

Department of Statistics
University of California, Los Angeles

Advanced Graphics in R

Ryan Rosario

January 27, 2010



Outline

- 1 Introduction
- 2 Intro to Customizing Graphics
- 3 Overlays and Monitors
- 4 Other Goodies: Math and Movies
- 5 Conclusion

- 1 Introduction
 - Early Days of R
 - Graphics Learning Curve
- 2 Intro to Customizing Graphics
- 3 Overlays and Monitors
- 4 Other Goodies: Math and Movies
- 5 Conclusion

Code for Graphics

The code to produce the graphics in this presentation is available at the following URL, for you to review at your leisure.

http://www.stat.ucla.edu/~rosario/scc/10w_agr_code.R

A copy of these slides (big) for following along is available at the following URL.

http://www.stat.ucla.edu/~rosario/scc/10w_agr-big.pdf

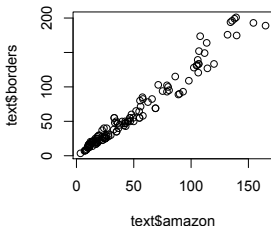
A copy of these slides (handout) for printing at home is available at the following URL.

http://www.stat.ucla.edu/~rosario/scc/10w_agr.pdf

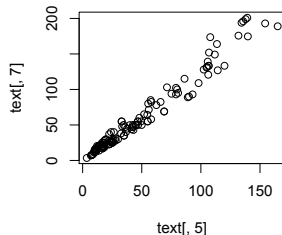
Early Days with R

Using the `plot` command is so simple, but when first getting started with R, something like the following is discouraging...

```
plot(text$Amazon.Sale.Price,  
      text$Borders.Price)
```



```
plot(text[,4],text[,7])
```



So, What's Wrong with That?

- there is no title to introduce the graphic.

So, What's Wrong with That?

- there is no title to introduce the graphic.
- the axes refer to data frame dimensions, rather than the context of the data.

So, What's Wrong with That?

- there is no title to introduce the graphic.
- the axes refer to data frame dimensions, rather than the context of the data.
- data points are too large as displayed.

So, What's Wrong with That?

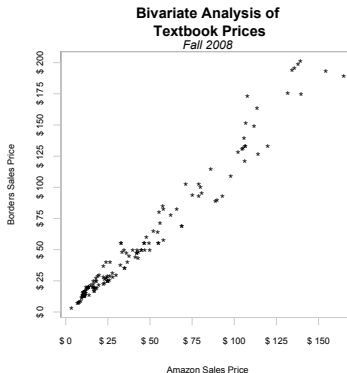
- there is no title to introduce the graphic.
- the axes refer to data frame dimensions, rather than the context of the data.
- data points are too large as displayed.
- data points are “clumped” which reduces signal to noise ratio in the plot.

So, What's Wrong with That?

- there is no title to introduce the graphic.
- the axes refer to data frame dimensions, rather than the context of the data.
- data points are too large as displayed.
- data points are “clumped” which reduces signal to noise ratio in the plot.
- there may be multiple classes of data points.

Now What?

Of course it is possible
to make beautiful
graphics in R.



The Learning Curve

...but at first it requires a lot of work!

```

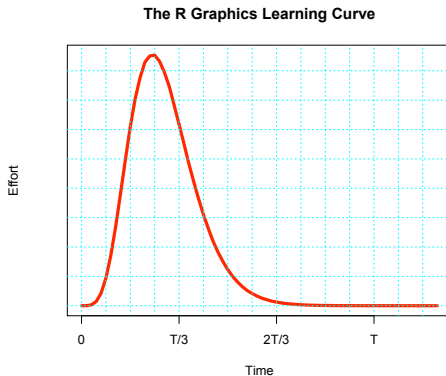
1 plot(text[,5],text[,7],main="Bivariate Analysis of
2 \n Textbook Prices",xlab="Amazon Sales Price",ylab=
   "Borders Sales Price",pch='*',xaxt="n",yaxt="n"
   ,cex.lab=0.75,box=FALSE,bg="grey")
3 mtext("Fall 2008",3,font=3)
4 axis(1,at=seq(0,200,25),paste("$",seq(0,200,25)),
   cex.axis=0.75,col="grey")
5 axis(2,at=seq(0,200,25),paste("$",seq(0,200,25)),
   cex.axis=0.75,col="grey")
6 axis(c(3:4),col="grey",tick=FALSE,labels=FALSE)
7 box("plot",col="grey")

```

Ouch...



The Learning Curve



- 1 Introduction
- 2 Intro to Customizing Graphics
 - par
 - Histograms: Not Just a Plot!
 - Manipulating Axes
 - Plot Types
 - Line Width
 - Color
- 3 Overlays and Monitors
- 4 Other Goodies: Math and Movies
- 5 Conclusion

In Intermediate Graphic in R we have already seen some ways to customize graphics:

col	lines	add	wireframe
main	density	identify	drape
pch	boxplot	ts	color.palette
legend	bwplot	mvtsplot	contour
pie	levelplot	xyplot	scatterplot.matrix
hist	curve	map	persp
abline	lwd	points	

We will skip most of these, and review some of them here.



The par Command

Graphics options can be passed directly to `par`, or to higher level plotting functions.

```
par(..., no.readonly = FALSE)
<highlevel plot> (... , <tag> = <value>)
```

We will stick with the second method for now. We will discuss the first method later.

The par Command

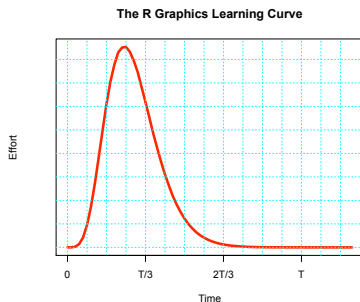
We use a graphics parameter by calling `plot`, or one of its friends (`hist`, `boxplot` etc.) with a comma separated list of *named* options.

```
1 plot(text[,5],text[,7],main="Bivariate Analysis of
   \n Textbook Prices",xlab="Amazon Sales Price",
   ylab="Borders Sales Price",pch='*',xaxt="n",
   yaxt="n",cex.lab=0.75)
```

The above `plot` command contains the following parameters: `xlab`, `ylab`, `pch`, `xaxt`, `yaxt`, `cex.lab`, `box`.

A Motivating Example

There is a lot of material, so let's start with an example and see where it takes us. Let's look at the Learning Curve graphic.

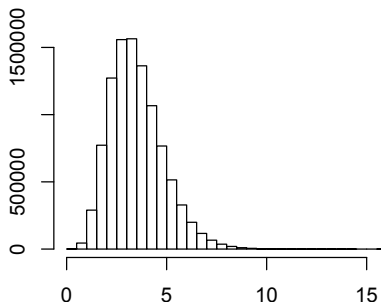


This graphic is an example of how I can express this trend using the graphical parameters in R.

