

Most Popular R Packages using Spark & R package dependencies visualization

Xavier Capdepon - *Data Scientist*

Bio:

Master in Urban Engineering (Hydraulic & Fluid mechanics)

Master in Corporate Finance

Actuarial Candidate ASA/CAS (passed 2 exams)

NYC Data Science Fellow : Bootcamp #2 – June to August 2015

R, Python, Data Mining, Machine Learning, Hadoop, Spark

Portfolio of 5 projects in Python, R, Spark and Machine learning



NYC DATA SCIENCE
ACADEMY

Projects' blogs:

<http://blog.nydatascience.com/author/chabir/>

Email:

Xavier.Capdepon@gmail.com

LinkedIn:

<https://www.linkedin.com/in/564738960482746278596>

I'm looking for a Data Science / Data Mining Job !!

Project Guidelines:

Time to delivery:

7 days initially (it took 13+ days including 1.5 days to install spark locally...)

Advises:

"You need to deliver a visualization on the due date. You need to rescale your project accordingly."

"It is fine not to build the most efficient code as long as the code works."

"You need to have a story to tell with your visualization."

Jason, NYC Data Science Academy's Teacher

Github: <https://github.com/chabir/Most-Popular-R-packages>

Project Files & Spark installation:

Project Github account:

<https://github.com/chabir/Most-Popular-R-packages>

Useful links to install Spark for IPython on PC:

<http://ysinjab.com/2015/03/28/hello-spark/>

<https://spark.apache.org/downloads.html>

<http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.SparkContext>

To launch Spark in Ipython:

```
from pyspark import SparkContext  
sc = SparkContext(master='local[7]', appName = 'learnSpark')
```

master: Cluster URL to connect to (e.g. mesos://host:port, spark://host:port, local[4]).

"7": corresponds to the number of cpu on local computer

appName: A name for your application, to display on the cluster web UI

Origin of the project:


KDnuggets™ Data Mining, Analytics, Big Data, and Data Science
Subscribe to [KDnuggets News](#) | Follow [@KDnuggets](#) | [Contact](#)

search KDnuggets

[Data Mining Software](#) | [News](#) | [Jobs](#) | [Academic](#) | [Companies](#) | [Courses](#) | [Datasets](#) | [Data Mining Course](#) | [Education](#) | [Meetings](#) | [Polls](#) | [Webcasts](#)

Latest News


- SlideRule: Data Science Course Mentor (online)
- Spark SQL for Real Time Analytics – Part Two
- Upcoming Webcasts on Analytics, Big Data, Data Science - Sep 22 and beyond
- AIMRx: Data Scientist Consultant - Immediate Need
- TMA Predictive Analytics Data Mining Training [San Jose, Dec 7-11]

**DEPAUL UNIVERSITY**
SCHOOL OF COMPUTING

PREDICTIVE ANALYTICS

☒ Innovative Courses
☒ Advanced Skills in Data Mining
☒ Focus on Real World Applications

DePaul U. MS in Predictive Analytics
4 Concentrations



Top 20 R packages by popularity

[f](#) [in](#) [G+](#) 16 [Share](#) 36 [Tweet](#) 188

[◀ Previous post](#) Tags: CRAN, R, R Packages, Top list [Next post ▶](#)

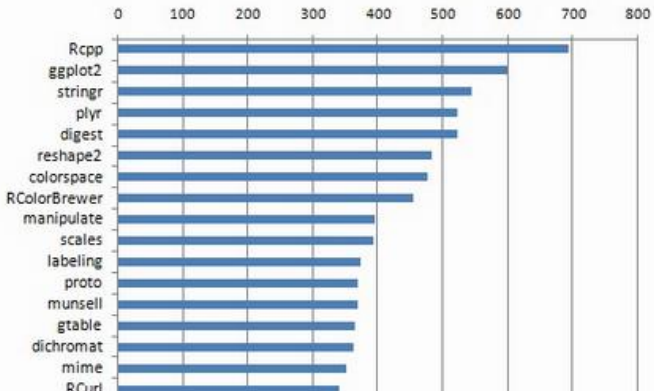
Wondering which are the most popular R packages? Here's an analysis based on most downloaded R packages from Jan to May 2015 to identify the top trending packages in the R world!

[comments](#)


By Bhavya Geethika, June 2015

The [CRAN Package repository](#) features 6778 active packages. Which of these should you know? Here is an analysis of the daily download logs of the CRAN mirror from Jan-May 2015. See a link to full data at the bottom of the post.


Downloads (000) from CRAN , Jan-May 2015



Package	Downloads (000)
Rcpp	700
ggplot2	600
stringr	550
plyr	520
digest	510
reshape2	480
colorspace	450
RColorBrewer	440
manipulate	420
scales	400
labeling	380
proto	370
munsell	360
gtable	350
dichromat	340
mime	330
RCurl	320

 **rapidminer**

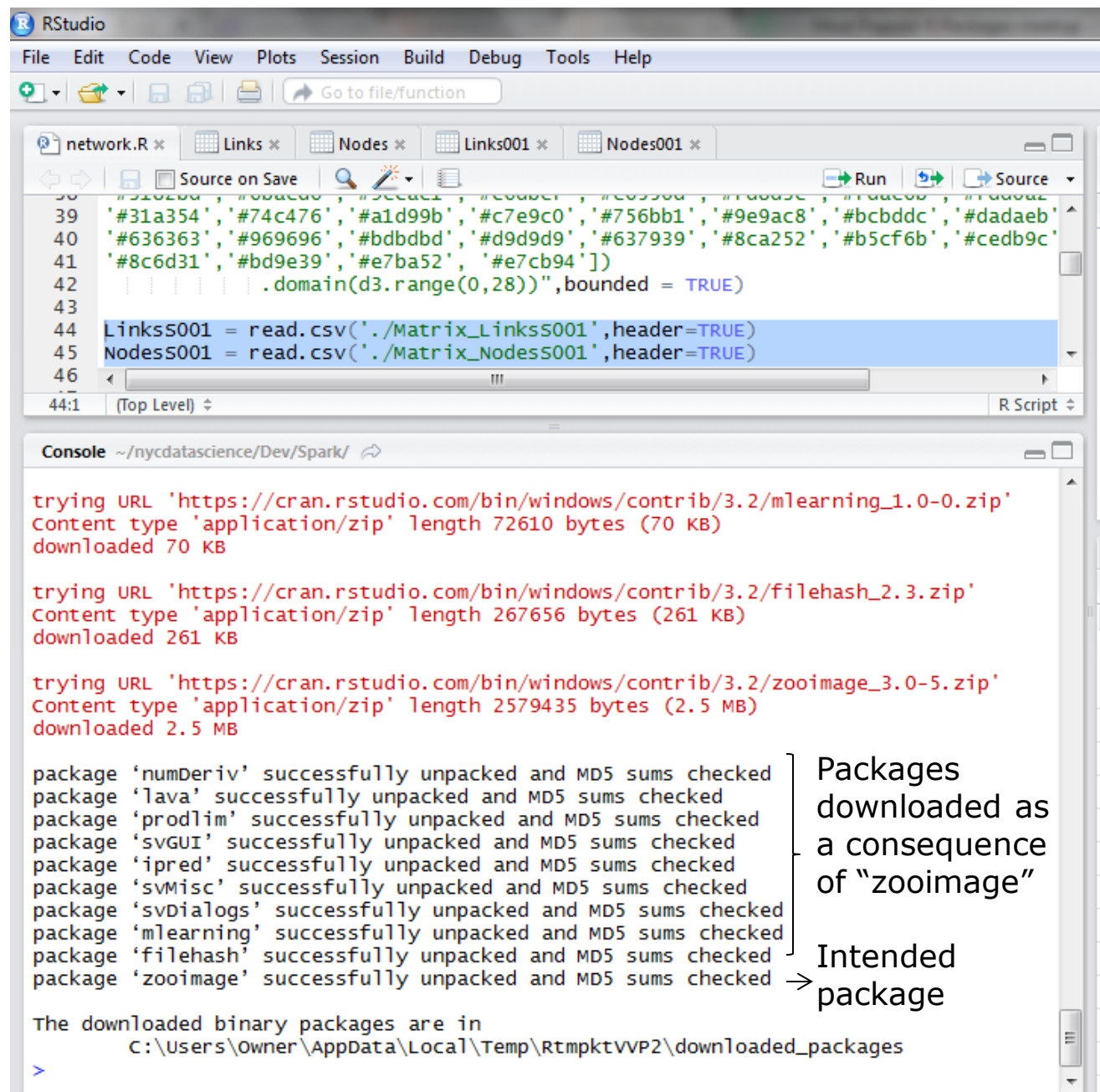
Just Released!
RapidMiner Studio 6.5

[DOWNLOAD NOW](#)
VERSION 6.5 

Source: <http://www.kdnuggets.com/2015/06/top-20-r-packages.html>

The capabilities of R are extended through user-created packages, which, primarily, allow specialized statistical techniques and graphical devices.

