UNIVERSITÀ DEGLI STUDI DI CAGLIARI

FACOLTÀ DI STUDI UMANISTICI

CORSO DI LAUREA IN SCIENZE E TECNICHE PSICOLOGICHE

L'AMBIENTE INTERATTIVO R PER L'ANALISI STATISTICA

UNA PROSPETTIVA PSICO-SOCIALE SULLA STORIA E LA PROGETTAZIONE DI UN SOFTWARE LIBERO

Relatore:
Dott. Gianmarco Altoè

Tesi di Laurea:
Francesco Cabiddu

Anno Accademico 2011 - 2012

INDICE

Sommario

Il presente lavoro di tesi è proposto partendo da una riflessione di fondo riguardante l'influenza dell'ideologia di mercato sul processo di innovazione scientifica e tecnologica. Viene affrontata un'analisi del successo di R (ambiente/software libero per l'analisi statistica dei dati) in ambito prevalentemente accademico e di ricerca, che non potrà prescindere dalla considerazione dei paradigmi etici che hanno gettato le basi per un lavoro comunitario giustamente osannato dalla psicologia sociale.

Verrà rimarcata spesse volte l'importanza delle scelte fondazionali adottate in primis dal movimento sociale del Free Software, per rendere conto del passaggio dal "perché" al "come" di un fenomeno sociale, culturale e organizzativo che ha trovato terreno fertile in molti settori di ricerca e di lavoro. A questo proposito è necessario sottolineare come l'intervento di una prospettiva Psico-Sociale – che funga da lente di ingrandimento per l'osservazione delle vicende intorno al progetto R - dia l'opportunità di prendere coscienza di quei processi organizzativi che, diventando maggiormente gestibili, potrebbero tradursi agevolmente in un incremento delle funzionalità produttive.

Inoltre, il tema della proprietà intellettuale - inscindibile dalle dinamiche relative ai modelli di sviluppo collaborativo che presenteremo (Free Software e Open Source) - permetterà di approfondire in termini operativi le conseguenze di un ideologia utilitarista che conserva, fingendo di ignorarle, contraddizioni di varia natura.

Il primo capitolo fornisce un'introduzione storica al movimento sociale del Free Software, punto di partenza per l'esplicitazione di un modello di sviluppo organizzativo (pertecipativo) retto dal pilastro legale del Copyleft. Infine, viene proposto un chiarimento circa le differenze sia teoriche che pratiche tra il movimento sociale del Free Software e la strategia economica dell'Open Source.

Il secondo capitolo, entra nel merito di un caso specifico di progetto basato sul modello di sviluppo del Free Software/Open Source, The R Project for statistical computing, analizzando i fattori che alimentano e ostacolano la partecipazione comunitaria e la produttivà da un punto di vista socio-organizzativo.

Il terzo capitolo introduce le funzionalità generali dell'ambiente/software R per l'analisi statistica dei dati, chiarendone le possibilità di utilizzo nella ricerca psicologica. Inoltre, viene brevemente trattata una recente integrazione GUI per R che ha suscitato l'interesse di numerosi membri appartenenti alla comunità di ricerca statistica.

Capitolo 1

Il movimento sociale del Free Software

"Distribuisci presto. Distribuisci spesso. E presta ascolto agli utenti".
(Raymond, The Cathedral and the Bazaar, 2000)

Nella prima sezione del presente capitolo si intende discutere il modello di sviluppo del Free Software alla luce delle vicende storiche che ne hanno anticipato il successo e delineato gli aspetti paradigmatici. Successivamente verrà proposta una disamina sugli aspetti giuridici in tema di copyright che sostengono e proteggono il lavoro in comunità, mettendo l'accento sui fattori benefici per il processo di innovazione tecnologica. Infine si concluderà presentando le dinamiche di sostenibilità economica associate al modello di sviluppo aperto, prendendo in considerazione l'attuale divisione fra movimento sociale del Free Software e strategia commerciale Open Source.

1.1. Cenni storici sulla nascita del software libero

Il significato del concetto di Free Software è andato costruendosi in un contesto prevalentemente sociale. Risulterebbe difficile e sicuramente poco chiaro delinearne le principali vicende storiche eludendo le motivazioni e l'etica che hanno mosso le azioni dei protagonisti. L'innovazione generata dalla reciprocità e dalla ridistribuzione dei software si scontrerà agli inizi degli anni Settanta con il mercato e l'organizzazione d'impresa, con i diritti di proprietà, la segretezza. Questa contesa darà vita a quella pratica progettuale-conflittuale nota come Free Software movement (Ceri, prefazione in Berra & Meo, 2000).

Negli anni Sessanta, élite accademiche appartenenti a prestigiose istituzioni come il Massachusetts Institute of Tecnology (MIT) e le università di Stanford e Berkeley esplorano i sistemi informatici spinti unicamente da curiosità intellettuale e ottimismo scientifico. È l'inizio di una cultura antiburocratica di ricerca spontanea e creativa, volta a testare i limiti delle macchine e capirne a fondo il funzionamento. Il termine hacker, che negli anni Ottanta subirà una connotazione negativa, nasce all'interno dei laboratori del MIT per identificare un individuo che ama sperimentare e trovare soluzioni ai problemi seguendo vie inusuali e brillanti.

I primi esperimenti di hacking iniziano all'interno del Tech Model Railroad Club in Cambridge, con sperimentazioni sul Txo (primo computer a transistor) e successivamente con il Pdp-1 più interattivo, scambiando idee con sistemi informatici di Time Sharing che posero le basi per la nascita di Internet. All'interno di questo club di appassionati di tecnologia vengono creati programmi che facilitano l'uso delle macchine, i cui codici sorgenti sono resi accessibili e modificabili. In questo ambiente i giochi cooperativi e le relazioni non formali sono sostenuti dall'intima convinzione che l'informazione debba essere libera e l'accesso ad essa il più esteso possibile (Levy, 2002), senza restrizioni burocratiche o pianificazioni che ne possano influenzare negativamente la spontaneità. La spinta ad adattare i progetti alle proprie esigenze personali tipica di questo gruppo è riconducibile a quanto successivamente Eric S. Raymond (2000) identificherà come primo fattore interveniente nel successo del software libero: "ogni buon lavoro software inizia dalla frenesia personale di uno sviluppatore". Questo processo di costruzione guidato esclusivamente dalla necessità personale si traduce in una ricerca continua di punti di svolta anche quando non apportano un diretto vantaggio economico, differenziandosi da un mondo imprenditoriale con una tradizione che privilegia la cautela, la routine, gli adattamenti a dispetto dell'innovazione (Berra & Meo, 2000). Inoltre, lo stesso Raymond riterrà essenziale ai fini di uno sviluppo software efficace la possibilità di riscrivere sorgenti già esistenti, facilitando la gestione di una serie di difficoltà nella programmazione. Pur non limitandoci a considerare la libera accessibilità alle idee come unico fattore di un processo creativo, gli esempi che verranno presentati, come i casi di Richard Marshall Stallman e Linus Torvalds, partiranno da soluzioni parziali che faranno da impalcature per progetti innovativi.

Lo scambio collaborativo alla base del paradigma di sviluppo aperto non può prescindere dagli eventi che portarono alla nascita della rete Internet, influenzando radicalmente il modo di concepire la comunicazione.

Negli anni che vanno dal 1961 al 1969, gli Stati Uniti d'America sono impegnati in un confronto tecnologico con la Russia che porta al costituirsi, per volontà del Dipartimento della Difesa americano, dell'agenzia ARPA (Advanced Research Project Agency), un'organizzazione scientifica che ha come obiettivo principale quello di risolvere le difficoltà di comunicazione tra reparti militari, creando un sistema di comunicazione decentrato per difendersi da eventuali tentativi di spionaggio e di attacco nucleare.

Paul Baran, della RAND corporation, nella sua pubblicazione “On distributed communications networks” (1964) concepisce una rete di comunicazione senza un centro di controllo in cui i nodi della rete sono indipendenti l’uno dall’altro. Lo strumento utilizzato è la commutazione di pacchetto: i messaggi vengono divisi in pacchetti di uguali dimensioni e ogni pacchetto può seguire lungo i nodi della rete il percorso più conveniente per poi andare a ricostruire il messaggio originale una volta giunto a destinazione (Todon, 2004). Una serie di sperimentazioni dà vita alla rete ARPANET, che nel Dicembre del 1969 permette alle quattro università della California, di Santa Barbara, di Stanford e dello Utah di stabilire un collegamento. Fin da questa prima fase dello sviluppo della rete molti programmatori studiano sistemi per migliorare la comunicazione ideando programmi di posta elettronica con possibilità di scambiare messaggi in privato o in gruppo, dando vita a vere e proprie comunità con interessi sociali e ricreativi, discostandosi dai temi principali di ricerca informatico-militare. Negli anni successivi il numero di paesi e utenti che si uniscono alla rete cresce fortemente, fino ad arrivare a circa 200 nodi nel 1982. Le vicende posteriori vedranno tra gli elementi più significativi la sostituzione del protocollo NCP (Network Control Protocol) con il TCP/IP più sofisticato e adatto alle numerose tipologie di connessione, l’ideazione del noto meccanismo del “World Wide Web” e l’approfondimento di varie interfacce grafiche che miglioreranno la fruibilità della rete contribuendo in buona parte alla sua attuale espansione.

Sterling (1993), descrivendo la dinamicità di questo nuovo sistema di comunicazione come un “raro esempio di reale, moderna e funzionale anarchia”, o ancora come “un’istituzione che resiste alla istituzionalizzazione, che appartiene a tutti e a nessuno”, sottolinea un aspetto che per E. Raymond (2000) può essere ritenuto alla base del modello di sviluppo del software libero, con la sua peculiare organizzazione a “bazaar”, in cui all’apparente confusione per la miriade di progetti che si evolvono in contemporanea si affianca la volenterosa attenzione di un numero elevato di co-sviluppatori, traducendosi in una migliore creatività e risoluzione dei problemi.

Il quadro storico di riferimento che farà da premessa per la nascita del Free Software movement comprende altri eventi importanti e contemporanei ai lavori dell’ARPA e del MIT.

Nello stesso anno in cui vede la luce ARPANET, nei Bell laboratories dell’AT&T nasce Unix, un sistema operativo frutto del lavoro di Ken Thompson e Dannie Richie.

Essendo interamente costruito in un nuovo linguaggio chiamato “C”, Unix permette l’installazione in diverse tipologie di macchine presentando la stessa interfaccia e le stesse funzionalità. Le implicazioni di ciò appaiono enormi, poiché “gli utenti non avrebbero mai più dovuto pagare per nuovi software appositamente progettati ogni volta che una macchina fosse diventata obsoleta. Gli hacker erano in grado di utilizzare gli stessi strumenti software da una macchina all’altra, piuttosto che dover reinventare l’equivalente di fuoco e ruota ogni volta” (Raymond, 1999). La portabilità non è l’unico punto fondamentale del sistema, che prevede persino una rete di qualità e velocità modesta con un proprio protocollo, l’UUCP (Unix to Unix Copy Protocol). Da questa rete ideata da studenti appartenenti ad ambienti esclusi da ARPANET nasce Usenet, un sistema di teleconferenze per Unix che mette in contatto la comunità di utilizzatori e ne favorisce la crescita. Il potere politico derivato dalle conoscenze tecniche, unito al metodo della collaborazione peripatetica, fornisce la libertà necessaria a uno sviluppo dal ritmo incalzante, crescendo nel brodo di coltura di un sistema operativo aperto, Unix, vivificato da uno strumento appena nato: Internet (Ippolita, 2005).

Paradossalmente, da una situazione di scambio e libera circolazione di saperi, le vicende successive porteranno ad una progressiva chiusura dei codici e all’aumento della speculazione e commercializzazione dei software. È la stessa società telefonica AT&T che inizialmente rende inaccessibile il codice di Unix distribuendolo esclusivamente compilato, innalzando i costi delle licenze e impedendo la pratica delle patch. Agli inizi degli anni Ottanta sono presenti sul mercato varianti di Unix distribuite da diversi produttori. Gli ambiti accademici sono sempre più legati a partner commerciali che reclutano coder facendo sottoscrivere accordi di non divulgazione per controllare lo sviluppo dei programmi. La nascita e diffusione dei personal computer, sui quali Unix risulta di difficile implementazione, e la fortuna dei neonati sistemi operativi chiusi DOS e Mac favoriscono l’impoverimento di sorgenti accessibili per il lavoro degli hacker e conseguentemente l’affievolirsi dello spirito di comunità e collaborazione.

È in questo contesto che Richard Marshall Stallman inizia la sua battaglia politica per la libertà in campo digitale fondando nel 1985 la Free Software Foundation (FSF; http://www.fsf.org/), un’organizzazione no-profit per lo sviluppo e la distribuzione del software libero. Dopo aver abbandonato nel 1984 il laboratorio di Intelligenza artificiale del MIT ed aver assistito al lento disgregarsi della comunità hacker, Stallman avvia il progetto GNU (http://www.gnu.org/home.it.html), volto alla

scrittura di un codice per un sistema operativo completamente libero basato su Unix. Nonostante la sua incredibile competenza tecnica, che lo porta a sviluppare in solitudine primi software applicativi di notevole complessità come il compilatore GCC, e prima di questo l'editor di testo Emacs, la svolta politica più significativa, frutto di un lavoro di riflessione sul tema del copyright, riguarda l'ideazione di un sistema legale di licenza che prende il nome di GNU General Public License – di cui si chiariranno i punti essenziali più avanti – che permette di rilasciare i programmi GNU garantendone la totale accessibilità e modificabilità a patto che l'utente distribuisca ogni eventuale modifica alle stesse condizioni. La chiarezza di espressione del codice intellettuale – corrispondente ai valori dell'etica Hacker – riprodotto in una forma di codice legale, permette in questo frangente la protezione delle relazioni cooperative alla base dei progetti, nonché la possibilità di preservare il potenziale valore intrinseco degli stessi eliminando lo spauracchio della non condivisibilità.

Agli inizi degli anni Novanta sono presenti oramai svariati applicativi per il sistema operativo. Tuttavia la comunità intorno al progetto GNU incontra ancora delle difficoltà nella costruzione di un kernel – il programma di controllo centrale di tutti i sistemi Unix capace di determinare quali periferiche e applicazioni debbano avere accesso al microprocessore e quando – che renderebbe l'opera completamente funzionante.

