Data Visualization in R
2. Standard graphics in R

Michael Friendly
SCS Short Course
Sep/Oct, 2018
http://datavis.ca/courses/RGraphics/

Outline: Session 2

• Session 2: Standard graphics in R
  ▪ R object-oriented design
  ▪ Tweaking graphs: control graphic parameters
    • Colors, point symbols, line styles
    • Labels and titles
  ▪ Annotating graphs
    • Add fitted lines, confidence envelopes
    • Add text, legends, point labels

R graphics systems

• Two graphics worlds
  ▪ “graphics”— traditional or base graphics
  ▪ “grid”— new style graphics
• Things work very differently in these
• Infrastructure for both is “grDevices“ — the R graphics engine
  ▪ Graphics devices,
  ▪ colors, fonts

  e.g.,
  • the Cairo graphics device can create high-quality vector (PDF, PostScript and SVG) and bitmap output (PNG, JPEG, TIFF)
  • the tikz device uses the LaTeX tikz package and LaTeX fonts, colors, etc.
Base graphics functions: high & low

- Graphics functions are mostly one of two types:
  - High-level functions → complete plots
    - plot(), boxplot(), dotplot(), mosaicplot(), ...
  - Low-level functions → add to an existing plot
    - lines(), points(), legend(), arrows(), polygon(), text()
- Some functions can work either way, via an argument add=TRUE/FALSE
  - symbols(x, y, ..., add=TRUE)
  - car::dataEllipse(x, y, add=TRUE)

The many faces of plot()

- plot() is the most important function in traditional graphics
- It is designed as a generic function, that does different things with numeric data (x, y), factors (FAC), matrices (MAT),...
  - plot(x) - index plot of x[i] vs i
  - plot(x, y) – scatterplot
  - plot(FAC, y) – boxplots
  - plot(x, FAC) – stripchart
  - plot(FAC, FAC) – spineplot, barchart
  - plot(MAT) – scatterplot matrix -> pairs()

Object-oriented approach in R

- Everything in R is an object, and has a class
  - data sets: class “data.frame”
  - statistical models: class “lm”, “glm”, ...
- Fit a model: obj <- lm(...) → a “lm” model object
  - print(obj) & summary(obj) → numerical results
  - anova(obj) & Anova(obj) → tests for model terms
  - update(obj), add1(obj), drop1(obj) model selection

Objects & methods

Method dispatch: The S3 object system

- Functions return objects of a given class
  - Anova/regression: lm() → an "lm" object
  - Generalized linear models: glm() → c("glm", "lm") – also inherits from lm()
  - Loglinear models: loglm() → a "loglm" object
- Class-specific methods have names of the form method.class
  - plot.lm(), plot.glm() – model diagnostic plots
- Generic functions – print(), plot(), summary() call the appropriate method for the class
  - plot(Eff(obj)) – calls plot.eff() effect plots
  - plot(influence(obj)) – calls plot.influence() for influence plots
  - plot(prcomp(obj)) – plots a PCA solution for a “prcomp” object
> data(Duncan, package="car")
> class(Duncan)
[1] "data.frame"
> duncan.mod <- lm(prestige ~ income + education, data=Duncan)
> class(duncan.mod)
[1] "lm"
> print(duncan.mod)
Call:
  lm(formula = prestige ~ income + education, data = Duncan)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.065      0.599  -10.11  1.1e-05 ***
income       0.599      0.146   4.17  3.7e-05 ***
education     0.546      0.146   3.72  3.7e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> Anova(duncan.mod)
Analysis of Variance Table