Any R user is consistently downloading additional packages on its computer and these packages often depend on other packages that also need to be downloaded as a consequence of the targeted package.



The screenshot shows the RStudio interface. The top pane displays a script named 'network.R' with the following code:

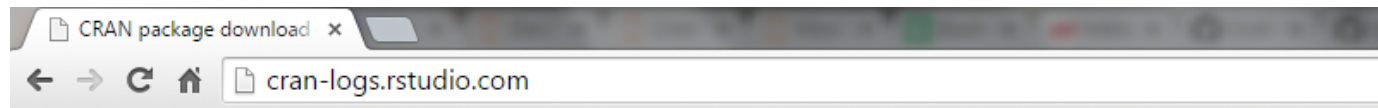
```
39 '#31a354', '#74c476', '#a1d99b', '#c7e9c0', '#756bb1', '#9e9ac8', '#bcbddc', '#dadaeb',  
40 '#636363', '#969696', '#bdbdbd', '#d9d9d9', '#637939', '#8ca252', '#b5cf6b', '#cedb9c',  
41 '#8c6d31', '#bd9e39', '#e7ba52', '#e7cb94'])  
42 .domain(d3.range(0,28))", bounded = TRUE)  
43  
44 Linkss001 = read.csv('./Matrix_Linkss001', header=TRUE)  
45 Nodes001 = read.csv('./Matrix_Nodes001', header=TRUE)  
46
```

The bottom pane shows the console output, which includes the download of several packages as a consequence of installing 'zooimage':

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/mlearning_1.0-0.zip'  
Content type 'application/zip' length 72610 bytes (70 KB)  
downloaded 70 KB  
  
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/filehash_2.3.zip'  
Content type 'application/zip' length 267656 bytes (261 KB)  
downloaded 261 KB  
  
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/zooimage_3.0-5.zip'  
Content type 'application/zip' length 2579435 bytes (2.5 MB)  
downloaded 2.5 MB  
  
package 'numDeriv' successfully unpacked and MD5 sums checked  
package 'lava' successfully unpacked and MD5 sums checked  
package 'prodlm' successfully unpacked and MD5 sums checked  
package 'svgUI' successfully unpacked and MD5 sums checked  
package 'ipred' successfully unpacked and MD5 sums checked  
package 'svMisc' successfully unpacked and MD5 sums checked  
package 'svdialogs' successfully unpacked and MD5 sums checked  
package 'mlearning' successfully unpacked and MD5 sums checked  
package 'filehash' successfully unpacked and MD5 sums checked  
package 'zooimage' successfully unpacked and MD5 sums checked  
  
The downloaded binary packages are in  
C:\Users\Owner\AppData\Local\Temp\RtmpktvVP2\downloaded_packages  
>
```

Annotations on the right side of the console output indicate that the first seven packages (numDeriv through zooimage) were downloaded as a consequence of installing 'zooimage', and that 'zooimage' is the intended package.

Package download logs:



CRAN package download logs

These log files contain all hits to <http://cran.rstudio.com/> related to packages. The raw log files have been parsed into CSV a

Daily package downloads

- Oct 2012: [2012-10-01](#), [2012-10-02](#), [2012-10-03](#), [2012-10-04](#), [2012-10-05](#), [2012-10-06](#), [2012-10-07](#), [2012-10-08](#), [2012-10-09](#), [2012-10-10](#), [2012-10-11](#), [2012-10-12](#), [2012-10-13](#), [2012-10-14](#), [2012-10-15](#), [2012-10-16](#), [2012-10-17](#), [2012-10-18](#), [2012-10-19](#), [2012-10-20](#), [2012-10-21](#), [2012-10-22](#), [2012-10-23](#), [2012-10-24](#), [2012-10-25](#), [2012-10-26](#), [2012-10-27](#), [2012-10-28](#), [2012-10-29](#), [2012-10-30](#), [2012-10-31](#)
- Nov 2012: [2012-11-01](#), [2012-11-02](#), [2012-11-03](#), [2012-11-04](#), [2012-11-05](#), [2012-11-06](#), [2012-11-07](#), [2012-11-08](#), [2012-11-09](#), [2012-11-10](#), [2012-11-11](#), [2012-11-12](#), [2012-11-13](#), [2012-11-14](#), [2012-11-15](#), [2012-11-16](#), [2012-11-17](#), [2012-11-18](#), [2012-11-19](#), [2012-11-20](#), [2012-11-21](#), [2012-11-22](#), [2012-11-23](#), [2012-11-24](#), [2012-11-25](#), [2012-11-26](#), [2012-11-27](#), [2012-11-28](#), [2012-11-29](#), [2012-11-30](#)
- Dec 2012: [2012-12-01](#), [2012-12-02](#), [2012-12-03](#), [2012-12-04](#), [2012-12-05](#), [2012-12-06](#), [2012-12-07](#), [2012-12-08](#), [2012-12-09](#), [2012-12-10](#), [2012-12-11](#), [2012-12-12](#), [2012-12-13](#), [2012-12-14](#), [2012-12-15](#), [2012-12-16](#), [2012-12-17](#), [2012-12-18](#), [2012-12-19](#), [2012-12-20](#), [2012-12-21](#), [2012-12-22](#), [2012-12-23](#), [2012-12-24](#), [2012-12-25](#), [2012-12-26](#), [2012-12-27](#), [2012-12-28](#), [2012-12-29](#), [2012-12-30](#), [2012-12-31](#)
- Jan 2013: [2013-01-01](#), [2013-01-02](#), [2013-01-03](#), [2013-01-04](#), [2013-01-05](#), [2013-01-06](#), [2013-01-07](#), [2013-01-08](#), [2013-01-09](#), [2013-01-10](#), [2013-01-11](#), [2013-01-12](#), [2013-01-13](#), [2013-01-14](#), [2013-01-15](#), [2013-01-16](#), [2013-01-17](#), [2013-01-18](#), [2013-01-19](#), [2013-01-20](#), [2013-01-21](#), [2013-01-22](#), [2013-01-23](#), [2013-01-24](#), [2013-01-25](#), [2013-01-26](#), [2013-01-27](#), [2013-01-28](#), [2013-01-29](#), [2013-01-30](#), [2013-01-31](#)
- Feb 2013: [2013-02-01](#), [2013-02-02](#), [2013-02-03](#), [2013-02-04](#), [2013-02-05](#), [2013-02-06](#), [2013-02-07](#), [2013-02-08](#), [2013-02-09](#), [2013-02-10](#), [2013-02-11](#), [2013-02-12](#), [2013-02-13](#), [2013-02-14](#), [2013-02-15](#), [2013-02-16](#), [2013-02-17](#), [2013-02-18](#), [2013-02-19](#), [2013-02-20](#), [2013-02-21](#), [2013-02-22](#), [2013-02-23](#), [2013-02-24](#), [2013-02-25](#), [2013-02-26](#), [2013-02-27](#), [2013-02-28](#)
- Mar 2013: [2013-03-01](#), [2013-03-02](#), [2013-03-03](#), [2013-03-04](#), [2013-03-05](#), [2013-03-06](#), [2013-03-07](#), [2013-03-08](#), [2013-03-09](#), [2013-03-10](#), [2013-03-11](#), [2013-03-12](#), [2013-03-13](#), [2013-03-14](#), [2013-03-15](#), [2013-03-16](#), [2013-03-17](#), [2013-03-18](#), [2013-03-19](#), [2013-03-20](#), [2013-03-21](#), [2013-03-22](#), [2013-03-23](#), [2013-03-24](#), [2013-03-25](#), [2013-03-26](#), [2013-03-27](#), [2013-03-28](#), [2013-03-29](#), [2013-03-30](#), [2013-03-31](#)
- Apr 2013: [2013-04-01](#), [2013-04-02](#), [2013-04-03](#), [2013-04-04](#), [2013-04-05](#), [2013-04-06](#), [2013-04-07](#), [2013-04-08](#), [2013-04-09](#), [2013-04-10](#), [2013-04-11](#), [2013-04-12](#), [2013-04-13](#), [2013-04-14](#), [2013-04-15](#), [2013-04-16](#), [2013-04-17](#), [2013-04-18](#), [2013-04-19](#), [2013-04-20](#), [2013-04-21](#), [2013-04-22](#), [2013-04-23](#), [2013-04-24](#), [2013-04-25](#), [2013-04-26](#), [2013-04-27](#), [2013-04-28](#), [2013-04-29](#), [2013-04-30](#)
- May 2013: [2013-05-01](#), [2013-05-02](#), [2013-05-03](#), [2013-05-04](#), [2013-05-05](#), [2013-05-06](#), [2013-05-07](#), [2013-05-08](#), [2013-05-09](#), [2013-05-10](#), [2013-05-11](#), [2013-05-12](#), [2013-05-13](#), [2013-05-14](#), [2013-05-15](#), [2013-05-16](#), [2013-05-17](#), [2013-05-18](#), [2013-05-19](#), [2013-05-20](#), [2013-05-21](#), [2013-05-22](#), [2013-05-23](#), [2013-05-24](#), [2013-05-25](#), [2013-05-26](#), [2013-05-27](#), [2013-05-28](#), [2013-05-29](#), [2013-05-30](#), [2013-05-31](#)

Source: <http://cran-logs.rstudio.com/>

Log Screen:

Log Processing using the depend/import matrix

Extract relevant information from package download logs

