

R programming for beginners

Ni Shuai

Computational Genome Biology
German Cancer Research Center (DKFZ)

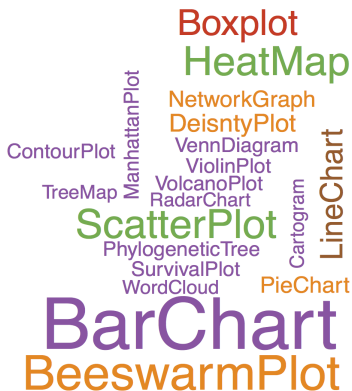
November, 2016



Graphics in R

"A picture is worth a thousand words"

- Various inbuilt functions and packages to present your data
- Build elegant and complicated plots with little effort
- High quality and high resolution graphs for publication



Graphics in R

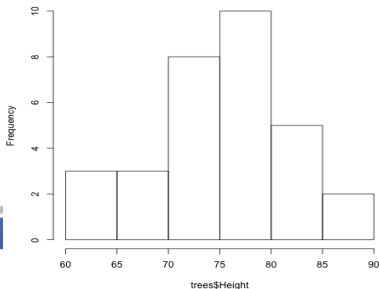
Let's look at the demos of the basic plots that can be generated in R

```
#demo(graphics)
```

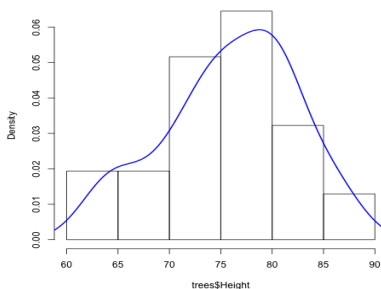
There are two kinds of graphical functions:

- High-level functions
Initializing graphical device and create a **new** plot
- Low-level functions
Plotting functions add more information to an **existing** plot

Hight distribution of 31 trees



Hight distribution of 31 trees



Hig-level graphics functions

Frequently used high-level graphics functions:

<code>plot()</code>	Depend on input data type
<code>boxplot()</code>	Show the distribution of a vector
<code>barplot()</code>	Show in bars values in a vector
<code>pie()</code>	Illustrate numerical proportion
<code>hist()</code>	Show histogram of a numeric vector
<code>pairs()</code>	Scatter plot for each column in matrix
<code>curve()</code>	Draw a curve corresponding to a function
<code>qqplot()</code>	Scatter plot of qualtiles for two vectors

Usually calling a highly-level graphical function will erase the current plot. Some functions can detect the type of its input object and generate plot accordingly, for example let's try:

```
plot(iris$Sepal.Length)
plot(iris$Species)
plot(iris$Species, iris$Sepal.Length)
plot(iris$Sepal.Width, iris$Sepal.Length)
```



Low-level graphics functions

Sometimes high-level plotting functions cannot produce the exact kind of plot you desire. In this case, low-level plotting commands can be used to add extra pices (such as points, lines or text) to the current plot. Some of the most used low-level plotting functions are:

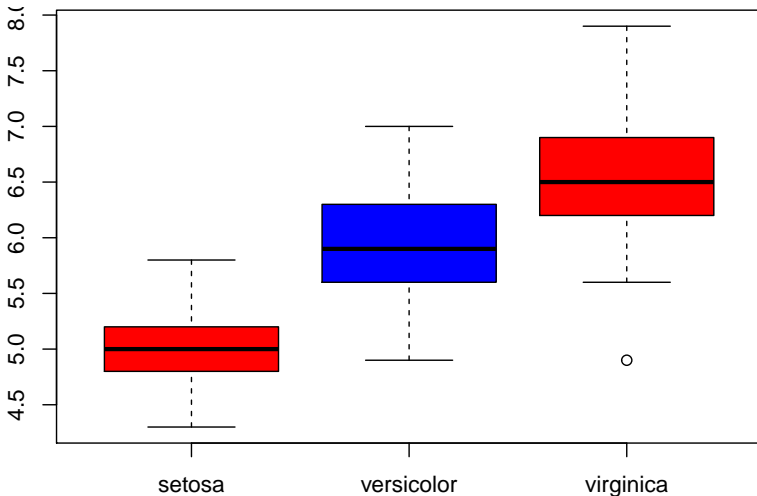
<code>points(x, y)</code>	Add points to the current plot
<code>lines(x, y)</code>	Add connected lines to the current plot
<code>abline(a, b)</code>	Add a line of slope b and intercept a to the current plot
<code>abline(h=y)</code>	Add a horizontal linen
<code>abline(v=x)</code>	Add a vertical line
<code>legend(...)</code>	Add a legend to the current plot
<code>title(...)</code>	Add a title to the current plot
<code>axis(...)</code>	Add an axis to the current plot



