

# Modern R Quick Start for System & Network Administrators

Part II of “Diagnosing Multi-Vendor Network Issues with an  
Introduction to R for System & Network Administrators”

Jon Meek  
meekjt (at) gmail.com  
meekj (at) ieee.org

4-Feb-2016 / LOPSA-NJ

# Why R?

- The standard statistics package in academia and industry
- Lots of users, lots of packages (7,845 on CRAN, at last count)
- Can be very fast - vectorized functions & more
- Interactive data analysis
- Batch jobs as well
- IEEE language ranking, #6 in 2015, up from #9
- R Consortium - Microsoft, Google, Oracle, HPE, ...

# Why Not R?

- Steep learning curve - Still true ?
- In general, all data need to be in memory
- Can be very slow
- Have to be careful when programming, e.g. loops are slow
- But, can always use C++ (Rcpp package)
- Command line flexibility is lacking
- Competition
  - ▶ SciPy / IPython - Probably good for Python experts
  - ▶ Julia - Supposedly fast, but Rcpp may be the more versatile choice

# Using R in the Modern Way

- This is a “Quick Start” guide and “Roadmap”
- You will need additional reference material, soon
- “It’s just one man’s opinion” F. Sinatra
- Avoid Many Base Features:
  - ▶ Base graphics
  - ▶ Functions: lapply, sapply, tapply, aggregate, subset, ...
- Avoid specific time series objects and methods (zoo, xts)
- Use the Hadleyverse!
- These suggestions should save time and reduce future refactoring

# The Hadleyverse

- Extremely popular packages by Hadley Wickham
  - ▶ RStudio, Rice University, R Foundation
- ggplot2 - Flexible and beautiful plotting, but avoid qplot
- dplyr - Data wrangling (avoid the older plyr)
- readr - Flexible (mostly) and fairly fast data file reading
- tidyr - Reshape data, long or wide
- lubridate - Date & time manipulation, but try the base functions first
- stringr - But try the base string functions first

# Running R

- Interactive data analysis & development
  - ▶ ESS - Emacs Speaks Statistics
  - ▶ RStudio - Best for non-Emacs users
  - ▶ In both, code can be selected then executed
- Running batch jobs
  - ▶ R CMD Sweave `~/wpl/talks/lopsanj-2016-1/modernR.Rnw`
  - ▶ `#!/usr/local/bin/Rscript`
  - ▶ `#!/usr/bin/env /usr/local/bin/r` - “littler” better CLI
- Reports
  - ▶ R +  $\text{\LaTeX}$  + Sweave  $\rightarrow$  publication quality PDF
  - ▶ R + Markdown + knitr  $\rightsquigarrow$  nice HTML output
- Interactive Web Applications
  - ▶ Shiny - Full featured interactive applications
  - ▶ ggvis, GoogleVis, etc - JavaScript active graphics

## Example 1 - Small Number of Data Points - Inline

```
IPsraw <- 'Time tcp_syn.dropped tcp_syn.forwarded
2015-09-17T21:33 49050 48483
2015-09-17T21:36 87309 85551
2015-09-17T21:39 163092 99578
2015-09-17T21:42 247875 114235
2015-09-17T21:45 335129 129098
2015-09-17T21:48 405430 135635
2015-09-17T21:51 473972 143137
2015-09-17T21:54 541985 149912
2015-09-17T21:57 651037 158139
2015-09-17T22:00 759434 166562
2015-09-17T22:03 812341 170244
2015-09-17T22:06 877437 174047
2015-09-17T22:09 942562 177693
2015-09-17T22:12 987158 180936'
```