Response: prestige
          Df  Sum Sq Mean Sq F value  Pr(>F)
income     1 4474.3  4474.3  25.000 1.1e-05 ***
education  1  5516.0  5516.0  30.900 1.7e-06 ***
Residuals 42  7507.0    178.7
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> library(car)
> methods(class="lm")
[1] add1                alias               anova               Anova               
[5] avPlot              Boot                ...                boxCox               
[9] case.names          ceresPlot           coerce              confidenceEllipse
[13] confint             cooks.distance     crPlot              deltaMethod         
[17] deviance            dfbeta             dfbetas             dummy.coef           
[21] effects             drop1               dummy.coef           durbinWatsonTest   
[25] family              extractAIC          family              influencePlot       
[29] hatvalues           hcm                 influencePlot       inverseResponsePlot
[33] influencePlot       initialize          leveragePlot        linearHypothesis   
[37] labels              leveneTest          leveragePlot        linearHypothesis   
[41] logLik              mcPlot              rep                 model.frame        
[45] model.matrix        nestFest            nextBoot            nobs                
[49] outlierTest         plot                powerTransform      predict              
[53] print               proj                qqPlot              residuals            
[57] qr                  residualPlot        residPlot           residuals            
[61] standard           residPlot           residPlot           residuals            
[65] simulate            slotsFrom3          summary             
[69] variable.names      vcov                see '?methods' for accessing help and source code

Some methods for “lm” objects (in the base and car packages):

> library(car)
> methods(class="lm")
[1] add1                alias               anova               Anova               
[5] avPlot              Boot                ...                boxCox               
[9] case.names          ceresPlot           coerce              confidenceEllipse
[13] confint             cooks.distance     crPlot              deltaMethod         
[17] deviance            dfbeta             dfbetas             dummy.coef           
[21] effects             drop1               dummy.coef           durbinWatsonTest   
[25] family              extractAIC          family              influencePlot       
[29] hatvalues           hcm                 influencePlot       inverseResponsePlot
[33] influencePlot       initialize          leveragePlot        linearHypothesis   
[37] labels              leveneTest          leveragePlot        linearHypothesis   
[41] logLik              mcPlot              rep                 model.frame        
[45] model.matrix        nestFest            nextBoot            nobs                
[49] outlierTest         plot                powerTransform      predict              
[53] print               proj                qqPlot              residuals            
[57] qr                  residualPlot        residPlot           residuals            
[61] standard           residPlot           residPlot           residuals            
[65] simulate            slotsFrom3          summary             
[69] variable.names      vcov                see '?methods' for accessing help and source code

Some plot methods produce multiple plots.
You can control the layout with par() settings

op <- par(mfrow=c(1,4)) # change layout parameters
plot(duncan.mod) # regression diagnostic plots
par(op) # restore old parameters

avPlots(duncan.mod, id.n=2, pch=16, ellipse=TRUE, 
        ellipse.args=list(levels=0.68, fill=TRUE, fill.alpha=0.1))
Graphic parameters

• All graphic functions take arguments that control details
  - colors (col=)
  - point symbols (pch=)
  - line type (lty=); line width (lwd=)
• Often these have default values in the function definition
  - col="black"; col=par("col"); col=palette()[1]
  - lwd=2, lty=1
• Most high-level graphic functions have a "..." argument that allow passing other arguments to the plotting functions

Some graphics parameters can be set globally for all graphs in your session, using the `par()` function

- `par(mar=c(4,4,1,1))` – plot margins
- `par(cex.lab=1.5)` – make axis labels 50% larger
- `par(cex=2)` – make text & point symbols 2x larger
- Graphics functions often use these as defaults

Most can be set in calls to high-level plotting functions

- `avPlots(duncan.mod, pch=16, cex=2, cex.lab=1.5, ...)`

The most commonly used graphic parameters:

These colors are the default, palette) 8 packages specifically related to color: colorspace, colorRamps, RColorBrewer, ...