```
In [122]: import gzip
with gzip.open('data/2012-10-03.csv.gz', 'rb') as f:
    log_content = pd.read_csv(f)
```

```
In [123]: log_content[:10]
```

```
Out[123]:
```

	date	time	size	r_version	r_arch	r_os	package	version	country	ip_id
0	2012-10-03	01:51:54	167303	2.15.1	x86_64	linux-gnu	formatR	0.6	US	1
1	2012-10-03	01:51:54	435497	2.15.1	x86_64	linux-gnu	knitr	0.8	US	1
2	2012-10-03	01:51:53	11150	2.15.1	x86_64	linux-gnu	evaluate	0.4.2	US	1
3	2012-10-03	01:23:01	4977	2.14.1	x86_64	linux-gnu	knn	1.1	US	2
4	2012-10-03	07:41:05	337101	2.15.1	x86_64	linux-gnu	colorspace	1.1-1	US	3
5	2012-10-03	07:40:41	574454	2.15.1	x86_64	linux-gnu	Hmisc	3.9-3	US	3
6	2012-10-03	07:39:59	364811	2.15.1	x86_64	linux-gnu	lattice	0.20-10	US	3
7	2012-10-03	07:37:43	2094445	2.15.1	x86_64	linux-gnu	mosaic	0.6-2	US	3
8	2012-10-03	07:41:05	712307	2.15.1	x86_64	linux-gnu	vcd	1.2-13	US	3
9	2012-10-03	07:42:10	325683	2.15.1	x86_64	linux-gnu	abd	0.2-4	US	3

Proposed methodology:

a. IP_ID[k] downloaded the packages {a, b, c, d, e, f} during a period of 24 hours

b. It happens that package "b" depends on {c, d, e, g}

=> IP_ID[k] downloaded the "root" packages: {a, b}

Then,

c. aggregate all "root" packages: {..., (a,b), ...}

d. Analysis of the list of root packages to determine the most downloaded ones:

Counter({a: k1, b:k2, ...})



Cran
website:

7000+
packages

Web scraping
using Python
and Spark

[CRAN](#)
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

[Documentation](#)
[Manuals](#)
[FAQs](#)
[Contributed](#)

Available CRAN Packages By Name

[A](#)[B](#)[C](#)[D](#)[E](#)[F](#)[G](#)[H](#)[I](#)[J](#)[K](#)[L](#)[M](#)[N](#)[O](#)[P](#)[Q](#)[R](#)[S](#)[T](#)[U](#)[V](#)[W](#)[X](#)[Y](#)[Z](#)

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
ABCanalysis	Computed ABC Analysis
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCOptim	Implementation of Artificial Bee Colony (ABC) Optimization
abcrf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abf2	Load Gap-Free Axon ABF2 Files
abind	Combine Multidimensional Arrays
abn	Data Modelling with Additive Bayesian Networks
abundant	Abundant regression and high-dimensional principal fitted components
acc	Processes Accelerometer Data
accelerometry	Functions for Processing Minute-to-Minute Accelerometer Data
AcceptanceSampling	Creation and Evaluation of Acceptance Sampling

Source: https://cran.r-project.org/web/packages/available_packages_by_name.html

ggplot2: An Implementation of the Grammar of Graphics

An implementation of the grammar of graphics in R. It combines the advantages of both base and lattice graphics: conditioning and shared axes are handled automatically, and you can still build up a plot step implements a sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes. See <http://ggplot2.org> for more information, documentation and examples.

Version: 1.0.1
Depends: R (≥ 2.14), stats, methods
Imports: [plyr](#) (≥ 1.7.1), [digest](#), [grid](#), [etable](#) (≥ 0.1.1), [reshape2](#), [scales](#) (≥ 0.2.3), [proto](#), [MASS](#)
Suggests: [quantreg](#), [Hmisc](#), [mapproj](#), [maps](#), [hexbin](#), [maptools](#), [multcomp](#), [nlme](#), [testthat](#), [knitr](#), [mgcv](#)
Enhances: [sp](#)
Published: 2015-03-17
Author: Hadley Wickham [aut, cre], Winston Chang [aut]
Maintainer: Hadley Wickham <h.wickham@gmail.com>
BugReports: <https://github.com/hadley/ggplot2/issues>
License: [GPL-2](#)
URL: <http://ggplot2.org>, <https://github.com/hadley/ggplot2>
NeedsCompilation: no
Citation: [ggplot2 citation info](#)
Materials: [README NEWS](#)
In views: [Graphics](#), [Phylogenetics](#)
CRAN checks: [ggplot2 results](#)

Downloads:

Reference manual: [ggplot2.pdf](#)
Vignettes: [Contributing to ggplot2 development](#)
[ggplot2 release process](#)
Package source: [ggplot2 1.0.1.tar.gz](#)
Windows binaries: r-devel: [ggplot2 1.0.1.zip](#), r-release: [ggplot2 1.0.1.zip](#), r-oldrel: [ggplot2 1.0.1.zip](#)
OS X Snow Leopard binaries: r-release: [ggplot2 1.0.1.tgz](#), r-oldrel: [ggplot2 1.0.1.tgz](#)
OS X Mavericks binaries: r-release: [ggplot2 1.0.1.tgz](#)
Old sources: [ggplot2 archive](#)

Reverse dependencies:

Reverse depends: [alphanull](#), [AmpliconDuo](#), [aoristic](#), [apsimr](#), [arqas](#), [bde](#), [benchmark](#), [biomod2](#), [bootnet](#), [brms](#), [caret](#), [catenary](#), [chemosensors](#), [CINOEDV](#), [cjoint](#), [ClimClass](#), [climwin](#), [clustrd](#), [coefplot](#), [confo](#), [Deducer](#), [DepthProc](#), [dfexplore](#), [diffR](#), [dMod](#), [dotwhisker](#), [dslice](#), [dtwclust](#), [DymNom](#), [earlywarnings](#), [eetools](#), [ESGtoolkit](#), [fbiroc](#), [fishmove](#), [freeparcord](#), [gapmap](#), [GenCAT](#), [gettingtoth](#), [ggswissmaps](#), [gettern](#), [getthemes](#), [GOplot](#), [gpmmap](#), [granovaGG](#), [gsDesign](#), [GSE](#), [Hmisc](#), [hyperSpec](#), [idm](#), [ifaTools](#), [interplot](#), [learnstats](#), [likeLTD](#), [likert](#), [lmms](#), [localgauss](#), [lsbclust](#), [MCMC](#), [meteoRam](#), [MissineDataGUI](#), [MINFIM](#), [mixOmics](#), [mlr](#), [mixR](#), [mosaic](#), [MRMR](#), [multilevelPSA](#), [ncappc](#), [NeatMap](#), [nullabor](#), [orgr](#), [OriGen](#), [OutbreakTools](#), [PairedData](#), [PASWR2](#), [pauv](#), [pequod](#), [perry](#), [perspective](#), [PhaseType](#), [pid](#), [pipe.design](#), [pitchRx](#), [PKeraph](#), [PKreport](#), [pointRes](#), [PopED](#), [popgraph](#), [PPTreeViz](#), [precuncon](#), [prevR](#), [PRISMA](#), [profileR](#), [ProgGUimR](#), [PSAb](#), [radiant](#), [RAM](#), [randomizeR](#), [Rcell](#), [RcmdrPlugin.KMespplot2](#), [rtPermute](#), [RGraphics](#), [RIGHT](#), [RJafroc](#), [rms](#), [robustHD](#), [rorutadis](#), [rotations](#), [rplos](#), [RSA](#), [RSDA](#), [rstan](#), [Rz](#), [SciencesPo](#), [sea](#), [SmarterPoland](#), [SMFIS](#), [snht](#), [soc.ca](#), [sparkTable](#), [SparseFactorAnalysis](#), [sparsereg](#), [spcosa](#), [spikeSlabGAM](#), [spoccutils](#), [sprm](#), [statebins](#), [SWMP](#), [synthpop](#), [tcR](#), [tdr](#), [timeline](#), [TriMatch](#), [Ti](#), [varian](#), [vde](#), [waffle](#), [walker](#), [xkcd](#), [zooraRch](#)
Reverse imports: [ACDm](#), [adezenet](#), [alm](#), [ANOM](#), [antitrust](#), [asremiPlus](#), [asVPC](#), [BACA](#), [backShift](#), [bamdit](#), [BBEST](#), [berm](#), [bdscale](#), [bdvis](#), [BioStatR](#), [blowtorch](#), [bmimix](#), [breakpoint](#), [broman](#), [BTSPAS](#), [capr](#), [choroplethr](#), [choroplethrAdmin1](#), [classify](#), [classfire](#), [clhs](#), [clifro](#), [CommT](#), [complanrob](#), [confidence](#), [cooccur](#), [cosmor](#), [CosmoPhotoz](#), [cplm](#), [cutoffR](#), [dcnr](#), [DescribeDisplay](#), [DFIT](#), [diveRsi](#), [dmsury](#), [EasyHTMLReport](#), [EcoGenetics](#), [edgar](#), [EffectLiteR](#), [ega](#), [egcm](#), [emil](#), [EpiDynamics](#), [eror](#), [evolug](#), [extracat](#), [ez](#), [ezsum](#), [FAOSTAT](#), [heatmap](#), [FinCal](#), [forestFloor](#), [FSRM](#), [G2Sc](#), [ggenealogy](#), [ggExtra](#), [ggparallel](#), [ggRandomForests](#), [ggsubplot](#), [gitter](#), [glvcanr](#), [googlesheets](#), [GraphPCA](#), [greport](#), [growcurves](#), [growfunctions](#), [hierarchicalDS](#), [HighDimOut](#), [HistDAWass](#), [InformationValue](#), [IntegratedJM](#), [intsvy](#), [kdetrees](#), [kobe](#), [kdatuning](#), [llama](#), [lmerTest](#), [LocFDRPois](#), [lsl](#), [mapDK](#), [marked](#), [marketeR](#), [marmap](#), [MAVIS](#), [MaxentVariableSelection](#), [merTools](#), [microbenchmark](#), [micromap](#), [mizer](#), [Mobilize](#), [morse](#), [multiDimBio](#), [MultiMeta](#), [myTAI](#), [netgen](#), [networkreporting](#), [NeuralNetTools](#), [ngramr](#), [NMF](#), [NORRRM](#), [oaxaca](#), [OpasnetUtils](#), [opti](#), [partialAR](#), [patPRO](#), [performanceEstimation](#), [plot2erouns](#), [PlotPriNetworks](#), [plotROC](#), [omc](#), [pozit](#), [PopGenReport](#), [popur](#), [predictmeans](#), [PRemiuM](#), [pRF](#), [primerTree](#), [proteomics](#), [pscore](#), [c](#)

} “import” and “depends”:
ggplot2 relies on these
packages to run.

“reverse depends” and
“reverse imports”: these
packages relies on ggplot2
to run.

We'll try to verify that ("Depends" Matrix) = t("Reverse Depends" Matrix)

Source: <https://cran.r-project.org/web/packages/ggplot2/index.html>

Programming in 3 steps:

1. Build Matrix of dependencies
2. Download all logs
3. Process all records to extract the “root” packages

Numbers associated with this project:

1. 7055 packages => 7055 webpages to scrap
2. Dependency matrix: 7055×7055 => 49+ Million elements to determine
3. 150 days of package log to download for a size of 2.5 GB
4. Every log is about 300,000 lines x 10 columns
=> 45+ Million line items to process

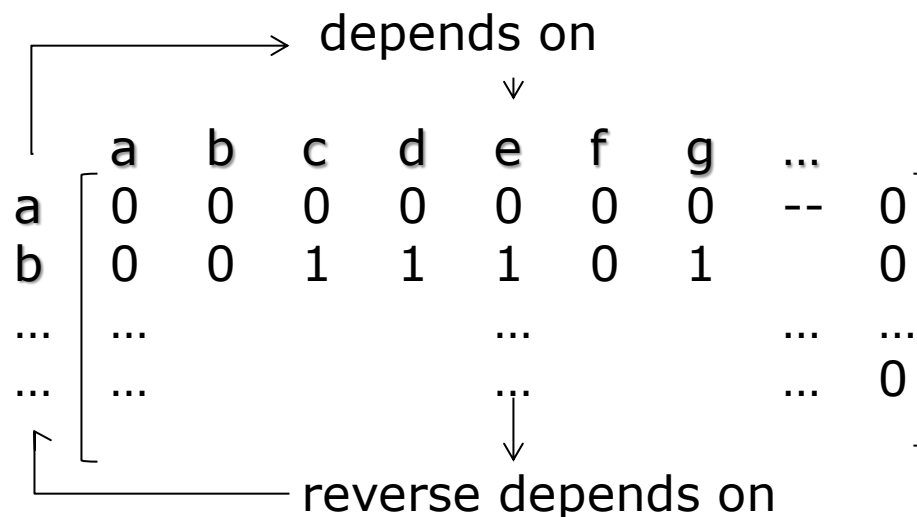
1. Script steps:

- List of all packages
- Build 2x Dictionaries: index -> name & name-> index: reduce memory and file size
- Web-scraping & Regular expression to grab the "depends", "imports", "reverse depends", "reverse imports" information
- Dependency matrix construction

How does the matrix construction works ?

- Package "a" doesn't depends on any other packages
- Package "b" depends on packages {c, d, e, g}

Matrix Size : 7055x7055



2. "Sparse Matrix":

Simplification of the matrix construction: **"Sparse Matrix"**

1. Package "a" doesn't depends on any other packages
2. Package "b" depends on packages {c, d, e, g}