## Example 1 - “Read” data and plot

```
library(readr)
library(ggplot2)

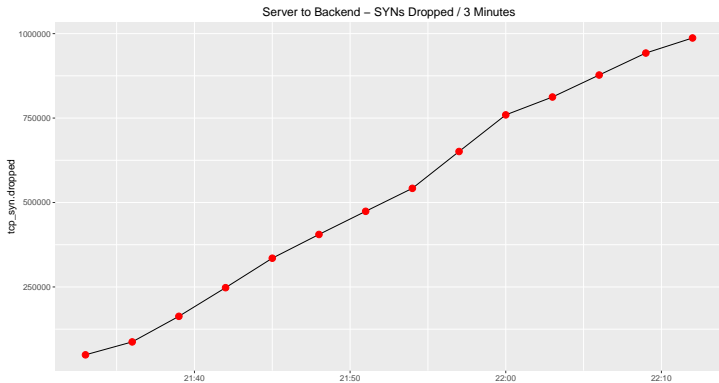
IPS <- read_delim(IPSraw, delim = ' ',
  col_types = list(Time = col_datetime('%Y-%m-%dT%H:%M'))))

PointSize <- 1.0 # Also used in Example 2 below

Title <- 'Server to Backend - SYNs Dropped / 3 Minutes'

ggplot(IPS) +
  geom_line(aes(x = Time, y = tcp_syn.dropped), size=0.1) +
  geom_point(aes(x = Time, y = tcp_syn.dropped),
    size=PointSize + 2, color = 'red', shape=19) +
  xlab('') +
  ggtitle(Title)
```





Example 1 Plot

# Have a Look at the Data

Just type variable name at the R prompt:

```
> IPS
```

```
Source: local data frame [14 x 3]
```

	Time (time)	tcp_syn.dropped (int)	tcp_syn.forwarded (int)
1	2015-09-17 21:33:00	49050	48483
2	2015-09-17 21:36:00	87309	85551
3	2015-09-17 21:39:00	163092	99578
4	2015-09-17 21:42:00	247875	114235
5	2015-09-17 21:45:00	335129	129098
6	2015-09-17 21:48:00	405430	135635
7	2015-09-17 21:51:00	473972	143137
8	2015-09-17 21:54:00	541985	149912
9	2015-09-17 21:57:00	651037	158139
10	2015-09-17 22:00:00	759434	166562
11	2015-09-17 22:03:00	812341	170244
12	2015-09-17 22:06:00	877437	174047
13	2015-09-17 22:09:00	942562	177693
14	2015-09-17 22:12:00	987158	180936

# Have a Look at the Data

Use the `str()` function to see the internal structure of the R object:

```
> str(IPS)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame': 14 obs. of  3 variables:
```

```
$ Time          : POSIXct, format: "2015-09-17 21:33:00" "2015-09-17 21:36:00" "2015-09-17 21:39:00" "2015-
$ tcp_syn.dropped : int  49050 87309 163092 247875 335129 405430 473972 541985 651037 759434 ...
$ tcp_syn.forwarded: int  48483 85551 99578 114235 129098 135635 143137 149912 158139 166562 ...
```

- This is a Data Frame
  - ▶ Usually the best way to organize and operate on data sets
  - ▶ Each column can be a different type
- Note that Time is POSIXct
  - ▶ POSIXct is best format for Date + Time
  - ▶ Avoid POSIXlt, and the special time series types and packages: (ts, zoo, xts, etc)

# Notes on Reading Data

- We used `read_delim` from the `readr` package
- Warning: `read_delim` with `delim = ' '` does not tolerate multiple spaces between fields
- Base R's `read.table` would have worked, but
  - ▶ It would have made `Time` be a “factor” data type
  - ▶ Converting to `POSIXct` would require one (ugly) line of code
  - ▶ `read.table` has no issue with multiple whitespace characters
- `readr` functions are much faster than Base R `read.table` and friends
- For extremely large data sets check out `fread` from the `data.table` package