![Graphic parameters](http://gastonsanchez.com/r-graphical-parameters-cheatsheet.pdf)
The functions `plot()`, `points()` and `lines()` understand a `type=` parameter and render the \((x, y)\) values in different ways.

```r
x <- -5:5
y <- -x^2 + 25
plot(x, y, type="p")
plot(x, y, type="l")
plot(x, y, type="b")
plot(x, y, type="o")
plot(x, y, type="h")
plot(x, y, type="s")
```

More on color

- Presentation graphs require careful choice of colors
  - Legible if copied in B/W?
  - Visible to those with color deficiency?
  - Mapping categorical or continuous variables to color scale
- R has a variety of ways to specify color
  - color names: see `colors()`
  - Hex RGB: red = "#FF0000", blue="#0000FF"
  - with transparency: #rrggbbaa
  - hsv(): hue, saturation, value (better as perceptual model)
  - `colorRamps`: `rainbow(n)`

Traditional R graphics: mental model

- R graphics functions add ink to a canvas – the "painter’s model"
  - new graphics elements overlay / obscure what is there before
  - only way to move or remove stuff is to re-draw in white (background color)
  - animated graphs re-do the whole plot in each frame
  - Transparent colors are often useful for filled areas
- Typically, create a graph with a high-level function, then add to it with low-level if desired
- I’ll illustrate by re-constructing two historical graphs
  1. Identify the graphical elements: points, lines, text, ...
  2. Start with a basic plot, then add to it
Building a custom graph

Custom graphs can be constructed by adding graphical elements (points, lines, text, arrows, etc.) to a basic `plot()`

John Arbuthnot: data on male/female sex ratios:

```r
> data(Arbuthnot, package="HistData")
> head(Arbuthnot[,c(1:3,6,7)])

Year Males Females Ratio Total
1 1629 5218 4683 1.114 9.901
2 1630 4858 4457 1.090 9.315
3 1631 4422 4102 1.078 8.524
4 1632 4994 4590 1.088 9.584
5 1633 5158 4839 1.066 9.997
6 1634 5035 4820 1.045 9.855

Arbuthnot didn’t make a graph. He simply calculated the probability that in 81 years from 1629—1710, the sex ratio would always be > 1
The first significance test!
```

Follow along

- Select all (ctrl+A) and copy (ctrl+C) to the clipboard
- In R Studio, open a new R script file (ctrl+shift+N)
- Paste the contents (ctrl+V)
- Run the lines (ctrl+Enter) to along with me

(You could instead save that file to your lab HOME directory and open it from there.)

Building a custom graph

1. Start with a basic plot of points

```r
plot(Ratio ~ Year, data=Arbuthnot,
     pch=16,
     ylim=c(1, 1.20),
     cex.lab = 1.3,
     ylab="Sex Ratio (M/F)")
```

Code details:
- `pch`: I like filled circles (16=●) for points
- `ylim`: allow more vertical space for caption
- `cex.lab`: make axis labels larger

2. Add gray lines

```r
lines(Ratio ~ Year, data=Arbuthnot, col="gray")
```

Code details:
- I could have used type="b" or type="o"
- But I wanted the lines in gray, not black
Building a custom graph

```r
plot(Ratio ~ Year, data=Arbuthnot,
     pch=16,
     ylim=c(1, 1.20),
     cex.lab = 1.3,
     ylab="Sex Ratio (M/F)"
# connect points by lines
lines(Ratio ~ Year, data=Arbuthnot, col="gray")
# add reference line
abline(h=1, col="red", lwd=3)
text(1640, 1, "Males = Females", col="red")
```

3. Add horizontal reference line & label

```
plot(Ratio ~ Year, data=Arbuthnot,
     pch=16,
     ylim=c(1, 1.20),
     cex.lab = 1.3,
     ylab="Sex Ratio (M/F)"
# connect points by lines
lines(Ratio ~ Year, data=Arbuthnot, col="gray")
# add reference line
abline(h=1, col="red", lwd=3)
text(1640, 1, "Males = Females", col="red")
```

4. Add regression & smoothed lines

```
plot(Ratio ~ Year, data=Arbuthnot,
     pch=16,
     ylim=c(1, 1.20),
     cex.lab = 1.3,
     ylab="Sex Ratio (M/F)"
# connect points by lines
lines(Ratio ~ Year, data=Arbuthnot, col="gray")
# add reference line
abline(h=1, col="red", lwd=3)
text(1640, 1, "Males = Females", col="red")
# add linear regression line
abline(lm(Ratio ~ Year, data=Arbuthnot),
col="darkgreen")
# add loess smooth
Arb.smooth <- with(Arbuthnot,
                   loess.smooth(Year, Ratio))
lines(Arb.smooth$x, Arb.smooth$y,
col="blue", lwd=2)
```