```
## dummy matrix (index(a) = 0, index(b)=1)
rows = [1,1,1,1,...,n] #sources for node visualization
cols  = [2,3,4,6,...,k] #targets for node visualization

data = [1,1,1,1,...,1]
```

Sparse Matrix is not a mathematical concept but a computer science concept for matrix where most of the element are zeros.

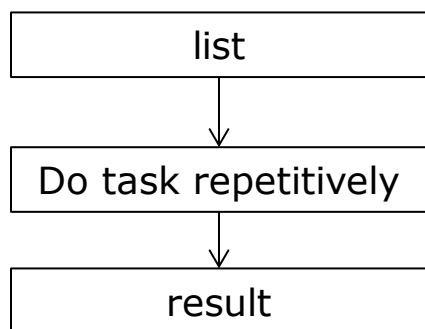
```
# Build a full matrix using coo_matrix scipy package
from scipy.sparse import coo_matrix
Full_matrix = coo_matrix((data, (row, col)), shape=(7055, 7055)).toarray()

#Retrieve the dependant packages index:
col_index = set(Full_matrix[1704].nonzero()[0])
```

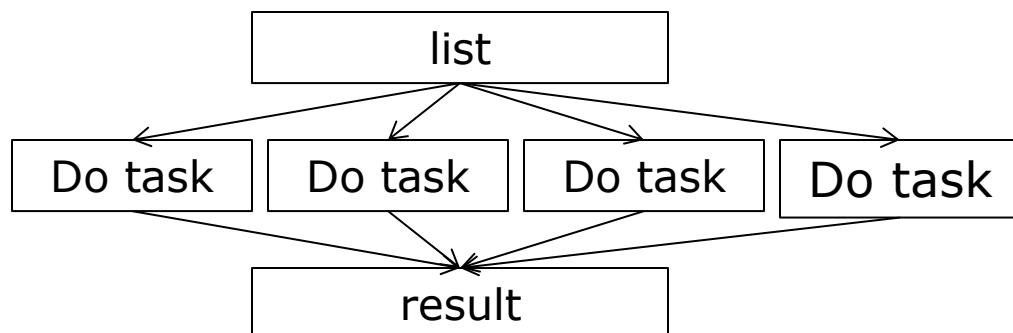
coo_matrix: a sparse matrix in COOrdinate format: 1. facilitate conversion between sparse matrix format 2. permits duplicate entries

3. Python versus Spark

Basic concept of Python vs Python + Spark:



```
for k in list:  
    do task
```



```
from pyspark import SparkContext  
sc = SparkContext(master='local[7]',  
    appName = 'learnSpark')
```