A Motivating Example

First, I generated 100,000 random numbers from a gamma distribution with $k = 7$ and $\theta = 2$ to construct this “trend.” The histogram below displays the gamma distribution.

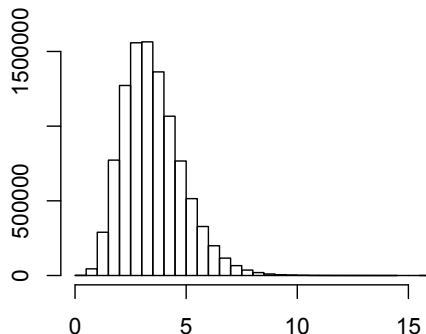
```
my.gamma <- rgamma(100000,7,2)
```



A Motivating Example

I suppress the default axis labels and default plot title by passing an empty string "" to some graphical parameters.

```
1 hist(my.gamma, xlab="", ylab="", main="")
```



A Motivating Example

But this is not the type of graphic I want. Instead, I want to plot the density using a curve. We could use `density` or `dgamma`, but let's work with the histogram. A histogram is an *object* of type `hist`. We can see what goodies this object contains using the `attributes` function.

```
1 attributes(hist(my.gamma))
```

```
$names
```

```
[1] "breaks"      "counts"      "intensities"
     "density"   "mids"        "xname"       "equidist"
```

```
$class
```

```
[1] "histogram"
```

Extracting Information from hist

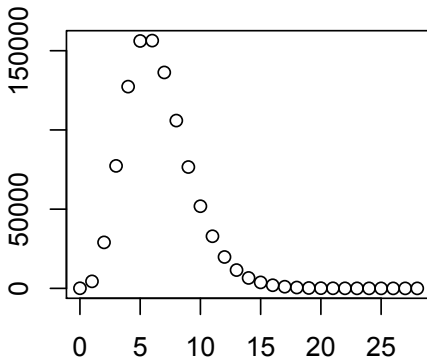
`hist(my.gamma,plot=FALSE)$counts` returns a vector of counts for each *bin* in the histogram, and I use this as the y axis. The number of bins in the histogram can be modified by adding the parameter `br` to the `hist` call.

```
1 y <- hist(my.gamma,plot=FALSE)$counts
2 #plot=FALSE suppresses plot, constructs object
3 x <- seq(0, length(y)-1)
4 #I used seq to create a dummy axis.
5 plot(x,y,xlab="",ylab="", main="")
```

The number of bins (or *breaks*) can be controlled using the `br` parameter in the `hist` call.

An Aside: Extracting Information from hist

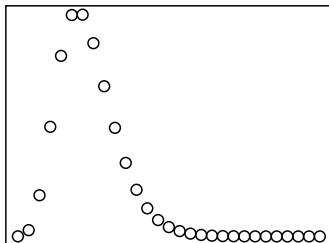
Let's check our progress...



Manipulating Axis

Recall that my x axis has no units and currently does not make sense, so let's replace it with something more appropriate. First we must remove it. Also, my y axis really has no practical meaning, so let's just remove it altogether.

```
1 plot(x,y,xlab="",ylab="", main="", xaxt="n", yaxt="n")
```



Manipulating Axis

We can then add back new axes that look how we want using `axis`.

- ① can put labels at specific places on the x axis using the `at` parameter
- ② can give these tick marks labels given in parameter `labels`.
- ③ the first parameter in `axis` indicates where to put the axis.

The side Argument (first argument of `axis`)

1 = bottom (x), 2 = left (y), 3 = above, 4 = right works for other functions such as `mtext`.

Manipulating Axis

```

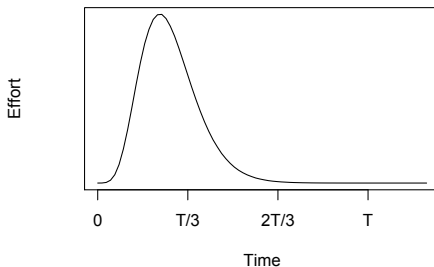
1  #Learning Curve graphic
2  y <- hist(my.gamma,br=100,plot=FALSE)$counts
3  x <- seq(0,max(my.gamma),length=length(y))
4  plot(y~x,xlab="Time",ylab="Effort",xaxt="n",
       yaxt="n", main="The R Graphics Learning
       Curve")
5  axis(1,at=seq(0,max(x),length=4), labels=c(0,"
       T/3","2T/3","T"))
6  #Cyan grid.
7  abline(v=seq(0,max(x),length=15),lty=3,col="
       cyan")
8  abline(h=seq(0,max(y),length=15),lty=3,col="
       cyan")

```

Some More Tinkering: Plot Types

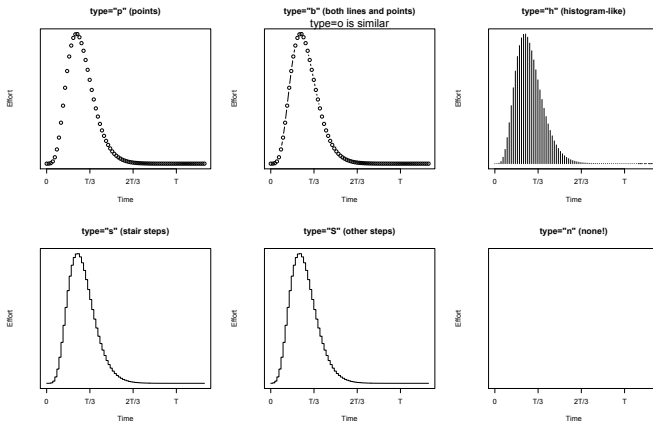
We can create a continuous curve by simply changing the plot type to `type="l"`.

The R Graphics Learning Curve



Some More Tinkering: Plot Types

Note: plot types are part of the `plot` function, not `par`.

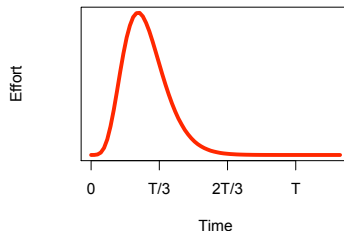


Some More Tinkering: Line Width and Color

We can change the line width using the `lwd` parameter. `lwd` defaults to 1, and larger integer values provide thicker lines. I use `lwd=4`.

We can also change the color of the line using the `col` parameter. I use `col="red"`, a named color.

The R Graphics Learning Curve



Color

Colors can be specified by name (i.e. "red"), by palette code (i.e. 10), or by RGB content #10AF09.

A full list of color names is available using the `colors()` command. We can also convert a color name into its corresponding RGB value using the `col2rgb` function.