Il definitivo successo del movimento prende avvio in seguito al lavoro di Linus Torvalds, che inizia a sviluppare il kernel Linux nell'agosto 1991 in seguito all'incontro con Richard Stallman al politecnico di Helsinki. Rinnovando l'architettura di Minix, una versione leggera di Unix sviluppata a scopo didattico dal professore universitario olandese Andrew Tannenbaum, Torvalds crea un kernel che permette di far girare un sistema simile a Unix su un PC. In due anni riesce a formare una comunità molto attiva sfruttando Internet e il rilascio continuo dei codici prima ancora che il progetto sia terminato. Questo scambio apparentemente caotico di informazioni, che mette in relazione parecchi individui permettendo il contributo tramite feedback di funzionamento anche agli utilizzatori, permette di testare efficacemente il codice e farlo evolvere rapidamente. Eric S. Raymond (2000) commenterà così questo sistema progettuale: "Credo che l'opera di hacking più sagace e coerente di Linus non sia stata la costruzione del kernel in sé, quanto piuttosto l'invenzione di quel nuovo modello di sviluppo". Nella sua opera "The Cathedral and the Bazaar", Raymond mette a confronto i metodi di costruzione di software utilizzati rispettivamente da Stallman e da Torvalds: i programmi GNU sembravano "cattedrali", monumenti hacker pianificati in modo

centralizzato, costruiti per durare nel tempo; Linux, d'altra parte, era più simile a un grande "bazaar" caotico, un programma sviluppato grazie alle dinamiche decentrate offerte da Internet (Williams, 2003).

La nascita del sistema operativo GNU/Linux è forse l'evento significativo che dà inizio e reale influenza ad un movimento che può essere definito un caso emblematico di un problema più ampio che riguarda la proprietà intellettuale pubblica e privata.

In questa sezione si è voluto mettere l'accento sui fattori portanti di un modello di sviluppo e di costruzione del sapere che in seguito verranno ripresi al fine di comprenderne gli effetti attraverso l'analisi di un caso specifico.

In base al quadro storico presentato, è possibile costruire una tabella riassuntiva (vedi tabella 1.1), allo scopo di facilitare la comprensione della sezione.

Tabella 1.1

Eventi storici significativi per la nascita del movimento del software libero

Periodo	Evento	Luogo
1960-1970	Nascita e sviluppo dell'etica e cultura Hacker	MIT – Stanford - Berkeley - Carnegie Mellon
1969	Arpanet va online mettendo in comunicazione quattro centri di ricerca	California/Santa Barbara/Standford/Utah
1969	K. Thomson e D. Richie sviluppano il linguaggio C e il S.O. Unix	Bell Laboratories
1981	Sentenza Corte Suprema USA che sancisce la brevettualità dei software (da forme espressive a invenzioni)	
1985	R. Stallman fonda la Free Software Foundation	
1991	L. Torvalds sviluppa il kernel linux	

1.2. Copyleft: GNU – General Public License

Dalla prima Rivoluzione Industriale in poi il concetto di proprietà privata sulle invenzioni tecnologiche o artistiche è divenuto strategico per l'attività economica. Strumenti di protezione delle idee come copyright, brevetti, marchi registrati e accordi di segretezza, comunemente associati a valori come la tutela e la migliore diffusione delle conoscenze, sono diventati i nuovi idoli di un'economia che realisticamente ha portato scarsi benefici ai settori di ricerca. Per citare degli esempi, negli Usa gli investimenti in ricerca e sviluppo sono diminuiti (come in quasi tutti i paesi ad alto tasso di innovazione tecnologica) negli anni Novanta, mentre il numero di brevetti è cresciuto ininterrottamente dalla metà degli anni Ottanta in poi (Gruppo Laser, 2005; Kortum &

Lerner, 1998; Cohen & Goto, 2002; Lanjouw & Lerner, 1997; Moser, 2003; Jaffe, 2000). “Altrettanto inutile, sul piano dello stimolo alla ricerca, sembra essere stata la riforma del sistema brevettuale giapponese del 1988, che non ha aumentato significativamente la spesa in innovazione malgrado l’estensione del monopolio attribuito ai detentori dei brevetti” (Gruppo Laser, 2005; Cohen & Goto, 2002; Sakakibara & Branstetter, 2001). Anche in Italia la crescita complessiva del numero dei brevetti è rimasta circoscritta a due periodi storici – dal 1913 al 1929 e dal 1963 al 1983 – senza tuttavia fare da traino a un incremento significativo nella ricerca e sviluppo (sia privati ma soprattutto pubblici) nello stesso periodo storico (Gruppo Laser, 2005; Giannetti, 1988).

Per quanto riguarda il settore dei software, è agli inizi degli anni Ottanta – periodo coincidente con le possibilità di commercializzazione su larga scala, favorite dalla distribuzione di Personal Computer economicamente accessibili alla massa – che gli algoritmi alla base dei programmi sono stati resi brevettabili. Se pur circoscritto al campo informatico, il movimento del Free Software ha creato un’alternativa tecnologicamente competitiva e giuridicamente sostenibile alla privatizzazione della creatività, dimostrando che il progresso può fare a meno della proprietà intellettuale. La General Public License, meglio nota come GPL, è prima di tutto un manifesto politico della Free Software Foundation, che si oppone all’idea secondo cui il progresso tecnologico motore dello sviluppo si arresterebbe se gli inventori non fossero remunerati con il monopolio dei propri risultati (Gruppo Laser, 2005). Il termine Copyleft, scelto ironicamente da R. Stallman come simbolo del rovesciamento del diritto d’autore, rimanda al principale aspetto di genialità che è stato attribuito a quest’opera legale, espresso efficacemente dallo stesso autore: “Gli sviluppatori di software proprietario ricorrono al copyright per rubare agli utenti la propria libertà; noi usiamo il copyright per tutelare quella libertà” (Stallman, 2002). Per rendere dunque un software permanentemente libero è sufficiente dichiararlo sotto copyright per poi riversare le garanzie di libertà per l’utente all’interno della licenza, ribaltando così il ruolo della stessa e creando un vincolo di tipo legale fra la disponibilità del codice e le tre libertà fondamentali: di utilizzo, di modifica, e di ridistribuzione (Aliprandi, 2005). La scelta di ricorrere a questo meccanismo di protezione di diritti nasce dall’impossibilità di divulgare il software ad un livello massimo di libertà (public domain), che rischierebbe di sfociare nella realizzazione di programmi proprietari con la relativa inaccessibilità dei sorgenti. Verranno analizzati di seguito i contenuti più importanti del testo del 1991 (http://www.gnu.org/licenses/gpl-2.0.html), versione di

licenza rilevante ai fini del nostro discorso.

Nel Preambolo, oltre agli aspetti già accennati, vengono introdotte le funzioni e le norme giuridiche più importanti per un corretto utilizzo, legate imprescindibilmente all'intento etico sottostante. Nello spirito del copyleft inteso come "permesso di copia", la Free Software Foundation specifica con una nota iniziale quanto questa prima sezione sia essenziale, e obbligatoriamente riportabile insieme a tutti i commi successivi. Questo punto viene chiarito nella sezione nove specificando che la FSF, detentrice del copyright sulla licenza, sia la sola a poter effettuare eventuali revisioni che "potranno differire nei dettagli al fine di coprire nuovi problemi e nuove situazioni".

La sezione zero fornisce le definizioni di "programma", di "modifica", e di cosa si intenda per "opera basata sul programma". In questo modo i punti successivi potranno essere correttamente intesi alla luce di queste specificazioni. Inoltre in questa parte si definisce l'ambito di utilizzo della licenza circoscritto alle questioni riguardanti l'esecuzione di un programma, la copia, la modifica e la distribuzione.

La sezione uno tratta la possibilità di copiare e distribuire copie letterali di un programma, a condizione che il destinatario prenda visione della licenza e che "venga riprodotta chiaramente su ogni copia una appropriata nota di copyright e di assenza di garanzia". In merito all'interpretazione erronea del termine "free", inteso col significato di gratis e non di libero, viene presentata l'importante possibilità di richiedere un pagamento per la distribuzione del programma o per funzioni di assistenza relative ad esso. Possiamo chiarire meglio questo punto cogliendo una differenza tra "permesso d'autore" e "diritto d'autore": è possibile richiedere un pagamento per la distribuzione, ma non è lecito pretendere che tutti quelli che ricevono il software debbano pagare un prezzo; in questo senso una volta distribuita la copia a pagamento il destinatario potrebbe a sua volta decidere di ridistribuire gratuitamente senza incorrere in violazioni legali.

La necessità di inserire la nota di copyright è ribadita nella sezione due anche in relazione alla modifica e distribuzione di un programma; in questo modo versioni differenti potranno essere adeguatamente distinte preservando la reputazione di ciascun sviluppatore. In realtà questo aspetto ha subìto un'attenuazione, poiché inizialmente prevedeva l'obbligo di sottoporre le modifiche a giudizio dell'autore originale. Tuttavia sono state presto riscontrate notevoli difficoltà di cooperazione per il modello di sviluppo software conseguenti al controllo centralizzato delle informazioni.

La questione all'interno del comma uno, riferita all'assenza di garanzia, è trattata estesamente nelle sezioni undici e dodici: esse rappresentano "la parte più determinante

dal punto di vista del diritto civile in generale, trattandosi di uno ‘scarico di responsabilità’ per coloro che hanno partecipato allo sviluppo e alla distribuzione del software libero” (Aliprandi, 2005). Si sottolinea infatti come “l’intero rischio concernente la qualità e le prestazioni del programma sia a carico dell’acquirente”.

La sezione tre specifica che qualora si proceda alla distribuzione di un programma in base ai primi due commi, è obbligatorio rendere disponibile il codice sorgente secondo determinate condizioni. È proprio la possibilità di modificare un software avendo libero accesso ai sorgenti che differenzia nettamente un free software da un software proprietario. Successivamente viene fornita una definizione di “codice sorgente completo” sia in senso concettuale – “la forma preferenziale usata per modificare un’opera” – che in senso tecnico. È interessante notare come la prima definizione generale permetta di utilizzare la licenza anche per scopi non concernenti i software (come i manuali e la documentazione), purché sia chiaramente identificabile ciò che si vuole intendere come codice sorgente. Tuttavia la FSF consiglia l’utilizzo della GNU Free Documentation License, che affronta il tema in maniera appropriata.

Altri aspetti interessanti trattati nelle rimanenti sezioni riguardano le dinamiche di accettazione implicita della licenza ed eventuali problematiche che potrebbero sorgere nell’applicazione dei termini in particolari regimi legislativi.

La caratteristica significativa di “viralità” è ribadita nell’appendice finale: “I programmi coperti da questa Licenza Pubblica Generica non possono essere incorporati all’interno di programmi non liberi”. Perciò, nel caso in cui uno sviluppatore disponga di una parte di codice coperto da GPL, dovrà assicurarsi di associarlo esclusivamente a codici sotto GPL o sotto un altro tipo di licenza libera previa richiesta all’autore del programma e verifica di compatibilità. Il noto caso delle librerie software tuttavia ha sollevato delle difficoltà portando la FSF a stilare una seconda licenza (Library General Public License), funzionale alle esigenze di programmazione. Lo stesso testo della LGPL fornisce una definizione chiara di “libreria” intendendo “una raccolta di funzioni software e/o dati predisposti in modo da poter essere facilmente collegati con programmi applicativi (che utilizzano alcune di queste funzioni e dati) così da poter formare degli eseguibili”; queste routine di funzioni permettono di risparmiare tempo durante la programmazione. Questa particolare licenza, dando la possibilità di inserire codice libero in pacchetti proprietari, evita di occultare il valore di librerie che sotto GPL non potrebbero essere utilizzate da sviluppatori di software proprietario. Tuttavia la FSF sottolinea che programmi diversi dalle librerie, se venissero distribuiti sotto LGPL, non rientrerebbero più nella categoria di software libero.

L’incompatibilità con progetti proprietari, alla base della lotta per i diritti dell’utente, sarà oggetto di critica da parte del movimento dell’Open Source. Le accuse di inflessibilità e di scarso adattamento al mercato saranno centrali e verranno chiarite in seguito. Il concetto di “Open Source” è stato fino ad ora volutamente ignorato poiché può essere compreso esclusivamente alla luce delle vicende relative al Free Software.

1.3. Modelli di sviluppo a confronto: Free Software e Open Source

Le connotazioni etiche del software sono definite da interazioni fra le modalità di distribuzione, le metodologie di sviluppo e le licenze applicate. Non dipendono dunque dal software in sé, non sono inscritte nel codice, ma sono frutto di un sistema di sinergie (Ippolita, 2005). In quest’ottica, si propone una discussione concernente le implicazioni pratiche relative al Free Software, volta a chiarire ulteriormente il dinamismo del modello. Successivamente verrà presentata la metodologia di sviluppo Open Source, definendone l’ambito di interesse e i punti di divergenza col movimento del software libero.

1.3.1. Pratiche di comunità e strategia economica

Risulterà agevole fornire una definizione precisa del concetto di software libero, avendone precedentemente sviscerato i tratti essenziali. Il termine “libertà” si riferisce al diritto di eseguire, copiare, distribuire, studiare, apportare cambiamenti e migliorie al programma. Oltre ad essere un valore centrale, è fortemente provocatorio e interpretabile come una condanna contro quegli strumenti di abuso che escludono gli individui dal controllo effettivo del software e delle sue funzioni. Come abbiamo visto nel capitolo precedente, ciò si concretizza efficacemente nel momento in cui vengono escogitate delle soluzioni per rendere permanenti e irrevocabili queste libertà per l’utente. I requisiti chiariti dalla FSF per poter definire un programma “libero” danno un forte peso agli scopi di personalizzazione che dovrebbero essere preservati in qualunque caso. Per fare un esempio, come fa notare Alessandro Rubini (1999), utilizzare free software all’interno di un centro di formazione, oltre a non garantire un risparmio significativo, richiede una competenza tecnica maggiore per quanto riguarda la gestione; tuttavia rende possibile investire in sviluppo e ricerca piuttosto che pagare per pubblicizzare le aziende produttrici. La necessità di risorse umane per fronteggiare la richiesta di investimento creativo unita alla flessibilità del modello di sviluppo soddisfa la sempre crescente esigenza di nuove funzionalità spesso estremamente differenti a

seconda del contesto. Prendere in seria considerazione la diversità conduce facilmente alla conclusione che un unico pacchetto software non sia quasi mai in grado di soddisfare tutte le esigenze degli utilizzatori. Come afferma Brian Behlendorf (1999), "quello stesso utente preferirebbe tornare al vecchio modello, cioè essere costretto a fidarsi del proprio fornitore di OS (operative system) proprietario binary-only, una volta che abbia sperimentato la scelta e la libertà del nuovo modello? Difficilmente".

