrysler Sebring,-12,69357, Cars
17, Dodge Stratus,-10,51993, Cars
```

Which looks like this on the left: (I added the highlights for the edited cells)

I changed the names of IDs 12 and 14, as well as their quantities. And I also adjusted the quantities of IDs 8 and 11 so that the total for the unitsSold column would be the same as the original file.

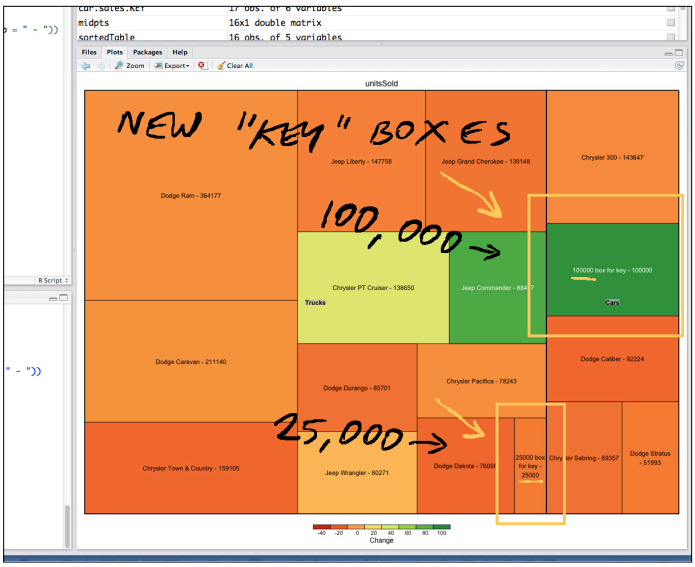
Basic tree map code

After importing the new data set into R:
car.sales.KEY.txt
Run the same code as before but with the modified file name: car.sales.KEY.r

which looks like this:

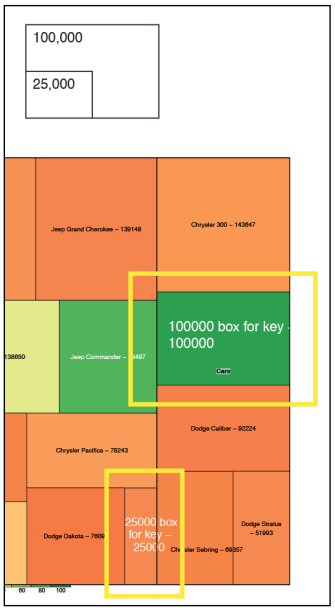
```
# Creating a new variable that contains the Model and Units Sold in one columns.
# R creates a new column in the data set, called "modelchange"

field <- c("Model", "unitsSold")
car.sales.KEY$modelchange <- do.call("paste", c(car.sales.KEY[field], sep = " - "))

