Getting up and running with R and R Studio

Presented by: Derek Kane

Overview of Topics

- Introduction to the R language
- Installation and Configuration
- R Studio Basics
  - Navigation Panes
  - Menu Descriptions
- R Script Development
  - Installing Packages
  - R Model Components
- Additional R Tips

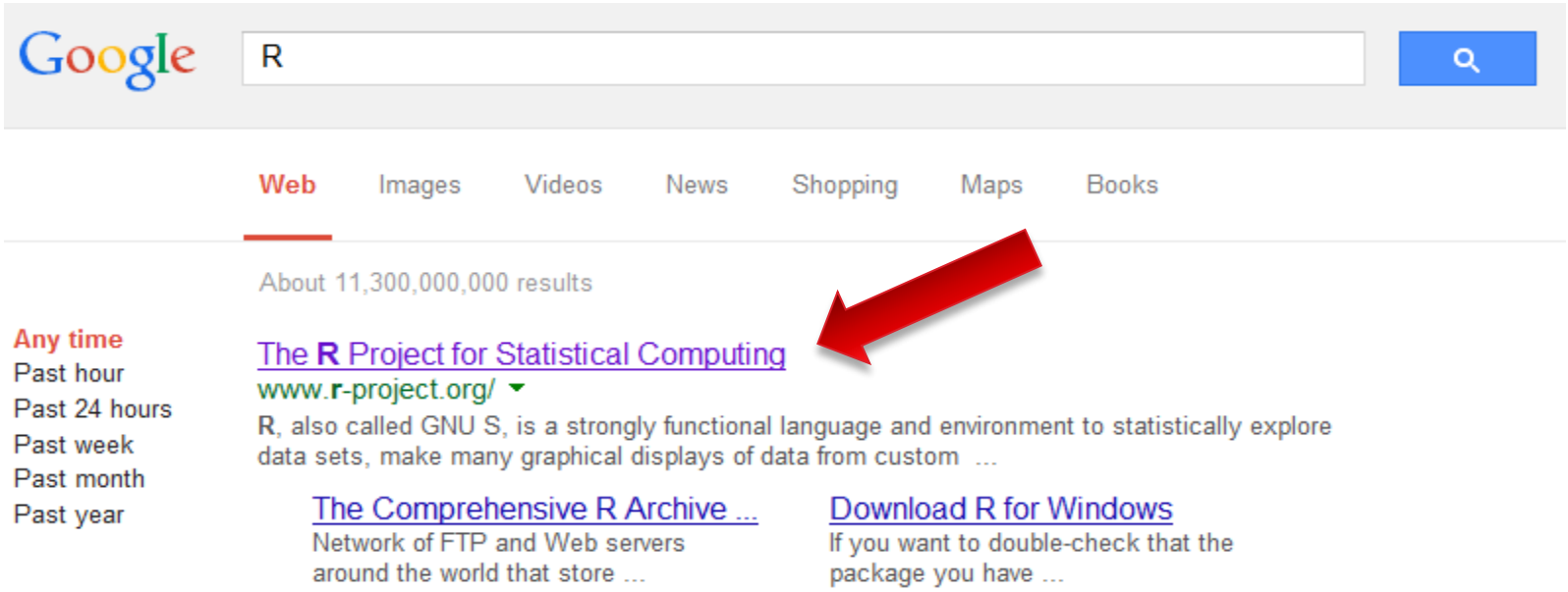
The Benefits of Using R

- Ideal investment cost. There is no upfront cost for using the technologies.
- Open Source – No black box mysteries, no proprietary lockdown into a specific tool.
- Most powerful statistical programming language.
- Easy to share across a business.
- Often works better/faster than Microsoft or Oracle products for data and analysis.
- Infinitely customizable to your problem and your products – vertical integration.
- Large support group of users worldwide. Most widely used data analysis software.
- Highly credible due to submission standards and university usage.
- Relatively easy to learn.

Thinking about R

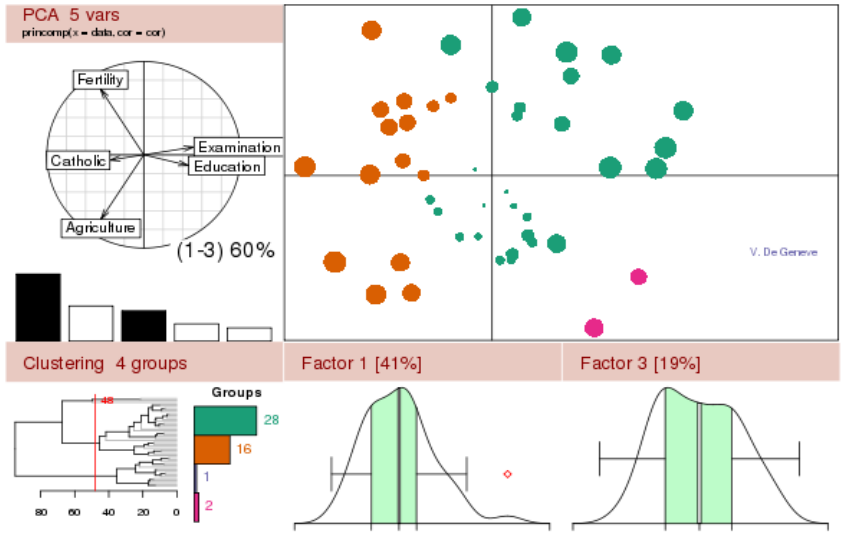
- One way to approach thinking about the R language would be to compare the language to our smartphones.
- The iPhone IOS interface has a lot of incredible built in features that are native to the phone. (Ex. Safari, music player, alarm clocks, quick menus, etc…)
- However, the real power of the iPhone is unlocked through the use of 3rd party applications. (Ex. Facebook, Google Maps, Weather.com, PvZ, etc…)
- In this regard R is very similar. Instead of running applications, we install the applications (called packages) and run them when we need them. We call these “libraries”.
- The key to being a great R user is to know which packages/libraries to run in tandem with each other for a specific machine learning / predictive modeling task.