```
list_RDD = sc.parallelize(list)  
result = list_RDD.map(task_fct).collect()
```

4. Results

Matrix construction by web-scraping of the 7,055 pages:

- Size of "depends" dependancy matrix: 14,196 elements
- Size of "reverse" dependancy matrix: 14,458 elements
 - > sparse matrix length: 28,654 elements
 - > coo_matrix (constitution of the full matrix and removal of duplicates)
 - >>> sparse matrix final: 14,435 elements

Time to process:

- 40 min on Spark on the "82" school server with 7 processors after 11pm
- Calculated approx. 8 hours with my personal computer

5. Findings:

A. 3 Most dependent R packages:

Vmsbase: GUI Tools to Process, Analyze and Plot Fisheries Data	27
PopGenReport: A Simple Framework to Analyse Population Genetic Data	23
RAM: R for Amplicon-Sequencing-Based Microbial-Ecology	23

(depends on # packages)

B. 3 most reverse dependent R packages:

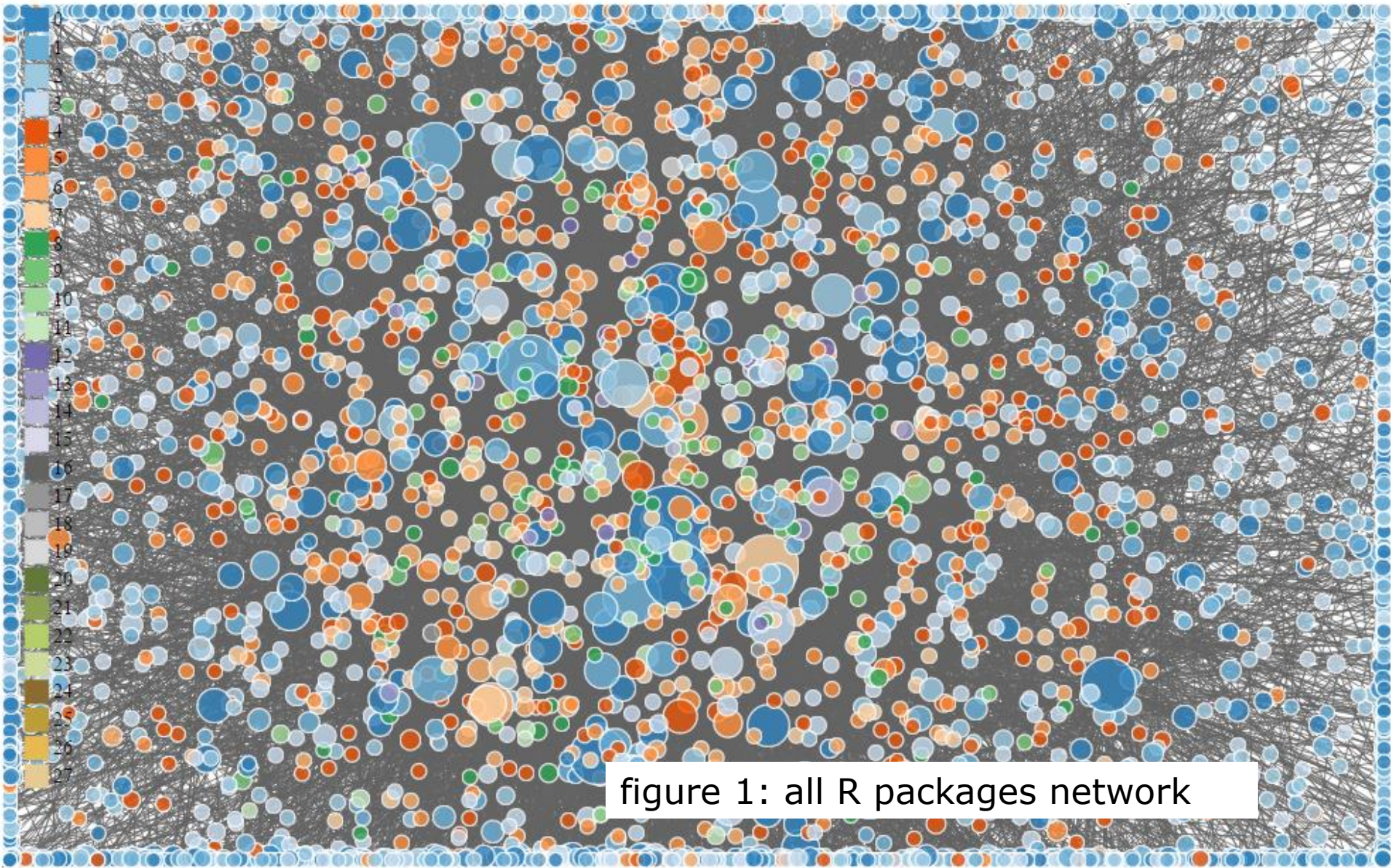
MASS: Support Functions and Datasets for Venables and Ripley's Modern Applied Statistic with S	679
Rcpp: Seamless R and C++ Integration	416