L'abbattimento dei monopoli creativi, generando la diversificazione della domanda, si traduce in un diverso approccio di mercato altamente sostenibile per professionisti autonomi e piccole società che sfruttano i nuovi bisogni di consulenza, assistenza e formazione incentivati dalla minore concorrenza. È interessante notare come questo discorso sia facilmente estendibile agli equilibri del mercato globale, che vede tuttora i paesi in via di sviluppo accettare tacitamente le condizioni imposte per la tutela della proprietà intellettuale al fine di sostenere le economie nazionali, acquisendo di fatto una posizione di schiavitù e dipendenza. Non a caso ultimamente il movimento del software libero sta riscuotendo parecchio successo proprio in quei paesi che disponendo di poco denaro e molte risorse umane trovano una valida alternativa in questa strategia di sviluppo slegata dai meccanismi brevettuali.

Secondo gli autori del manifesto GNU un'autentica libertà di scelta, il codice libero, e lo spostamento dell'attenzione (anche economica) al servizio spingono a diminuire il peso della burocrazia e delle attività non produttive, in primis di marketing (Ippolita, 2005). La ricerca di ergonomia è evidente su più fronti. Un primo esempio si riferisce alla pratica tanto informale quanto consolidata di documentarsi sugli sviluppi recenti di altri progetti al fine di non correre il rischio di sviluppare applicativi o librerie già esistenti; gli sviluppatori, infatti, ritengono più utile dare dei contributi in lavori già avviati, in virtù e in difesa del principio di interdipendenza. Questo procedimento di monitoraggio trova le sue ragioni nei meccanismi di efficienza e risparmio di energie, fondandosi nella consapevolezza di una migliore qualità di produzione derivante dalla comunicazione peripatetica non competitiva. Un altro esempio significativo di questo meccanismo tipico della cultura hacker arriva da alcune raccomandazioni da parte della Free Software Foundation: si richiede di visionare attentamente le licenze già esistenti prima di crearne di nuove, al fine di favorire una certa facilità di consultazione ed ovviare a problemi di dispersione. Inoltre, sempre per ragioni di efficienza, è consigliato cedere alla FSF i diritti d'autore dei lavori che sono inseriti all'interno del progetto GNU, in modo da avere una migliore gestione delle violazioni di copyright e preservare la libertà dei codici.

Nonostante vi siano differenze fra modelli collaborativi messi in atto dalle comunità di sviluppo, una determinata gestione dei ruoli accomuna ognuno di essi. In particolare è risultato fruttuoso mantenere una struttura gerarchica che permetta di gestire i flussi comunicativi quanto basta per preservare l'aspetto di decentralizzazione, fulcro dalla correlazione tra progetti. I legami trasversali sono spesso favoriti da individui che partecipano contemporaneamente a diversi progetti e sono in grado di far viaggiare gli aggiornamenti fra comunità, contribuendo ad arricchire il processo di ricerca. Il caso delle librerie, già citato per le questioni di licenza, è forse l'esempio migliore di questa reticolarità metodologica. "Le librerie sono progetti sviluppati in maniera diffusa da comunità il cui intento è fornire strumenti generici e trasversali per risolvere problemi complessi. Esattamente come un software viene scritto con l'obiettivo di raggiungere più utenti possibile, una libreria è fatta per arrivare a quante più comunità possibile" (Ippolita, 2005).

1.3.2. Open Source come strategia commerciale

Già verso la fine degli anni Ottanta, la commercializzazione del pacchetto Unix da parte della compagnia AT&T vede nascere una variante di licenza d'uso che prevede la possibilità di usare i sorgenti tutelati per sviluppare software proprietari, la Berkeley Software Distribution license (BSD). Questo aspetto di chiusura tradisce fondamentalmente le basi del paradigma del software libero, non assicurando la protezione permanente delle libertà dell'utente. Le ragioni che porteranno al consolidarsi di una tale scelta di flessibilità sono il fulcro stesso del modello Open Source. L'avvicinamento del mondo commerciale alla cultura hacker intacca l'etica alla base del movimento del free software, a favore di una prospettiva totalmente interna alle logiche di mercato. La domanda che ci si pone in questo frangente è la seguente: Qual è la strategia migliore per guadagnare col software libero? Una delle risposte è sicuramente accettare all'interno del proprio progetto open componenti e passaggi proprietari, garantendo una maggiore competitività di mercato. Commentiamo di seguito le vicende che hanno contribuito alla creazione della metodologia di sviluppo Open Source.

Nel 1998 la Netscape dà inizio al progetto di liberazione dei sorgenti del proprio browser internet Navigator, dando dimostrazione di come il lavoro di comunità sia conciliabile con una valida strategia di mercato. La Mozilla Public License (MPL) viene studiata per dare la possibilità di inframezzare il sorgente con codice di natura proprietaria, assicurandosi che le modifiche apportate al software vengano licenziate

sotto MPL per essere nuovamente disponibili al programma. Alcuni personaggi noti, come Tim O'Reilly ed Eric Raymond, avendo attivamente partecipato al progetto, si rendono conto delle potenzialità del fenomeno. L'avvicinamento delle aziende e la ricerca di visibilità inizialmente si scontrano con i problemi di ambiguità del termine "free" e la scomodità dei valori etici in contrasto con le logiche di mercato. Raymond, in "Colonizzare la noosfera" (1998), afferma che la GPL è da considerarsi come uno strumento anziché come un fine ultimo; non tanto un'arma contro i costrittori ma uno strumento per incoraggiare la condivisione del software e la crescita secondo il modello bazaar. Tuttavia, la piena efficacia di questo spostamento di attenzione dal valore di condivisione al valore di produzione non è del tutto confermata dalla realtà dei fatti. In dieci anni di collaborazione la comunità del progetto Mozilla-Firefox ha sviluppato un'amministrazione fortemente verticale, tendendo a sviluppare programmi senza appoggiarsi ad altri progetti di applicativi o librerie. Questo approccio gerarchico di sviluppo è fortemente influenzato dalla ricerca di affidabilità che ha visto il codice del browser rimanere sostanzialmente inalterato, con conseguenti aspetti di lentezza e pesantezza derivati dalla scarsa comunicazione con l'intero sistema comunitario di rete. Mozilla-Firefox diventa in questo modo relegato a funzionalità che possono soddisfare un'utenza medio-bassa, perdendo gli attributi qualitativi correlati all'attività creativa (Ippolita, 2005).

Robert Young (1999) riconosce che il controllo che l'utente acquisisce sul software open, grazie ai suoi vantaggi di flessibilità, stabilità e personalizzabilità può contribuire notevolmente alla creazione di un marchio di successo. L'obiettivo centrale di posizionamento di mercato spinge Young, ed altri promotori dell'Open Source, ad affermare che qualsiasi licenza vada bene purché preservi quelle qualità che favoriscono il profitto. Richard M. Stallman critica fortemente l'oscuramento del valore di libertà a favore della convenienza tecnica, mettendo l'accento sulla fallacia dell'affidabilità per il modello di sviluppo aperto. In effetti, laddove un software proprietario risulti migliore di un software Open Source, non ritenendo centrale la libertà di condivisione ma l'affidabilità, un sostenitore del sorgente aperto non avrà difficoltà ad usufruire della soluzione proprietaria negandosi l'esercizio del diritto di controllo sulla conoscenza e sulla scoperta.

Alla fine del 1998 Raymond e alcuni seguaci che avevano contribuito al progetto Mozilla, dopo aver introdotto il termine "Open Source" eticamente moderato, fondano la Open Source Iniziative (OSI; http://opensource.org/). Il tentativo di istituzionalizzare la nuova metodologia di sviluppo si traduce in un documento che individua dieci criteri

necessari per poter definire un software Open Source, la Open Source Definition (OSD; http://opensource.org/docs/osd).

L'azione di questa corporazione no profit è rivolta principalmente a monitorare le licenze in circolazione per definire quali siano compatibili col modello open, certificando i programmi con un marchio apposito di notevole efficacia grafica (OSI certified). Come afferma Bruce Perens (1999), "la Open Source Definition è una carta dei diritti dell'utente di computer. Definisce certi diritti che una licenza software deve garantire per poter essere certificata come Open Source". Tuttavia, l'enfasi del rispetto del diritto è posta sulle implicazioni relative al raggiungimento di nuovi mercati, alle possibilità di fare prezzi concorrenziali.

La scelta di registrazione di un marchio di certificazione è stata criticata dalla FSF per via dell'aumento delle difficoltà burocratiche derivate da un ulteriore livello di controllo giuridico, ma soprattutto per la confusione concettuale causata da molte aziende che hanno aggirato il sistema di certificazione proponendo i propri programmi come OSD-compatibili senza rientrare di fatto nei criteri OSD.

Per concludere questa introduzione ai modelli di sviluppo aperto possiamo dire che ormai da tempo sono apprezzate le dinamiche di collaborazione reticolare: "è la stessa vecchia legge di Metcalfe, l'ideatore delle reti ethernet, ad affermare che il valore di una rete è proporzionale al quadrato delle persone/nodi che collega. L'Open Source sembra offrire alcune garanzie rilevanti nello sviluppo di reti ad alto valore aggiunto: da una parte consente al software di rimanere in qualche modo un bene "pubblico" (adotta lo sviluppo aperto e si avvale di comunità di supporto); dall'altra mantiene molto bassi quelli che vengono definiti gli switching costs, ovvero i costi derivanti dal passaggio da un sistema all'altro, in particolare dai modelli close a quelli open" (Ippolita, 2005).

Capitolo 2

Introduzione ad R

Ko Maru kai atu
Ko Maru kai mai
Ka ngohe ngohe

"Give as well as take and all will be right".
(Proverbio Maori, tratto da Taylor, Te Ika a Maui, 1855)

Nel secondo capitolo, dopo una breve introduzione riguardante la nascita del Progetto R, si presenterà un quadro generale relativo alle dinamiche organizzative della comunità di R. In particolare, nella seconda sezione verranno discussi gli aspetti motivazionali che orientano i comportamenti partecipativi e stimolano il senso di appartenenza. Infine, verrà proposta una riflessione su una serie di fattori correlati al tema della produttività, accennando a possibili soluzioni per problemi organizzativi attualmente aperti.

2.1. Il progetto R per l'analisi statistica

Gli argomenti affrontati nel capitolo precedente fanno da premessa esplicativa alle dinamiche del progetto R (http://www.r-project.org), uno dei tanti progetti nati sulla scia di un modello di sviluppo telematico scaturito da un'iniziale cultura elitaria e successivamente da un vero e proprio movimento sociale, economico e organizzativo. Nonostante R sia distribuito sotto Licenza Pubblica Generica GNU, l'evoluzione del progetto fa emergere delle contraddizioni in quella apparentemente chiara demarcazione concettuale che divide ciò che possiamo considerare Free Software da ciò che rientra nella categoria Open Source. Inoltre, essendo la comunità di R in continuo mutamento, le considerazioni che verranno proposte in seguito riflettono l'esigenza di addentrarsi nel caso specifico, per tentare di comprendere ancora meglio la cornice teorica e metodologica di riferimento.

Come per molti progetti Open Source, R nasce dalla curiosità e dal bisogno di adattare uno strumento informatico a delle esigenze personali.

Agli inizi degli anni Novanta, Ross Ihaka ha la possibilità di sperimentare dapprima il linguaggio di programmazione Scheme (sviluppato negli anni Settanta da Guy L. Steele e Gerald Jay Sussman), poi S (sviluppato nei Bell Labs da John Chambers e colleghi). I primi esperimenti riguardano i tentativi di ottenere delle proprie variabili impratichendosi con la sintassi di programmazione di questi linguaggi. Entrato all'Università di Auckland come docente, decide di scrivere insieme al collega Robert Gentleman un interprete per Scheme (un programma in grado di eseguire altri programmi a partire direttamente dal relativo codice sorgente), dotato di un'interfaccia basata sulla sintassi di S. Questa scelta sarà importante dal punto di vista della diffusione di R, poiché faciliterà il suo utilizzo al bacino di utenti già abili con il linguaggio S.

I buoni progressi iniziali nello sviluppo del programma, che inizia ad essere utilizzato da Ihaka e Gentleman in laboratori di insegnamento, portano alla divulgazione dei suoi sorgenti nel 1995 sotto GPL. I miglioramenti repentini derivati dai feedback e dalla collaborazione di volontari sono resi possibili grazie al costituirsi di luoghi virtuali di discussione, come la prima mailing list (r-testers) nel 1996, sostituita già un anno dopo da vari newsgroup. Inoltre viene creato un archivio formale per facilitare la distribuzione del programma. La quantità di contributi cresce a tal punto che nel 1997 si rende necessaria l'estensione del gruppo di sviluppatori (R Development Core Team) per poter soddisfare le numerose richieste di modifiche al software. Dagli sviluppatori viene infine istituita The R Foundation for Statistical Computing, un'organizzazione no-profit che diventa un punto di riferimento per individui, istituzioni e imprese commerciali che decidano di interagire con la comunità di sviluppo di R. La fondazione ha come intenti principali la promozione della ricerca in ambito statistico-informatico e l'amministrazione del copyright di R sia per gli aspetti software che per la documentazione. Ulteriori aspetti circa l'evoluzione del progetto R verranno discussi nella prossima sezione.

2.2. Il modello di sviluppo di R: aspetti socio-organizzativi

Come specificato dall'R Development Core Team (2012), è preferibile definire R un "ambiente" integrato e coerente per il calcolo statistico e l'elaborazione di grafici. Alcuni esempi applicativi che giustificano questa scelta interpretativa sono identificabili nelle caratteristiche di flessibilità ed estensibilità del sistema: essendo progettato come linguaggio informatico, permette di aggiungere nuovi strumenti di lavoro costruendo

nuove funzioni. Il sistema dei “packages” incoraggia il feedback, la condivisione, e la personalizzazione dei programmi. Il rispetto della eterogeneità delle esigenze dell’utenza, fattore che peraltro è stato già individuato come decisivo per il successo del software libero, è rispecchiato dagli oltre duemila pacchetti scaricabili dal CRAN (The Comprehensive R Archive Network). I termini della GNU General Public License, sotto i quali è utilizzabile R, preservano e pongono le basi per il lavoro cooperativo in comunità.

L’implicito valore aggiunto attribuito ad R (Ricci, 2004), insito nel termine “ambiente”, facilita già da un punto di vista semantico l’utilizzo di una chiave di lettura interazionista per le questioni relative al suo modello di sviluppo. Una prospettiva psico-sociale appare evidentemente appropriata per far luce sugli aspetti comunemente affrontati in relazione al successo di R e alle sue problematiche. Temi come la presa di decisione, la comunicazione, la motivazione, l’identità, le gerarchie di ruolo e di status diventano centrali per approfondire le modalità di funzionamento dei gruppi di lavoro che formano la comunità intorno al progetto R.