Installing R Software



Installing R Software

The R Project for Statistical Computing



About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Installing R Software

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

0-Cloud	
http://cran.rstudio.com/	Rstudio, automatic redirection to servers worldwide
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
http://cran.csiro.au/	CSIRO
http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	
http://cran.at.r-project.org/	Wirtschaftsuniversitaet Wien
Belgium	
http://www.freestatistics.org/cran/	K.U.Leuven Association
Brazil	
http://nbcgib.uesc.br/mirrors/cran/	Center for Comp. Biol. at Universidade Estadual de Santa Cruz
http://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Canada	
http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://mirror.its.dal.ca/cran/	Dalhousie University, Halifax
http://cran.utstat.utoronto.ca/	University of Toronto
http://cran.skazkaforyou.com/	iWeb, Montreal
http://cran.parentingamerica.com/	iWeb, Montreal

The Comprehensive R Archive Network

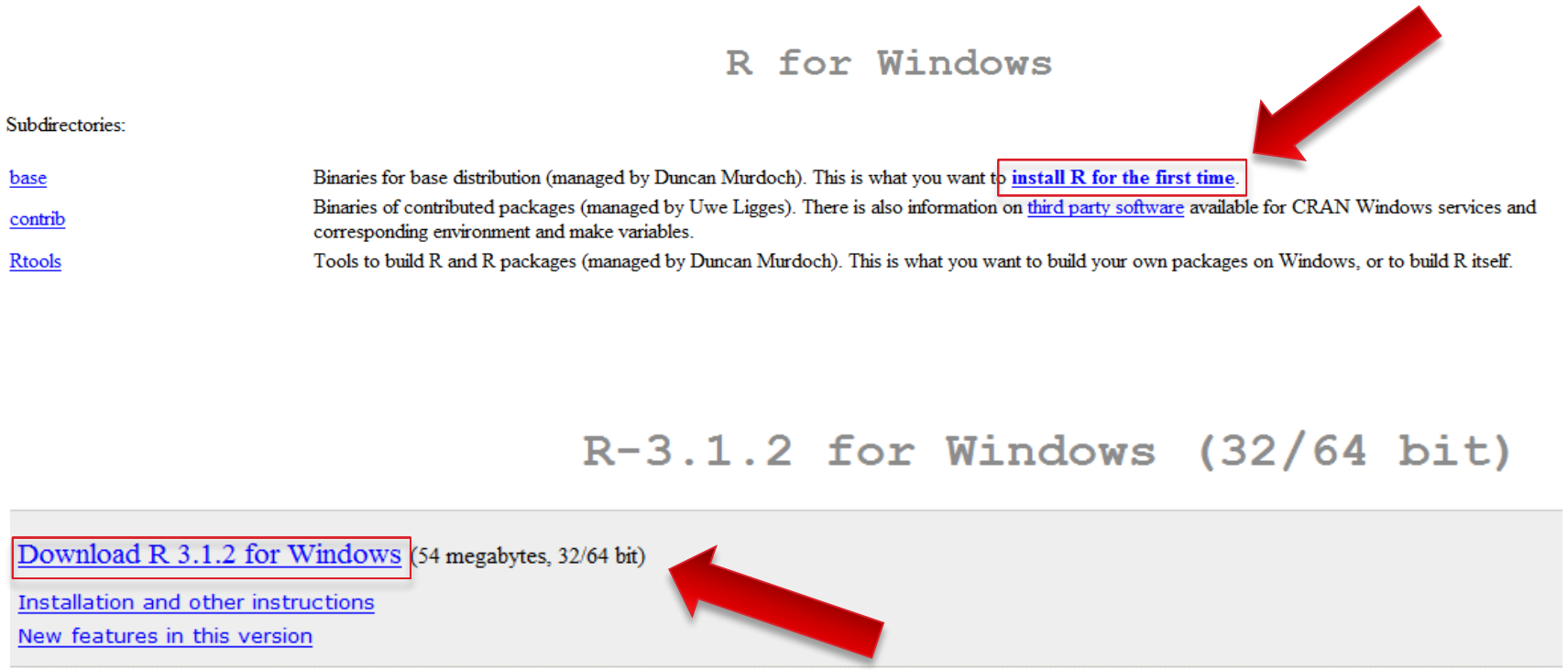
Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

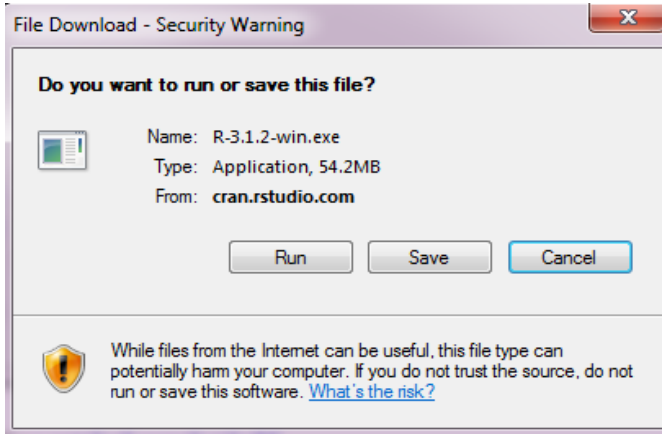
- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

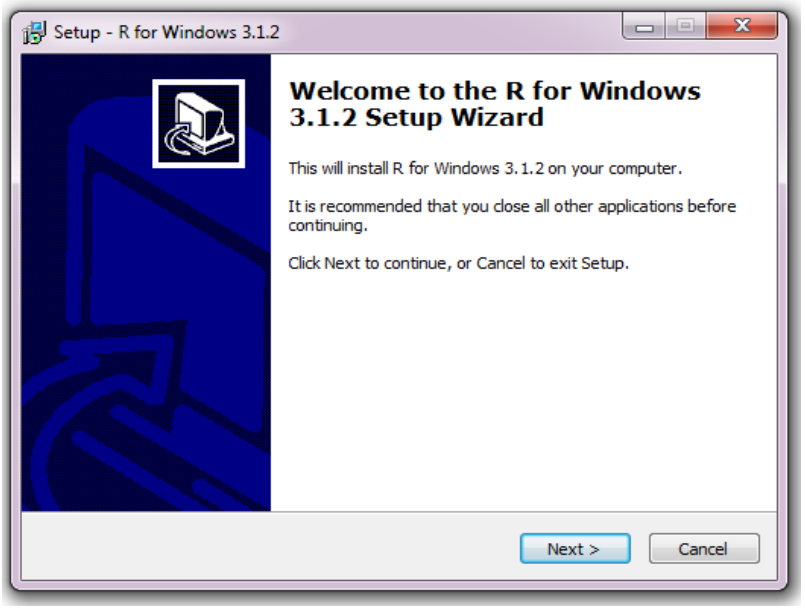
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Installing R Software