A full index of R colors can be found at <http://research.stowers-institute.org/efg/R/Color/Chart/>. You can create your own index by using the following command:

```
1 source("http://research.stowers-institute.org
    /efg/R/Color/Chart/ColorChart.R")
```



Color - Palette Codes

R colors — Sorted by Hue, Saturation, Value

24	153	261	154	262	155	263	156	264	157	265	158	266	159	267	160	268	161	269	162	270	163	271	164	272
165	273	166	274	167	275	168	276	169	277	170	278	171	279	172	280	173	281	174	282	175	283	176	284	177
285	178	286	179	287	180	288	181	289	182	290	183	291	184	292	185	293	186	294	187	295	188	296	189	297
190	298	191	299	192	300	193	301	126	127	194	302	195	303	196	304	197	305	198	306	199	307	200	308	201
309	202	310	203	311	204	312	205	313	206	314	207	315	208	316	209	317	210	318	211	319	212	320	213	321
214	322	215	323	216	324	217	325	218	326	219	327	80	82	220	328	221	329	222	330	223	331	224	332	225
333	226	334	227	335	152	260	228	336	229	337	230	338	231	339	232	340	233	341	234	342	235	343	416	418
236	344	237	345	238	346	239	347	140	240	348	241	349	242	350	243	351	244	352	245	353	246	354	247	355
248	356	249	357	651	250	358	251	359	252	360	1	253	361	609	608	605	606	607	557	558	559	560	561	404
372	376	374	373	375	32	36	33	35	34	133	134	137	136	135	100	556	555	554	552	553	482	479	480	568
481	483	633	632	630	631	634	60	61	59	58	570	571	572	569	101	507	506	57	505	503	504	426	424	425
428	427	587	586	585	584	588	579	580	54	55	53	52	56	567	582	581	583	573	532	533	530	531	534	621
624	622	535	623	92	94	93	91	449	22	19	20	21	90	6	4	41	5	39	38	40	23	37	3	620
7	487	485	486	25	488	488	529	484	500	498	499	502	850	501	646	492	647	649	648	138	76	77	78	79
75	151	147	148	149	150	63	64	65	66	412	411	413	414	146	67	410	142	143	145	144	396	397	394	395
382	513	386	383	385	384	398	83	381	380	377	378	379	18	443	444	445	446	447	415	656	655	654	652	653
497	494	493	495	496	657	85	86	89	87	88	259	47	48	50	51	49	393	366	365	362	363	364	102	103
104	105	106	514	417	516	517	518	515	139	448	81	258	257	256	254	255	576	575	574	578	577	472	612	614
613	610	611	478	474	471	460	8	9	10	12	635	428	475	17	16	13	14	15	405	406	407	408	409	519
523	520	522	521	108	113	110	109	111	112	72	74	71	70	68	69	114	42	639	637	638	636	46	44	45
43	546	403	399	124	400	401	121	122	125	123	402	589	433	432	431	434	430	592	590	591	593	617	618	615
616	619	2	131	128	129	130	132	436	437	599	604	602	600	601	603	442	441	440	438	439	62	565	564	563
562	566	141	387	477	490	491	30	73	29	461	28	26	27	597	595	594	598	435	596	107	473	471	469	470
467	468	548	31	549	550	551	547	99	97	96	95	98	115	465	464	463	466	462	629	625	628	626	627	545
542	544	543	541	640	84	454	453	452	450	451	512	508	511	509	510	641	459	476	458	457	456	120	118	116
117	119	367	371	368	369	345	642	643	644	370	392	455	390	391	388	389	525	524	527	526	528	540	538	539
537	536	419	421	422	423	420																		

Color - Sample Index¹

127. dimgrey	#696969	105	105	105	177. gray24	#303030	61	61	61
128. dodgerblue	#1E90FF	30	144	255	178. gray25	#404040	64	64	64
129. dodgerblue1	#1E90FF	30	144	255	179. gray26	#424242	66	66	66
130. dodgerblue2	#1C86EE	28	134	238	180. gray27	#454545	69	69	69
131. dodgerblue3	#1874CD	24	116	205	181. gray28	#474747	71	71	71
132. dodgerblue4	#104E8B	16	78	139	182. gray29	#4A4A4A	74	74	74
133. firebrick	#B22222	178	34	34	183. gray30	#4D4D4D	77	77	77
134. firebrick1	#FF3030	255	48	48	184. gray31	#4F4F4F	79	79	79
135. firebrick2	#EE2C2C	238	44	44	185. gray32	#525252	82	82	82
136. firebrick3	#CD2626	205	38	38	186. gray33	#545454	84	84	84
137. firebrick4	#8B1A1A	139	26	26	187. gray34	#575757	87	87	87
138. floralwhite	#FFFAF0	255	250	240	188. gray35	#595959	89	89	89
139. forestgreen	#228B22	34	139	34	189. gray36	#5C5C5C	92	92	92
140. gainsboro	#DCDCDC	220	220	220	190. gray37	#5E5E5E	94	94	94
141. ghostwhite	#F8F8FF	248	248	255	191. gray38	#616161	97	97	97
142. gold	#FFD700	255	215	0	192. gray39	#636363	99	99	99
143. gold1	#FFD700	255	215	0	193. gray40	#666666	102	102	102
144. gold2	#EBC900	238	201	0	194. gray41	#696969	105	105	105
145. gold3	#CDAD00	205	173	0	195. gray42	#6B6B6B	107	107	107
146. gold4	#9B7500	139	117	0	196. gray43	#6E6E6E	110	110	110
147. goldenrod	#DAA520	218	165	32	197. gray44	#707070	112	112	112
148. goldenrod1	#FFC125	255	193	37	198. gray45	#737373	115	115	115
149. goldenrod2	#EEB422	238	180	34	199. gray46	#757575	117	117	117
150. goldenrod3	#CD9B1D	205	155	29	200. gray47	#787878	120	120	120

¹Image from Earl F. Glynn, Stowers Institute for Medical Research
<http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf>

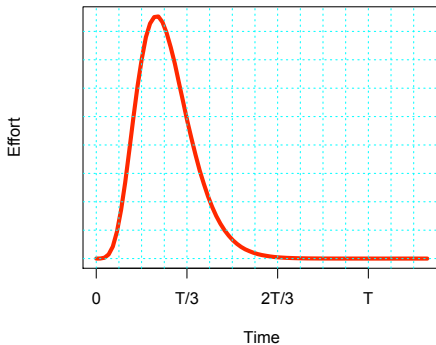
Final Result

After some last few touches using `abline` covered in the previous minicourse, we get the following code:

```
1  #Learning Curve graphic
2  y <- hist(my.gamma,br=100,plot=FALSE)$counts
3  x <- seq(0,max(my.gamma),length=length(y))
4  plot(y~x,type="l",lwd=4,col="red",xlab="Time",ylab="
    Effort",xaxt="n",yaxt="n", main="The R Graphics
    Learning Curve")
5  axis(1,at=seq(0,max(x),length=4), labels=c(0,"T/3","2T/3"
    ,"T"))
6  #Cyan grid.
7  abline(v=seq(0,max(x),length=15),lty=3,col="cyan")
8  abline(h=seq(0,max(y),length=15),lty=3,col="cyan")
```

Final Result

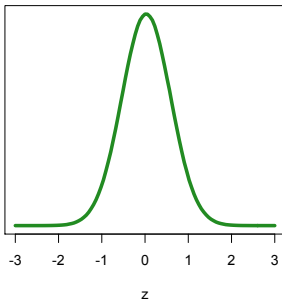
The R Graphics Learning Curve



Exercise 1

Create a plot of the normal distribution. The curve should be thicker than the default, should be colored *Forest Green* (Hint: go 3 slides back). The x axis should represent z scores, and the y axis should be blank. Add the title “My Normal Distribution”, add a label to the x axis “ z ” and leave the y axis blank.