# Creating the treemap
# Using the newly generated column "modelchange"

tmPlot(car.sales.KEY,
       index=c("Category","modelchange"),
       vSize=c("unitsSold"),
       vColor=c("Change"),
       type="value",
       range=c(-30,100),
       palette=brewer.pal(n=8,"RdYlGn"),
       algorithm="pivotSize",
       sortID="-size")
```

This will generate the new scrap plot with the correct rectangles for 25K and 100K. See next picture:



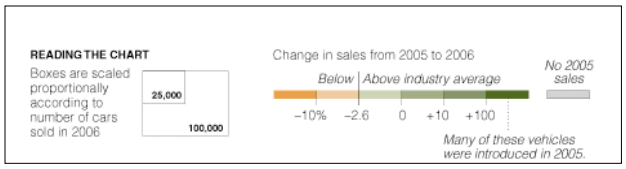
The 100K rectangle looks good, but the 25K one is too skinny and a different shape, so I will redraw that in Illustrator simply by scaling the 100K rectangle by 50% (= 1/4 of the area).

See the PDF of the original plot as well as the PDF of the new Illustrator file for more details and instructions, but it will look like this below:



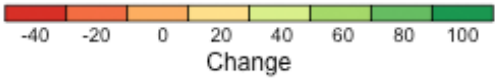
CUSTOM COLORS and LEGEND

Notice how the original tree map (and the legend) colored in green those models that in 2006 did better than the industry average for that year, which was -2.6. So the designer used that value as the departing point for the diverging scale, instead of the default zero point. This is shown below:

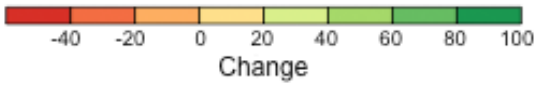


By contrast, the color scale we used so far:
`palette=brewer.pal(n=8,"RdYlGn"),`

Outputs four reds on the left and four greens on the right:



Note also that even though the selected color brewer palette has eleven values, we restricted the choices to eight in the `n=8` code before the name of the palette. The resulting palette is OK but has some limitations. The colors diverge equally from the middle (not ideal here) and it has the defect of labeling the swatches in the middle instead of locating the labels at the breaks between each swatch. As is, it's not clear if the labels should be moved to the left or to the right to align with the boundaries (separators) between each swatch. I am guessing that it should really be like this:



You can always make the necessary modifications in Illustrator when the only change required is this shifting of the labels, but if you need to customize your colors, you can use the method using "bucket" described below and suggested by Amanda Cox, the real author of the original The New York Times tree map.

Once again, import the data set into R (from text file). At this point we'll not worry about the key. Just use the process above once you have created the new customized tree map.
`car.sales.txt`
you could also import it from URL:
`http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/data_set/car.sales.txt`

Download the basic R code to create the tree map with the custom "bucket" colors:
`http://unixlab.sfsu.edu/~trogu/523/02_2013_fall/demo/treemap_key/R_code/treemapCategories.r`

Custom color (bucket) tree map code

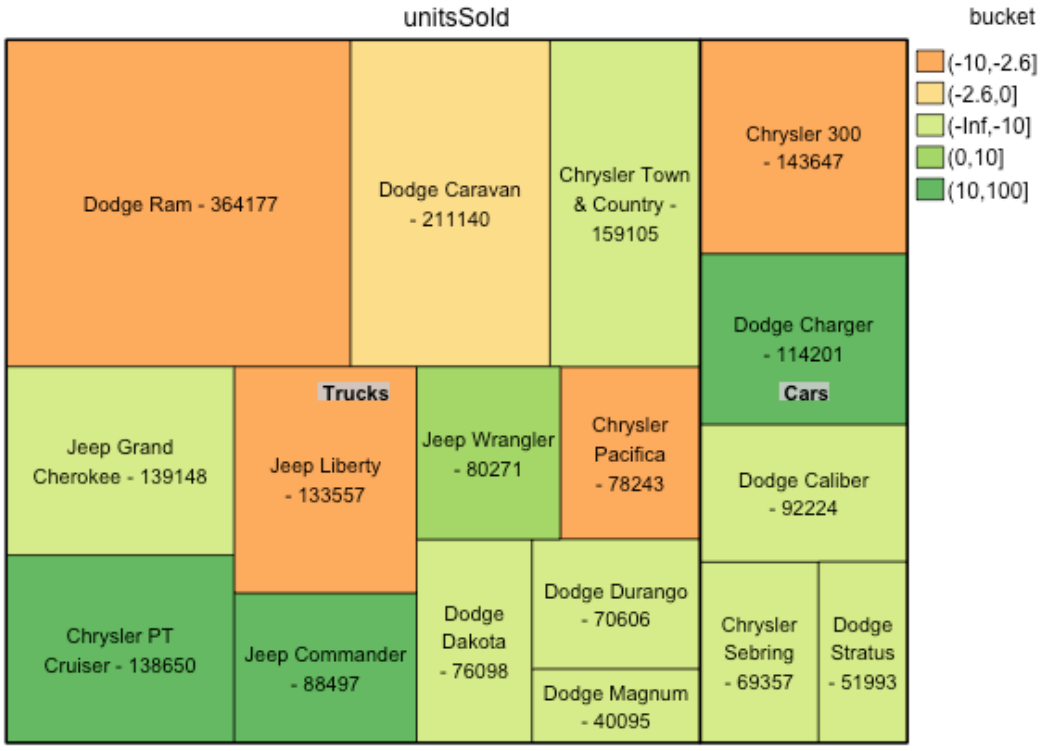
The code looks like this:

```
library(treemap)

field <- c("Model", "unitsSold")
car.sales$modelchange <- do.call("paste", c(car.sales[field], sep = " \n "))

car.sales$bucket <- cut(car.sales$Change, breaks = c(-100, -10, -2.6, 0, 10, 100, 200));