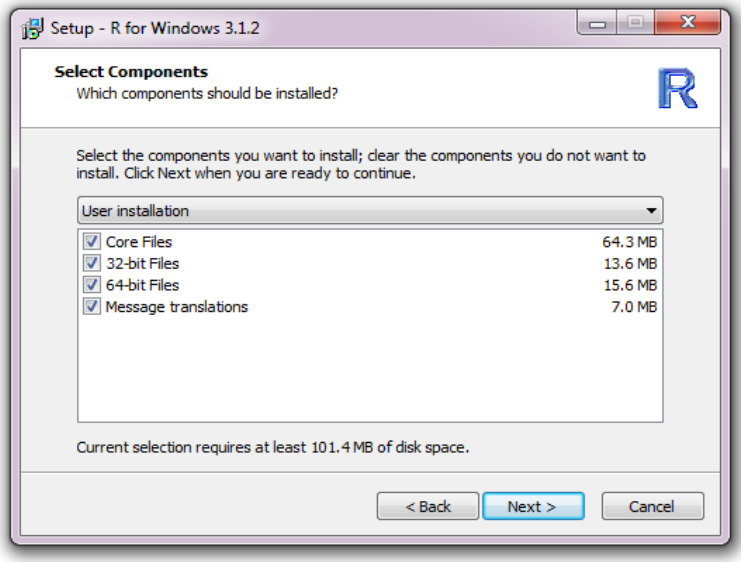


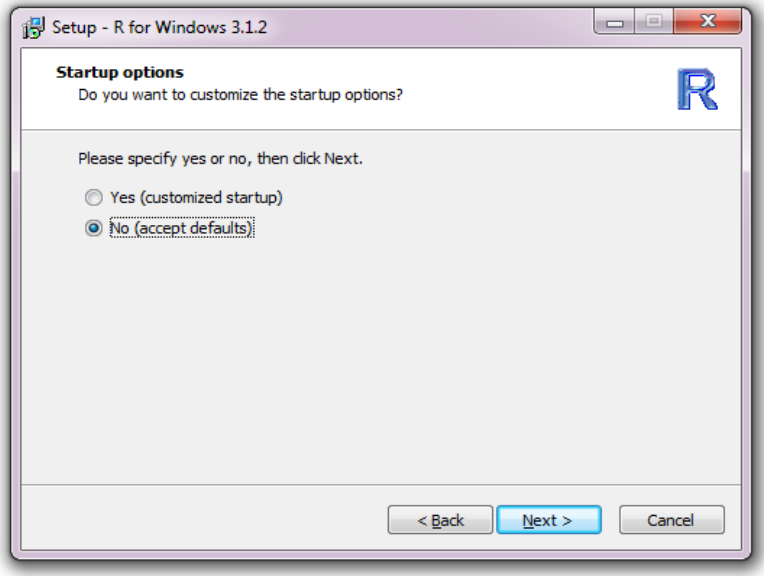
Installing R Software

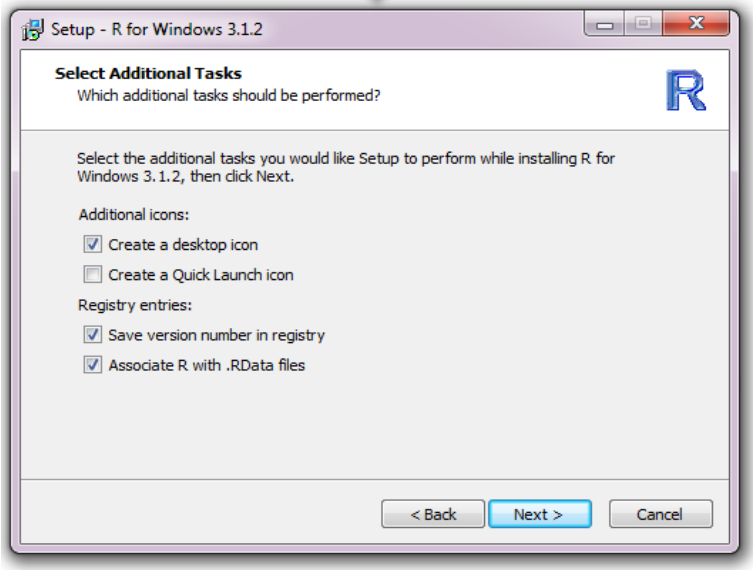




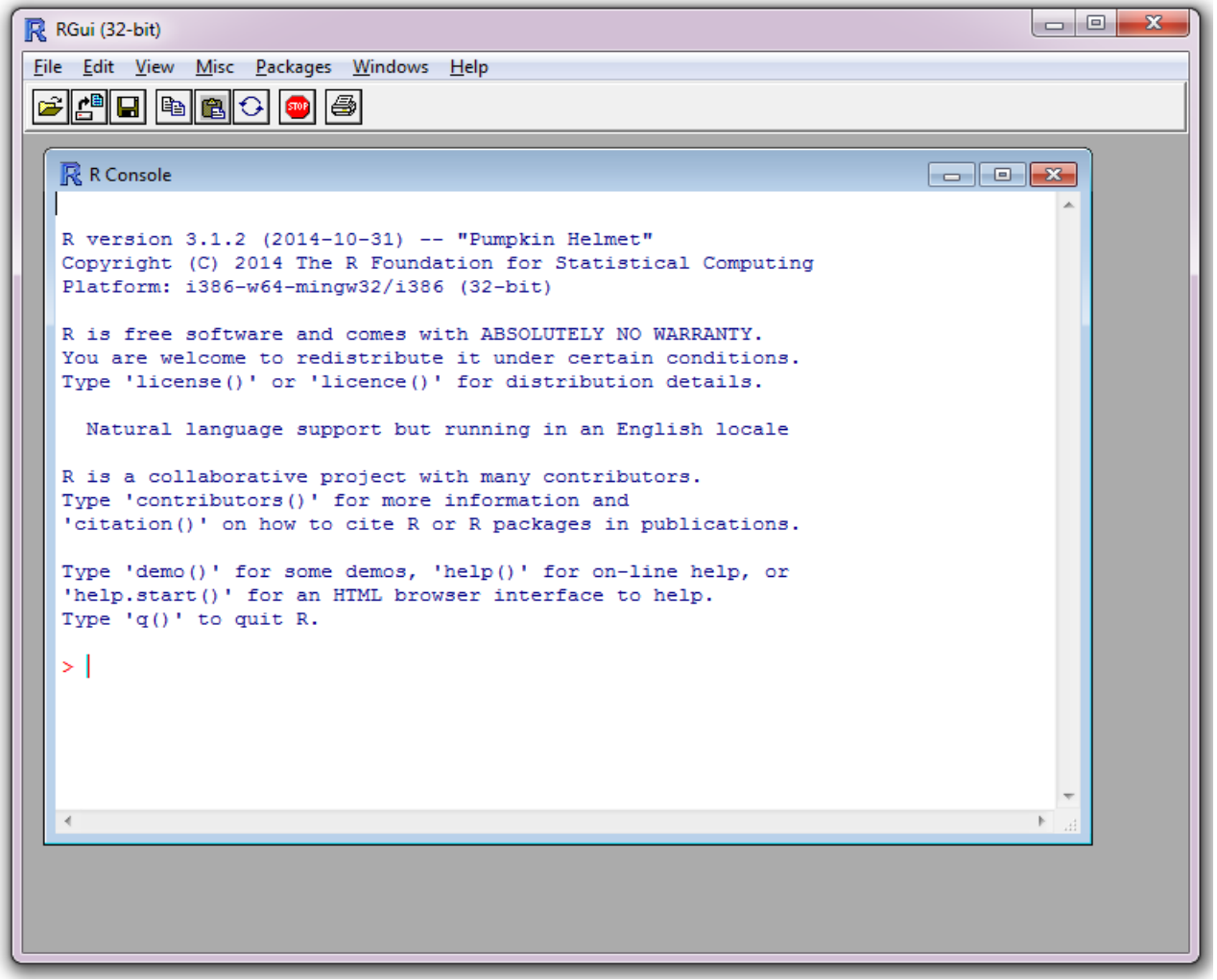
Installing R Software



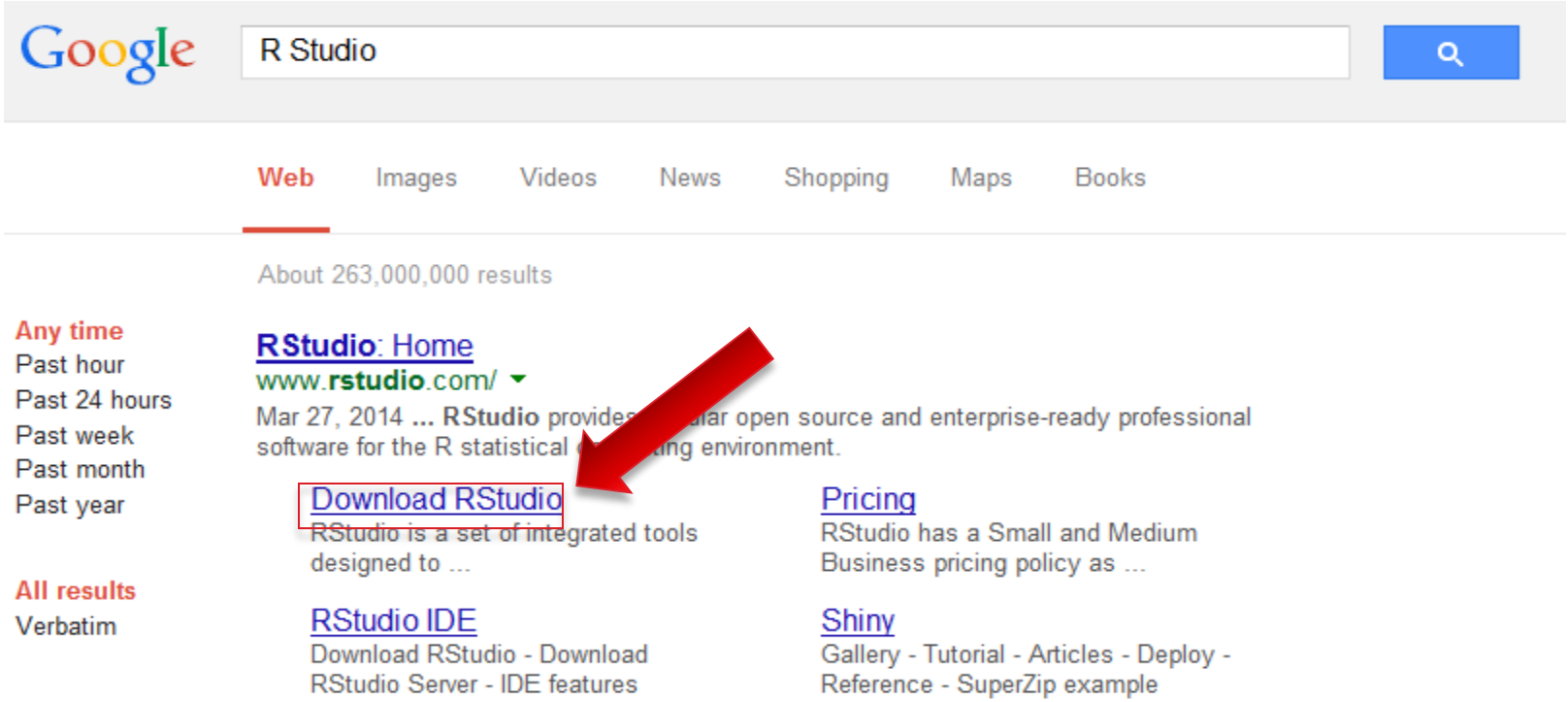




Installing R Software



Installing R Studio



Installing R Studio

RStudio Products Resources Pricing About Us Blog

Download RStudio

Home / Overview / RStudio / Download RStudio/

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

If you run R on a Linux server and want to enable users to remotely access RStudio using a web browser please download RStudio Server.