## Example 2 - Data Wrangling - The Data File

UNIX Seconds - Server Name - %CPU - %Memory - Concurrent Users

```
utime Server CPU Memory Users
1447718220 chn-bc-sg-01 4 22 542
1447718220 chn-bc-sg-02 6 22 510
1447718220 lon-bc-sg-01 8 21 1806
1447718220 lon-bc-sg-02 9 20 1566
1447718220 tok-bc-ce-01 2 35 331
1447718220 tok-bc-sg-01 23 24 1714
1447718220 snd-bc-sg-01 8 24 405
1447718220 snd-bc-sg-02 8 23 360
1447718220 bos-bc-ce-03 1 17 0
1447718220 bos-bc-ce-04 1 24 0
1447718220 bos-bc-drp-01 24 55 11166
1447718220 bos-bc-drp-02 23 55 11182
1447718220 bos-bc-rp-01 10 16 74
1447718220 bos-bc-sg-01 15 23 1675
1447718220 bos-bc-sg-02 15 20 1631
1447718220 bos-bc-sg-03 14 20 1503
1447718220 sng-bc-ce-01 1 25 218
1447718220 nyc-bc-drp-01 0 20 0
1447718220 nyc-bc-drp-02 0 20 0
1447718220 nyc-bc-rp-01 2 24 0
1447718220 nyc-bc-rp-02 1 16 1
1447718220 nyc-bc-sg-01 14 21 1906
1447718220 nyc-bc-sg-02 20 22 1873
1447718220 nyc-bc-sg-03 12 22 1684
1447718220 syd-bc-ce-01 18 65 479
... 53,278 lines of data
```

We will select only the \*-drp-\* server data

## Example 2 - Data Wrangling - The Code

```
library(dplyr) # Provides %>% and functions used for filtering and aggregation - See Data Wrangling Cheatsheet

site_servers <- list() # Servers of interest
site_servers[["nyc"]] <- c("nyc-bc-drp-01", "nyc-bc-drp-02") # A form of hash
site_servers[["bos"]] <- c("bos-bc-drp-01", "bos-bc-drp-02")

servers_of_interest <- unique(unlist(site_servers)) # Vector of server names

File <- '/home/meekj/wpl/talks/lopsanj-2016-1/si20151117.dat'

bc <- read.table(File, header = TRUE) # Base R read.table
bc <- bc %>% filter(Server %in% servers_of_interest) # Keep only servers of interest
bc$Time <- as.POSIXct(bc$utime, tz="UTC", origin="1970-01-01") # Convert UNIX seconds to POSIXct

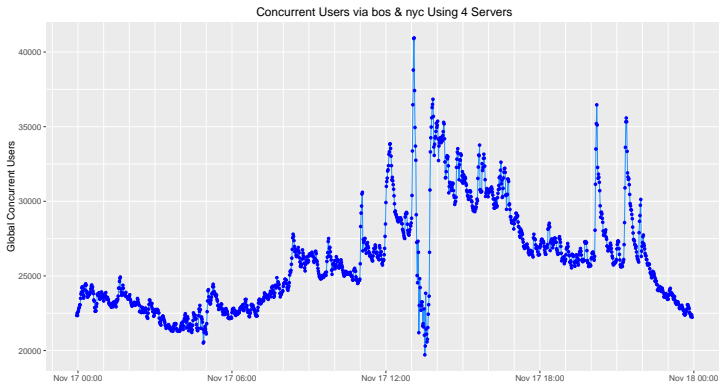
global_agg <- bc %>% group_by(Time) %>% summarise(TotalUsers = sum(Users), ServerCount = n()) # Pipe-like

MedianServerCount <- median(global_agg$ServerCount) # How many servers are active?
global_agg <- global_agg %>% filter(ServerCount == MedianServerCount) # Drop times with missing server data

bc$Egress <- substr(bc$Server, 1, 3) # Get data center site code from server name
SiteCodes <- unique(bc$Egress) # Character vector of unique DC names
cSiteCodes <- paste(SiteCodes, collapse = ' & ') # Collapse into simple string

Title <- paste('Concurrent Users via', cSiteCodes, 'Using', MedianServerCount, 'Servers')

ggplot(global_agg) +
  geom_line(aes(x = Time, y = TotalUsers), size=PointSize - 0.6, color = 'dodgerblue') + # Line first
  geom_point(aes(x = Time, y = TotalUsers), size=PointSize, color = 'blue', shape=19) + # Points on top
  xlab("") + ylab("Global Concurrent Users") +
  scale_x_datetime(date_minor_breaks = "1 hour") + # ggplot2 2.0.0, remove 'date_' for previous versions!
  ggtitle(Title)
```