Graphics in R

Example: Box plot with `boxplot()`

```
boxplot(iris$Sepal.Length ~ iris$Species, col=(c("red","blue")))
```



Graphics in R

Arguments to function `boxplot()`:

```
?boxplot
```

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,  
        notch = FALSE, outline = TRUE, names, plot = TRUE,  
        border = par("fg"), col = NULL, log = "",  
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),  
        horizontal = FALSE, add = FALSE, at = NULL)
```

Exercises:

- Plot the same bar plot with three colors: gold, darkgreen and skyblue.
- Name the plot 'Sepal length in 3 species'
- Add a horizontal line to the plot shows the average sepal length



Save your plot

3 steps to save your plot to file:

- 1 Choose the format that you want to use and initialize the device, e.g. `png("c:/MyFirstPlot.png", res=300, width = 480, height = 480)`, here you can make adjustment to the size and resolution
- 2 Draw a plot with high-level and low-level graphical functions, you will not actually see the plot, because its saved in the choosen disvice
- 3 Close the device: `dev.off()`

Some useful devices:

- screen graphics: when you use interactive R
- pdf: Adobe Portable Document Format
- png: PNG graphics
- jpeg: small size, not recommended



Graphics

Exercises:

- Draw the function

$$E = mc^2$$

when $0 < m < 10$ ($c = 299792458$ m/s)

- Draw a histogram of 1000 random numbers from a normal distribution, give it a title
- Suppose your monthly expenditure is as follows,

Housing	Food	Cloths	Entainment	Other
600	200	250	350	200

Present it in a pie chart.

- In the scatter plot Sepal.Length VS. Sepal.Width in iris, extend the x axis to include a new observation, Sepal.Length=3.2, Sepal.Width=2.5
- Save the scatter plot as 'lengthVSwidth.pdf'



Global Parameters

Arguments given to a plot function only changes the plot settings locally, what if we would like some setting to be applied to all plots? This way we use function `"par()"`.

`par()` returns all current graphical parameters as a named list. Setting in each high-level plotting function will override the defaults set in `par()`

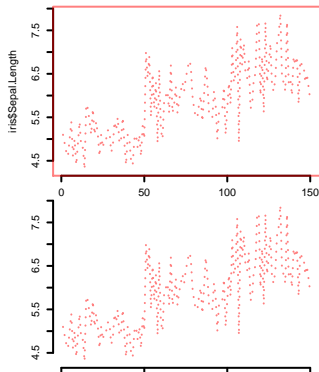
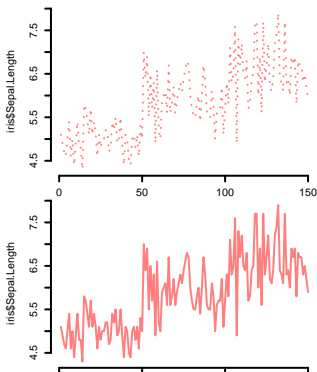
There are some parameters that can only be set by a call to `par()`

<code>ask</code>	Ask the user before a new figure is drawn
<code>mar</code>	<code>c(bottom, left, top, right)</code> , margins on 4 sides of the plot
<code>mfc</code>	<code>c(nr, nc)</code> , number of rows and columns of a multi-paneled plot
<code>mfw</code>	Figures will be drawn by row instead by columns
<code>xlog</code>	A local value indicating should a logarithmic scale to be used
<code>bg</code>	background color of the plot

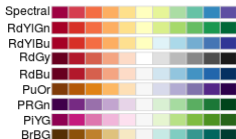


Global Parameters

