Graphics

Fritz Scholz

April 29, 2015
A well constructed graph is worth a thousand words.
Many people use R mainly for obtaining effective graphs.
I usually use functions to build up the graph I want.
That way I can reuse them for other but similar situations, say for different data sets.
I have used it to make technical drawings for illustrations.
You can annotate graphs in many ways.
You can even use mathematical expressions in annotations.
There are many generic plot commands.
Many further commands add graphics elements to plots.
We illustrate by example, use documentation to explore more.
See also: R Graphics by Paul Murrell, Chapman & Hall/CRC.
The following drawing generated in R was part of a collaboration on Tolerance Analysis between IBM and Boeing.
Scatter Plot: `plot(faithful)`

RStudio saves plots in various formats: \(\Rightarrow\) Plots \(\Rightarrow\) Export
• **faithful** is a data frame (native to R) with 2 columns: eruptions and waiting in minutes.
• From the data frame nature of 2 columns the `plot` command knows to plot column 2 against column 1.
• Default plot symbol is a hollow circle.
• Normal usage is `plot(x, y)` with x and y numerical vectors of equal length.
• Note the resulting difference in the following commands

```r
plot(faithful[,1], faithful[,2])
plot(faithful[,1], faithful[,2],
 xlab="eruption length (min)",
 ylab="waiting time to next eruption (min)"
)
plot(faithful[,1], faithful[,2])
plot(faithful[,1], faithful[,2]),
xlab="eruption length (min)", ylab =
"waiting time to next eruption (min)"
fit <- lm(waiting ~ eruptions, data=faithful)
plot(faithful[,1],faithful[,2],
     xlab="eruption length (min)",
     ylab="waiting time to next eruption (min)")
lines(faithful$eruptions,fitted(fit),
      col="red",lwd=2,lty=4)
abline(fit)

- **lm fits waiting as linear function of eruptions**
- **data=faithful indicates reference data set,**
  **avoids using faithful$waiting, etc.**
- **fitted(fit) gives the fitted values (y-coordinates).**
- **abline(fit) draws a straight line with a and b from fit.**
Faithful Eruptions and Fitted Line: The Plot
attach(USArrests)
plot(Assault, Murder, cex=UrbanPop/20,
    main="arrests/100000")
legend("topleft","circle diameter =
    proportional to percent urban
    population",bty="n")

cex=UrbanPop/20 ⇒ 3rd dimension, cex = character expansion.

bty="n" no bounding box around legend
Crime Statistics for 50 States

Circle diameter = proportional to percent urban population

arrests/100000

Assault

Murder
plot(0, 0, xlim=c(-10, 10), ylim=c(-10, 10), type="n", xlab="", ylab="")
for(i in 1:20) abline(i-10, 1, lty=i)
lty=0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash
Using Previously Imported `ReactionTime` Data Frame

```r
head(ReactionTime)
    Reaction Station Shift
1      86    ST64   B
2     182    ST64   B
3     132    ST64   B
4     196    ST64   B
5     160    ST64   A
6      3    ST65   A

hist(ReactionTime$Reaction,breaks=seq(0,400,5))
# histogram bins cover [0,400] in increments of 5.

hist(ReactionTime$Reaction,breaks=seq(0,400,5),
    main="EMS Turnout Time", xlab="time (seconds)",
    col=c("blue","orange"))
```
hist(ReactionTime$Reaction, breaks=seq(0,400,5))
hist(ReactionTime$Reaction, breaks=seq(0,400,5), main="EMS Turnout Time", xlab="time (seconds)", col=c("blue","orange"))
Plotting Symbols and Colors:

```r
pcol <- c("green", "yellow", "purple", "red", "blue", "gold", "black", "cyan", "orange")
plot(1:25, 1:25, pch=1:25, col=pcol)
```

See all 657 color names via `colors()`.
x <- rnorm(100)
# x is a standard normal random sample, n=100

qqnorm(x, pch=16, cex=.5)
# makes QQ-plot of sample

> qqline(x)
# adds a fitted line to the previous plot.
# line is fitted through 1st and 3rd quartiles
# of sample and standard normal distribution
The $p$-quantile of a normal population with mean $\mu$ and standard deviation $\sigma$ is $x_p = \mu + \sigma z_p$.

$P(X \leq x_p) = p$, by definition.

The standard normal ($\mu = 0$ and $\sigma = 1$) $p$-quantile is $z_p$.

In R: $z_p = \text{qnorm}(p)$

The sample median is close to the population median.

This follows from the law of large numbers (LLN).

If $X_{(i)}$ represents the $i^{th}$ smallest sample value, then it can be viewed as $p_i$-quantile of the sample of size $n$, $p_i = \frac{i}{n+1}$.

Similarly, the sample $p$-quantile is close to the population $p$-quantile, at least when $0 \ll p \ll 1$.

Thus the $X_{(i)}$ should be in a rough linear relationship to $z_{p_i}$, slope $\approx \sigma$, intercept $\approx \mu$. 
qqnorm(x, pch=16, cex=.5)
qqline(x)
Normal QQ-Plot $n = 30$