2.2.1. Cooperazione e appartenenza

Una prima questione di carattere generale riguarda le motivazioni che spingono gruppi di volontari a cooperare in progetti open-source come R, in cui non è presente un diretto ritorno economico. Questo tema è intrinsecamente legato alle concezioni fortemente radicate nella cultura occidentale di un’economia capitalista, della proprietà intellettuale, e più estesamente di un’ideologia utilitarista che come afferma Marco Aime (Mauss, 2002) considera il perseguimento del proprio interesse egoistico un elemento centrale affinché la società funzioni. Le evidenze empiriche contraddittorie rispetto a questo paradigma individualista venivano messe in luce già negli studi antropologici di Marcel Mauss (2002), in cui “l’onore, il disinteresse, la solidarietà corporativa non sono in esse una vana parola, né si rivelano in contrasto con le necessità del lavoro”. Ai fini del nostro discorso è interessante notare come nelle culture e nelle economie primitive basate sul dono, il fine dello scambio è quello di stabilire rapporti sociali duraturi. In queste società il meccanismo di indebitamento continuo generato dall’offerta ha lo scopo di mantenere attivo il legame tra le parti. La relazione diventa conseguentemente più importante del bene stesso (Mauss, 2002). Un evidente richiamo di questa prospettiva socio-antropologica è osservabile negli intenti etici e programmatici della Free Software Foundation; lo stesso Stallman, propone un esempio calzante allo scopo di ribadire le differenze fra la concezione del Free Software e quella

dell'Open Source; un sostenitore della FSF, qualora si trovasse davanti ad un programma proprietario più potente e affidabile di qualsiasi altro software libero in circolazione, sarebbe propenso ad affermare: "Il vostro programma è molto attraente, ma non può valere il prezzo della mia libertà, per cui devo farne a meno. Piuttosto darò il mio supporto ad un progetto che sviluppi un'alternativa libera. Se noi consideriamo un valore la nostra libertà, allora agiremo in modo da mantenerla e difenderla". In quest'ottica la libertà è sinonimo di protezione di quei rapporti sociali che permettono la condivisione e la collaborazione.

Robert B. Cialdini (2010) considera il "principio" psicologico della reciprocità come un fattore motivazionale importante che orienta e dirige il comportamento umano spesso in maniera automatica e stereotipata. Il sentimento di contraccambio generato dalla poca rigidità nella regolamentazione dello scambio, come nel caso delle collaborazioni informali che contraddistinguono i progetti open-source, si traduce in schemi pressoché fissi d'azione volti a saldare obbligatoriamente il debito. In proposito una delle risposte fornite dai membri dell'R Core Team per motivare il proprio impegno di condivisione volontaria è la seguente:

> *"Sento di ottenere grandi benefici dal software di tipo open-source. Questo è estremamente importante per me, potendo utilizzare tutti questi strumenti, e sento l'obbligo sia morale che pratico di dover contribuire nuovamente in questo mare di strumenti che sono, a mio avviso, molto importanti per lo sviluppo della nostra professione" (Fox, 2009).*

Cialdini mette in gioco la costituzione di norme sociali rilevanti per commentare il sentimento d'obbligo di sdebitarsi che si verifica anche quando non c'è una richiesta esplicita di restituzione: il sistema sociale umano avrebbe sviluppato questa strategia per permettere agli individui di "prendere l'iniziativa" nelle transazioni senza correre il rischio di non essere contraccambiati. Inoltre possiamo aggiungere che la non regolazione dell'equivalenza determinata dall'informalità degli scambi produce un circolo di dono/controdono che difficilmente si interrompe; come detto in precedenza, questo meccanismo è particolarmente funzionale al mantenimento e consolidamento di alleanze sociali.

Il legame sociale in sé tuttavia non giustifica il successo di R e la repentina crescita della sua comunità. Raymond (1998), analizzando il tipo cultura organizzativa in progetti Open Source, sposta l'attenzione sul fattore reputazione, considerandolo

centrale per la spinta collaborativa. In questo senso ad una struttura di ruolo paritaria si affiancherebbe l'aspetto competitivo intragruppo di ricerca di uno status elevato. A sostegno di questa ipotesi Festinger (1954) con la sua teoria dei confronti sociali, conferma come in numerose culture occidentali si attribuisca valore alla prestazione migliore; per effetto di una irresistibile "pulsione verso l'alto", gli individui troverebbero motivazione nel superare i risultati degli altri. In un quadro maggiormente interattivo l'aspetto fondamentale dello status si identifica col prestigio consensuale, ovvero una valutazione positiva o classificazione positiva da parte degli altri membri del gruppo (Brown, 2000).

La visione di un confronto competitivo intragruppo entra nettamente in contrasto con gli effetti reali di un sistema di lavoro altamente collaborativo, vigente nella maggior parte dei progetti open-source, incluso R. In effetti, il contesto cooperativo oltre ad abbassare notevolmente il bisogno di confronto sociale intragruppo "induce una quota maggiore di coesione, una maggiore divisione del lavoro a cui corrisponde una maggiore coordinazione degli sforzi, una comunicazione più efficace con una maggiore accettazione delle idee altrui, il desiderio di stabilire rapporti basati sul reciproco rispetto e anche una maggiore produttività oltre che un tasso di soddisfazione più elevato" (Serri, 2004).

Tuttavia, spostando l'attenzione da fattore motivazione ad aspetto strutturale, è interessante notare come la reputazione risulti essere un valore predominante nella maggior parte delle culture organizzative cibernetiche. Proponendo una breve digressione che prenderà corpo più avanti, si può dire che le specificità della comunicazione mediata da computer (CMC), influenzino anche le gerarchie di status in quanto le posizioni di leadership vengono sempre ricoperte da coloro che dimostrano di possedere le competenze necessarie al raggiungimento degli obiettivi organizzativi; i riconoscimenti formali di leadership invece (ruoli), decadono spesso e facilmente in virtù del basso grado di influenza normativa esercitata in un contesto in cui l'individuo è in parte de-individuato. Come vedremo, il fatto che un'influenza di tipo informativo sembri verificarsi in misura maggiore durante le interazioni di gruppi virtuali, da un lato contribuisce a mantenere aperte le strade al cambiamento e alla creatività, dall'altro potrebbe rallentare i tentativi di ricerca di consenso per la partecipazione e l'impegno ai progetti. È necessario, a questo punto, passare ad una prospettiva intergruppo che badi alla complessità dei meccanismi di funzionamento sociale intervenienti.

Partendo dall'idea che le affiliazioni abbiano un ruolo essenziale per la costruzione dell'identità personale e sociale, Tajfel e Turner (1986) estendono la teoria

dei confronti sociali di Festinger (1954) considerando la correlazione fra il valore attribuito all'ingroup risultante dal confronto con l'outgroup, e il grado di autostima del soggetto membro. "Secondo Tajfel, infatti, per quanto ricca e complessa possa essere l'immagine che gli individui hanno di se stessi, alcuni aspetti di essa si identificano con l'appartenenza a certi gruppi o a certe categorie sociali attribuendosi le connotazioni sociali che tali gruppi o categorie possiedono" (Bertani, Manetti, 2007). Coerentemente l'identità sociale può essere ritenuta una parte dell'immagine di sé derivante dalla conoscenza della propria appartenenza ad uno o più gruppi, unitamente al valore e al significato emotivo riconosciuti a tale appartenenza (Tajfel, 1978).

La combinazione di vari aspetti strutturali, dei quali si discuterà in seguito, favorendo notevolmente la produttività, potrebbe aver senz'altro contribuito indirettamente ad incrementare il senso di soddisfazione percepito in coloro che decidono di prestare tempo e impegno alla realizzazione del progetto R. Spesso le disamine riguardanti non solo i progetti open-source ma anche nel nostro caso i software statistici si traducono in una serie di confronti di tipo tecnico e ideologico fra movimenti sociali. Non dovrebbe sorprendere la risultante in termini di guadagno identitario associata all'appartenenza ad un progetto innovativo come R. Alla luce di ciò risultano ancora più chiare le implicazioni di una decisione come quella di definire R un "ambiente" piuttosto che un semplice software, muovendo una neanche tanto velata critica nei confronti di un panorama (intergruppi) di pacchetti statistici spesso caratterizzati da funzionalità rigide e inflessibili. Tutta una serie di dichiarazioni da parte dei membri dell'R Core Team lasciano trasparire quel senso di soddisfazione tipico di chi stabilendo delle specificità positive per il proprio gruppo di appartenenza può trarne beneficio "brillando di luce riflessa". Per citare alcuni esempi:

> *"Una delle cose migliori è che altre persone nelle Filippine, in Bolivia o in Messico possano avere un sistema software di livello mondiale quando non avrebbero mai potuto permettersi un sistema software commerciale"(Fox, 2009).*

O ancora:

> *"Si vuole contribuire effettivamente al bene comune. Intendo che è per questo che facciamo questo tipo di lavoro, nella speranza che risulti importante per alcune persone e che possa cambiare la loro situazione" (Fox, 2009).*

Infine discutiamo brevemente alcuni fattori che in un contesto di comunicazione mediata da computer (CMC) aumenterebbero il senso di identificazione personale col il gruppo e il livello di attrazione reciproca fra individui, incentivando la motivazione a collaborare in progetti comuni.

Secondo la Social Identity De-individuation theory (SIDE) di Spears e Lea (1992), gli individui immersi in ambienti telematici potrebbero essere fortemente suscettibili alle norme sociali nel caso in cui i valori del gruppo di appartenenza siano particolarmente salienti per i membri stessi. L'assenza di riferimenti concreti nel contesto relazionale (tono e volume della voce, aspetti non verbali, aspetti contestuali fisici), favorirebbe dinamiche di de-individuazione, dando risalto ad aspetti strutturali come i valori, i rapporti gerarchici, le norme (Bertani & Manetti, 2007). Inoltre, seguendo Reicher (1984), Spears e Lea affermano che l'anonimato visivo, diminuendo la percezione delle differenze e delle caratteristiche personali degli individui membri, potrebbe aumentare l'importanza dell'identità del gruppo nel suo insieme.

A questi aspetti di identificazione e influenza sociale si affianca la possibilità di una maggiore attrazione fra individui, tipica in contesti di comunicazione elettronica. Walther (1996), in linea con la SIDE, propone una serie di componenti che contribuirebbero ad aumentare la coesione intragruppale; di seguito citiamo i più significativi:

- La possibilità di comunicazioni asincrone fornisce più tempo agli interlocutori per negoziare e mantenere coerente l'immagine del sé.
- Durante le interazioni comunicative mediate da computer il ricevente attribuisce un valore più intenso ai segnali inviati dal mittente rispetto agli scambi relazionali faccia a faccia. L'assenza di segnali sociali e il processo di categorizzazione determinano una percezione esagerata di similarità fra i membri del gruppo (Bertani & Manetti, 2007).
- Solitamente l'immagine identitaria in costruzione ottiene feedback positivi che incentivano il proseguimento delle relazioni.

Il secondo fattore può essere chiarito richiamando la definizione di coesione derivata dalla teoria dell'autocategorizzazione di Turner (1987) secondo la quale l'assimilazione delle caratteristiche prototipiche del gruppo interno e il contrasto rispetto ai prototipi esterni costituiscono i processi di base che determinano il comportamento di ogni gruppo. In questo senso per coesione s'intende l'attrazione dei membri all'idea del gruppo, alla sua immagine prototipica condivisa e al modo in cui essa si riflette nelle

caratteristiche e nella condotta del membro tipico; pertanto, un gruppo risulta coeso nella misura in cui i suoi comportamenti si identificano nelle sue caratteristiche e nei suoi ideali distintivi (Hogg, 1992). È possibile che il livellamento delle differenze tra individui lasci emergere perlopiù atteggiamenti e comportamenti interpersonali in linea con i valori organizzativi, diminuendo notevolmente gli scarti fra immagini identitarie dei membri e immagine prototipica. Come vedremo più avanti, saranno proprio le implicazioni derivanti dall'anonimato che verranno prese in considerazione per analizzare alcune difficoltà nel raggiungimento di obiettivi organizzativi da parte delle comunità intorno al progetto R.

2.2.2. Comunicazione, distribuzione dei ruoli e Leadership

La differenziazione dei ruoli in ambito organizzativo risponde direttamente alle esigenze di divisione del lavoro e delle responsabilità fra i componenti del gruppo. Nel corso dello sviluppo del progetto R, in particolare con la costituzione del Core Team avvenuta nel 1997, la struttura dei ruoli è progredita verso un carattere di semi-formalità, mantenendo pur sempre la sua natura situazionale (vedi tabella 2.1). Questo grado di duttilità è sicuramente un elemento di forza, in quanto un modello di distribuzione dei ruoli troppo rigido può ostacolare l'adattamento a nuovi contesti (Gersick & Hackman, 1990).

Fox (2009), tuttavia, definendo spesso confusa e accidentale l'assegnazione dei ruoli nei gruppi di lavoro contribuenti al progetto R, ne ipotizza l'effetto negativo sul modello di sviluppo elencando quegli aspetti che attualmente risultano più problematici: il perseguimento di piani a lungo termine, una negoziazione chiara di obiettivi e la ricerca del consenso all'avvio dei progetti.

Nonostante sia l'intero gruppo che co-costruisce i valori fondamentali e le norme dominanti, questa serie di difficoltà organizzative sono sicuramente attribuibili ad un aspetto essenziale concernente la coordinazione dei disegni progettuali, ovvero lo stile di leadership.

La figura del leader è indispensabile poiché la sua presenza costituisce un punto di riferimento per tutti i membri del team di lavoro, garantendo conseguentemente l'esistenza del gruppo stesso (Serri, 2004).

Il successo di R trova certamente giustificazione nella possibilità di costruire dinamicamente diverse strutture di ruolo e di interazione ogni volta che inizia un nuovo progetto. In effetti, una delle componenti più importanti della leadership, la capacità di influenzare gli altri, è frutto di caratteristiche di personalità che possono diventare

ottimali oppure disfunzionali a seconda della situazione e del particolare tipo di compito che il leader stesso si trova ad affrontare. In questo senso la malleabilità strutturale della quale si è precedentemente discusso è sicuramente utile al "selezionamento" del leader appropriato che viene "creato" di volta in volta (Serri, 2004).