Do you need support or a commercial license?
Check out our commercial offerings

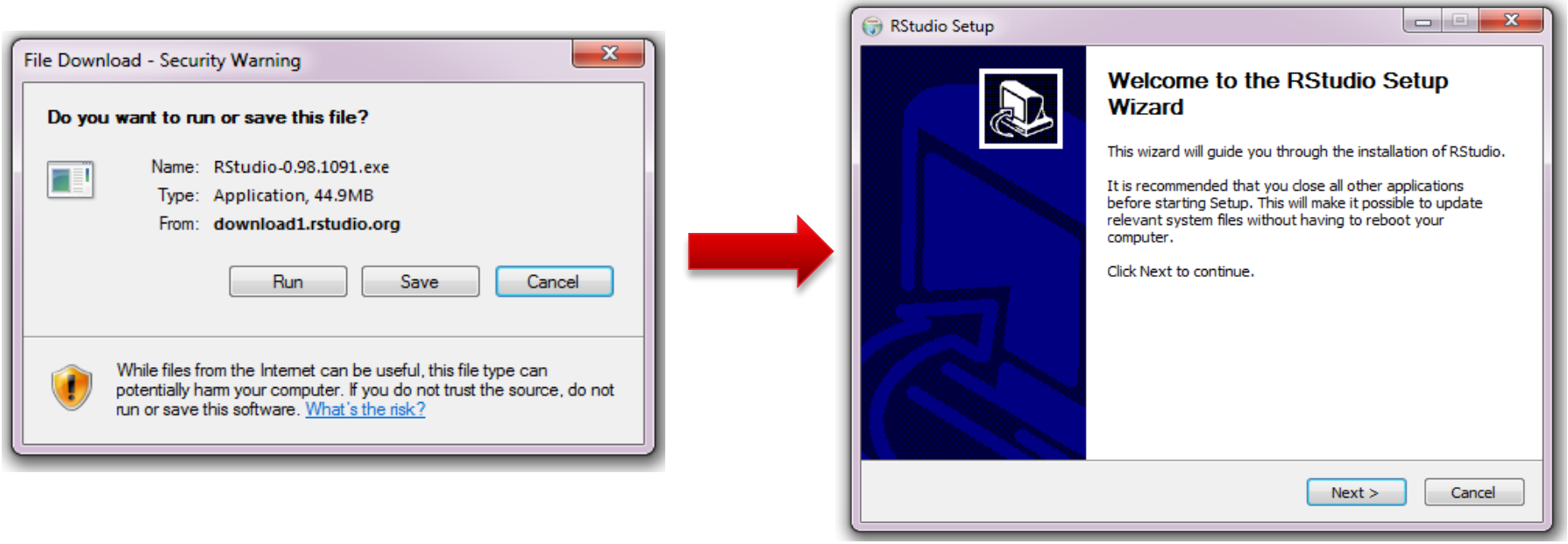
Download RStudio Desktop v0.98.1091 — Release Notes

RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it here.

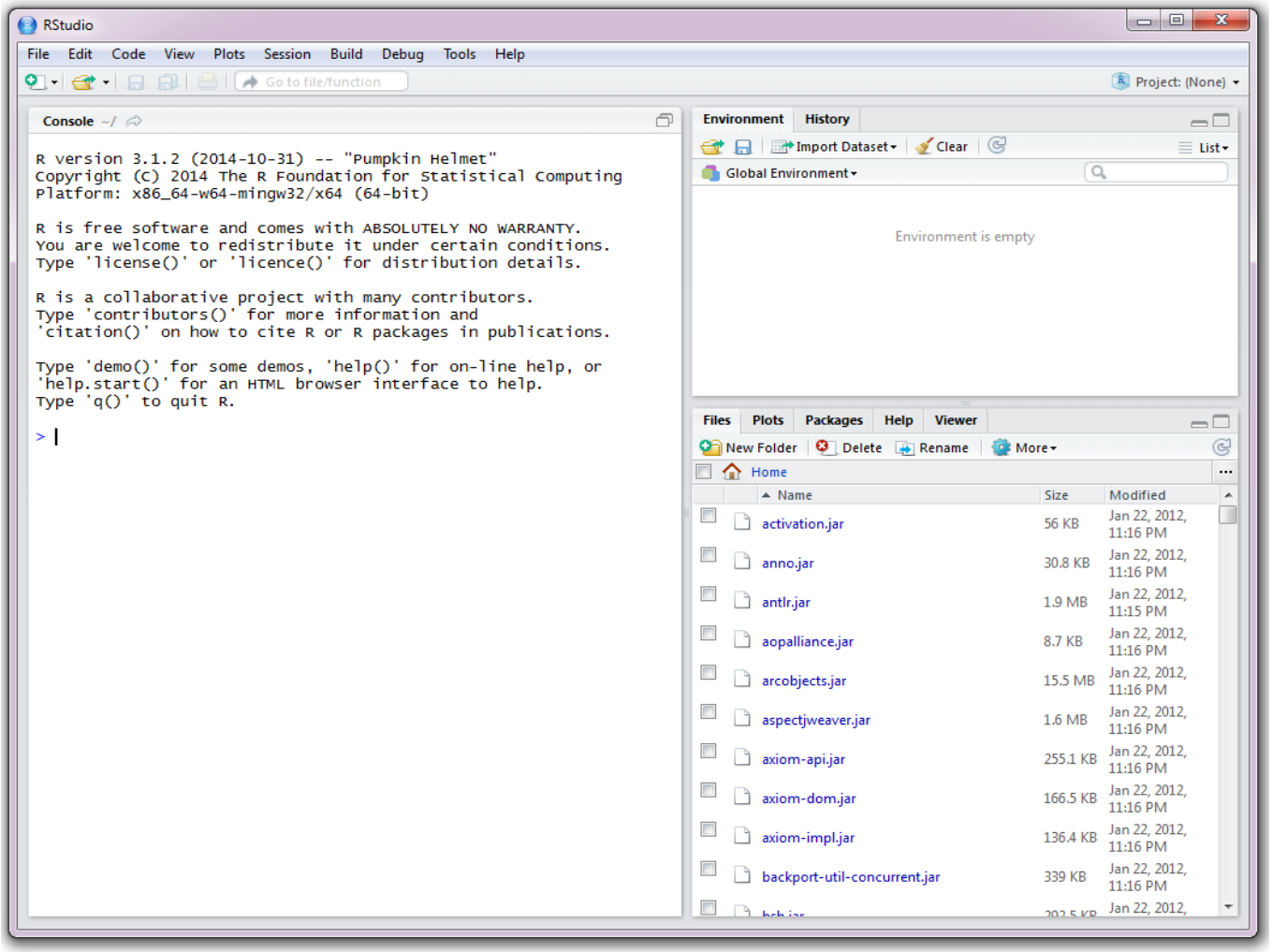
Installers for ALL Platforms

Installers	Size	Date	MD5
RStudio 0.98.1091 - Windows XP/Vista/7/8	45 MB	2014-11-06	910fba345c0555597bda498cad1302b0
RStudio 0.98.1091 - Mac OS X 10.6+ (64-bit)	38.4 MB	2014-11-06	9c7d2cea702cf478a4a774b79134b3ee
RStudio 0.98.1091 - Debian 6+/Ubuntu 10.04+ (32-bit)	53 MB	2014-11-06	0bc579cbee43a514e3fb4569959a0ada
RStudio 0.98.1091 - Debian 6+/Ubuntu 10.04+ (64-bit)	54.9 MB	2014-11-06	1e88e6775993daa8cf7d4d89f76af7e0
RStudio 0.98.1091 - Fedora 13+/RedHat 7+/openSUSE 11.4+ (32-bit)	53.4 MB	2014-11-06	3ae5923956166f90ecc1cb721b02f90f
RStudio 0.98.1091 - Fedora 13+/RedHat 7+/openSUSE 11.4+ (64-bit)	55 MB	2014-11-06	6d1ac08ceed731f5750f3de9a911511b