tmPlot(car.sales,
       index=c("Category","modelchange"),
       vSize=c("unitsSold"),
       vColor=c("bucket"),
       type="categorical",
       range=c(-15, 95),
       palette=c("#fdae61", "#fee08b", "#d9ef8b", "#a6d96a", "#66bd63", "#1a9850"),
       algorithm="pivotSize",
       sortID="-size"
)
```

Note the following:
1. the new separator in the field function `\n` which will insert a hard return between the model name and the number of units sold
2. the `car.sales$bucket` function specifies exactly the breaks between each color in the map and in the legend. These numbers are based on the actual data in the (percentage) Change column.

3. the color palette `palette=c` is also specified exactly with colors in hexadecimal notation. These are picked from the existing color brewer scale that was used before: `RdYlGn` but eliminating some of the steps, and more importantly selecting only two reds and four greens.

Run the code in R.
Go here for a fully commented version of the same code, or go to the end of this document. The result looks like this:



Here is a link to the color brewer palettes and their names. Mike Bostock has also posted the complete color brewer palette, and on the same page you will find a link to the javascript notation and the hexadecimal values of all the swatches used in all the scales. Since they are pretty nice, it will be a good place to start when selecting specific colors.

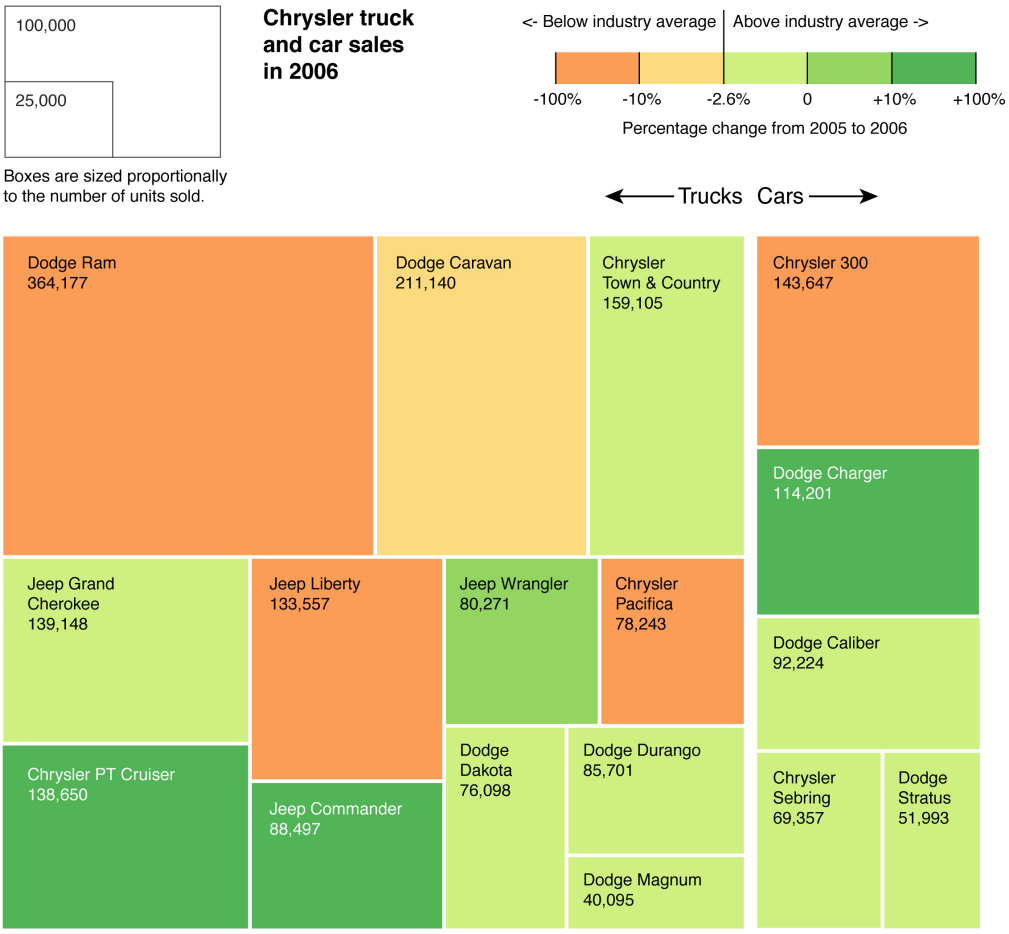
Also, keep in mind that your tree map might require a single gradient of colors, in that case simply choose the appropriate palette.

You can play with your bucket settings and the color swatches to achieve what you need. Note that the labels in the legend in the example above are slightly off, while the color order is correct. That's probably because R is arranging them alphabetically. Simply fix this in the final Illustrator layout, as shown on page 8. There are also only five color swatches instead of the expected six, probably because there is no data between 100 and 200 in the (percentage) Change column.

Please note the other adjustments I made in Illustrator:
1. A thick, vertical white line separating the two main groups (trucks and cars)
2. Thinner white lines separating each model box
3. Horizontal legend and proper labeling
4. Refined typography for the labels, flush left and in the upper left corners of the boxes
5. The main labels for the two groups are moved outside the main rectangles on top
6. Added back the commas (thousands separator) in the numbers in the labels for each box
7. Added the key generated earlier (for 25K and 100K), though I had to adapt the overall shape.

Keep in mind that it's OK to change the shape of the tree map as long as it's done all at once after the graph has been generated, and that the typography is not distorted.

Final version in Illustrator



See full commented code for basic tree map (no bucket) on page 9 and for customized color tree map (bucket) on pages 10 and 11.

Basic tree map code – with comments

```
# in iLearn, this code is referred to as "Gabriela code" (GIRL)