```r
par(mfrow=c(2,3))
x <- rnorm(30); qqnorm(x); qqline(x)
x <- rnorm(30); qqnorm(x); qqline(x)
x <- rnorm(30); qqnorm(x); qqline(x)
x <- rnorm(30); qqnorm(x); qqline(x)
x <- rnorm(30); qqnorm(x); qqline(x)
x <- rnorm(30); qqnorm(x); qqline(x)
```

- The `par` function controls many plotting parameters.  
  $\Rightarrow$ ?par.
- Some plotting parameters work within the plotting function, others only within a prior `par(...)` call.
- The `;` separation allows several commands on one line.
Judging Normality Takes Lots of Practice

The following diagrams illustrate normal Q-Q plots, where the theoretical quantiles are compared against the sample quantiles. The points on the plots fall close to the diagonal line, indicating that the data is normally distributed.
par(mfrow=c(2,3))
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
x <- rnorm(100); qqnorm(x, pch=16, cex=.5); qqline(x)
Increasing $n$ to 100 Helps
Empirical Distribution Function (EDF) Comparison with Normal CDF

```r
> x <- sort(rnorm(20))
> plot(stepfun(x, (0:20)/20), ylab="empirical cdf")
> lines(seq(-4,4,.1),pnorm(seq(-4,4,.1),col="blue")
> lines(seq(-4,4,.1),pnorm(seq(-4,4,.1),mean(x),
>     sqrt(var(x))),col="green")
# the green curve uses the fitted normal (closer)
```

![Graph showing Empirical Distribution Function (EDF) comparison with Normal CDF](attachment:graph.png)
Comparing Two Samples

```r
qqplot(rnorm(500), rnorm(800))
abline(0, 1)
```
Comparing Two Samples (Shift)

```r
qqplot(rnorm(500), rnorm(800) + 1)
abline(0, 1)
```
Comparing Two Normal Samples (Scale Factor)

```r
qqplot(rnorm(500), rnorm(800) * .7)
abline(0, 1)
```
Comparing Normal & t Samples

```r
qqplot(rnorm(500), rt(600, 3))
abline(0, 1)
```
Comparing Normal & $t$ Samples Using \texttt{ad.test}

\begin{verbatim}
> x <- rnorm(500)
> y <- rt(600,3)
> ad.test(x,y)

Anderson-Darling k-sample test.

Number of samples: 2
Sample sizes: 500, 600
Number of ties: 0

Mean of Anderson-Darling Criterion: 1
Standard deviation of Anderson-Darling Criterion: 0.76007

T.AD = \frac{\text{Anderson-Darling Criterion - mean}}{\sigma}

Null Hypothesis: All samples come from a common population.

\begin{tabular}{ccc}
AD & T.AD & asympt. P-value \\
version 1: & 5.7848 & 6.2952 & 0.0011862 \\
version 2: & 5.8000 & 6.3200 & 0.0011660 \\
\end{tabular}
\end{verbatim}
Comparing Normal & t Samples