```
par(mfcol=c(2,2), lty=3, col=rgb(1,0,0, alpha=0.5), bty='n', cex=0.3)  
plot(iris$Sepal.Length, type='l')  
plot(iris$Sepal.Length, type='l', lty=1)  
plot(iris$Sepal.Length, type='l', bty='o')  
plot(iris$Sepal.Length, type='l', ann=FALSE)
```

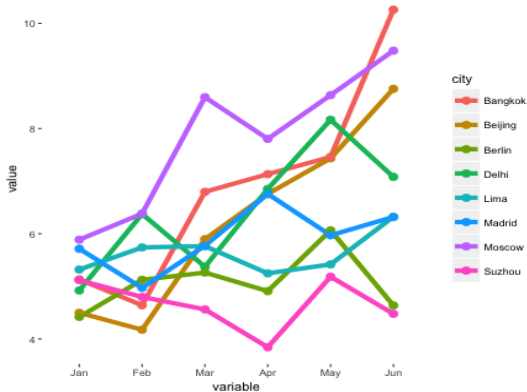


Choose your colors



What if you are drawing 8 lines in a plot and having trouble choosing a sensible colour scheme? RColorBrewer helps you to do this

The main function to extract the colors:
`brewer.pal(number, "Setname")`



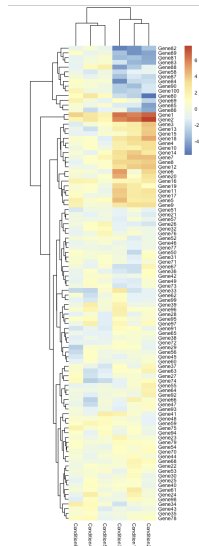
Heatmap

Build a heatmap with **Pheatmap** package

Three steps towards a nice heatmap:

- Formatting the data into a matrix
- Annotating the rows and columns
- Draw your heatmap using **pheatmap()** and save it

```
library(pheatmap)
genes <- matrix(rnorm(600), nrow=100)
rownames(genes) <- paste0("Gene", 1:100)
colnames(genes) <- paste0("Condition", 1:6)
genes[1:2,] = genes[1:2,]+3
genes[1:20, 1:3] = genes[1:20, 1:3]+3
genes[80:90, 1:3] = genes[80:90, 1:3]-3
pheatmap(genes, cellwidth = 20, fontsize = 7)
pheatmap(genes, filename = 'Myheatmap.pdf')
```



Heatmap

Need even more colors?

Usually 8 colors are not enough to represent the tiniest changes of values in a heatmap, function `colorRampPalette()` can help us to convert hand-designed color schemes into color ramps with finer scales

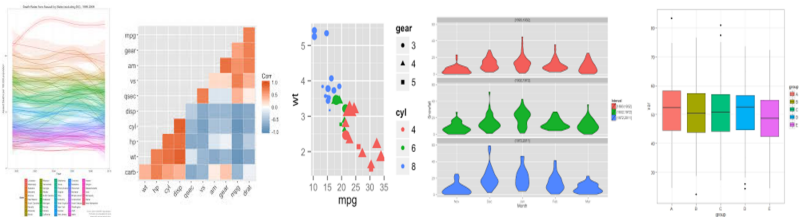
```
library(grDevices)
Finer_color = colorRampPalette(brewer.pal(n = 7, name = "RdBu"))(100)
Mymatrix=matrix(1:5000, nrow=10)
pheatmap(Mymatrix, cellheight = 10, color=brewer.pal(n = 7, name='RdBu'),
          cluster_rows = FALSE, cluster_cols = FALSE,
          main = 'Heatmap with 7 individual colors')
pheatmap(Mymatrix, color = Finer_color, cellheight = 10,
          cluster_rows = FALSE, cluster_cols = FALSE,
          main = 'Heatmap with 847 individual colors')
```



ggplot

ggplot - a new plotting system in R

- Intrinsically nice looking
- Powerful and smart
- Complete plot system and consistent grammar
- Complicated with simple plot, simple with complicated plot
- Actively maintained and developed



Essencial components in a ggplot

- Input dataset, should always be a `data.frame`
- X and Y axis - mapping, grouping, coloring
- Layer: the geometric object (plot type)
- Layer: statistical representation of the data
- Position adjustment: `dodge`, `jitter`, `stack`
- Annotations: addons, lines, borders
- Scales: axis, limits, colors
- Themes: existing themes

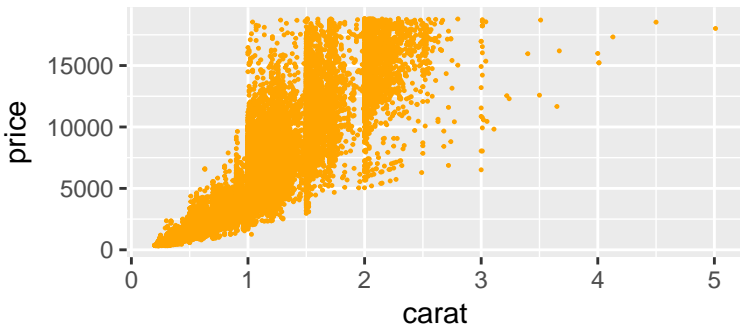
The layer `geom` and `stat` is always exchangeable in different situations based on the emphasis



Grammar

All ggplot2 plots with a call to `ggplot()`, supplying default data and aesthetic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

```
library(ggplot2)
ggplot(data=diamonds, aes(x=carat, y=price)) +
  geom_point(color='orange', size= 0.2)
## ggsave('Myggplot.pdf')
## ggsave("Myggplot.png")
```



Aesthetic mapping

Aesthetic mappings is the central part of the plot, it describes what variable in the data to be represented and should the plot elements be grouped by variables.