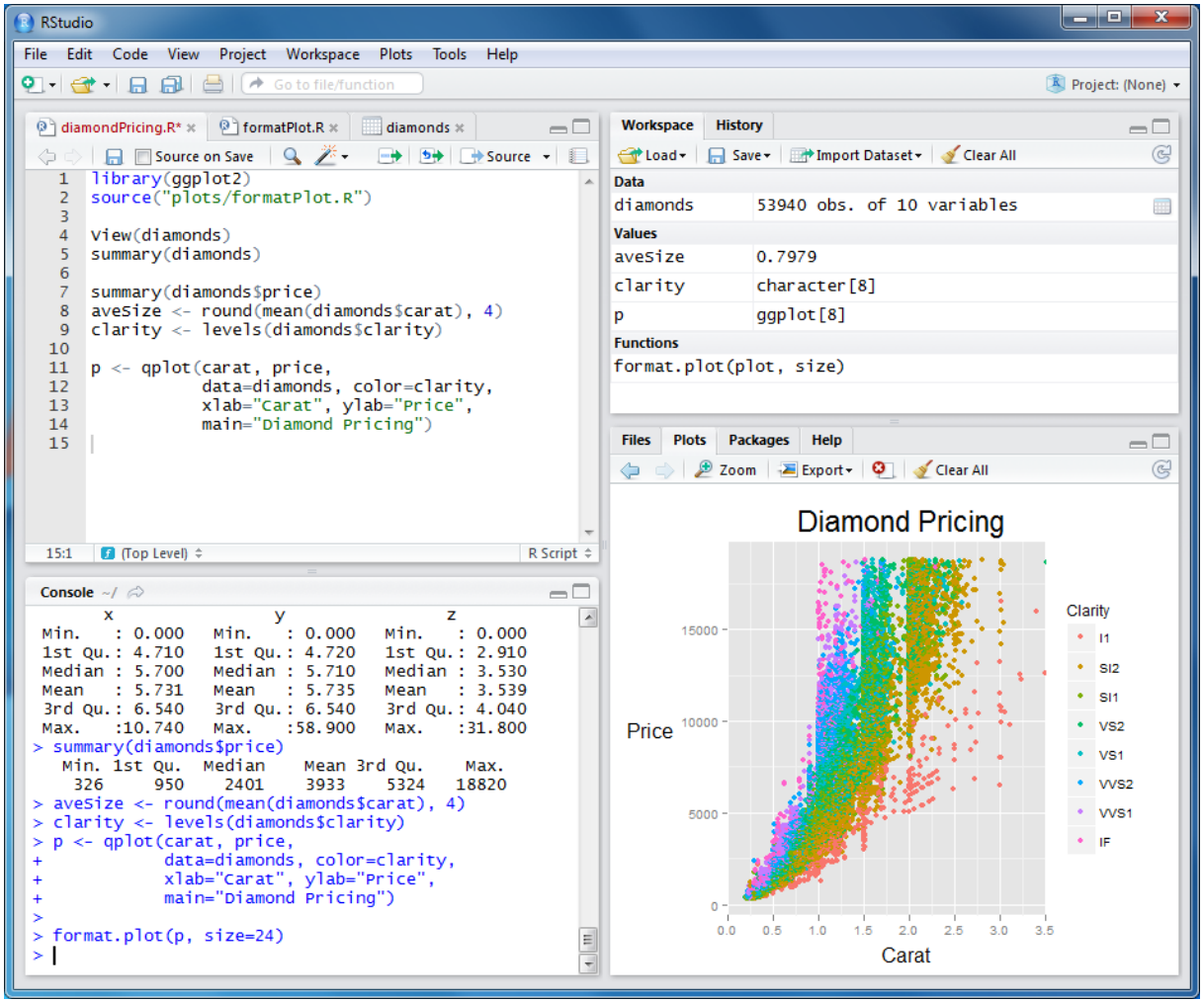
Installing R Studio



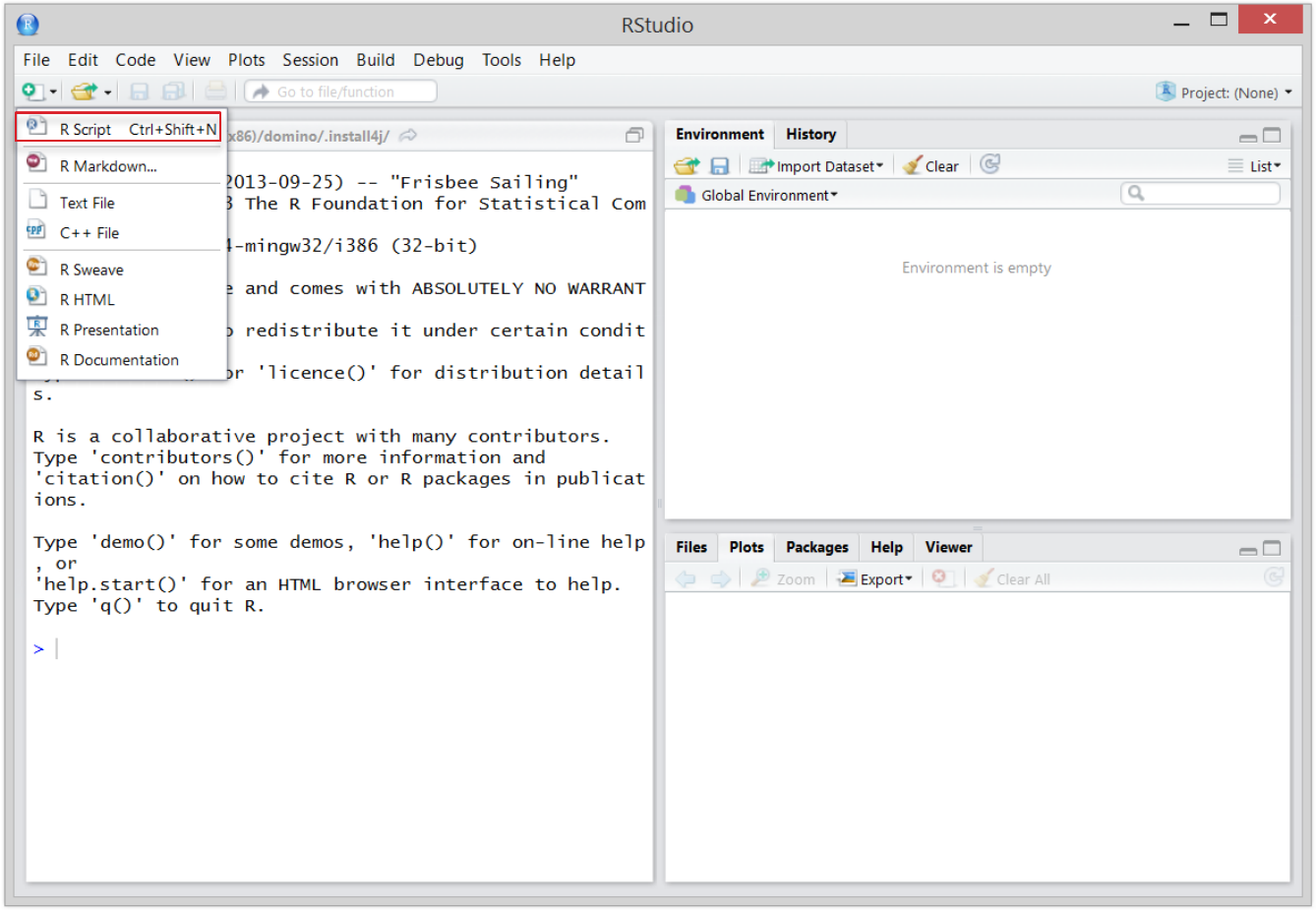
Installing R Studio



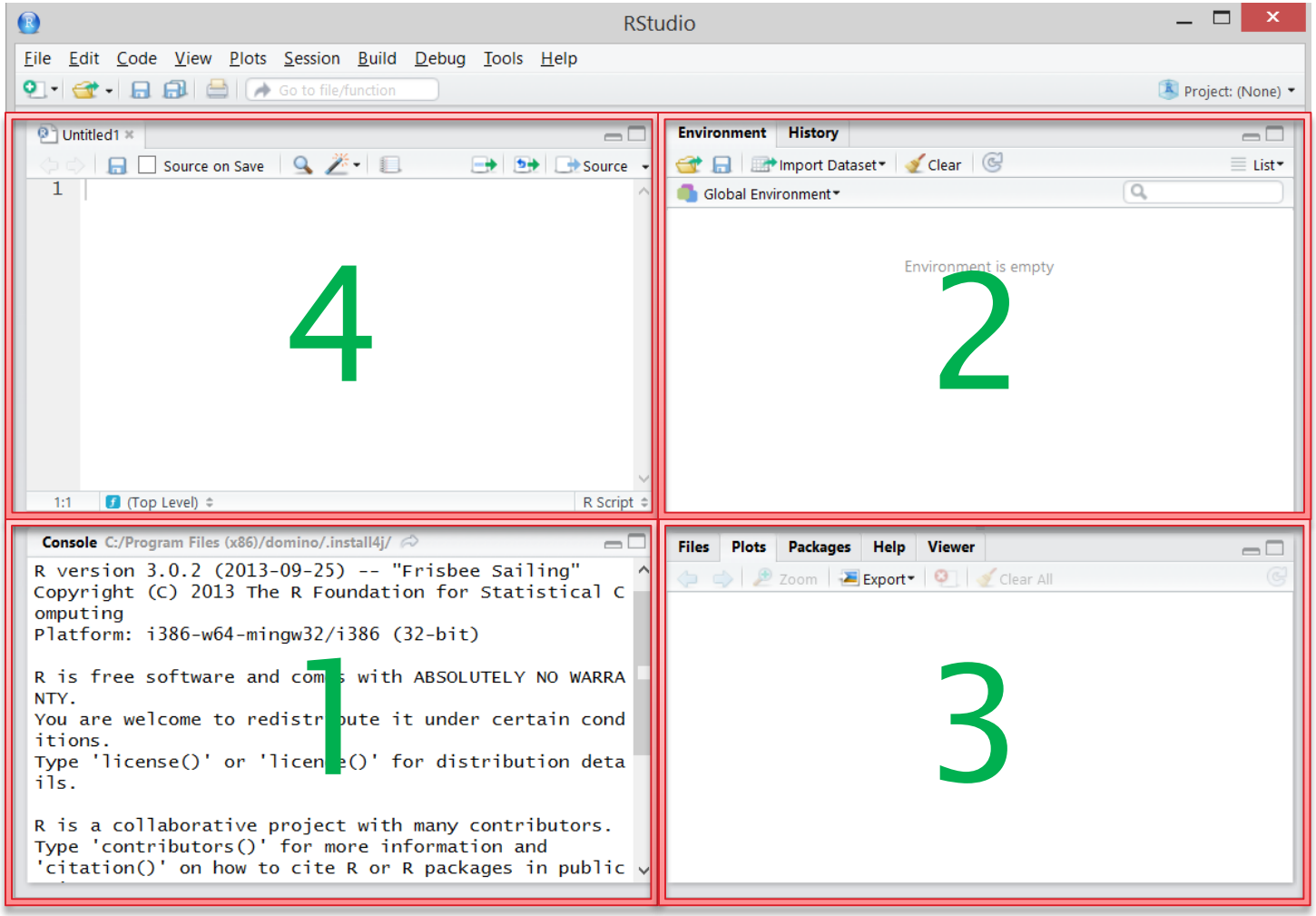
R Studio Environment



R Studio Environment

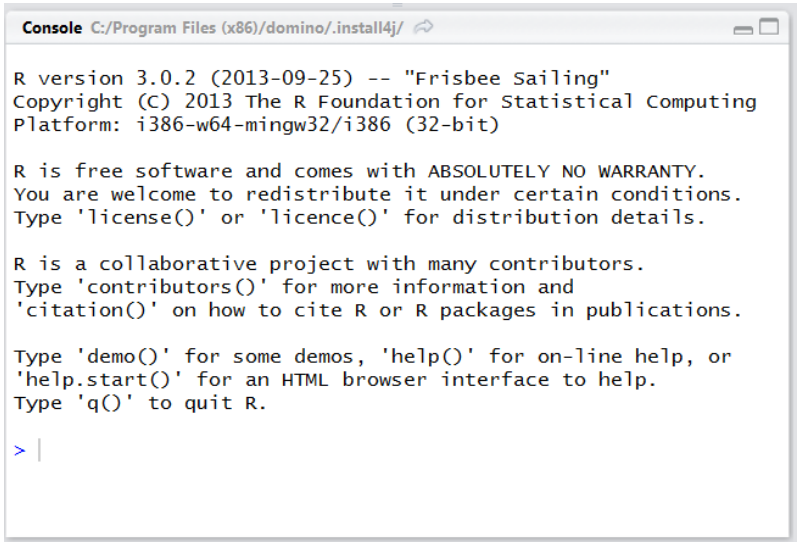