My Normal Distribution



Exercise 1 Solution

```
1  my.norm <- rnorm(10000000,0,1)
2  #easiest to use standard normal!
3  #large number makes the curve smooth.
4  y <- hist(my.norm,br=100,plot=FALSE)$counts
5  #dummy x axis.
6  x <- seq(-3,3,length=length(y))
7  x.ticks <- seq(-3,3,1)
8  #Can specify the color in many different ways:
9  #With color name string
10 plot(y~x,type="l",lwd=4,col="forestgreen",yaxt="n",xaxt="
    n",xlab="z",ylab="",main="My Normal Distribution")
11 #With palette code
12 plot(y~x,type="l",lwd=4,col=139,yaxt="n",xaxt="n",xlab="z
    ",ylab="",main="My Normal Distribution")
13 #Or with RGB color content
14 plot(y~x,type="l",lwd=4,col="#228B22",yaxt="n",xaxt="n",
    xlab="z",ylab="",main="My Normal Distribution")
15 axis(1,at=x.ticks,labels=x.ticks)
```


Exercise 1 Solution: Is there a Better Way?

Yes! Instead of using the features of the `hist` object, we can construct the normal distribution directly using the `dnorm` function

```
1 x <- seq(-3,3,by=0.01)
2 plot(dnorm(x)~x,type="l",...)
```

and then we do not need to fudge the x axis. Or even better,

```
1 curve(dnorm(x),...)
```

- 1 Introduction
- 2 Intro to Customizing Graphics
- 3 Overlays and Monitors
 - Example Dataset
 - Using Multiple Plotting Windows
 - Multiple Plots in One Plotting Window
 - Multiple Plots in One Plot Frame
- 4 Other Goodies: Math and Movies
- 5 Conclusion

Data for this Example

These two datasets come from Facebook. Each dataset contains information about users during two time periods: 2007 and 2009.

```
1 group.1 <- read.csv("http://www.stat.ucla.edu
  /~rosario/scc/facebook-2007.csv",
  header=TRUE)
2 group.2 <- read.csv("http://www.stat.ucla.edu
  /~rosario/scc/facebook-2009.csv",
  header=TRUE)
```

Monitors/Plotting Windows

When we execute `plot` (or similar), a new graphics window pops up. If we execute `plot` again, the current graphic is replaced with a new graphic. We can also *open a new window* for the new plot instead, using `dev.new`

We can can specify the `height` and `width` in inches of the new plotting window. This is good when we want to produce several graphics with identical dimensions.

Using Multiple Plotting Windows

```

1 dev.new(height=4,width=4)
2 plot(group.1$Wall.Posts~group.1$Friends,pch='.',
      main="Facebook 2007",xlab="Friends",ylab="Wall
      Posts")
3 dev.new(height=4,width=4)
4 plot(group.2$Wall.Posts~group.2$Friends,pch='.',
      main="Facebook 2009",xlab="Friends",ylab="Wall
      Posts")

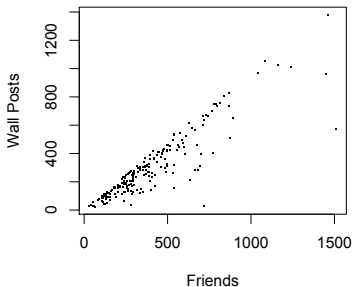
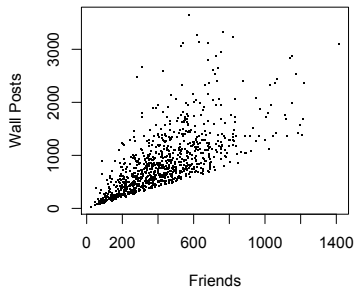
```

Multiple Plots in one Plotting Window

Calling `par(mfrow=c(m,n))` will produce a display containing m rows and n columns, and plots will appear row-wise, from left to right. (`mfcol` is similar except plots appear column-wise, from top to bottom, and then right.)

```
1  par(mfrow=c(1,2))
2  plot(group.1$Wall.Posts~group.1$Friends,pch='.',
      main="Facebook 2007",xlab="Friends",ylab="Wall
      Posts")
3  plot(group.2$Wall.Posts~group.2$Friends,pch='.',
      main="Facebook 2009",xlab="Friends",ylab="Wall
      Posts")
```

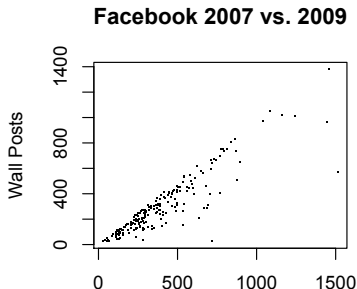
Multiple Plots in one Plotting Window

Facebook 2007**Facebook 2009**

Two Plots for the Price of One

We can also plot both datasets on the same plot. First, construct a plot for the first graphic.

```
1 plot(group.1$Wall.Posts~group.1$Friends ,pch='.',  
      main="Facebook 2007 vs. 2009",xlab="Friends",  
      ylab="Wall Posts")
```



Two Plots for the Price of One

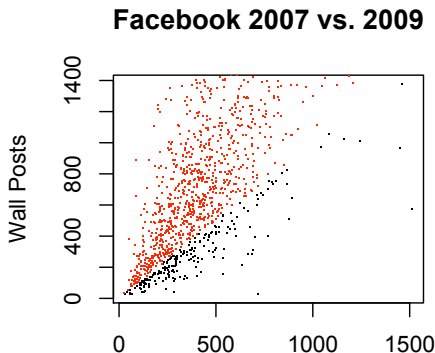
Recall that once a `plot` is constructed, the plot will be replaced if we construct another one. Instead, we need to *overlay* another plot on top of this one. To add more data onto this plot, use the `points` function.

```
points(x, y = NULL, type = "p", ...)
```

`x` and `y` are vectors containing the `x` and `y` coordinates of the values to overlay.