ggplot2: An Implementation of the Grammar of Graphics	351

(# other packages depends this package)

C. Longest path between R packages: 12 packages

The longest path is interesting as it can serve as an upper limit in the iteration to build a n-degree dependency matrix.

6. Visualization:



Visualization using R networkD3 library
(modification of color panel in Java script)

Color code:
of dependents



Circle Size:
Reverse dependent
(# of other
packages based on
this package)

figure 2: 3 most reverse dependent R packages:
MASS, Rcpp, ggplot2

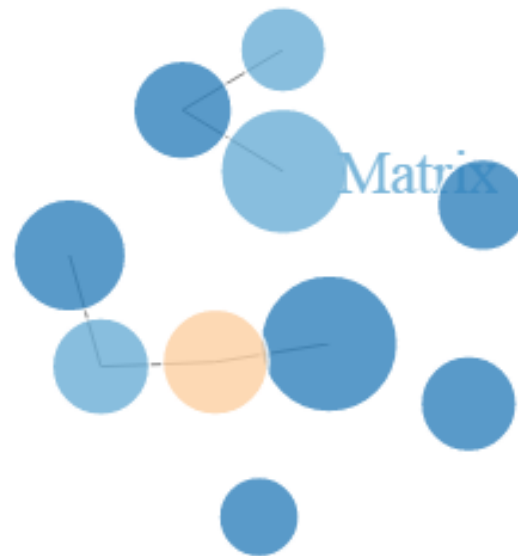
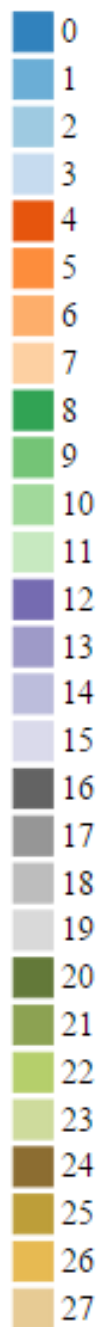
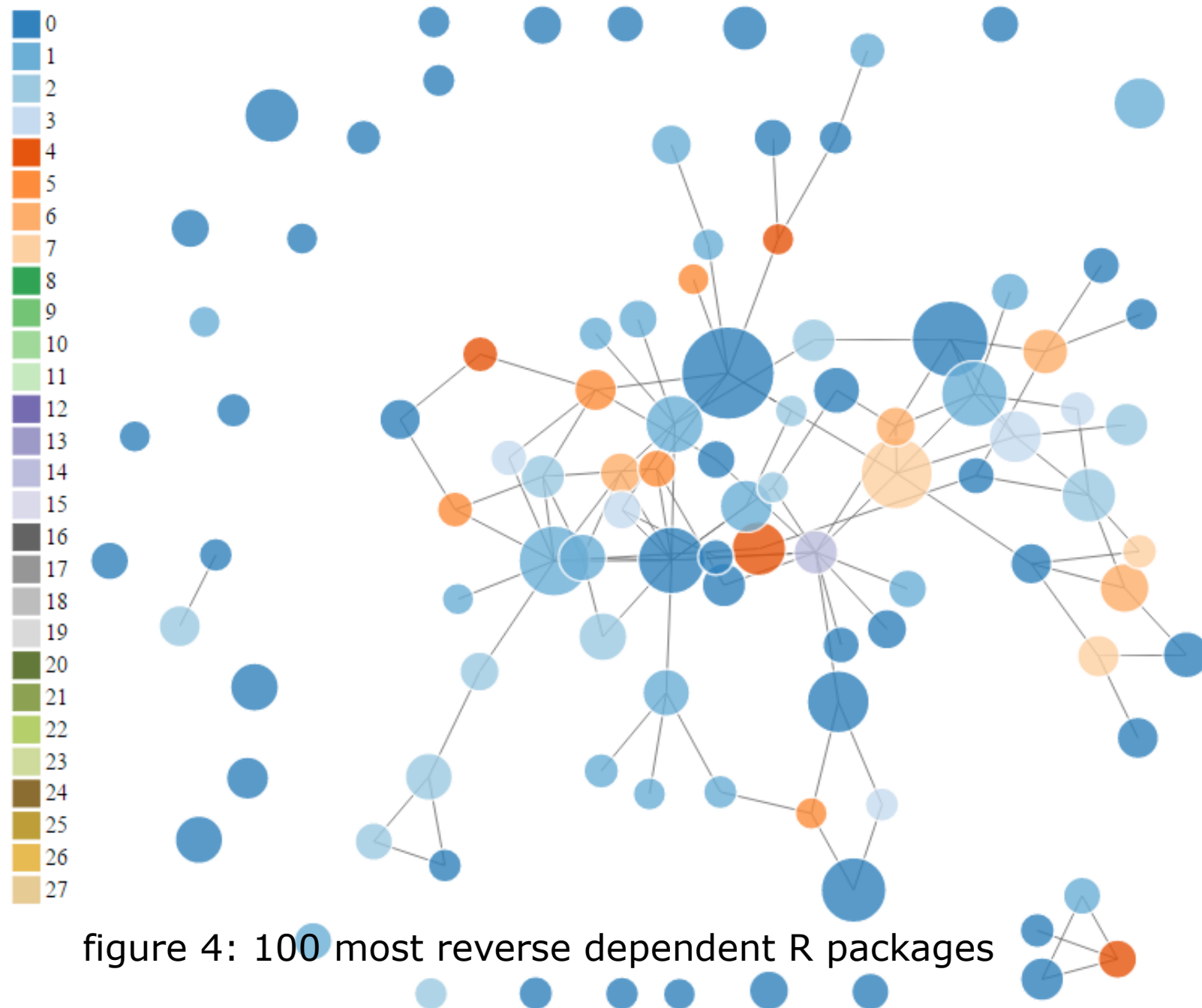


figure 3: 10 most reverse dependent R packages



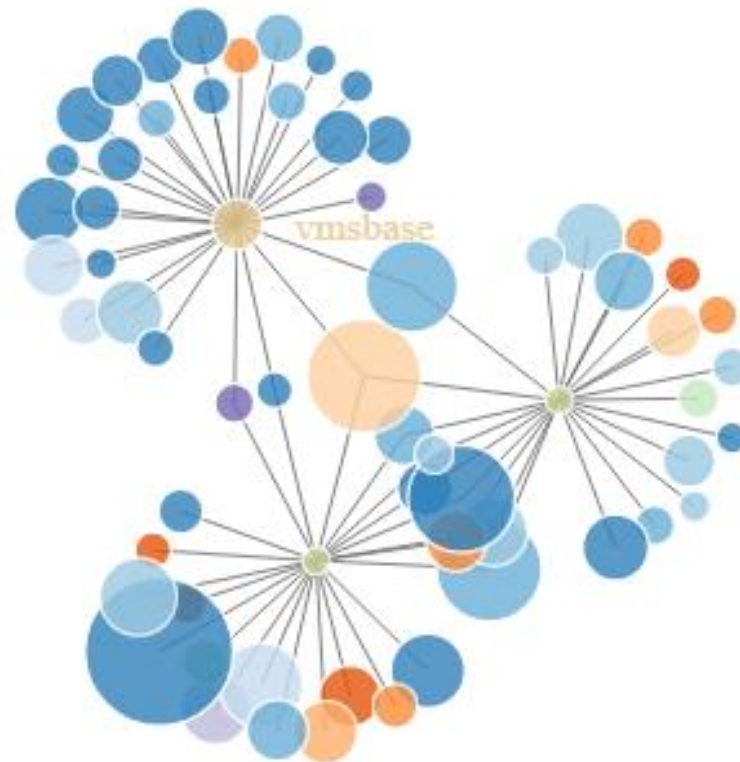
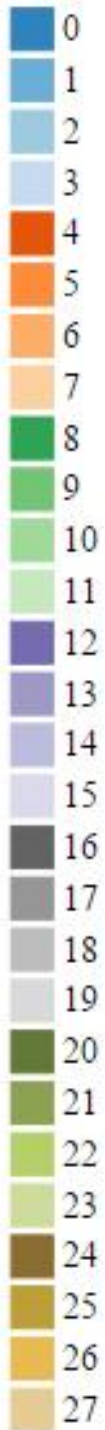


figure 5: 3 most dependent R packages

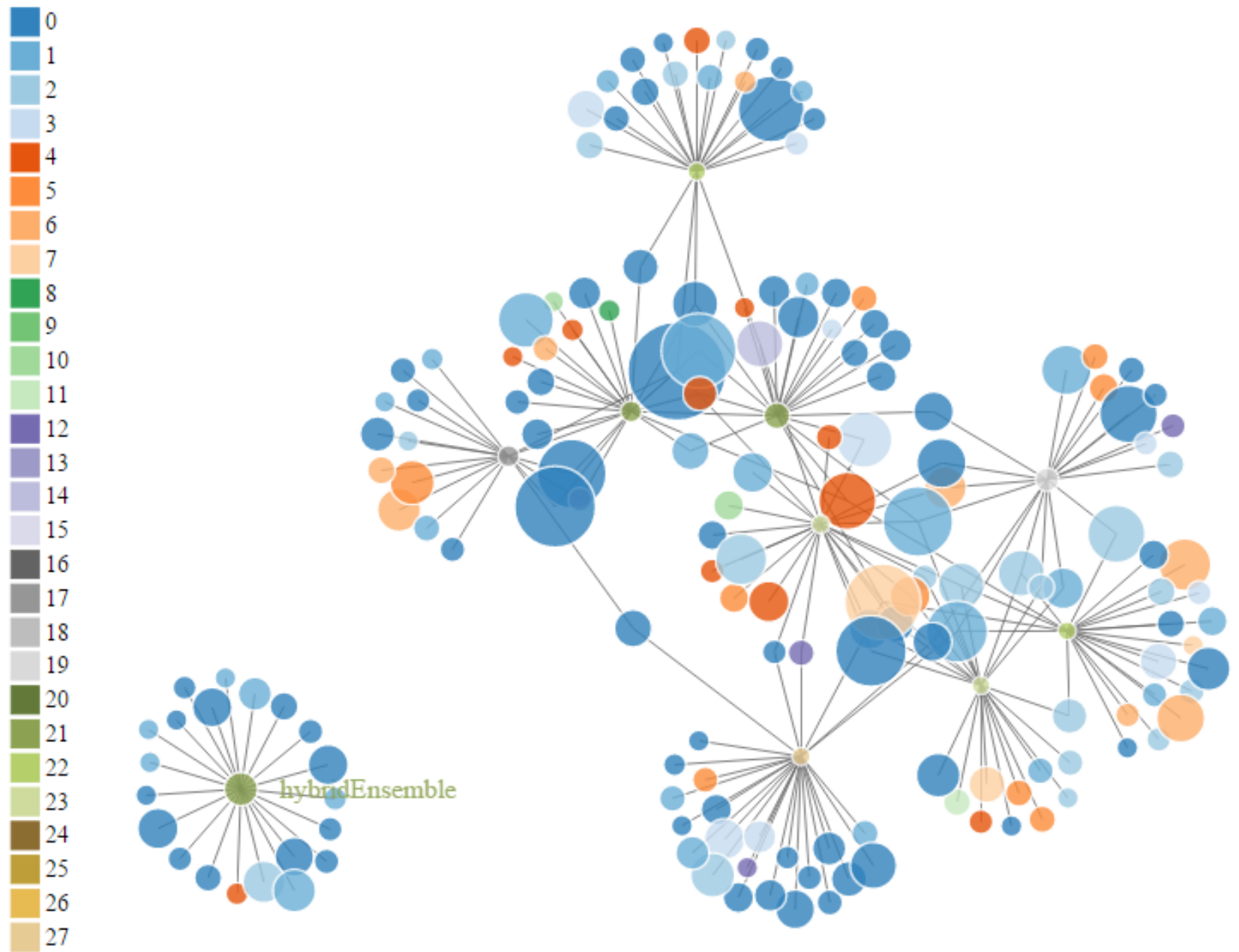
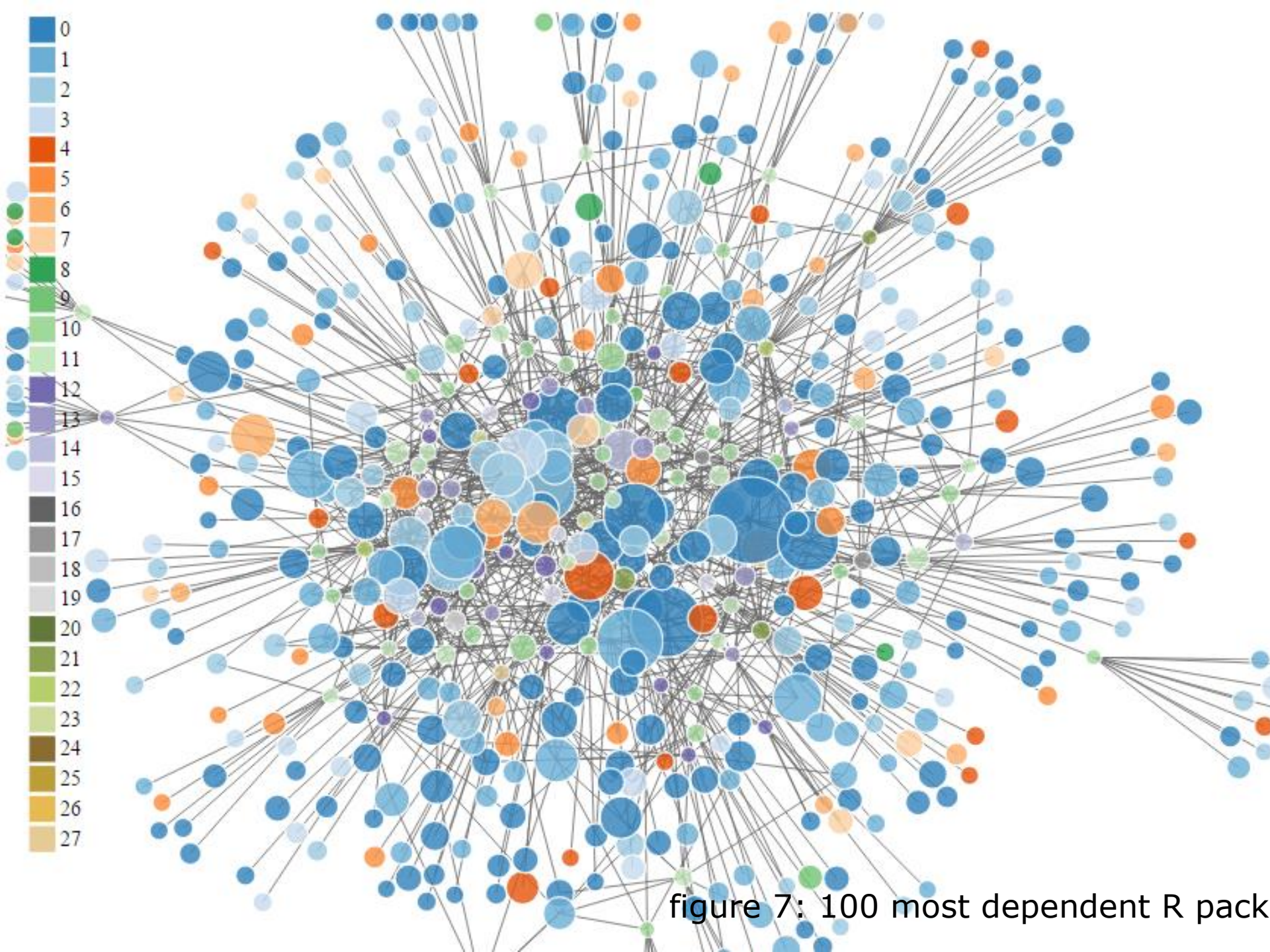
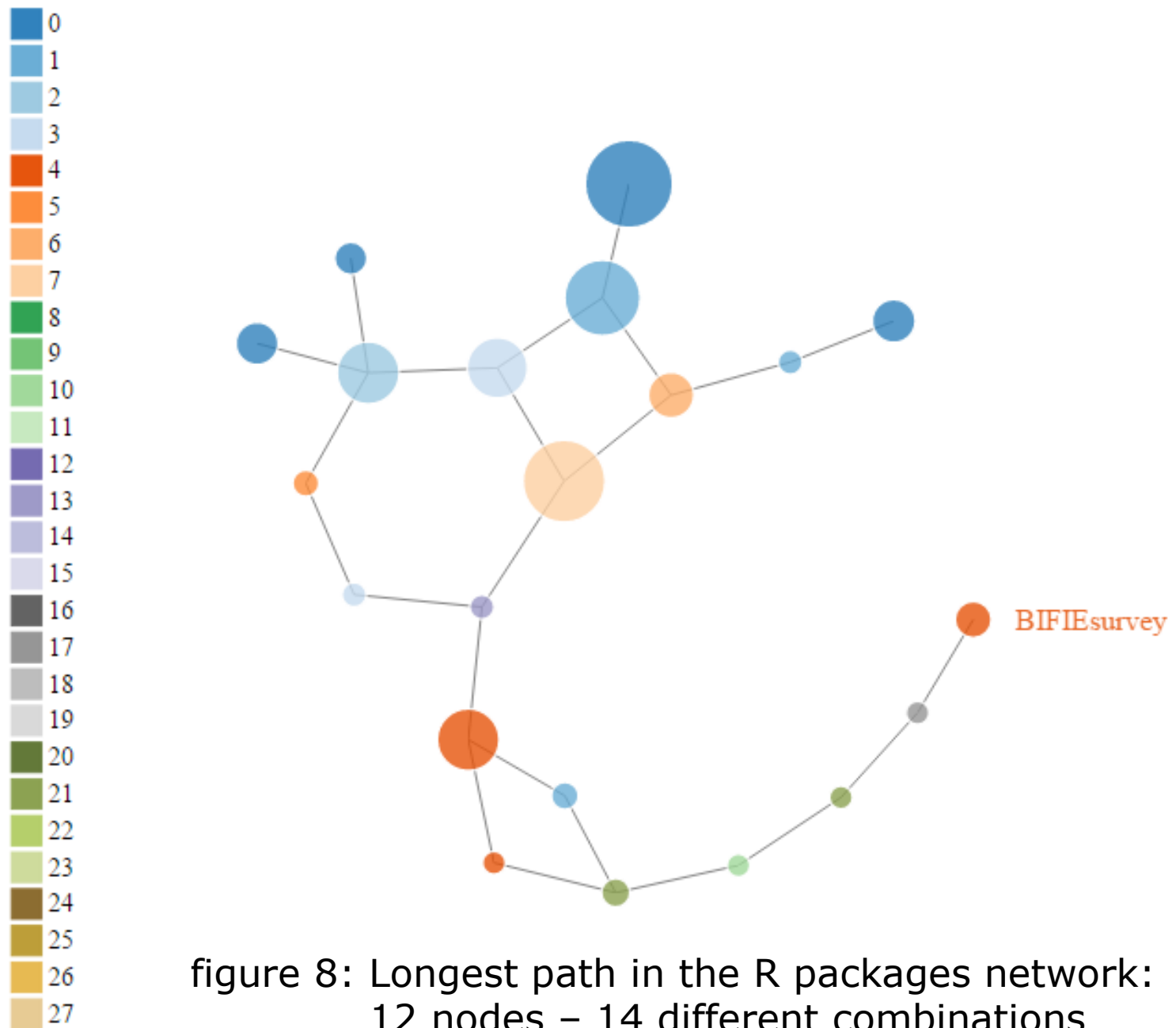


figure 6: 10 most dependent R packages





II. Download the logs:

In []:

```
# Before running, update url_log_full_paths file.

#####
### FINAL SCRIPT FOR SPARK TO DOWNLOAD LOG FILES
#####

#from pyspark import SparkContext
#sc = SparkContext(master='local[7]', appName = 'LearnSpark')

import numpy as np
from bs4 import BeautifulSoup
import requests
import re
import csv

# fonction to download and save the log files / ready for parallelisation:
import urllib
def download_log(url_log):

    path_with_log_name = './data/' + re.sub('(http:\\\\cran-logs.rstudio.com\\/([0-9]+\\/))','',url_log)
    urllib.urlretrieve(url_log, path_with_log_name)

url_log_full_paths = [line.strip() for line in open("url_log_full_paths.txt", 'r')]

print 'Spark started'

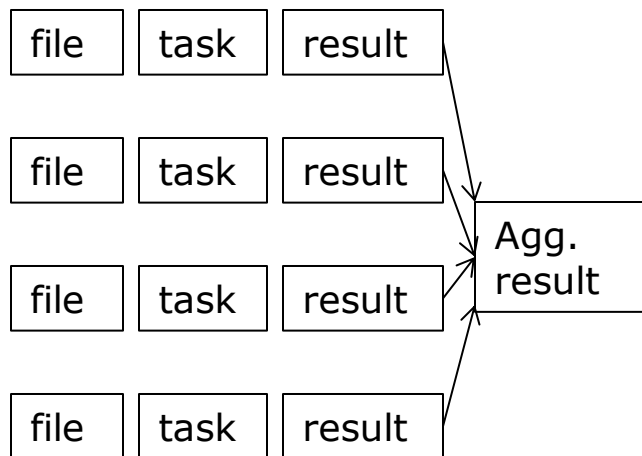
doc_log= sc.parallelize(url_log_full_paths[1003:1007]) #1004:1054
doc_log.map(download_log).collect()