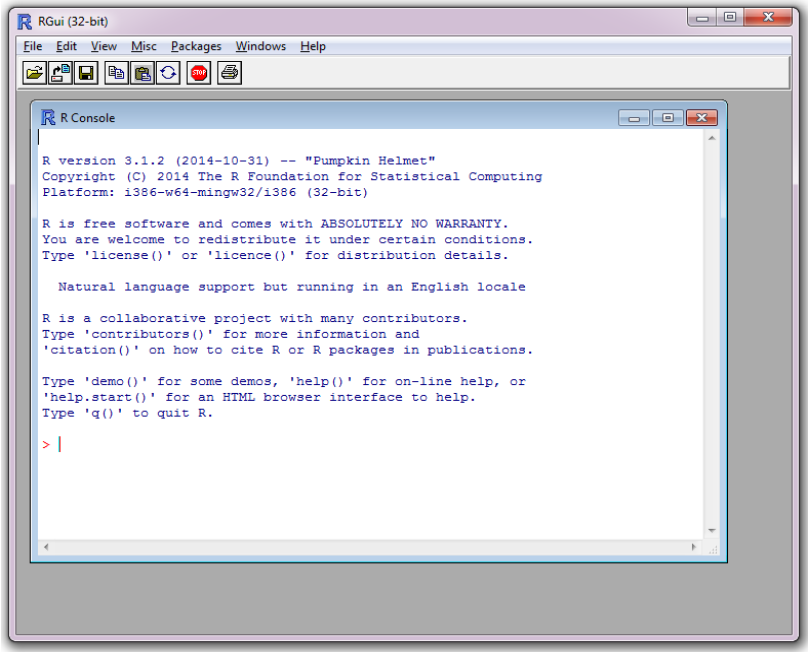
R Studio Environment



R Studio Environment – Pane 1

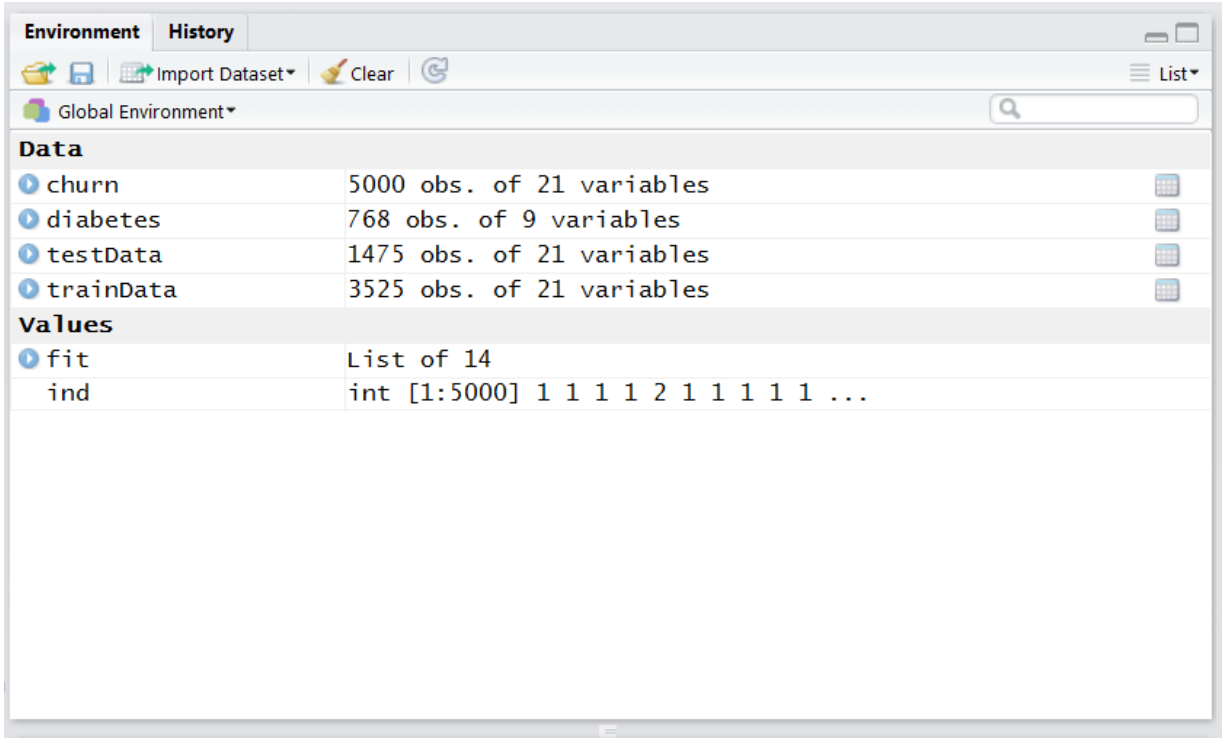
- The pane at the bottom left hand side of the R Management Studio shows the R console as it would appear in the base R environment.