Two Plots for the Price of One

- 1 `plot(group.1$Wall.Posts~group.1$Friends,pch='.',main="Facebook 2007 vs. 2009",xlab="Friends",ylab="Wall Posts")`
- 2 `points(group.2$Wall.Posts~group.2$Friends,pch='.',col="red")`

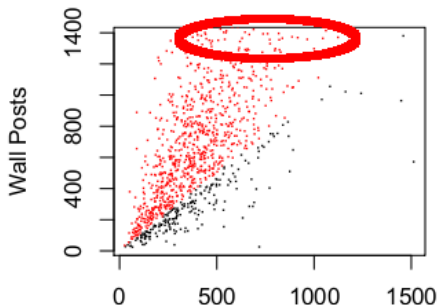


Two Plots for the Price of One

But, there's a problem...

Data points are truncated on the plot, because the new points were laid on top of the *existing* coordinate system...

Facebook 2007 vs. 2009



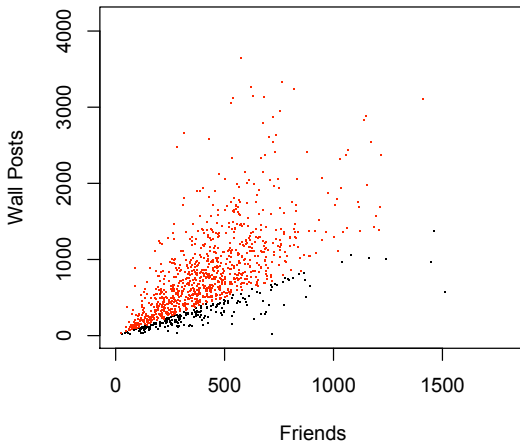
Two Plots for the Price of One

We can fix this problem by expanding the axes using `xlim` and/or `ylim`, using something like the following:

```
1 plot(group.1$Wall.Posts~group.1$Friends,pch='.',
      main="Facebook 2007 vs. 2009",xlab="Friends",
      ylab="Wall Posts",xlim=c(0,max(group.2$Friends)
      +400),ylim=c(0,max(group.2$Wall.Posts)+500))
2 #Set the coordinate system w/r/t the dataset that
  exceeds the bound.
3 points(group.2$Wall.Posts~group.2$Friends,pch='.',
      col="red")
```

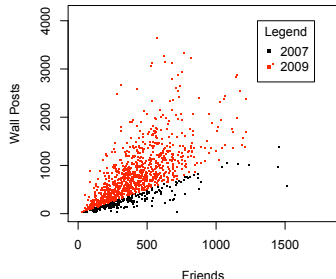
Multiple Plots in One Plot Frame

Facebook 2007 vs. 2009



We can also add a legend to the plot. The first two parameters are the x and y locations of the legend *with respect to the data*. The third parameter is a vector containing the text labels in the legend. `col` indicates the colors of the items in the key, `pch` indicates the character to use for the key and `pt.cex` blows up the points by a factor of 5, so they are visible. `inset` pushes the legend box a bit further into the plot.

Facebook 2007 vs. 2009



```
1 legend(1300, 4000, c(2007, 2009),
      col=c("black", "red"),
      pch=c('.', '.'), inset=1,
      title="Legend", pt.cex=5)
```

Exercise 2

Load in the UCLA textbook price comparison data from <http://www.stat.ucla.edu/~rosario/scc/textbooks.csv>. It is a CSV file with a header. Plot the Amazon list price vs. the Amazon sales price and Amazon list price vs. Barnes & Noble price on the same plot. Use different plotting symbols for Amazon and Barnes and Noble. Add a grey dashed line (Hint: `?abline`) representing the location on the plot where the sales price of a book is equal to the list price. Highlight in red those Amazon books that differ from the Amazon list price by more than 25%. Label the plot and the axes.

Solution Exercise 2

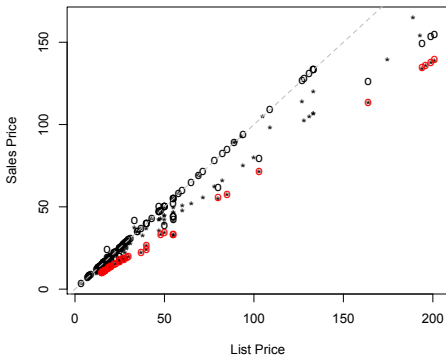
```

1 plot(Amazon.Sale.Price~Amazon.List.Price,pch='*',
      main="Comparison of UCLA Textbook Prices:
      Amazon vs. BN",xlab="List Price",ylab="Sales
      Price")
2 points(Barnes.Noble.Price~Amazon.List.Price,pch='o'
      )
3 abline(a=0,b=1,lty=2,col="grey")
4 good.deals <- which(abs((Amazon.List.Price-Amazon.
      Sale.Price)/Amazon.List.Price) >= 0.25)
5 points(Amazon.Sale.Price[good.deals]~Amazon.List.
      Price[good.deals],pch='o',col="green")

```


Solution Exercise 2

Comparison of UCLA Textbook Prices: Amazon vs. BN



- 1 Introduction
- 2 Intro to Customizing Graphics
- 3 Overlays and Monitors
- 4 Other Goodies: Math and Movies**
 - Math Typesetting in Graphics
 - Writing Plots to Disk
 - Movies
- 5 Conclusion

Integrals

In this section we will take a look at using **math typesetting** in graphics as well as constructing a movie displaying changes over time graphically.

Task: Consider the Riemann integral, or definite integral, of the function $f(\theta) = \cos^3 \theta$ can be defined as

$$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^3 \theta d\theta$$

That is, we can fill attempt to fill the area under the curve with a bunch of rectangles of some width. As the width of these little rectangles goes to zero, we have can fill the area under the curve up to the curve.



The curve Function

First, we need to plot the curve $\cos^3 \theta d\theta$. To do this, we use the curve function. Let's exclude all plotting options for now.

```
1 f <- function(x) { return(cos(x)^3) }
2 curve(f, from=-pi, to=pi, n=10000, lwd=4, col="green",
      xaxt="n", yaxt="n", main="", xlab="", ylab="")
```

curve and plot

curve acts like plot. It generates a **new** plotting window. To overlay a curve on an existing plot, we need to add the parameter `add=TRUE` to the call to curve.

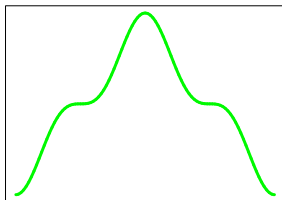
The curve Function

curve has a few different options than plot.

```
curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", ylab = NULL, log = NULL, xlim = NULL, ...)
```

- `expr` is an expression in terms of x , OR, a function f
- `from` is a the minimum value of x to be plotted.
- `to` is the maximum value of x to be plotted.
- `n` is the number of data points to plot, defaults to 101.
- the other options have been covered.