Example 2 Plot

# Generating Reports

- These slides were made with R +  $\text{\LaTeX}$  + Beamer + Sweave
- R +  $\text{\LaTeX}$  + Sweave
  - ▶ Publication quality PDF
  - ▶ Page breaks are a pain !
  - ▶  $\text{\LaTeX}$  is somewhat complex, and can be a pain with syntax errors
- R + Markdown + knitr  $\rightsquigarrow$  nice HTML output
  - ▶ Simple syntax
  - ▶ Easy to show R code
  - ▶ No page constraints
- For all of above, self-contained flexibility is limited, a template method may be required if there are a variable number of plots, complex CLI arguments, etc.



# Resources

- Get R: <https://cran.r-project.org/>
- Daily news: <http://www.r-bloggers.com/>
- <https://www.r-consortium.org/>
- Cheatsheets:  
<https://www.rstudio.com/resources/cheatsheets/>
  - ▶ Data Visualization Cheat Sheet
  - ▶ Data Wrangling Cheat Sheet
  - ▶ and others
- Many books are available, a few to get started
  - ▶ Project documentation
  - ▶ Base R programming: The Art of R Programming by Norman Matloff
  - ▶ Visualization: R Graphics Cookbook by Winston Chang
  - ▶ Modern R: Mastering Data Analysis with R by Gergely Daróczi
  - ▶ Advanced R Programming by Hadley Wickham:  
<http://adv-r.had.co.nz/>
- Google → Stackoverflow are your friends, as expected
- Many courses are available, watch out for age and topics covered

# Installing R

- MacOS and Windows (& personal Linux), just get from <https://cran.r-project.org/>
- Then add a few packages
  - ▶ `sudo R --no-site-file --no-init-file --quiet`
  - ▶ `install.packages(c("ggplot2", "plyr", "reshape2", "dplyr", "tidyr"))`
  - ▶ `install.packages(c("lubridate", "readr", "stringr", "knitr"))`
  - ▶ To exit R: 'q()' (don't save the workspace)

# Almost All of the Packages I Install

## Dependencies will cause other packages to be installed

```
install.packages(c("ggplot2", "plyr", "reshape2", "dplyr", "tidyr", "lubridate", "readr", "stringr", "knitr"))
install.packages(c("digest", "gtable", "munsell", "pspline", "hash", "getopt", "littler"))
install.packages(c("devtools", "RCurl", "RColorBrewer", "rbenchmark", "quantmod", "httr"))
install.packages(c("forecast", "RSQLite", "data.table", "ggvis"))
install.packages("caret")
install.packages(c("googleVis", "gridExtra"))
install.packages(c("TTR", "maps", "gdata", "spc", "scales"))
install.packages(c("xtable", "zoo", "xts", "chron", "lattice"))
```

```
install.packages(c("doParallel", "doMC", "labeling", "microbenchmark", "shiny", "jsonlite", "iptools"))
install.packages(c("wmtsa", "fftw", "fftwtools", "signal", "numDeriv", "ChemometricsWithR", "ptw"))
install.packages(c("hyperSpec", "sfsmisc", "baseline", "Peaks", "robfilter", "FKF", "KFAS"))
install.packages(c("StreamMetabolism", "PerformanceAnalytics", "qcc", "scatterplot3d"))
install.packages(c("BenfordTests", "RcppDE", "pryr", "purrr", "timeline", "tm", "wordcloud"))
install.packages(c("svglite", "rsvg", "webp", "", "", "", "", ""))
```

```
library(devtools)
devtools::install_github("twitter/BreakoutDetection")
devtools::install_github("twitter/AnomalyDetection")
devtools::install_github("slowkow/ggrepel")
install_github("hrbrmstr/resolv")
install_github("jayjacobs/tldextract")
install_github("hrbrmstr/ipapi")
install_github("hrbrmstr/taucharts@dev")
```

```
source("http://bioconductor.org/biocLite.R")
biocLite("aroma.light")
```