5. Add figure caption

```
plot(Ratio ~ Year, data=Arbuthnot,
     pch=16,
     ylim=c(1, 1.20),
     cex.lab = 1.3,
     ylab="Sex Ratio (M/F)"
# connect points by lines
lines(Ratio ~ Year, data=Arbuthnot, col="gray")
# add reference line
abline(h=1, col="red", lwd=3)
text(1640, 1, "Males = Females", col="red")
# add linear regression line
abline(lm(Ratio ~ Year, data=Arbuthnot),
col="darkgreen")
# add loess smooth
Arb.smooth <- with(Arbuthnot,
                   loess.smooth(Year, Ratio))
lines(Arb.smooth$x, Arb.smooth$y,
col="blue", lwd=2)
```

The same graph, using ggplot2

ggplot2 has a totally different idea about constructing graphs

The syntax adds elements and layers to a graph with functions connected with "+" signs.

Details in a following lecture

```
ggplot(Arbuthnot, aes(x=Year, y=Ratio)) +
ylim(1, 1.20) +
ylab("Sex Ratio (M/F)") +
gg_point(pch=16) +
gg_line(color="gray") +
gg_smooth(method="loess", color="blue", fill="blue", alpha=0.2) +
gg_smooth(method="lm", color="darkgreen", se=FALSE) +
gg_line(intercept=1, color="red", size=2) +
annotate("text", x=1640, y=1.005, label="Males = Females", color="red", size=4) +
annotate("text", x=1690, y=1.19,
        label="Arbuthnot's data on the Male / Female Sex Ratio", size=6) +
theme_bw()
```

```
Playfair’s wheat

William Playfair (1759—1836) invented most of the forms of modern data graphics: the bar chart, line graph and pie chart.

- This multivariate chart shows the price of wheat (bars), wages of a good mechanic (line graph), and the reigns of British monarchs over 250 years, 1565—1830
- Playfair’s goal: Show that workers were better off now than at any time in the past.

Did Playfair succeed?

What can you read from this chart re: wages vs. price of wheat?

Reproducing Playfair’s chart

To try to reproduce this chart:
- Identify the graphical elements: 3 time series, cartouche caption, grid lines, ...
- Make a basic plot setting up (x,y) range, axis labels, ...
- Use low-level functions to add graphical elements

Capitol: plot using multiline text()

Monarchs: draw using segments(), label with text()

Time series of wheat: plot as step function: type="s"

Wages: draw using lines()

Reproducing Playfair’s chart

Playfair’s data was digitized from his chart. The HistData package records this as two data frames.

```r
> str(Wheat)
'data.frame': 53 obs. of 3 variables:
$ Year : int 1565 1570 1575 1580 1585 1590 1595 1600 1605 1610 ...
$ Wheat: num 41 45 42 49 41.5 47 64 27 33 32 ...
$ Wages: num 5 5.05 5.08 5.12 5.15 5.25 5.54 5.61 5.69 5.78 ...