What we Need to Do...



- 1 add a title
- 2 add axes
- 3 add axis labels
- 4 overlay bounds for the definite integral, $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

Math Typesetting in Graphics: Some Functions

Our title will be:

Computing the Integral $\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^3 \theta d\theta$

Enclose *anything* that may contain math text in the `expression` function. To concatenate text with a math object, use the `paste` function.



Math Typesetting in Graphics: Some Functions

The `symbol` function takes its parameter and prints it as a symbol (i.e. `symbol(theta)` displays as θ).

The `plain` function takes its parameter and prints it as standard text.

Some valid expressions are on the next slide, and can be generated using `demo(plotmath)`.



Math Typesetting in Graphics: Options

Arithmetic Operators		Radicals	
$x + y$	$x + y$	\sqrt{x}	\sqrt{x}
$x - y$	$x - y$	$\sqrt[n]{x, y}$	$\sqrt[n]{x}$
$x * y$	xy	Relations	
x/y	x/y	$x == y$	$x = y$
$x \% + \% y$	$x \pm y$	$x \approx y$	$x \approx y$
$x \% / \% y$	$x \div y$	$x < y$	$x < y$
$x \% * \% y$	$x \times y$	$x \leq y$	$x \leq y$
$x \% \times \% y$	$x \cdot y$	$x > y$	$x > y$
$-x$	$-x$	$x \geq y$	$x \geq y$
$+x$	$+x$	$x \sim y$	$x \sim y$
Sub/Superscripts		$x \approx y$	$x \approx y$
$x[i]$	x_i	$x \approx y$	$x \approx y$
x^2	x^2	$x \approx y$	$x \approx y$
Juxtaposition	Typeface		
$x * y$	xy	$\text{plain}(x)$	x
$\text{paste}(x, y, z)$	xyz	$\text{italic}(x)$	x
Lists		$\text{bold}(x)$	\mathbf{x}
$\text{list}(x, y, z)$	x, y, z	$\text{bolditalic}(x)$	\mathbf{x}
		$\text{underline}(x)$	\underline{x}

(a)

Ellipsis		Arrows	
$\text{list}(x[1], \dots, x[n])$	x_1, \dots, x_n	$x \leftarrow y$	$x \leftarrow y$
$x[1] + \dots + x[n]$	$x_1 + \dots + x_n$	$x \rightarrow y$	$x \rightarrow y$
$\text{list}(x[1], \text{cdots}, x[n])$	x_1, \dots, x_n	$x \leftarrow y$	$x \leftarrow y$
$x[1] + \text{ldots} + x[n]$	$x_1 + \dots + x_n$	$x \uparrow y$	$x \uparrow y$
Set Relations		$x \downarrow y$	$x \downarrow y$
$x \subset y$	$x \subset y$	$x \leftrightarrow y$	$x \leftrightarrow y$
$x \subseteq y$	$x \subseteq y$	$x \Rightarrow y$	$x \Rightarrow y$
$x \supseteq y$	$x \supseteq y$	$x \Leftarrow y$	$x \Leftarrow y$
$x \supset y$	$x \supset y$	$x \Updownarrow y$	$x \Updownarrow y$
$x \not\subset y$	$x \not\subset y$	$x \Downarrow y$	$x \Downarrow y$
$x \in y$	$x \in y$	Symbolic Names	
$x \notin y$	$x \notin y$	Alpha - Omega	$\Lambda - \Omega$
Accents		alpha - omega	$\alpha - \omega$
$\text{hat}(x)$	\hat{x}	$\phi/1 + \sigma/1$	$\varphi + \varsigma$
$\text{tild}(x)$	\tilde{x}	Upsilon1	Υ
$\text{ring}(x)$	\dot{x}	infinity	∞
$\text{bar}(xy)$	\overline{xy}	32 * degree	32°
$\text{widehat}(xy)$	\widehat{xy}	60 * minute	$60'$
$\text{widetilde}(xy)$	\widetilde{xy}	30 * second	$30''$

(b)



Math Typesetting in Graphics: Options

Style	
<code>displaystyle(x)</code>	x
<code>textstyle(x)</code>	x
<code>scriptstyle(x)</code>	x
<code>scriptscriptstyle(x)</code>	x
Spacing	
<code>x ~ -y</code>	$x \ y$

<code>x + phantom(0) + y</code>	$x + \ +y$
<code>x + over(1, phantom(0))</code>	$x + \frac{1}{-}$

Fractions	
<code>frac(x, y)</code>	$\frac{x}{y}$
<code>over(x, y)</code>	$\frac{x}{y}$
<code>atop(x, y)</code>	$\frac{x}{y}$

(c)

Big Operators	
<code>sum(x[i], i = 1, n)</code>	$\sum_1^n x_i$
<code>prod(plain(P)(X == x), x)</code>	$\prod_x P(X = x)$
<code>integral(f(x) * dx, a, b)</code>	$\int_a^b f(x) dx$
<code>union(A[i], i = 1, n)</code>	$\bigcup_{i=1}^n A_i$
<code>intersect(A[i], i = 1, n)</code>	$\bigcap_{i=1}^n A_i$
<code>lim(f(x), x %>= 0)</code>	$\lim_{x \rightarrow 0} f(x)$
<code>min(g(x), x >= 0)</code>	$\min_{x \geq 0} g(x)$
<code>inf(S)</code>	$\inf S$
<code>sup(S)</code>	$\sup S$

(d)

Math Typesetting in Graphics: Options

Grouping	
$(x + y) * z$	$(x + y)z$
$x^a y + z$	$x^y + z$
$x^a(y + z)$	$x^{(y+z)}$
$x^a\{y + z\}$	x^{y+z}
<code>group("(", list(a, b), ")")</code>	(a, b)
<code>bgroup("(", atop(x, y), ")")</code>	$\begin{pmatrix} x \\ y \end{pmatrix}$
<code>group(lcell, x, rcell)</code>	$[x]$
<code>group(lfloor, x, rfloor)</code>	$\lfloor x \rfloor$
<code>group(" ", x, " ")</code>	$ x $

(e)

Using the tables from the previous slides, we can produce the plot title, using the main parameter. Here I store the syntax for the title in a variable called `my.title`.

```
1 my.title <- expression(paste("Computing the
  Integral ", integral(plain(cos)^3*symbol(theta)
  *d*symbol(theta), -symbol(pi)/2, symbol(pi)/2))
  )
```

Producing axis labels is easier. I store axis labels in the variables `x.label` and `y.label`.