# Building R on Linux

- Good for full control, and retention of previous versions
- Under tcsh (sorry)

```
set BUILDDIR=/home/meekj/build
set TARDIR=~Downloads
set RVERSION=3.2.3
cd $BUILDDIR
tar xzf $TARDIR/R-$RVERSION.tar.gz
cd R-$RVERSION

./configure --enable-R-shlib --prefix /usr/local/R-$RVERSION
make
make check
sudo make install

sudo /usr/local/R-$RVERSION/bin/R --no-site-file --no-init-file
```

- Install packages as shown on previous slides

# Building R on Linux - Finish

- Test as needed using full paths
- Change default R version when ready, Update `/usr/bin/` only if needed

Check current symlinks, or actual files (rename if needed):

```
ls -l /usr/bin/R /usr/bin/Rscript /usr/bin/r
ls -l /usr/local/bin/R /usr/local/bin/Rscript /usr/local/bin/r

sudo rm /usr/bin/R /usr/bin/Rscript /usr/bin/r
sudo rm /usr/local/bin/R /usr/local/bin/Rscript /usr/local/bin/r

sudo ln -s /usr/local/R-$RVERSION/bin/R /usr/bin/R
sudo ln -s /usr/local/R-$RVERSION/bin/Rscript /usr/bin/Rscript
sudo ln -s /usr/local/R-$RVERSION/lib/R/library/littler/bin/r /usr/bin/r

sudo ln -s /usr/local/R-$RVERSION/bin/R /usr/local/bin/R
sudo ln -s /usr/local/R-$RVERSION/bin/Rscript /usr/local/bin/Rscript
sudo ln -s /usr/local/R-$RVERSION/lib/R/library/littler/bin/r /usr/local/bin/r

ls -l /usr/bin/R /usr/bin/Rscript /usr/bin/r
ls -l /usr/local/bin/R /usr/local/bin/Rscript /usr/local/bin/r
```

- On Redhat the library path may be `/usr/local/R-$RVERSION/lib64`

# Loading Libraries in a “Good” Order

- Function overloading can be a problem
- Note that wmtsa is not a commonly used package, but it loads the more often used MASS package
- I usually run the lines below in new interactive sessions
- And use a similar order in batch jobs

```
## Load common libraries in a good order
## not complete

library(wmtsa) # Wavelet Methods for peak detection, uses MASS which overloads dplyr::select

library(ggplot2)
library(plyr) # plyr, followed by dplyr
library(dplyr)
library(readr)
library(lubridate)

Sys.setenv(TZ="UTC") # Yes, timezone issues can arise, nip them in the bud
```

# A Sample .Rprofile File - Mostly commented out

```
# http://www.r-bloggers.com/fun-with-rprofile-and-customizing-r-startup/

# General options
options(tab.width = 2)
options(width = 130)

options(max.print=500)
options(scipen=10)

q <- function (save="no", ...) { # Don't ask to save workspace
  quit(save=save, ...)
}

.First <- function(){
# library(Hmisc)
# library(R2HTML)

#   if(interactive()){
#     library(ggplot2)
#     library(dplyr)
#     library(readr)
#     library(stringr)
#     library(lubridate)
#   }

  cat("\nWelcome Jon at", date(), "\n")
}

.Last <- function(){
  cat("\nGoodbye Jon at ", date(), "\n")
}

message("\n*** Successfully loaded .Rprofile ***\n")
```