The R Project for Statistical Computing

Philip Bjorge (philipbjorge@gmail.com)

ggplot2 Plots

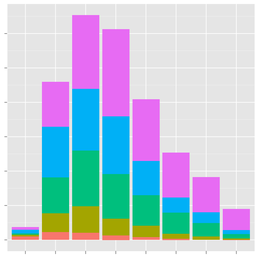

Figure 1: A simple stacked bar chart.

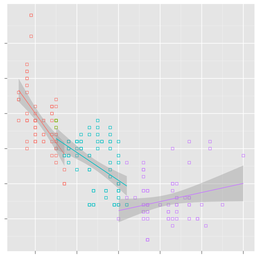

Figure 2: Linear regression lines over groups of points.

High Level Overview

Beautiful Graphics
Visualize data and save your results in vector, raster, and web formats. Even animations are possible through external packages.

Data Reshaping
Has specialized data structures and syntactic sugar reminiscient of APL and Fortran for data manipulation.

New Perspective
Exploratory graphics are simple to make, encouraging tinkering. The terminal interface encourages step-by-step statistical exploration.

Prolific Community
A single R statement gives access to a repository of over 2000 R packages (package count has seen an exponential growth rate for the last 10 years).

Industry Standard
R is being used by the New York Times, Google, Pfizer, and the Fred Hutchinson Cancer Research Center.

Bleeding Edge
R is the tool of choice for academic stasticians, which means the latest techniques tend to be seen in R packages first.

Cross-Platform
R is being developed for the Unix-like, Windows and Mac families of operating systems.

Open-Source
Thousands of dollars cheaper than comparable statistical packages. A GNU Project released with the GPL V2 license.

Visualize Anything

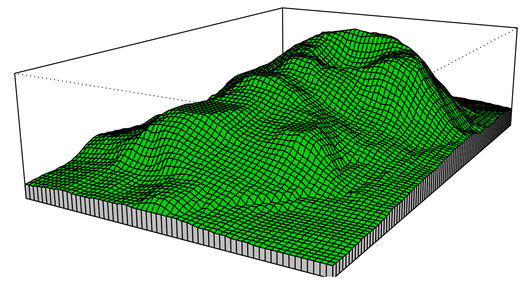

Figure 4: Built-in voxel rendering capabilities.

Technical Overview A

Problem Domain
Statistics, Data Mining and Visualization.

Functional
R is a functional language in spirit. However it does nothing to prevent external side effects, external influences, or assignments.

How Functional?
R allows direct access to parsed expressions and functions. R allows you to alter and subsequently execute them, or create new functions from scratch. The R engine is very Lisp-like.

Scripting
R is a scripting language with over 2000 libraries, the ability to import a huge variety of data sources, and interface with standalone and shared library code.

Scoping
Lexical (R is functional and in some cases it uses dynamic scoping rules).

Parameter Passing
Has "Promise" objects. These have 3 slots: a value, an expression, and an environment (a reference to the caller's environment for when we evaluate the parameter). When a parameter is accessed, the stored expression is evaluated in the stored environment and the result is returned.

References
Technically no. However, R is not a language of absolutes and references have been implemented in libraries using environments.

Visualizing Facebook

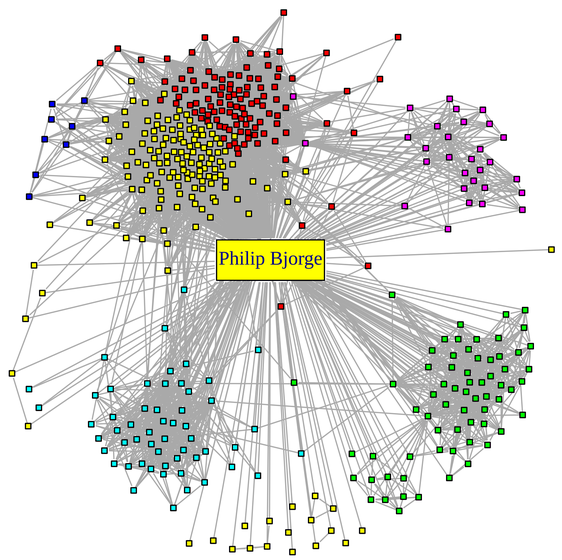

Figure 3: Used the Fruchterman-Reingold layout algorithm. Communities were colored using the fast greedy modularity optimization algorithm.

The R Code

```r
# require("igraph") require("RCurl") require("rjson")
facebook <- function( path = "me", access_token, options){
    if( !missing(options) ){
        options <- sprintf("?%s", paste( names(options),
                                "=", unlist(options),
                                collapse = "&", sep = "" ) )
    } else {
        options <- ""
    }
    data <- getURL( sprintf("https://g.facebook.com/%s%s&access_token=%s",
                            path, options, access_token) )
    fromJSON(data)
}

# scrape our friends list
friends <- facebook(path="me/friends", access_token=access_token)
friends.id <- sapply(friends$data, function(x) x$id)

# Build our adjacency matrix
N <- length(friends.id)
friendship.matrix <- matrix(0,N,N)
for (i in 1:N) {
  tmp <- facebook( path=paste("me/mutualfriends", friends.id[i], sep="/") )
  mutualfriends <- sapply(tmp$data, function(x) x$id)
  friendship.matrix[i,friends.id %in% mutualfriends] <- 1
}

# Create our graph and change the drawing properties
G <- graph.adjacency(friendship.matrix, mode="undirected", diag=FALSE)
V(G)$size <- 2
E(G)$width <- 1

# Running a greedy community finding algorithm to color
fc <- fastgreedy.community(G)
com <- community.to.membership(G,fc$merges,steps= which.max(fc$modularity)-1)
V(G)$color <- com$membership+1
G$layout <- layout.fruchterman.reingold

plot(G)
# Access token, styling, and some strings reduced or removed.
```

Technical Overview B

OOP
Everything is an object including control structures (these are functions, which are objects). OOP is implemented with generic functions which carry tags to the objects they work on.

Whitespace
Whitespace characters aren't technically considered tokens, but they can be used in cases of ambiguity.

Concurrency
R supports a variety of concurrent programming models through its package system (CUDA, MPI, parallel map-style functions, beowulf and other clusters).

NULL
R has a single NULL object (to which all instances refer) that is only used to indicate when an object is absent. R uses NA objects to refer to missing values in a statistical sense and NaN objects for *not* numbers.

Data Frames
Data frame objects are tabular structures with column names and rows as data entries. A data frame is like a list where the components are columns of a data table.

Useful Syntax

```r
best_friends <- fb[fb$posts_ct > 10]
likes_desc <- fb[order(fb[,"likes_ct"])]
zero_row[12,] <- 0     # row 12 all set to 0
L <- positive_df < 0   # Logical is vector[T/F]
positive_df[L] <- 0    # set all -n to 0
```

Listing 1: Data Frames

```r
x <- "hi"
```

Listing 2: Sub caption

References

Facebook JSON function -

http://romainfrancois.blog.free.fr

Figure 4 -

http://addictedtor.free.fr/graphiques/sources/source_25.R

R Language Spec -

http://cran.r-project.org/doc/manuals/R-lang.html