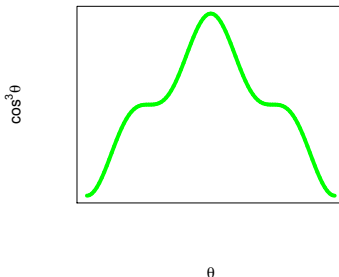
```
1 x.label <- expression(symbol(theta))
2 y.label <- expression(plain(cos)^3*symbol(theta))
```

Now I can pass the variables `my.title`, `x.label`, `y.label` as the values for the `main`, `xlab` and `ylab` parameters respectively.

Now we have

```
1 curve(f, from=-pi, to=pi, n=10000, lwd=4, col="green",
      xaxt="n", yaxt="n", main=my.title, xlab=x.label,
      ylab=y.label)
```

Computing the Integral $\int_{-\pi/2}^{\pi/2} \cos^3 \theta d\theta$



Now, we need to think about adding back the axes, but let's use the common values of θ for trigonometric functions. For this function, the y axis does not have any special values (but if you're ambitious, you can add the more complicated values to the y axis).

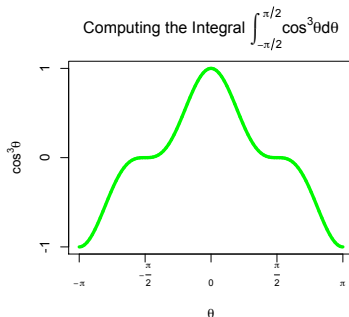
at value	-pi	-pi/2	0	pi/2	pi
labels value	$-\pi$	$-\frac{\pi}{2}$	0	$\frac{\pi}{2}$	π

Let's store the ticks we want to use for the x and y axes:

```
1 x.tick.locations <- seq(-pi,pi,pi/2)
2 x.tick.labels <- c(expression(-pi),expression(-frac
  (symbol(pi),2)),0,expression(frac(symbol(pi),2)
  ),expression(symbol(pi)))
3 #y axis is even easier.
4 y.tick.locations <- c(-1,0,1)
5 y.tick.labels <- y.tick.locations
```

Now, we can add back the axes:

- 1 `axis(1,at=x.tick.locations ,labels=x.tick.labels ,
cex.axis=0.5)`
- 2 `axis(2,at=y.tick.locations ,labels=y.tick.labels ,
cex.axis=0.5)`



Using cex and Friends

If the tick labels are too large for your liking, you can shrink them using the `cex.axis` parameter. The value of this parameter is the percentage of the current object size. To shrink, set `cex.axis` less than 1, and to enlarge, set greater than 1.

Other Variants of `cex`

`cex` controls all text and symbols. `cex.lab` controls axis labels (`xlab` and `ylab`), `cex.main` controls the size of the title of the plot, and `cex.sub` controls the size of the subtitle of the plot.

The segments Function

Using `segments`, we can draw a line segments from a point (x_0, y_0) to another point (x_1, y_1) .

```
segments(x0, y0, x1, y1,...)
```

I added dashed lines in grey for to denote the limits of integration.

```
1 segments(-pi/2, -1, -pi/2, 0, lty=2, col="grey")
2 segments(pi/2, -1, pi/2, 0, lty=2, col="grey")
```

`lty` controls the line type. Note that `segments` is similar to `lines`.



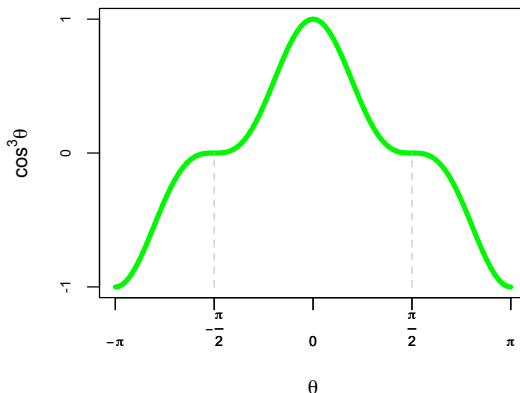
Line Types: lty

The `lty` parameter is a number or string representing the type of line to draw.

- 0, "blank"
- 1, "solid"
- 2, "dashed"
- 3, "dotted"
- 4, "dotdash"
- 5, "longdash"
- 6, "twodash"

The Final Integral Graphic

Computing the Integral $\int_{-\pi/2}^{\pi/2} \cos^3 \theta d\theta$



The Final Integral Graphic

```

1 curve(f, from=-pi, to=pi, n=10000, lwd=4, col="green",
      xaxt="n", yaxt="n", main=my.title, xlab=x.label,
      ylab=y.label)
2 axis(1, at=x.tick.locations, labels=x.tick.labels,
      cex.axis=0.5)
3 axis(2, at=y.tick.locations, labels=y.tick.labels,
      cex.axis=0.5)
4 segments(-pi/2, -1, -pi/2, 0, lty=2, col="grey")
5 segments(pi/2, -1, pi/2, 0, lty=2, col="grey")

```

Integration: The Movie

First, wrap the mega-code for my plot into a function called `my.plot`. It takes a parameter `i` representing the current iteration.

```
1 my.plot <- function(i) {  
2   curve(f, from=-pi, to=pi, n=10000, lwd=4, col="green", xaxt="n"  
         , yaxt="n", main=my.title, xlab=x.label, ylab=y.label)  
3   axis(1, at=x.tick.locations, labels=x.tick.labels, cex.  
         axis=0.5)  
4   axis(2, at=y.tick.locations, labels=y.tick.labels, cex.  
         axis=0.5)  
5   segments(-pi/2, -1, -pi/2, 0, lty=2, col="grey")  
6   segments(pi/2, -1, pi/2, 0, lty=2, col="grey")  
7   e <- (pi/4)*(1/(2**i))  
8   rect(-pi/2+e*seq(0, (2**(i+2))-1), f(-pi), -pi/2+e*seq(1, (2  
         *(i+2))), f(-pi/2+e*seq(1, (2**(i+2))))), col="black")  
9 }
```

Integration: The Movie

Next, I will call this function in a loop...

```

1  setwd(tempdir()) #set working dir to a temp
    directory.
2  for (i in 1:5) {
3    filename <- paste("plot",i,".jpg",sep="")
4    #Create a JPEG with name "filename"
5    jpeg(file=filename)
6    #make a "frame" (one plot)
7    my.plot(i)
8    #turn the device off
9    dev.off()
10 }
```

Instead of jpeg, you can also use png, tiff, gif etc. They all have similar options. pdf is a bit different...

Producing Graphics on Disk rather than Screen

In the previous slide we construct JPEGs for each frame in the movie using the `jpeg` function. We can also print a graphic to disk rather than to the screen. This is commonly done to create PDFs.

```

1 pdf(file="mypdf.pdf",height=7,width=7,onefile=TRUE)
2 #height and width specify size of graphic, in
  inches.
3 #onefile=TRUE - all plots will be in same file
  instead of multiple files.
4 #execute your plotting commands here...
5 plot(1:10,1:10)
6 #turn the plotting "device" off.
7 dev.off()
8 #like closing a file in a programming language.
```

Then, to stitch together the plot frames into a movie, I use the following code from the `rgl` library help.

```
1  make.mov <- function() {
2      unlink("plot.mpg")
3      system("convert -delay 0.25 plot*.jpg
              plot.mpg")
4  }
```

This function deletes (`unlink`) file `plot.mpg` if it exists. Then `system` executes the string passed to it, as if it were typed at the command line.