> str(Wheat.monarch)
'data.frame': 12 obs. of 4 variables:
$ name : Factor w/ 12 levels "Anne","Charles I",...: 5 10 2 4 3 11 12 1 6 7 ...
$ start : int 1565 1603 1625 1649 1660 1689 1702 1714 1727 ...
$ end : int 1603 1625 1649 1660 1685 1689 1702 1714 1727 1760 ...
$ commonwealth: int 0 0 0 1 0 0 0 0 0 ...
```

Reproducing Playfair’s chart

- Reproducing Playfair’s chart
- text label: srt=3 rotates the text 3°
- paste(s1, s2, ..., sep="\n") makes separate lines
- font=3: italic
- The decorative cartouche is drawn with text(), using a vector of strings

Consulting for Playfair

WP: Can you help me make a better graph?
SCS: Yes, plot the ratio of Wheat / Wages: the labor cost to buy a quarter of wheat

This clearly shows that wheat was becoming cheaper in terms of the amount of labor required
Plotting data was so new that Playfair did not think of plotting a derived value.

Consulting for Playfair

Code for this plot:

Wheat1 <- within(na.omit(Wheat), {Labor=Wheat/Wages})
with(Wheat1, {
plot(Year, Labor, type="b", pch=16, cex=1.5, lwd=1.5, ylab="Labor cost of a Quarter of Wheat (weeks)", ylim=c(1,12.5), xlim=c(1560,1823), cex.axis=1.2, cex.lab=1.5, lab=c(12,5,7)
); lines(lowess(Year, Labor), col="red", lwd=3)
})

The remainder of the code is similar to that for the original plot
In 1875 Francis Galton studied heredity of physical traits. In one experiment, he sent packets of sweet peas of 7 different sizes to friends, and measured the sizes of their offspring.

His first attempt was a semi-graphic table, tabulating the number of parent-child seeds in each combination of values. He noted that both distributions followed the “law of frequency of error” (Normal distribution).

Galton’s (1877) presentation graph:
- Plotted mean diameter of child seeds vs. mean of parents
- Noticed these were nearly in a line– An “Ah ha” moment!
- The slope of the line said something about heredity

But, the slope of the line < 1 → “reversion” toward mean → children of large/small parents less extreme than their parents
Later used the term “regression” for this phenomenon, and statistical explanation.

How Galton got there – the untold story
His friend, JFW Herschel said, "Why don’t you make a scatterplot?"
He fired up R on his Babbage machine ...
... but was initially disappointed in the result: too much overplotting

NB: Galton was careful to
- Set aspect ratio = 1
- Use explicit axis labels

SCS: Ah! your data are discrete.

Galton thoughtfully met with an SCS consultant, who said: “Show me your data!!!”
Galton’s peas: a text-table plot

Galton: Ah! Maybe I’ll just go back to my original table

SCS consultant: Good, but make it into a plot also: use text()

Here’s a good graphic trick: make font size ~ f(n)

plot(child ~ parent, data=peas, ...)  
with(peas.freq,  
text(parent, child,  
  count, col="red", cex=log(count)))

Galton’s peas: Sunflower plots

Perhaps better: use point symbols that show explicitly the number of observations at each (x, y) location

A sunflower plot uses symbols with the number of rays = # of obs at each (x, y)

Now, he could see the upward trend – sort of

Galton’s peas: jittering

Another possibility is to jitter() the plotted points by adding little random #s

But, he also needed to calculate and plot the line of means and the trend line

plot(jitter(child) ~ jitter(parent), data=peas,  
pch=16, cex.lab=1.25,  
asp=1, xlim=c(14, 23),  
ylab="Child seed diameter (0.01 in.)")

# add line of means  
means <- aggregate(child ~ parent, data=peas,  
  FUN=mean)  
lines(child ~ parent, data=means, type="b",  
pch="+", cex=2, lwd=7, col="darkgreen")

text(15, 15.3, "means", col="darkgreen", cex=1.4,  
pos=2)

# calculate & draw the regression line  
peas.mod <- lm(child ~ parent, data=peas)  
abline(peas.mod, lwd=2, col="blue")

Plotting discrete data: Galton’s peas

Making Galton’s argument visually clearer:
• Label the regression line with its slope
• Show the comparison line (slope=1) if there was no regression toward the mean

41

42

43

44
library(devtools)
install_github("Gibbsdavidl/CatterPlots")

# plot random cats
multicat(x=-10:10, ys=rnorm(21),
cat=c(1,2,3,4,5,6,7,8,9,10),
catcolor=list(c(0,0,0,1)),
canvas=c(-0.1,1.1, -0.1, 1.1),
xlabel="some cats",
ylabel="other cats",
main="Random Cats")

How this works:
• 11 cat shape images saved as PNG
• Calls plot(x, y, ...) – set up plot frame
• rasterImage(catImg, ...) – plot each cat

R has special methods for dealing with time series data
The sunspots data set records monthly mean relative sunspot numbers from 1749 to 1983.

> data(sunspots)
> str(sunspots)
Time-Series [1:2820] from 1749 to 1984: 58 62.6 70 55.7 85 83.5 94.8 75.5 ...

plot(sunspots, cex.lab=1.5, ylab="Number of sunspots")

But the aspect ratio (V/H) of the plot is often important. A systematic pattern is revealed when the average local trend is \( \sim 45^\circ \)

Lag plots show a time series against lagged versions of themselves. This helps visualizing "auto-dependence".

plot(sunspots, lag(sunspots, 1), cex=0.7, col="blue")

There is a strong dependence between this year’s sunspots and last.

Often, we want to see dependence across a range of lag values. lag.plot(series) does this quite flexibly

lag.plot(sqrt(sunspots), set = c(1:4, 9:12), layout=c(2,4), col="blue", cex=0.7 )

dependence is persistent, but weakens over lags.
Time series: Seasonal patterns

Data UKLungDeaths: monthly deaths from bronchitis, emphysema and asthma in the UK, 1974–1979

data(UKLungDeaths)
plot(ldeaths, lwd=2, main="UK Lung Deaths")

Looks like cycles, peaking in winter

The acf() function calculates and plots autocorrelations of a time series across various lags

acf(ldeaths, lwd=5, main="Autocorrelations of UK Lung Deaths")

This gives a compact view of the seasonal pattern

Other time-series graphs can show other details

Saving image files: R Studio

From the R Studio Plots tab, you can save any image in a variety of types

Some options are available in the menu to control the details of size, shape & image format

For publication purposes, you will often want more control: plot margins, font sizes, figure shape, etc.

Saving image files: R scripts

• The default graphics device in R is your computer screen.
• In an R script, there are 3 steps:
  1. Open a graphics device, with desired parameters
     • Call png(), jpg(), pdf(), ...
  2. Create the plot
  3. Close the graphics device: dev.off()