Tuttavia potremmo ipotizzare che gli stili di leadership che tendenzialmente vanno delineandosi nei gruppi di lavoro del progetto R abbiano caratteristiche fra la democraticità e la permissività, ostacolando in parte l'innovazione e la formazione di nuove prospettive di sviluppo.

Seguendo Lewin, la leadership democratica – stile forse più appropriato per il contesto di lavoro telematico – nonostante determini una bassa dipendenza nei confronti del leader, un buon grado di soddisfazione, l'accettazione di punti di vista devianti, bassa aggressività intragruppo e rendimento qualitativamente buono, comporta livelli modesti di produttività in termini quantitativi. Ancor più se lo stile di leadership è sbilanciato sul fronte permissivo – ipotesi interpretativa più appropriata alla struttura di lavoro tipica all'interno della comunità di R – in cui si troverà un'emergenza di alte proposte creative con un rendimento complessivo significativamente basso (Serri, 2004). Proprio quest'ultimo aspetto viene ribadito da Fox (2009), che osserva come spesso molte proposte di sviluppo si interrompano sul nascere, o proseguano grazie al contributo di un ridotta fetta di volontari, accompagnate dall'acquiescenza generale.

Anche B. Bertani e M. Manetti (2007), per ciò che concerne gli ambienti virtuali, ribadiscono gli effetti negativi sulla prestazione del gruppo determinati da uno stile di leadership eccessivamente tollerante. In effetti, l'assenza dei segnali sociali crea la necessità di continui richiami al "dovere" da portare a termine; il leader virtuale dovrebbe adempiere perciò non solo al compito fondamentale di disegnare le strategie organizzative, ma avere anche la responsabilità di "tener le fila" delle attività del gruppo stesso.

Il retaggio culturale proveniente dalle spinte libertarie dell'etica hacker, a favore di un processo di sviluppo totalmente spontaneo, anarchico, privo di ogni elemento organizzativo e burocratico che possa chiudere le strade all'innovazione, sembra procurare delle difficoltà in particolare riguardo alla concentrazione e ottimizzazione degli sforzi durante la negoziazione e la messa in atto di piani di lavoro. Secondo Pier Giorgio Gabassi (2006), la pianificazione ottimale di un metodo di lavoro comporta la rinuncia ad una parte di originalità, di creatività, di identità individuale: "È il metodo infatti, che conforma il gruppo, che lega un membro all'altro, nell'aiuto reciproco, nella condivisione delle responsabilità ma anche nella dipendenza e nel vincolo

omogeneizzante. Esso ordina il lavoro comune ma richiede anche l'accettazione di un pensiero e di un comportamento sovra individuali". Emerge inevitabilmente la necessità di una definizione chiara – seppur non troppo rigida ai fini della sopravvivenza del gruppo – di norme che orientino efficacemente i progetti; dalle aspettative di comportamento si passa conseguentemente alla chiarificazione dei ruoli e degli obiettivi, in un'ottica di co-costruzione incentivata dal supporto di una figura appropriata di leader.

I fattori positivi di leadership associati a gruppi di lavoro virtuali conducono ad un'integrazione fra una leadership di servizio (Quaglino et al., 1992), che comporta esperienza relazionale e intervento finalizzato al successo del gruppo piuttosto che all'espressione delle proprie capacità, ed un tipo di leadership distribuito (Lumsden & Lumsden, 1996), che permette il verificarsi di una rotazione del ruolo di leader a seconda del contesto e delle competenze richieste relative all'obiettivo. Ricercando un compromesso fra soddisfacimento di bisogni soggettivi e scopi comuni attraverso la negoziazione di norme condivise, risulta particolarmente importante in contesti informatici la chiarificazione degli obiettivi organizzativi, che può avvenire esclusivamente tramite una buona definizione dei ruoli mediata dal supporto del leader.

La definizione chiara di obiettivi, la costruzione funzionale di aspettative di pensiero, emozione e comportamento, unite alla mediazione efficace del leader emergono e possono essere considerati esclusivamente alla luce della qualità degli scambi comunicativi, che pongono in essere gli aspetti strutturali e processuali citati. Shaw (1964), revisionando i lavori sulle reti di comunicazione proposti da Bavelas e colleghi, mostra le differenze in termini di efficienza fra reti di comunicazione centralizzate e de-centralizzate in gruppi di lavoro. In una visione topologica della comunicazione, viene proposto un indice di centralità, da intendersi come la misura in cui il flusso di informazioni nel gruppo è centralizzato in un persona o è disperso in modo più uniforme fra i membri (Leavitt, 1951). Uno studio accurato di diciotto esperimenti condotto da Shaw mostrò che gruppi centralizzati risultano maggiormente produttivi qualora vengano affrontati compiti semplici. Il motivo di ciò sta nella possibilità da parte del leader di gestire efficacemente un flusso di informazioni che sostanzialmente non richiede un'elevata dose di elaborazione. Infatti, laddove il compito richieda un'integrazione di informazioni superiore, il gruppo de-centralizzato risulta più produttivo dal momento che preserva il leader da un possibile sovraccarico cognitivo a cui conseguirebbe il deterioramento della prestazione. Questi studi, riguardando gruppi reali con interazioni faccia a faccia, lasciano aperti dei quesiti sull'applicabilità degli

stessi concetti in contesti di comunicazione mediata da computer. Rispetto agli elementi presi in esame finora si può affermare che la CMC può offrire un'elevata flessibilità delle strutture comunicative, determinando un migliore adattamento a seconda dei compiti da affrontare. Per fare qualche esempio: l'uso della posta elettronica crea una comunicazione centralizzata che tuttavia può facilmente tradursi in de-centralizzata laddove vengano inserite persone che leggono i messaggi "per conoscenza"; nonostante i forum abbiano una struttura di comunicazione tendenzialmente de-centralizzata, questa si centralizza nei casi in cui una discussione venga portata avanti da pochi interlocutori, oppure quando un individuo funge da catalizzatore delle informazioni (Bertani & Manetti, 2007). Infine, discutiamo una delle considerazioni riportate da Fox (2009) a proposito della necessità di interazioni faccia a faccia come supplemento alle usuali vie di comunicazione utilizzate dalla comunità di R (e-mail, mailing list, forum, distribuzione di pacchetti via Internet unita ai lavori di coordinamento tramite version control). Molti membri dell'R Core Team sostengono che periodici meetings risultino cruciali per stabilire idee generali che guidino i percorsi di sviluppo.

Il celebre esperimento di Kurt Lewin del 1941 sulle abitudini alimentari è un esempio che dimostra la potenza della negoziazione in gruppi reali per influenzare le condotte individuali. I tentativi di incrementare l'acquisto di frattaglie di pollo da parte delle massaie portarono a verificare come i gruppi che avevano partecipato a lezioni frontali sul tema in questione avevano modificato le proprie abitudini alimentari in misura significativamente minore rispetto ai gruppi in cui si era privilegiata una situazione di interazione e discussione (Bertani & Manetti, 2007). Il motivo del cambiamento delle condotte viene analizzato da Cialdini (2010), che chiarisce come la negoziazione in gruppo sia intrinsecamente legata alla messa in gioco dell'immagine di sé. Un'identità che si mantiene coerente nel tempo ottiene tendenzialmente un giudizio sociale positivo; questo meccanismo ha un'influenza talmente forte sulla vita degli individui che Cialdini lo inserisce fra i suoi principi motivazionali di base che orientano il comportamento. In questo senso l'immagine di sé negoziata in gruppo durante una discussione in cui si prende pubblicamente un impegno comporta un grande investimento in termini identitari unito alle aspettative personali e sociali in termini di atteggiamento e comportamento. Questo investimento sarebbe minore nei contesti telematici poiché l'identità ha una minore salienza a causa dell'assenza di una parte dei segnali di comunicazione. Seguendo la teoria Reduced Social Cues (RSC) l'utilizzo della CMC, comportando un indebolimento del processo di categorizzazione sociale (propria e altrui), provocherebbe la riduzione della rilevanza dei processi di influenza

sociale (Kiesler et al., 1984). Sarebbero proprio i meccanismi deindividuanti – con relativo calo dell'impegno individuale nei progetti a lungo termine – a provocare l'esigenza di instaurare discussioni tramite interazioni faccia a faccia (meetings periodici), in cui i meccanismi di influenza risultano significativamente più forti, determinando dei miglioramenti per quanto riguarda la chiarificazione e il perseguimento degli obiettivi di comunità.

Tabella 2.1

Aspetti organizzativi strutturali e processuali dello sviluppo del progetto R

	Fasi		
	Iniziale	***Di transizione***	***R Core***
	1990-94	**1994-97**	**1997**
Reclutamento	Partecipazione spontanea di studenti	Dimostrazione di interesse	Su invito
Divisione del lavoro	Nessuna	In via di sviluppo	Semi-formale
Gerarchia di ruolo	Nessuna	Sviluppatori originali	Secondo partecipazione
Cooperazione	Diretta	Volontariato anarchico	Ruoli distinti e volontariato
Pianificazione obiettivi	Nessuna	Implicita	Esplicitazione parziale
Decision-Making	Di gruppo	In gran parte individuale	Tramite ricerca del consenso
Risoluzione conflitti	Discussione	Non necessaria	Discussione/prevenzione
Obiettivo principale	Sviluppo personale	Espansione di R	Vari, in parte conflittuali

Nota: adattato da Fox, 2009

2.2.3. Produttività

Le tematiche fino ad ora trattate sono integrabili e intrinsecamente legate al miglioramento o peggioramento delle prestazioni del gruppo. Badare alle fondamenta strutturali del funzionamento della comunità di R (chiarezza degli obiettivi, definizione dei ruoli, efficacia comunicativa), dà la possibilità di fare alcune puntualizzazioni per chiarire ancora meglio le risultanti in termini di produttività.

Brevemente, è possibile ragionare su due fronti, entrambi relativi al confronto fra la prestazione individuale e quella di gruppo: il primo si riferisce alle teorie del social loafing o inerzia sociale (Steiner, 1972; Latanè, 1981), che guardano alla collaborazione come un impedimento per la realizzazione del vero potenziale del gruppo; il secondo punto, invece, è relativo all'aspetto specifico del brainstorming e più in generale degli stimoli alla creatività (Sproull & Kiesler, 1991).

Le indagini di Steiner e Latanè, che indagano le perdite di processo (process

loss), arrivano ad identificare la presenza dell'altro sociale come prima fonte di inibizione sulla prestazione. Esperimenti successivi (Karau & Williams, 1993) offrono una visione più completa aggiungendo la salienza del compito e del gruppo per l'individuo come elemento che dà giustificazione dei vantaggi del lavoro cooperativo, permettendo di uscire da una visione prettamente individualista. È possibile tuttavia trattare la questione sotto gli aspetti della comunicazione virtuale, attribuendo ad essa alcune note positive per quanto riguarda il miglioramento generale della prestazione di gruppo.

Kiesler e Sproull (1992) ribadiscono una serie di fattori, in parte già trattati, che contribuiscono ad abbassare le perdite di processo:

- L'anonimato è sicuramente l'aspetto più incisivo, poiché permette all'individuo di perseguire gli obiettivi di gruppo mantenendosi in una sorta di spazio intimo individuale, proteggendo dagli impedimenti relativi a questioni di influenza e confronto sociale.
- La possibilità di comunicare di maniera asincrona, prendendo più tempo per rispondere all'interlocutore, si traduce in una maggiore concentrazione sul compito.
- Essendo gli spazi informatici sempre raggiungibili, rendono l'individuo indipendente dalla componente di spazio e di tempo (questo aspetto è strettamente connesso a quello precedente).

Possiamo dire che gli aspetti positivi della CMC che prevengono le perdite di processo, unite alla forte motivazione relativa alla salienza del compito e dei valori del gruppo, potrebbero essere stati fattori essenziali ai fini della qualità dei processi produttivi della comunità di R.

Concludiamo la sezione fornendo un ulteriore conferma delle potenzialità produttive della CMC, in particolare parlando di compiti di brainstorming (Lamm & Trommsdorff, 1973; Mullen et al., 1991). Quest'ultimo, in esperienze faccia a faccia, risulta più efficace quando viene eseguito prima in privato, usando successivamente il gruppo di interazione come un luogo di discussione pubblico per combinare e valutare queste idee prodotte individualmente.

Il brainstorming effettuato tramite CMC, invece, ha insito nella maggior parte dei casi (soprattutto se è presente un grado elevato di asincronicità comunicativa) il meccanismo di isolamento/gruppo funzionale. In un forum o tramite posta elettronica, per esempio, l'individuo mentre scrive il proprio messaggio lo sta in realtà mettendo al vaglio privatamente per poi condividerlo.

Capitolo 3

R: un ambiente/software libero per il calcolo statistico e l'elaborazione grafica

"If you give people a linear model function you give them something dangerous"
(John Fox, UseR!,Vienna, Maggio 2004)

Il terzo capitolo si propone di presentare un quadro di riferimento generale (non esaustivo) sulle funzionalità basilari dell'ambiente/software interattivo R. Il primo sottocapitolo esplora le potenzialità di R in ambito statistico. La seconda sezione invece offre una panoramica delle risorse disponibili che permettono di usufruire di R per la ricerca in Psicologia. Infine, si è voluta dedicare una breve sezione all'analisi introduttiva di una recente interfaccia per l'utente (Rstudio) che ha suscitato giudizi positivi poiché, non snaturando le funzioni di base di R, facilità il suo utilizzo ai principianti semplificando allo stesso tempo il lavoro dell'utente più esperto.

3.1. Uso di R per l'analisi statistica dei dati

R costituisce una valida alternativa ai più comuni software statistici commerciali. Inizialmente progettato per sistemi MacOS, oggi è un programma informatico multipiattaforma a tutti gli effetti, che può essere utilizzato su una grande varietà di architetture e su diversi sistemi operativi (tra i più conosciuti: Linux, Windows, MacOS, Unix). Il suo sviluppo è curato da ricercatori di fama internazionale, che contribuiscono ad incrementarne le possibilità di utilizzo. Il suo linguaggio di programmazione – anche chiamato GNU S, poiché basato sul linguaggio S ma reso per volontà degli sviluppatori originali un sistema aperto – gode di notevole consistenza e struttura sintattica. Una prova di ciò è sicuramente il prestigioso riconoscimento (The Association for Computing Machinery award for Software Systems) assegnato nel 1998 a John Chambers (oggi membro del R Core Team) per la definizione del linguaggio S. Oltre ai benefici in termini di produttività determinati da una comunità intera che collabora, altri frutti della scelta di rendere R un sistema aperto sono senz'altro:

➢ la ricchezza e la facilità di consultazione delle informazioni contenute nella funzione di aiuto richiamabile dal software

- `help();`
- `example();`
- `help.search("termine");`
- `help(package="nomepacchetto").`

➢ la disponibilità di informazioni e risorse (dall'assistenza alla vasta manualistica scaricabile) accessibili dal sito http://www.r-project.org. Inoltre, in quest'ultimo sono presenti i codici binari del software e il codice sorgente scritto in linguaggio C, Fortran (Backus, 1998) ed R.