Caveat

This is only known to work on Unix, Linux and MacOS X systems containing the ImageMagick package as well as the ffmpeg package.



It's Show Time!

We can call `make.mov` to create the movie on disk. Then, open it using the OS.

make.mov()



Alternatives for Movies

My solution is rather primitive. There are packages that can produce animations or movies in R, that may be cross-platform.

- 1 write.gif in package caTools
- 2 animation package on CRAN.
- 3 EBImage package in BioConductor

For more information, check out a related question on **StackOverflow.com**:

<http://stackoverflow.com/questions/1298100/creating-a-movie-from-a-series-of-plots-in-r>

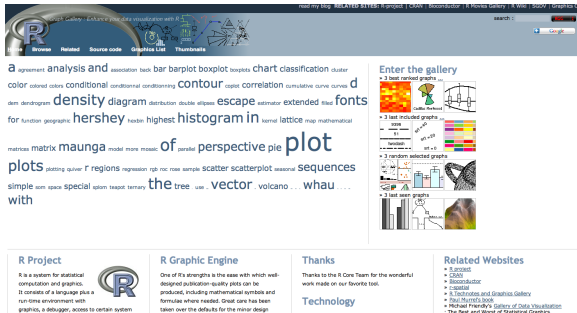
- 1 Introduction
- 2 Intro to Customizing Graphics
- 3 Overlays and Monitors
- 4 Other Goodies: Math and Movies
- 5 Conclusion



Other Resources for R Graphics

R Graph Gallery

<http://addictedtor.free.fr/graphiques/>



The screenshot shows the homepage of the R Graph Gallery. At the top, there's a navigation bar with links like "home", "Browse", "Related", "Source code", "Graphics List", and "Thumbnail". Below this is a large menu of R graphics categories, including agreement, analysis, association, bar, barplot, boxplot, boxplots, chart, classification, cluster, color, colored, colors, conditional, conditional, conditioning, contour, coplot, correlation, cumulative, curve, curves, d, dem, dendrogram, density, diagram, distribution, double, ellipse, escape, estimator, extended, file, fonts, for, function, geographs, hershey, hexbin, highest, histogram, in, kernel, lattice, map, mathematical, matrices, matrix, maunga, model, more, mosaic, of, parallel, perspective, pie, plot, plots, plotting, quiver, r, regions, regression, rgb, roc, rose, sample, scatter, scatterplot, seasonal, sequences, simple, son, space, Special, spm, tmap, ternary, the, tree, use, vector, volcano, whau, with. On the right side, there's a section titled "Enter the gallery" with sub-sections for "3 best related graphs", "3 last included graphs", "3 random selected graphs", and "3 best seen graphs". Each sub-section contains a grid of small thumbnail images of various R plots. At the bottom, there are four columns: "R Project" (describing R as a system for statistical computation and graphics), "R Graphic Engine" (describing the design of R graphics), "Thanks" (acknowledging the R Core Team), and "Related Websites" (listing various R-related resources).

R Project
R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system

R Graphic Engine
One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design

Thanks
Thanks to the R Core Team for the wonderful work made on our favorite tool.

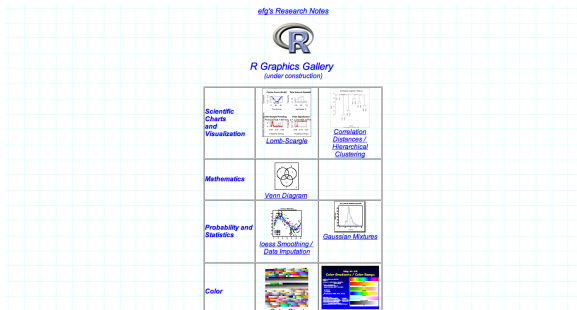
Technology

Related Websites
• R project
• CRAN
• Rdocumentation
• R techniques and Graphics Gallery
• R4u Rmetrics book
• Michael Friendly's Gallery of Data Visualization
• The Best and Worst of Statistical Graphics.

Other Resources for R Graphics

R Graphics Gallery

<http://research.stowers-institute.org/efg/R/>



Other Resources for R Graphics

Statistics with R

http://zoonek2.free.fr/UNIX/48_R/all.html



Warning

Here are the notes I took while discovering and using the statistical environment R. However, I do not claim any competence in the domains I tackle; I hope you will find those notes useful, but keep your eyes open – errors and bad advice are still lurking in those pages...

Should you want it, I have prepared a quick-and-dirty [PDF version](#) of this document.

The old, French version is still available, [in HTML](#), or as [a single file](#).

You may also want [all the code](#) in this document.

1. [Introduction to R](#)
2. [Programming in R](#)
3. [From Data to Graphics](#)
4. [Customizing graphics](#)
5. [Factorial methods: Around Principal Component Analysis \(PCA\)](#)
6. [Clustering](#)
7. [Probability Distributions](#)
8. [Estimators and Statistical Tests](#)
9. [Regression](#)

Other Resources for R Graphics

StackOverflow

<http://www.stackoverflow.com>

The screenshot shows the Stack Overflow homepage. At the top, there are navigation links: Questions, Tags, Users, Badges, and Unanswered. Below these, there's a section for 'Recent Questions' with a filter set to 'active'. The questions listed include:

- What is “ and ” equivalent in Winforms? (4s ago, 7,464 votes)
- Suggestions for WPF & WCF projects (8s ago, 1,102 votes)
- Asp.net - Creating 2 Linked Drop Down Lists from Database (12s ago, 846 votes)
- How to put the data in mysql if the file in pdf format using php? (22s ago, 9 votes)
- LINQ many to many Left Join Grouping (26s ago, 2,735 votes)
- What source/version control to use for home? (40s ago, 4,303 votes)
- Setting a <input> element's display text to "dimmed" (53s ago, 11.6k votes)
- Policy with catching std::bad_alloc (1m ago, 3,151 votes)
- Objective-C Bonjour/TCP Stack (1m ago, 374 votes)

On the right side, there's a section for 'Interesting Tags' with tags like machine-learning, natural-language, python, c++, twitter, hadoop, and r. Below that is a 'Got Server Questions?' section and a 'Try serverfault' link. At the bottom right, there's a 'Wanted: Software Development Engineer/User Interface - EC2 at Amazon Web Services (Ashburn, VA)' listing. The footer of the page shows 'Recent Tags'.

Thank you for your attention!