print 'Spark finished'
```

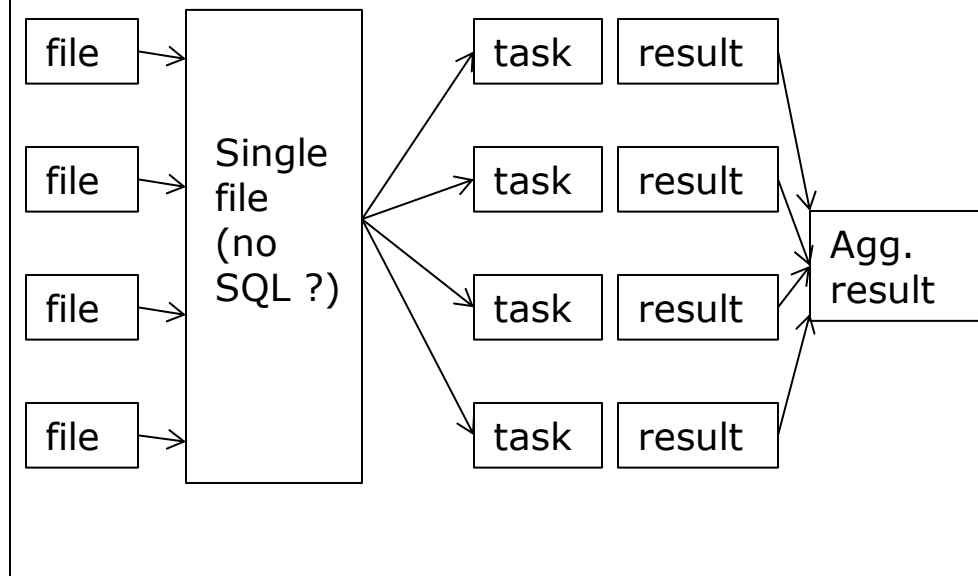
1. Process the logs

1. Construct a unique ID = IP+Date
2. List all packages downloaded by unique ID
3. Save "root" packages by using the dependency matrix

Adopted Solution using Spark



Other Solution using Spark



2. Results

Jan to May 2015:

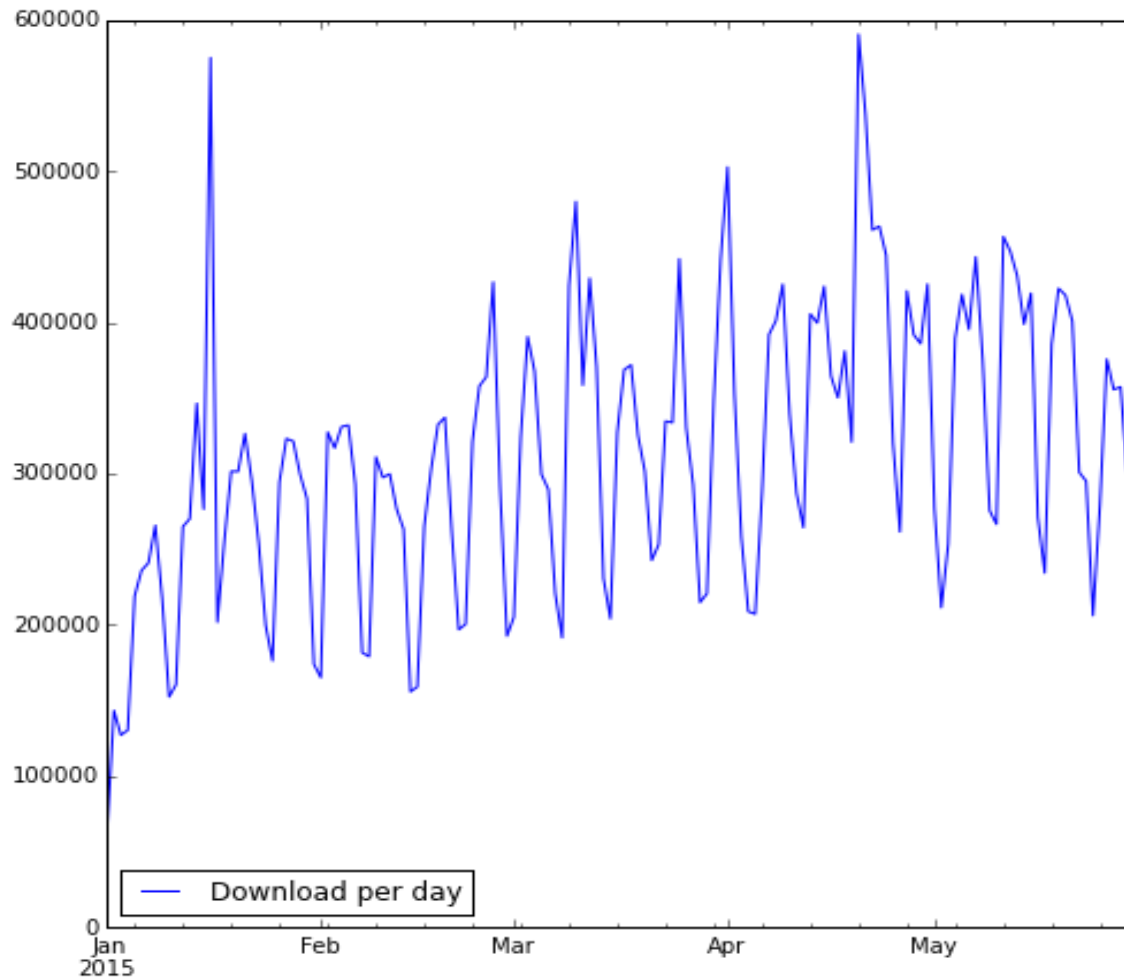
- 46,944,117 lines items to process
- Took 5 to 9 hours using Spark 10 cpu **overnight**

Results:

- “Root” package list: 18,716,714 (100MB)

Trick: all scripts are processed and are saving result using the package indexes to reduce file sizes and process time.

3. Finding #1: Download per day



Note:

Y2012 logs
contains about
100
downloads /
day

Vs

Y2015 logs
contains about
300,000
downloads /
day

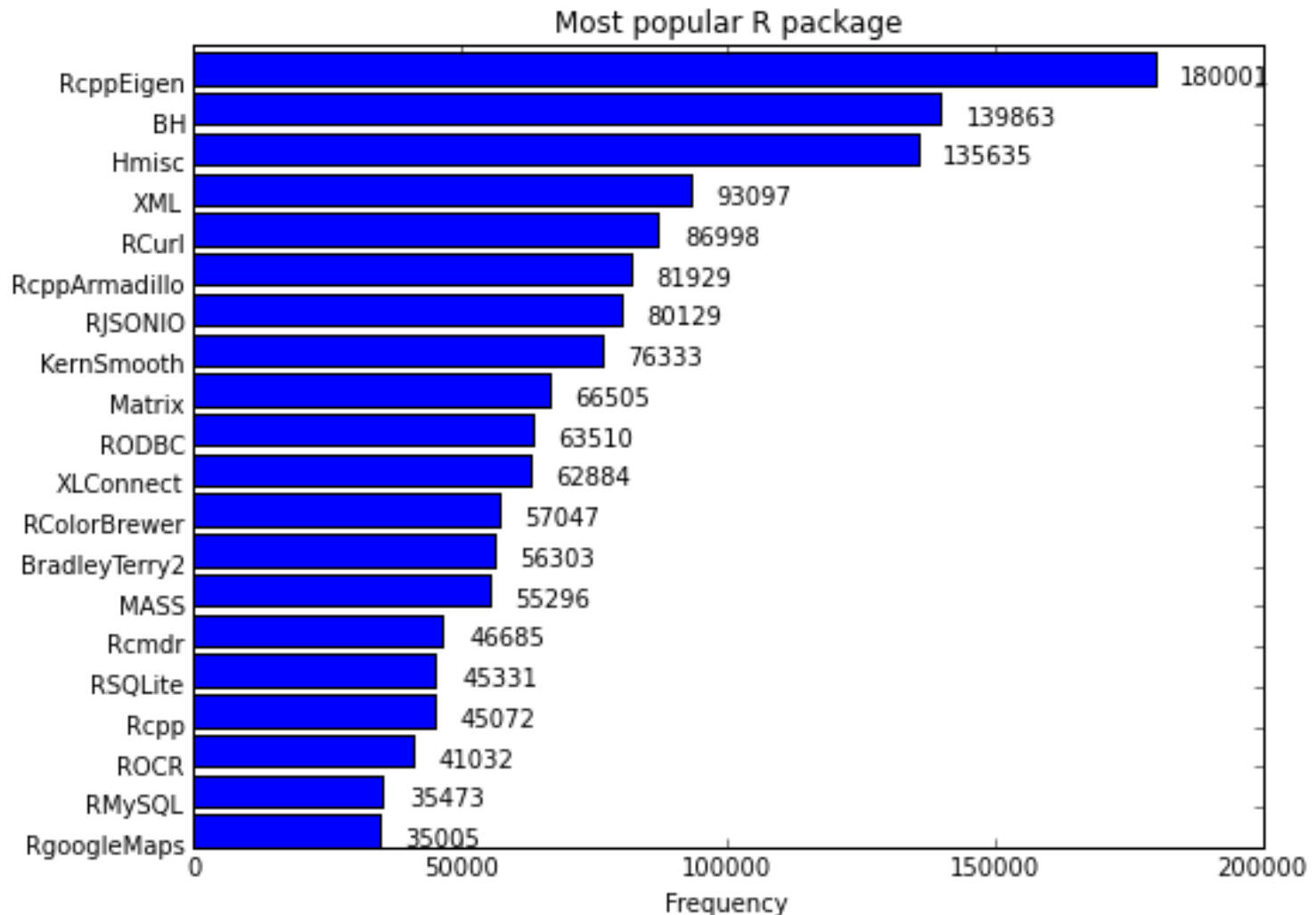
3. Finding #2: “CRAC” package

“CRAC”, Cosmology R Analysis Code Package,
developed by our excellent teacher Jason (Jiayi Liu).

The “CRAC” has been downloaded 1,423 times between
January and May 2015

“CRAC” is ranked 1,170th according to the present
methodology.

4. The most popular R packages



Studied period: January to May 2015

The testing took much more time than anticipated because:

- The CRAN website is updated every night (the package count changes by +/- 20 per day and tends to increase over time)
- The package authors have a tendency to increase the number of package functionalities by increasing the dependency of their own package.

Skepticism about my methodology:

- I used only 1st degree of dependency: I shall redo the calculation by increasing the degree of dependency, now that I calculated the longest path in the R package network (used as an upper limit in the iteration).
- Time difference between the R “day time” and foreign R users
=> inaccuracy of the unique ID (day+IP)
- Old packages in the logs are not referenced anymore by the Cran website