```r
qqplot(rnorm(500), rt(600, 3))
abline(0, 1)
```
Box Plots

```r
boxplot(Reaction~Station+Shift, data=ReactionTime, cex.axis=.5)
```
The horizontal box lines $\equiv$ 3 quartiles $Q(.25), Q(.5), Q(.75)$.

The dashed vertical lines extend to the adjacent values.
- Compute the interquartile range $IQR = Q(.75) - Q(.25)$.
- The upper adjacent value is the largest observation $\leq Q(.75) + 1.5 \times IQR$
- The lower adjacent value is the smallest observation $\geq Q(.25) - 1.5 \times IQR$

Points beyond adjacent values shown individually (outliers?)
- For $\mathcal{N}(\mu, \sigma^2) \approx .35\%$ are beyond each adjacent value.
- `data=ReactionTime` $\Rightarrow$ simpler reference to variables.
- `Reaction ~ Station + Shift` implies boxplots for all factor combinations of Station and Shift.
Many graphics functions allow fine tuning control as follows.

Plot dimensions are controlled by `xlim=c(a,b)` and `ylim=c(c,d)`, using your `a < b, c < d`.

Axis labels are controlled by `xlab="your x-label"` and `ylab="your y-label"`.

Set the main plot title by `main="Your Main Title"`.

Set the plot sub title by `sub="Your Sub Title"`.

See `par` for many graphics control options, like

- `cex, cex.axis, cex.main, cex.sub` character expansion factors.
- `col, col.axis, col.lab, col.main, col.sub` specifying colors.
- `font, font.axis, font.lab, font.main, font.sub` font choices, `1 = plain text (the default), 2 = bold face, 3 = italic and 4 = bold italic.`
Augmentation to Plots

- Some commands only work after a plot has been initiated.
- `abline(a, b)` draws line with intercept `a` and slope `b`.
- `segments(...)` draws line segment(s) from $P_1$ to $P_2$.
- `arrows(...)` draws arrow(s) from $P_1$ to $P_2$.
- `lines(...)` draws curves through points by line segments.
- `points(...)` plots symbols (`pch`) at specified locations.
- `polygon(...), rect(...)` draw polygons and rectangles.
- `text(...)` puts specified text at selected positions.
- `legend(...)` adds legends to plots.
- `mtext(...)` adds text to plot margins.
- and lots more ⇒ `help.start()` ⇒ Graphics ⇒ `aplot`.

Check the documentation on each, to get a rough usage sense.
par(mfrow=c(1,3))

plot(LakeHuron,type="l",main='type="l"')
# points connected by lines

plot(LakeHuron,type="p",main='type="p"')
# only points are plotted

plot(LakeHuron,type="b",main='type="b"')
# both points and lines are plotted

# see ?plot for more on the type argument
Fisher’s or Anderson’s Iris Data

str(iris)
'data.frame': 150 obs. of 5 variables:
$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species : Factor w/ 3 levels "setosa","versicolor",...

unique(iris$Species)
[1] setosa versicolor virginica
Levels: setosa versicolor virginica

pairs(iris,col=
  rep(c("red","blue","orange"),each=50))
An Attempt to Visualize Higher Dimensional Data
pairs(iris,col=rep(c("red","blue","orange"),each=50),upper.panel=NULL)
⇒ \texttt{?plotmath} gives documentation on it.

\texttt{> demo(plotmath)} gives examples by commands and results.

normalhist <- function(n=1000) {
    x <- rnorm(n)
    xx <- seq(-4,4,.1)
    hist(x,breaks=xx,probability=T,
         main="normal histogram")
    yy <- dnorm(xx)
    lines(xx,yy,col="blue")
    text(-4,.3,expression(varphi(x)==
                         over(1,sqrt(2*pi))*phantom(0)*
                         e^{-x^2/2}),adj=0,col="blue")
    xbar <- mean(x)
    s2 <- var(x)
    text(2,.3,substitute(bar(x)==xb,
                         list(xb=round(xbar,3))),adj=0)
    text(2,.25,substitute(s==ss,
                         list(ss=round(sqrt(s2),3))),adj=0)
}
\[ \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \]

\[ \mu = 0.009 \]

\[ \sigma = 1.016 \]
• We indicated the interactive way of saving plots within RStudio and within the R Gui.

• There are also various other ways by direct commands.
  
  \texttt{pdf(file="myplot.pdf", width=8, height=6)}
  opens pdf-file "myplot.pdf". \texttt{width, height} are in inches.

• Any subsequent graphics commands produce output to that file, until \texttt{dev.off()} is issued, or the R session terminates.

• Similar commands exist for other graphics formats
  \Rightarrow \texttt{?Devices}

  for tiff, jpeg, bmp, png, postscript, quartz (Mac).
A good reference is available in Wiki Books.


Part of Murrell’s book and especially the code for all his examples is available here.

Such code can serve as starting point for doing your own plots.

Some of the examples require extra packages to be installed, followed by `library(package.name)`.

We will try out a few.

Other references may be found on the web.
Add-on packages provide more graphics capabilities. We mention just three.

These are too complex to delve into here. Good as projects.

The `lattice` package.


The `ggplot2` package, not covered here, but see *R Graphics Cookbook* by Winston Chang, O’Reilly, 2013.