```
##city miles per gallon and highway miles per gallon  
ggplot(data=mpg, aes(x=cty, hwy))  
p = ggplot(data=mpg, aes(x=cty, hwy))  
summary(p)  
  
p + geom_point()
```

Aesthetic mappings can be set in `ggplot()` and in every `geom()` layers

```
ggplot(mpg) + geom_point(aes(x=cty, y=hwy))  
ggplot(mpg, aes(x=cty, y=hwy))
```



Aesthetic mapping

`aes()` is also used to set color and size by variables in dataset

```
ggplot(data=mpg, aes(x=cty, hwy))+  
  geom_point(aes(color=cty))  
  
ggplot(data=mpg, aes(x=cty, hwy))+  
  geom_point(aes(color=factor(cyl), size=cyl))+  
  ggtitle('title of my first graph')
```

You can also map aesthetics to functions of variables

```
ggplot(data=mpg, aes( x = cty^ 2, y = hwy / cyl))+  
  geom_point()
```



Aesthetic mapping

There can be only one variable in `aes()`, with suitable geom method

```
ggplot(mtcars, aes(mpg))  
ggplot(mtcars, aes(mpg))+geom_histogram(binwidth=5)  
  
ggplot(mtcars, aes(mpg))+geom_point()
```

geom labels will override color scheme from ggplot call

```
ggplot(mtcars, aes(x=wt, y=mpg, color=cyl))+  
  geom_point(size=5, color='green')  
ggplot(mtcars, aes(x=wt, y=mpg, color=factor(cyl)))+  
  geom_point(size=5)
```



Layers

We can set another layer of statistical representation, variable names should always indide `aes()`

```
ggplot(mpg, aes(x=cty, y=hwy))+  
  geom_point(aes(color=factor(year), size=displ))+  
  stat_smooth()
```

```
ggplot(mpg, aes(x=cty, y=hwy))+  
  geom_point(aes(color=factor(year), size=displ),  
            alpha=0.5, position= 'jitter')+  
  stat_smooth()+  
  scale_color_manual(values=c('gold','lightblue'))
```

Aesthetic mapping will not be shared between added layers

```
ggplot(mpg)+ geom_point(aes(x=cty, y=hwy))+stat_smooth()
```



Exercise

- Box plot

```
ggplot(mtcars, aes(x=factor(cyl), y=mpg, fill=cyl))+  
  geom_boxplot()  
ggplot(mtcars, aes(x=factor(cyl), y=mpg, fill=factor(cyl)))+  
  geom_boxplot()
```

- Violin plot

```
ggplot(mtcars, aes(x=factor(cyl), y=mpg, fill=factor(cyl)))+  
  geom_violin()
```

- Jitter plot

```
###jitter plot  
ggplot(diamonds, aes(x=clarity, y=price,color=clarity))+  
  geom_jitter()  
head(diamonds)
```



Exercise

• Line chart

```
scale_fill_manual(values=rep(brewer.pal(8, 'Pastel1')[c(1,2,5)], 5))
set.seed(100)
rainfall=data.frame(matrix(rnorm(48), 8, 6))
rainfall=rainfall+5
names(rainfall)=c('Jan','Feb','Mar','Apr','May','Jun')
rainfall[1:4,]=rainfall[1:4,] + rep(seq(0,9,length.out = 8), each=6)
rainfall$city=c('Beijing','Bangkok','Delhi',
                'Moscow','Suzhou','Lima', 'Berlin','Madrid')
rainfall=melt(rainfall)
ggplot(rainfall, aes(variable, value, color=city))+
  geom_line(aes(group=city), size=2 )+
  geom_point(size=3)+
  theme(panel.grid.minor=element_blank(),
        panel.grid.major=element_blank(),
        panel.background=element_blank(),
        panel.border=element_blank())+
  scale_fill_manual(values=brewer.pal(8, 'set2'))
```