Possiamo definire R un ambiente interattivo e versatile, che integra una serie di risorse software utili per manipolare dati, fare calcoli e costruire rappresentazioni grafiche.
Le sue caratteristiche principali comprendono:

- La flessibilità determinata dall'efficace linguaggio di programmazione che permette di creare istruzioni condizionali, loop, funzioni ricorsive definite dall'utente e strumenti di input/output;
- Semplicità di gestione e manipolazione di dati; inoltre R ha una buona capacità di memorizzazione di dati;
- Un insieme di operatori che consentono di effettuare calcoli su vettori, matrici e altre operazioni complesse;
- Una vasta raccolta di strumenti intermedi per l'analisi statistica. Permette una gestione ottimale sia dei modelli statistici più semplici che di quelli più complessi;
- Produzioni grafiche: consente di creare un numero elevato di rappresentazioni grafiche che possono essere definite individualmente oppure utilizzando diversi pacchetti basati su Java, OpenGL, etc.

3.1.1. Funzionalità di R

R opera tramite linea di comando ed eventualmente col supporto di diverse interfacce grafiche (GUI). Al suo avvio sono visualizzabili all'interno della console una nota di licenza e alcune brevi delucidazioni, seguite da un prompt (>). Il cursore di scrittura (|) indica lo spazio all'interno del quale è possibile iniziare a scrivere i comandi.

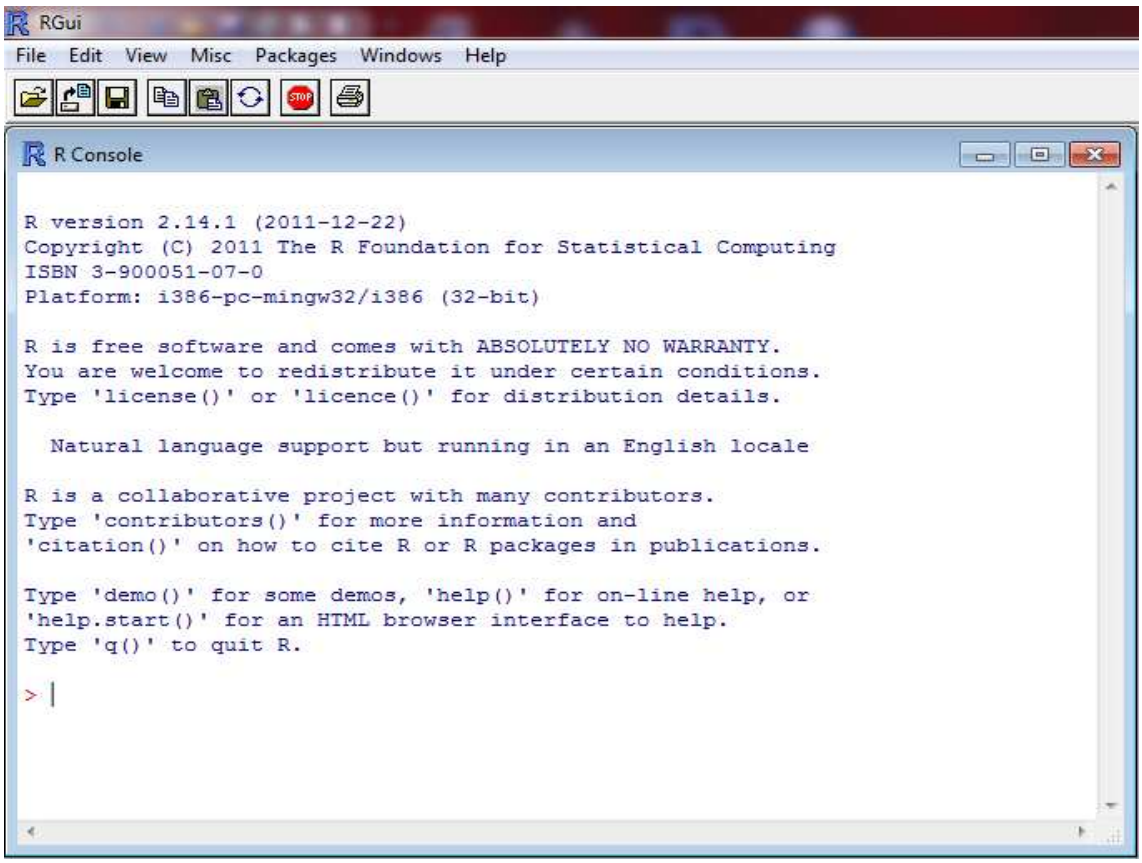

Figura 3.1. R per Windows, versione 2.14.1.

Fra le più conosciute interfacce grafiche (GUI) abbiamo:

- R-Commander GUI.

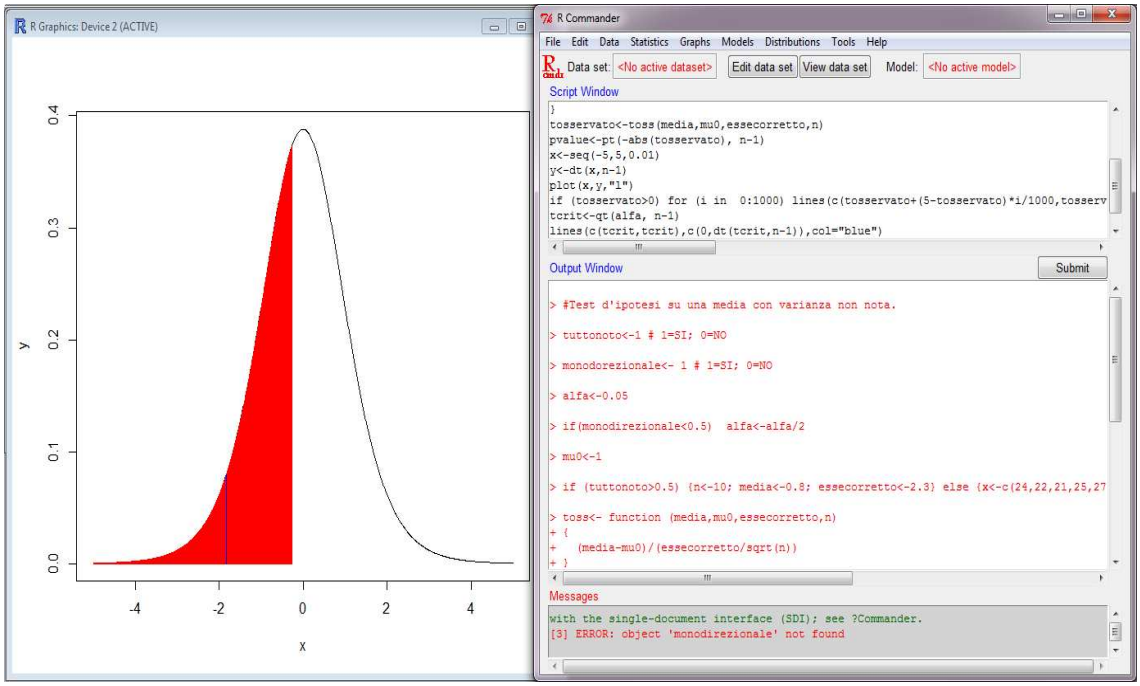

Figura 3.2. R Commander per Windows, versione 1.9-x.

- Tinn-R.

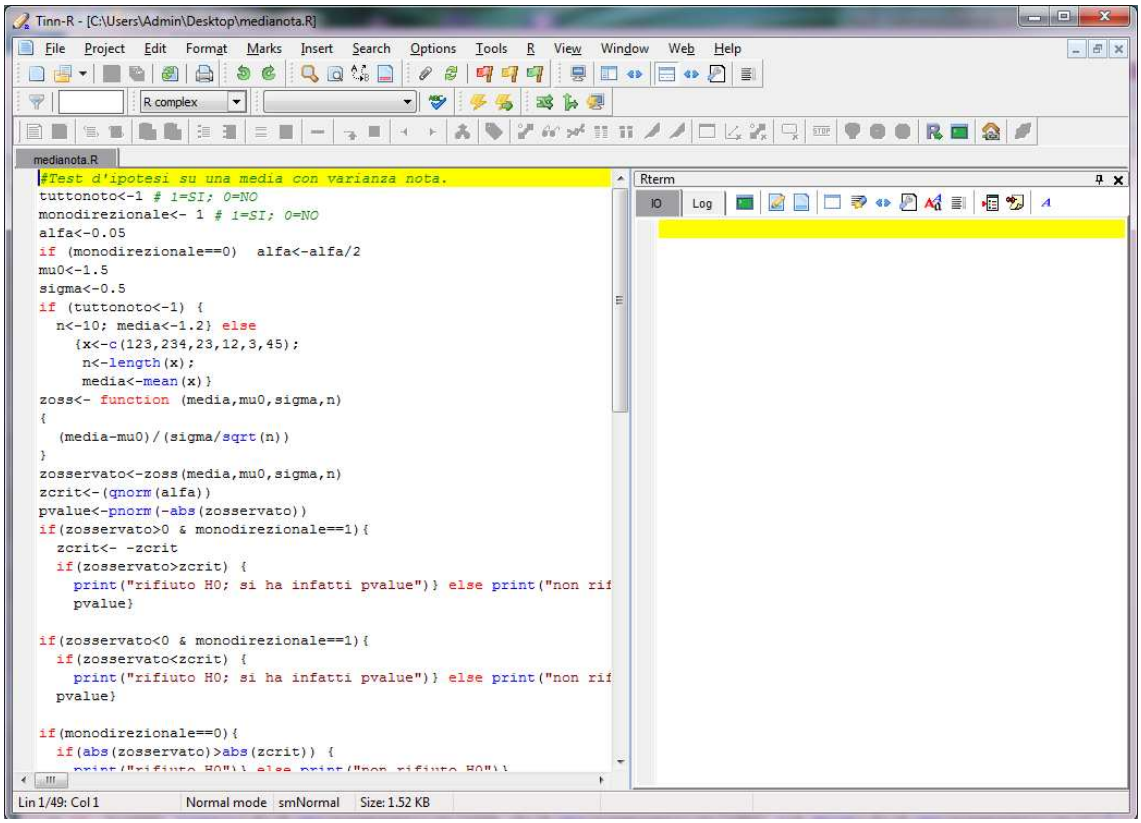

Figura 3.3. Tinn-R Editor Gui per Windows, versione 2.3.7.1.

- RStudio.

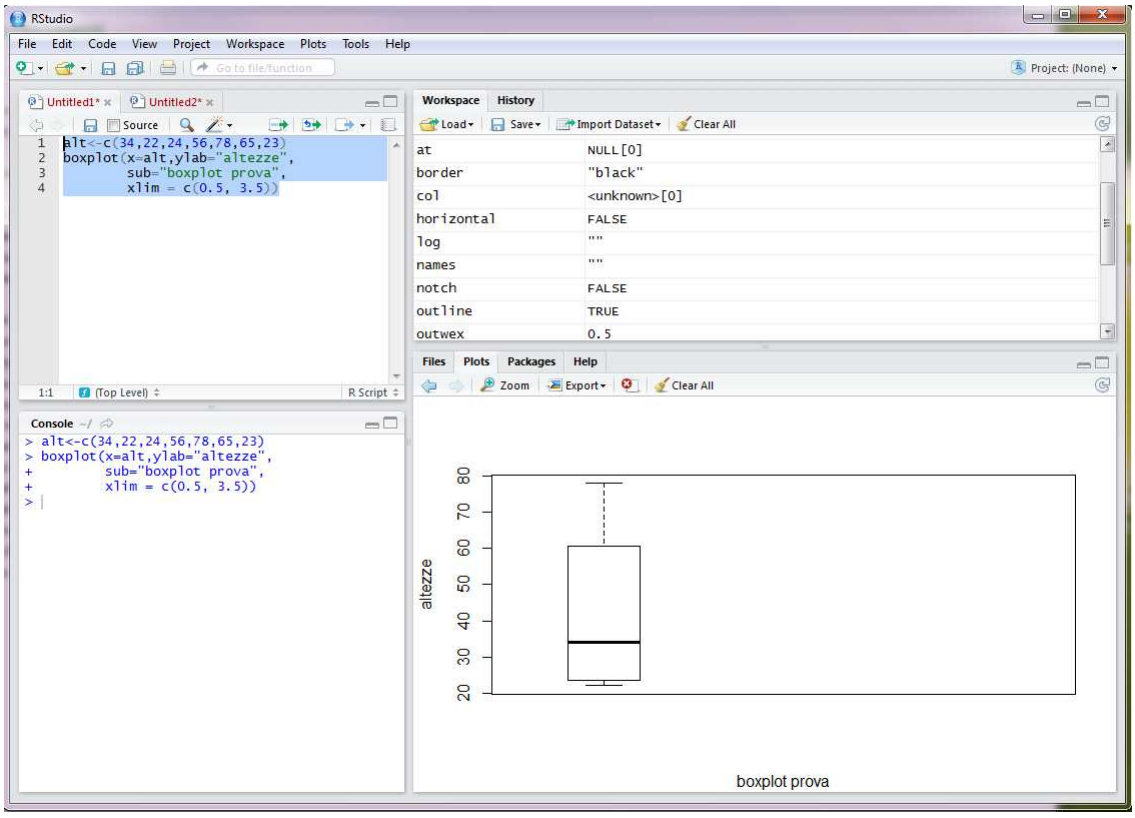

Figura 3.4. RStudio per Windows, versione 0.96.331.

Utilizzare un'interfaccia di supporto per l'utente GUI può essere molto utile in una fase introduttiva al software, poiché diminuisce notevolmente le possibilità di commettere errori durante la formulazione sintattica dei comandi. Tuttavia, allo stesso tempo possono verificarsi una diminuzione della flessibilità e dei rallentamenti, qualora sia necessario effettuare operazioni complesse. L'interfaccia minimalista a linea di comando è uno degli aspetti che differenzia R da altri software statistici di tipo commerciale, che in certi casi operano interamente tramite GUI.