# this code is used with the data set called "car.sales.txt"

# this code needs the treemap library. install it by using RStudio to look for it or
# type the following:

library(treemap)

# always make sure that the treemap box is checked in the library tab
# RColorBrewer is also need to run this code

# the first two lines of code create a new variable
# that combines both the Model name and the Units sold for each model in one single column.
# R creates a new (virtual) column in the data set, called "modelchange"
# note that you can add text (sep) between the two values
# in the example below it's the text between the quotation marks: a space, than a dash
# than another space. So anything can go there, because its' a string (text)
# however, you can use that to insert hard coded items, such as a hard break in the text.
# if you type "\n" instead of the dash
# then the unit sold value will "print" in a new line. try it.

field <- c("Model", "unitsSold")
car.sales$modelchange <- do.call("paste", c(car.sales[field], sep = " - "))

# once R has created the new column
# you can create the treemap
# using the newly generated column "modelchange"

tmPlot(car.sales,    # run treemap plot of data set car.sales
       index=c("Category","modelchange"),
       # builds boxes out of values in Category:
       # anything belonging to cars will go into one group;
       # anything belonging to trucks will go into another group

       vSize=c("unitsSold"), # sets the size of each rectangle based on units sold
       vColor=c("Change"), # sets the color shift (red to green) based on percentage

#      change from a previous year (not in data set) to the year shown (units sold)
#      the percentage change is a value that needs to be created in the data set
#      you might be lucky if it's already present, if not, you might need to calculate it
#      based on data from two separate years

       type="value", # based on the numerical value in Change
       range=c(-30,100), # sets the outer edges of the scale.
       palette=brewer.pal(n=8,"RdYlGn"), # uses a pre-defined diverging scale
       # from color brewer
       algorithm="pivotSize", # the formula used to draw the rectangles
       sortID="-size") # arranges the rectangle from largest to smallest
```

Custom color (bucket) tree map code – with comments

```
# in iLearn, this code is referred to as "Gabriel code" (BOY)

# this code is used with the data set called "car.sales.txt"

# this code needs the treemap library. install it by using RStudio to look for it or
# type the following:

library(treemap)

# always make sure that the treemap box is checked in the library tab
# RColorBrewer is also need to run this code

# the first two lines of code create a new variable
# that combines both the Model name and the Units sold for each model in one single column.
# R creates a new (virtual) column in the data set, called "modelchange"
# note that you can add text (sep) between the two values
# in the example below it's the text between the quotation marks: a space, than a dash
# than another space. So anything can go there, because its' a string (text)
# however, you can use that to insert hard coded items, such as a hard break in the text.
# if you type "\n" instead of the dash
# then the unit sold value will "print" in a new line. try it.

field <- c("Model", "unitsSold")
car.sales$modelchange <- do.call("paste", c(car.sales[field], sep = " - "))

# once R has created the new column
# you can run create the treemap
# using the newly generated column "modelchange"

field <- c("Model", "unitsSold")
car.sales$modelchange <- do.call("paste", c(car.sales[field], sep = " \n "))

# the method below was suggested by Amanda Cox, author of original NY Times tree map
# a new variable called bucket is created, specifying exactly each boundary
# or unit sold value (tick marks or separators) between the color swatches in the
# tree map as well as the legend

car.sales$bucket <- cut(car.sales$Change, breaks = c(-100, -10, -2.6, 0, 10, 100, 200));

# in the NY times example, the diverging origin is set at -2.6 (industry average
# for the year shown. later, the color swatches picked will switch from red
# to green exactly at that point (rather than at zero)

tmPlot(car.sales,
       index=c("Category","modelchange"),
       vSize=c("unitsSold"),
       vColor=c("bucket"),  # colors are based on the boundaries set in the new variable
       type="categorical",  # based on the new bucket info (normally this would be value)
       range=c(-20, 200), # range for the legend.
       palette=c("#fdae61", "#fee08b", "#d9ef8b", "#a6d96a", "#66bd63", "#1a9850"),
       # the hexadecimal color values were hand-picked
       # from the RdYlGn scale in colorbrewer - six values should fit inside the seven
       # bucket breaks.
       algorithm="pivotSize",
       sortID="-size"
)
```

```
# if you run it and the legend only displays five swatches instead of six
# it's because there is no data between 100 and 200
# also, note that the text of the labels does not match the colors of the swatches
# that's because it's ordering them alphabetically, but color order is correct
# fix that in illustrator. also, make legend horizontal
# with connected rectangles, not squares
# remove double labeling by positioning just one number under each boundary line
# (separator line or tick mark) instead of centering them under each swatch
# which is the default in R
# in Photoshop, you could double-check the accuracy of the legend by sampling its colors
# and selecting similar, then looking at the data set to see that the tree map
# is rendering the values correctly.
# selecting similar in photo
```