```
png(file="bowling.png", width=400, height=400)
x <- 1:10
y <- (x - 5)^2
plot(y ~ x, type="b",
xlab="Frame",
ylab="Bowling score")
dev.off()
```

Much easier with ggplot2: ggsave()
Saving image files: Margins, fonts

```r
png(file="bowling.png", width=400, height=400)
x <- 1:10
y <- (x - 5)^2
plot(y ~ x, type="b", pch=15, cex=2, cex.lab=1.5,
     xlab="Frame", ylab="Bowling score",
     main="Why do I have a slump?")
dev.off()
```

Set plot margins, font size for points & labels

```r
png(file="bowling0.png", width=400, height=400)
x <- 1:10
y <- (x - 5)^2
plot(y ~ x, type="b",
     xlab="Frame", ylab="Bowling score")
dev.off()
```

Saving image files: R markdown

- In R markdown files, use chunk options to control figures & other output
  - global options -- control all chunks: knitr::opts_chunk$set()
  - individual chunk options

```{r setup, include=FALSE, message=FALSE}
opts_chunk$set(fig.path="figs/",
              dev=c("png","pdf"), # devices for figs
              fig.width=6, fig.height=5, # fig size
              fig.align="center",
              dpi=300, # make high res.
              digits=4, …) # printed output
```

```{r wheat1, fig.width=9, fig.height=4}
data(Wheat)
plot(Wheat ~ Year, data=Wheat, type="s")
```

Set global options:

Change size for this figure:

Details: see [https://yihui.name/knitr/options/](https://yihui.name/knitr/options/)

Summary

- Standard R graphics
  - High-level (plot()) vs. low level (lines()) functions
  - Understand object-oriented methods
- Graphics parameters
  - Understand the basic ones: col, pch, lty, lwd
  - Use `help(par)` or cheat sheet to find others
  - For a high-level function, use `help(fun)`
- Building graphs
  - Think about graphic elements: points, lines, areas, ...
  - How these should be rendered: graphical attributes