Un'altra importante differenza fra R e altri software statistici è la possibilità di effettuare analisi tramite una serie di passi, con risultati intermedi che vengono immagazzinati in "oggetti". Così, mentre SAS o SPSS daranno una produzione copiosa di risultati relativi, R darà la minima produzione conservando i risultati, che potranno essere richiamati da altre funzioni di R (Mineo, 2003). Nello specifico, qualsiasi entità in R – numeri, variabili, funzioni e più in generale qualsiasi struttura costruita a partire da tali componenti – esiste come oggetto, avente specifiche proprietà di lunghezza, modo, appartenenza ad una classe.

Tutti gli oggetti che vengono creati durante una sessione di lavoro possono essere visualizzati nello spazio di lavoro (workspace; vedi Figura 3.4) ed eventualmente salvati per essere recuperati successivamente. R imposta automaticamente una cartella di default in cui verrà salvato il workspace al termine della sessione. In generale è utile utilizzare spazi di lavoro diversi per evitare confusione fra oggetti di analisi dei dati diverse. Tramite console è possibile utilizzare diversi comandi (in alternativa le interfacce per l'utente GUI sono dotate di funzioni manuali per effettuare le stesse operazioni):

- Per modificare la directory:

```
setwd("C:/nomecartella")
```

- Per salvare il workspace in maniera definitiva:

```
save.image()      oppure      save.image("nomefile.Rdata")
```

- Per salvare singoli oggetti:

```
save();  save(x, file = "x.Rdata");
save(x, y, z, file ="prova.Rdata")
```

Possiamo dire che fra tutti gli oggetti, le funzioni rivestono un ruolo rilevante poiché permettono di svolgere dalle operazioni più semplici alle tecniche statistiche più complesse. Ogni comando in R è una funzione con sintassi generica:

```
nomefunzione(argomento1, argomento2, argomento3,...)
```

Non è sempre necessario specificare ogni argomento di una funzione. Nemmeno l'ordine degli argomenti è importante, purché vengano specificati i nomi degli argomenti che si vogliono impiegare (vedi Figura 3.4, esempio di funzione boxplot). Funzioni che possono essere richiamate come `mean()`, `var()` o `table()` sono scritte in linguaggio R e non differiscono sostanzialmente dalle funzioni che l'utente può creare autonomamente a seconda delle esigenze di lavoro. Questo processo che permette di memorizzare oggetti che possono essere utilizzati in altre espressioni fa guadagnare al linguaggio R in termini di potenza, praticità ed eleganza. Imparare a scrivere funzioni in R è uno dei modi principali per rendere il suo utilizzo confortevole e produttivo (R Core Team, 2012).

R svolge principalmente operazioni utilizzando dati strutturati come vettori, matrici, liste e dataframe. Tuttavia, più semplicemente è possibile lavorare con dati di tipo numerico, di testo, e valori logici. In particolare è possibile utilizzare R come una calcolatrice svolgendo operazioni matematiche con simboli predefiniti (+ , - , * , / , ^), oppure per eseguire operazioni tramite operatori logici (vedi Tabella 3.1):

Tabella 3.1

Corrispondenza simbolo/operatore logico in R

<	Minore di
>	Maggior di
<=	Minore o uguale a
>=	Maggiore o uguale a
==	Uguale a
!=	Diverso da
&	And
|	Or
!	Not

In generale, invece di svolgere direttamente le operazioni semplici, è utile immagazzinare i risultati sotto forma di variabili. Perciò, se si volesse memorizzare l'operazione algebrica `3/10 + 2/5` si potrebbe creare un oggetto assegnando un nome alla variabile:

```
x <- 3/10 + 2/5
```

Richiamando l'oggetto "x" R riporterà il valore associato (anche presente nel workspace):

```
x
[1] 0.7
```

È importante ricordare che il linguaggio di R è case sensitive, ovvero distingue fra lettere maiuscole e minuscole; perciò, se si provasse a richiamare la variabile precedentemente definita scrivendo in console "X" il programma non riuscirebbe a identificare l'oggetto:

```
X
Error: object 'X' not found
```

Le variabili possono contenere informazioni di diverso formato:

- Numeri (scalari) sia interi che decimali, stringhe (testo tra virgolette), valori logici (TRUE, FALSE, T, F).
- Data vector (vettore dati): concatenazione di elementi di uno stesso tipo.

```
y <- c("ac", "rf", "yh", "pvt")
```

- Factors (fattori): variabili definite categorialmente a livello nominale o ordinale.

```
taglia <- c("m","s","s","xl","l","l","xxl","m")
tag <- factor(taglia)
tag
[1] m   s   s   xl  l   l   xxl m
Levels: l m s xl xxl
```

- Matrici: distribuzione strutturata degli elementi di un vettore in righe e colonne.

```
matrix(vettore, numrighe, numcolonne)

taglia <- matrix(taglia, 2, 4)
taglia
     [,1] [,2] [,3] [,4]
[1,] "m"  "s"  "l"  "xxl"
[2,] "s"  "xl" "l"  "m"
```

- Data-frame: una tipologia di matrice in cui è possibile inserire dati di tipo diverso. Solitamente ogni colonna identifica un tipo di variabile da analizzare, mentre le righe corrispondono ai casi statistici. Questa struttura di dati è molto utilizzata.

```
Numeri <- c(98, 57, 42)
Stringa <- c("Richard", "Marshall", "Stallman")
Df <- data.frame(num=numeri, stg=stringa)
df
  num      stg
1  98  Richard
2  57 Marshall
3  42 Stallman
```

- List (Liste o elenchi): è possibile raggruppare anche una serie di dati e tutti i tipi di variabili in un elenco.

```
Lista <- c(76,x,tag,df )
```

In R è possibile esportare ed importare dati tramite diverse funzioni a seconda del formato (R, .txt, .xls, .spss,…). In alcuni casi è necessario installare e caricare un pacchetto statistico aggiuntivo che permetta di svolgere queste operazioni. Un esempio è il pacchetto `foreign`, molto utile poiché permette di leggere file SPSS, SAS, Stata, Systat, Minitab. Effettivamente i “contributed packages” (pacchetti) ampliano notevolmente le potenzialità di lavoro in R, che di fatto vede le sue funzioni essenzialmente raccolte in librerie. Per poter usufruire di un determinato pacchetto statistico questo deve essere installato – R si collega automaticamente al repository

(deposito) dal quale è possibile scaricare il pacchetto che si desidera – e caricato. Al modulo base di R, che comprende le librerie di default già installate e caricate, è possibile aggiungere pacchetti aggiuntivi che sono già installati ma non caricati, oppure altri che è necessario scaricare in vari modi dal sito http://www.r-project.org/. Infine, concludiamo questa breve disamina sulle funzionalità basilari di R accennando alle sue potenzialità di sviluppo grafico.

- Le rappresentazioni grafiche rivestono un ruolo fondamentale in Statistica. Si tratta infatti di elaborazioni che vengono fatte su una serie di dati per trarne informazioni chiare (Mineo, 2003). I comandi per i grafici possono essere suddivisi generalmente in tre gruppi:

- Funzioni di alto livello, che permettono di creare grafici nuovi, visibili nella finestra grafica; il comando più utilizzato è `plot()`. Altri esempi di funzioni che producono grafici particolari possono essere: `qqnorm()`, `hist()`, `dotchart()`, `barplot()`, `boxplot()`.

- Funzioni di basso livello, utili per aggiungere particolari ad un grafico già esistente; alcuni esempi: `points(x, y)`, `lines(x, y)` per aggiungere punti o linee di collegamento al grafico corrente; `text(x, y, labels, …)` per aggiungere un testo nei punti dati da "`x`" e "`y`"; `abline(a, b)` per aggiungere una retta di pendenza "`b`" e intercetta "`a`"; `polygon(x, y, …)` per tracciare un poligono definito dai vertici ordinati (`x`, `y`); `legend(x, y, legend, …)` che aggiunge una legenda nella posizione specificata.

- Funzioni per grafici interattivi, che consentono di aggiungere o estrarre informazioni da un grafico esistente. Le principali funzioni sono: `locator(n, type)` che permette all'utente di selezionare le posizioni del grafico usando il tasto sinistro del mouse finché n punti non sono stati selezionati o il bottone centrale del mouse non è premuto. L'argomento "`type`" permette di disegnare i punti selezionati: il default è nessun grafico (`type="n"`); `identify(x, y, labels)` dà la possibilità all'utente di evidenziare alcuni punti definiti da (`x`, `y`) usando il tasto sinistro del mouse, disegnando le corrispondenti label vicino.

3.2. R in Psicologia

La Psicometria, ovvero l'insieme dei metodi d'indagine psicologica tendenti al raggiungimento di valutazioni quantitative del comportamento umano o animale, è una disciplina strettamente collegata alla Statistica. Negli ultimi 30 anni sono stati sviluppati dei software statistici con una forte connotazione rispetto all'analisi di dati psicologici, come ad esempio SPSS (Statistical Package for the Social Sciences) e SAS (Statistical Analysis System). Tali software conservano effettivamente una forte connessione con la storia della Psicometria (Leeuw & Mair, 2007).

Le grandi potenzialità di programmazione di linguaggi come R ed S+, sviluppati con l'obiettivo specifico di fare statistica, hanno reso maggiormente accessibile lo studio della Psicometria. Non è più necessario scrivere lunghi programmi in Fortran o fare affidamento su gruppi di pacchetti informatici proprietari creati per analisi particolari. Oggi è possibile utilizzare R per quasi tutte le esigenze psicometriche basilari o avanzate (Revelle, 2011). Inoltre, ponendo le basi per una comunicazione di livello internazionale fra persone di differenti culture, è possibile considerare R una sorta di lingua franca in ambito statistico.

All'indirizzo http://cran.r-project.org/web/views/ è possibile prendere visione delle varie tipologie di pacchetti presenti nel CRAN, divisi per settori di interesse (attualmente ne sono disponibili 29). In particolare "The Psychometric Task View" (http://cran.r-project.org/web/views/Psychometrics.html) racchiude dei pacchetti R concernenti la progettazione e l'analisi di ricerca finalizzate alla misurazione in ambito psicometrico. Questa sezione comprende novantaquattro package e alcune funzioni provenienti dal package `stats` (`factanal()`, `princomp()`, and `cmdscale()`). Fra questi, novantadue sono direttamente accessibili e due solamente menzionati (`cirt`, `mlirt`). Tra i pacchetti accessibili diciannove possono essere considerati principali (Core). The Psychometric Task View è inoltre suddiviso in sei aree metodologiche:

- Item Response Theory (IRT);
- Correspondence Analysis (CA);
- Structural Equation Models, Factor Analysis, Principal Component
- Analysis;
- Multidimensional Scaling (MDS);
- Classical Test Theory (CTT) ;
- Other related packages (latent class analysis, configural frequency analysis, interrater reliability and agreement, psychophysical data analysis, Bradley–Terry

models, elimination-by-aspects models, confidence intervals for standardized effect sizes, knowledge space theory, Fechnerian scaling).

3.3. Una recente integrazione per R: RStudio

Recentemente si è verificata una progressiva diffusione di Rstudio (http://rstudio.org/), superficialmente inserito fra le interfacce GUI user friendly, ma meglio definibile come un ambiente di sviluppo integrato per R (IDE: integrated development environment). Il suo team di sviluppo, composto da J. J. Allaire, J. Cheng, J. Paulson e P. DiCristina, ha come obiettivo centrale quello di fornire degli strumenti potenti, semplici e intuitivi che rendano R un ambiente accogliente sia per l'utente novizio che per l'esperto programmatore. RStudio viene distribuito sotto la GNU Affero Public License v3 (AGPL: http://www.gnu.org/licenses/agpl-3.0-standalone.html), nello spirito della Free Software Foundation e del progetto GNU. Questa può essere inserita fra le licenze con copyleft "forte", in quanto è sostanzialmente composta dai termini della GPL con una clausola aggiuntiva che impone l'obbligo di rendere disponibili i sorgenti, anche se il software non viene tecnicamente distribuito ma è utilizzato esclusivamente per offrire servizi online.

Per usufruire di RStudio è necessario aver precedentemente installato la versione standard di R, disponibile nel CRAN; è utilizzabile, oltre che nelle più comuni piattaforme (Windows, Mac, Linux), anche tramite un server remoto Linux che fornisce l'interfaccia su browser internet. Questo permette all'utente: di accedere al proprio spazio di lavoro da qualsiasi luogo; di scambiare velocemente dati, file e codici; di accedere a risorse di calcolo più potenti qualora il server sia ben equipaggiato; di installare e configurare in maniera centralizzata R, pacchetti R e TeX, e altre librerie di supporto.

RStudio integra in un unico ambiente tutti gli strumenti che vengono utilizzati durante una sessione di lavoro in R: source editor, console, workspace, grafici, R Help, Packages (vedi Figura 3.4.). La disponibilità di una grande varietà di strumenti interattivi dà la possibilità di migliorare significativamente la produttività. Esponiamo di seguito le principali funzionalità innovative (la maggior parte sono disponibili tramite shortcuts):

- Strumenti di coding:

- È possibile mantenere un history con tutti i comandi precedentemente digitati in console. Questi verranno visualizzati nel finestra History e potranno essere ricercati

utilizzando l’apposita barra di ricerca. Inoltre, selezionando il comando indicato, è possibile caricarlo in console o inserirlo all’interno del source editor.

- Durante il lavoro (sia in console che nel source editor) è possibile utilizzare il completamento automatico del codice, che fornisce non solo un suggerimento sui tipi di funzione disponibili, ma anche sulla sintassi degli argomenti. Inoltre permette di richiamare agevolmente degli oggetti presenti nel workspace.
- Rstudio permette di eseguire direttamente i comandi dal source editor.
- Il source editor dà la possibilità di aprire contemporaneamente file di formato diverso (R scripts, file testo, documento TeX, documento Sweave). Inoltre possono risultare molto utili le funzioni brevi find/replace (trova e sostituisce una parte di testo), extract function (selezionando una parte di codice è possibile convertirlo automaticamente in una funzione), comment/uncomment (converte una riga di codice in commento e viceversa), code folding (permette di nascondere parti di codice in modo da facilitarne la consultazione; utile per nascondere il corpo delle funzioni lasciando visibile il nome).

➢ Navigazione guidata all’interno del codice che permette di migliorarne la comprensibilità soprattutto quando viene scritto da altri, o nel caso di pacchetti esterni.

➢ Progetti multipli: RStudio dà la possibilità di creare contesti di lavoro separati, ognuno con il proprio spazio di lavoro, la propria directory di salvataggio, la propria history e i propri documenti di codice; l’utente può lavorare a progetti differenti contemporaneamente, associandoli a directory nuove o già esistenti. RStudio ha anche un supporto integrato per Git e Subversion. Sono comprese inoltre diverse opzioni di settaggio per la personalizzazione dei progetti.

➢ RStudio include un sistema potente e flessibile (Sweave) per la creazione di report dinamici e ricerche riproducibili con LaTeX. Sweave consente l'incorporamento di codice R all'interno di documenti LaTeX per generare un file PDF che contiene narrativa e analisi, grafica, codice, ed i risultati dei calcoli. Tramite `knitr` (pacchetto di R interamente supportato da RStudio) vengono aggiunte ulteriori funzionalità a Sweave. Per utilizzare Sweave e `knitr` per creare un report in formato PDF, è necessario avere LaTeX installato sul computer.

➢ Il pacchetto `manipulate` consente di aggiungere delle funzioni interattive per manipolare i grafici.

Conclusioni

Nel presente lavoro si è deciso di considerare in egual misura gli aspetti diacronici e sincronici alla base dei processi sociali e organizzativi analizzati. In questo senso una prospettiva d'indagine che ha dei connotati antropologici - volutamente conciliati con la Psicologia Sociale e del Lavoro - ha fornito una più chiara visione della qualità e della quantità di interazioni che hanno determinato la struttura di un macro-gruppo di lavoro come la comunità intorno al progetto R.

È stata evidentemente accolta la critica di Richard Stallman al movimento dell'Open Source, accusato di aver svalutato l'influenza di una prospettiva etica sui processi organizzativi e di sviluppo di intere comunità di lavoro. In realtà, dagli studi presentati è emerso che l'esplicitazione dei valori all'interno di organizzazioni telematiche diventi essenziale per il perseguimento efficace degli obiettivi, per la ricerca del consenso e dell'impegno nei progetti.

Possiamo affermare che, le evidenze empiriche e sperimentali a favore di un lavoro di tipo collaborativo, sono solo una parte della consapevolezza necessaria per poter gestire efficacemente il processo di cambiamento/adattamento che i gruppi sociali e lavorativi si trovano a dover affrontare continuamente. In questa sede è emerso come spesso cambiamenti radicali possano interessare le strutture organizzative, senza una conseguente cognizione di ciò da parte dei membri del gruppo.

Concludiamo sottolineando come l'ottica psicologico-sociale qui adottata ha permesso di interpretare le vicende relative allo sviluppo di R ricercando quel complesso di forze che vede interdipendenti e concatenati fattori etici, legali, sociali, produttivi.

Riferimenti bibliografici

Agostinelli, C. (2000). *Introduzione a R. Versione 0.3*. Ottobre. URL http://www.mat.uniroma3.it/didattica_interattiva/aa_01_02/st1/manuale.pdf

Aliprandi, S. (2005). *Copyleft & opencontent. L'altra faccia del copyright*. Lodi, PrimaOra società cooperativa, marzo.

Backus, J. (1998). The history of Fortran I, II, and III. *IEEE Annals of the History of Computing*. 20(4), 68–78.

Baran, P. (1964). *On distributed communications: I. Introduction to distributed communication networks. Santa Monica*. The RAND Corporation, agosto. URL http://www.rand.org/content/dam/rand/pubs/research_memoranda/2006/RM3420.pdf.

Behlendorf, B. (1999). Open Source come strategia commerciale. In C. di Bona, S. Hockman & M. Stone, *Open sources. Voci dalla rivoluzione Open Source.* Milano, Apogeo. URL http://www.apogeonline.com/openpress/libri/545/brianbeh.html.

Berra, M., Meo, A. R. (2000). *Il software libero: la qualità attraverso la collaborazione*. Quaderni di Sociologia, Torino, Rosenberg & Sellier, XLIV, 23, febbraio, 3-21.

Bertani, B., Manetti, M. (2007). *Psicologia dei gruppi. Teoria, contesti e metodologie d'intervento*. Milano, FrancoAngeli.

Brown, R. (2000). *Group Processes. Dynamics within and between Groups* (trad.it. Psicologia sociale dei gruppi. Bologna, il Mulino).

Cialdini, R. B. (2010). *Influence. The Psychology of Persuasion* (trad.it. Le armi della persuasione. Come e perché si finisce col dire di si. Firenze, Giunti Editore).

Cohen, W., Goto, A., Nagata, A., Nelson, R. R., Walsh, J. P. (2002). R&D spillovers, patents and the incentives to innovate in Japan and the United States. *Recent Policy*, 31, 1349-1367.

Festinger, L. (1954). A theory of social comparisation processes. *Human Relations*, 7, 117-140.

Fox, J. (2009). Aspects of the Social Organization and Trajectory of the R Project. *The R Journal*. 1/2, Dicembre. ISSN 2073-4859 URL http://journal.rproject.org/archive/2009-2/RJournal_2009-2_Fox.pdf.

Free Software Foundation (1991). *GNU General Public License*. Boston (trad.it. Associazione per il software libero, 2000) URL http://softwarelibero.it/gnudoc/gpl.it.txt.

Free Software Foundation (2012). *What is free software? The Free Software Definition*. URL http://www.gnu.org/philosophy/free-sw.en.html.

Gabassi, P. G. (2006). *Psicologia del lavoro nelle organizzazioni*. Milano, FrancoAngeli.

Gersick, C., J., Hackman, J. R. (1990). Habitual routines in task performing groups. *Organizational Behaviour and Human Decision Processes*, 47, 65-97.

Gruppo Laser (2005). *Il sapere liberato. Il movimento dell'open source e la ricerca scientifica*. Milano, Giangiacomo Feltrinelli.

Hogg, M. A. (1992). *The Social Psychology of Group Cohesiveness: From Attraction to Social Identity*. Harvester Wheatsheaf, Londra.

Ihaka, R. (1998). R: Past and Future History. URL http://cran.r-project.org/doc/html/interface98-paper/paper.html.

Ippolita (2005). *Open non è free. Comunità digitali tra etica Hacker e mercato globale*. Milano, Elèuthera.

Jaffe, A. B. (2000). The U.S. patent system in transition: policy innovation and the Innovation process. *Recent Policy*, 29, 531-557. URL http://www.rvm.gatech.edu/bozeman/rp/read/30304.pdf.

Karau, S., J., Williams, K. D. (1993). Social loafing: A meta-analytic review and theoretical integration. *Journal of Personality and Social Psychology*, 65, 681-706.

Kiesler, S., Siegel, J., McGuire, T. (1984). Social Psychological aspects of computer mediated communication. *American Psychologist*, 39, 1123-1134.

Kiesler, S., Sproull, L. (1992). Group Decision making and communication tecnology. *Organizational Behaviour and Human Decision Processes*, 52, 96-123.

Kortum, S., Lerner, J. (1998). What is behind the recent surge in patenting? *Recent Policy*, 27, Luglio, 273-284.

Lamm, H., Trommsdorff, G. (1973). Group versus individual performance in tasks requiring ideational proficiency (brainstorming): A review. *European Journal of Social Psychology*, 3, 361-388.

Lanjouw, J., Lerner, S. (1997). The enforcement of intellectual property rights: a survey of the empirical literature. *NBER Working Papers*. w6296, dicembre.

Latané, B. (1981). The social impact of majorities and minorities. *Psychological Review*, 88, 438-453.

Lea, M., Spears, R. (1992). Computer Mediated Communication, de-individuation and group decision making. *International Journal of Man-Machine studies*. 34, 283-301.

Leavitt, H. J. (1951). Some effects of certain communication, deindividuation and group decision-making. *Journal of Abnormal and Social Psychology*, 46, 38-50.

Leeuw, J., Mair, P. (2007). An Introduction to the Special Volume on "Psychometrics in R". *Journal of Statistical Software*, 20/1, Maggio.

Levy, S. (2002). *Hackers. Gli eroi della rivoluzione informatica*, IV, Milano, ShaKe, 33.

Lumsden, G., Lumsden, D. (1996). *Communicating in groups and teams: Sharing Leadership*. New York, Wadsworth Publishing.

Maindonald, J. H. (2008*). Using R for Data Analysis and Graphics Introduction, Code and Commentary*. URL http://cran.r-project.org/doc/contrib/usingR.pdf.

Massari, R. (2012). *Introduzione all'analisi statistica con R*. URL http://www.dis. uniroma1.it/~statistica/dispensaR.pdf.

Mauss, M. (2002). *Essai sur le don* (trad.it. Saggio sul dono. Forma e motivo dello scambio nelle società arcaiche. Introduzione di Marco Aime. Torino, Giulio Einaudi editore).

Mineo, A. M. (2003). *Una giuda all'utilizzo dell'ambiente statistico R*. URL http://cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf.

Moser, P. (2003). How do patent laws influence innovation? Evidence from nineteenth Century world fairs. *NBER Working Papers*. w9909, Agosto. URL http:// www.nber.org/papers/w9909.pdf.

Mullen, B., Johnson, C., Salas, E. (1991). Productivity loss in brainstorming groups: A meta-analytic integration. *Basic and Applied Social Psychology*, 12, 3-23.

Open Source Initiative (2012) *Mission*. URL http://opensource.org/.

Open Source Initiative (2012) *The Open Source Definition*. URL http://opensource.org /docs/osd.

Perens, B. (1999) The Open Source Definition. In C. di Bona, S. Hockman & M. Stone, *Open sources. Voci dalla rivoluzione Open Source*. Milano, Apogeo. URL http://www.apogeonline.com/openpress/libri/545/bruceper.html.

Pinzani, F. (2012) *Storia di Internet: gli anni '60*. URL http://www.pinzani.it/storiainternet.php.

Pinzone, F. (2011). *Manuale Introduttivo all'uso di R*. URL http://epidemiologia.asppalermo.org/manuale_introduttivo_all_uso_di_r.pdf.

Quaglino, G., P., Casagrande, C., Castellano A. (1992). *Gruppo di lavoro, lavoro di gruppo*. Milano, Raffaello Cortina editore.

R Development Core Team (2012). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

R Development Core Team. (2012). *An Introduction to R. Notes on R: Programming Environment for Data Analysis and Graphics*. *Version 2.15.1*. 22, Giugno. URL http://cran.r-project.org/doc/manuals/R-intro.pdf.

Raymond, E. S. (1999). Breve storia sugli hackers. In C. di Bona, S. Hockman & M. Stone, *Open source*. *Voci dalla rivoluzione Open Source*. Milano, Apogeo. URL http://www.apogeonline.com/openpress/libri/545/raymonda.html.

Raymond, E. S. (1999). *Homesteanding the Noosphere* (trad.it. Colonizzare la Noosfera, Milano, Apogeo, Giugno) URL http://www.apogeonline.com/openpress/homesteading.

Raymond, E. S. (2000). *The Cathedral and the Bazaar*. *3.0*. URL http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html.

Reicher, S. D. (1984). Social Influence in the crowd: Attitudinal and behaviour effects of de-individuation in conditions of high and low group salience. *British Journal of Social Psychology*. 23, 341-350.

Revelle, W. (2011). *An introduction to psychometric theory with applications in R*. URL http://www.personality-project.org/r/book/.

Ricci, V. (2004). R: un ambiente open source per l'analisi statistica dei dati. URL http://www.dsa.unipr.it/soliani/allegato.pdf.

Rossi, G. (2010). *L'uso di R in psicologia 0.7.1*. URL http://www.germanorossi.it/mi/file/disp/RxPsi.pdf.

Rubini, A. (1999). *Il software libero è economicamente sostenibile*. Apogeo, 19, luglio. URL http://www.apogeonline.com/openpress/articoli/art_21.html.

Serri, F. (2004). *Gruppi, comunicazione e processi decisionali*. Cagliari, CUEC Cooperativa Universitaria Editrice Cagliaritana, Gennaio.

Shaw, M. E. (1964). Communication networks. In Berkowitz, *Advances in Experimental Social Psychology*, New york, Academic Press, 1.

Sproull, L., Kiesler, S. (1991). *Connections. New ways of working in the networked organization*. Mit press, Cambridge.

Stallman, R. M. (2002). *Cos'è il Copyleft? Free Software, Free Society: The Selected Essays of Richard M. Stallman.* GNU Press. URL http://www.faberbox.com/shared/docs/software_libero_pensiero_libero/cosacopyleft.html.

Stallman, R. M. (2012). *Why Open Source misses the point of Free Software*. URL http://www.gnu.org/philosophy/open-source-misses-the-point.en.html.

Steiner, I. D. (1972). *Group Process and Productivity*. New York, Academic Press.

Sterling, B. (1993). A short history of the Internet. *Magazine of fantasy and science fiction*. Cornwall, febbraio. URL http://www.library.yale.edu/div/instruct/internet/history.htm.

Tajfel, H. (1978). *Differentiation between Social Groups: Studies in the Social Psychology of Intergroups Relations*. Londra, Academic Press.

Tajfel, H., Turner, J. C. (1986). An integrative theory of social conflict. In S. Worchel e W. Austin, *Psychology of Intergroup Relations*, Nelson Hall, Chicago, I11.

Todon, A. (2004). *Il software libero/open source. Una dimensione sociale*. URL http://softwarelibero.it/files/todon_software_libero_open.pdf.

Turner, J. C. (1987). The analysis of social influence. In J.C. Turner, M.A. Hogg, P.J. Oakes, S.D. Reicher e M.S. Wetherell, *Rediscovering the Social Group: A Self-Categorization Theory*, Blackwell, Oxford.

Ünlü, A., Yanagida, T. (2010). R you ready for R?: The CRAN Psychometric Task View. *British Journal of Mathematical and Statistical Psychology*, 64, 182-186.

Walther, J. B. (1996). Computer-Mediated Communication: Impersonal, Interpersonal, and Hyperpersonal Interaction. *Communication Research*, 19, 52-90.

Williams, S. (2003). *Free as in Freedom: Richard Stallman's Crusade for Free Software* (trad.it. Codice libero. Richard Stallman e la crociata per il software libero. Milano, Apogeo, Febbraio) URL file:///C:/Users/Admin/Documents/tesi/Open%20source/CodiceLibero%20%20Richard%20Stallman/CodiceLibero/index.html.

Young, R. (1999). Regalato! Come Red Hat Software si trovò fra le mani un nuovo modello economico e contribuì a migliorare un'industria. In C. di Bona, S. Hockman & M. Stone, *Open sources. Voci dalla rivoluzione Open Source.*

Milano, Apogeo. URL http://www.apogeonline.com/openpress/libri/545/young.html.

Zeltser, L. (1995). *The World Wide Web: Origins and Beyond*. URL http://zeltser.com/web-history/.