- This section shows us the executed results of our code as well as any error messages in the execution of our code. We wont actually be performing our work here but this section is critical to review when developing in R.





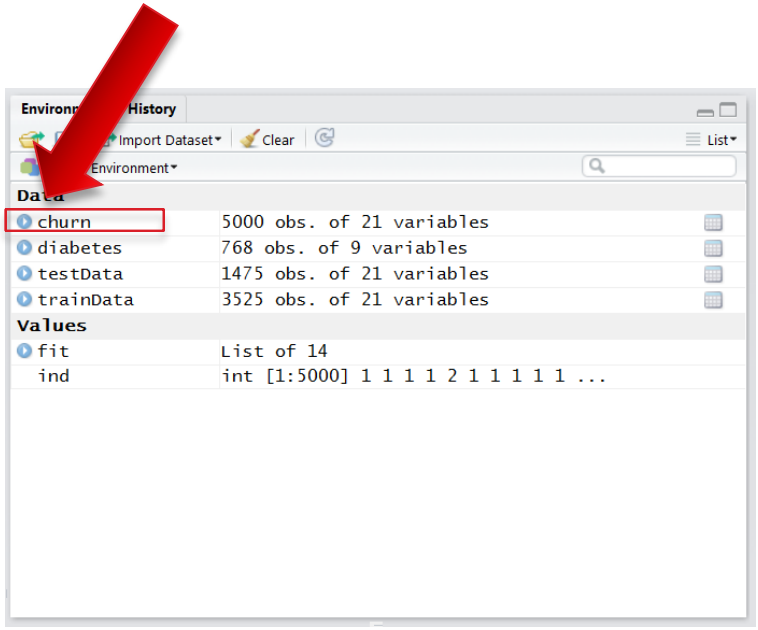
R Studio Environment – Pane 2

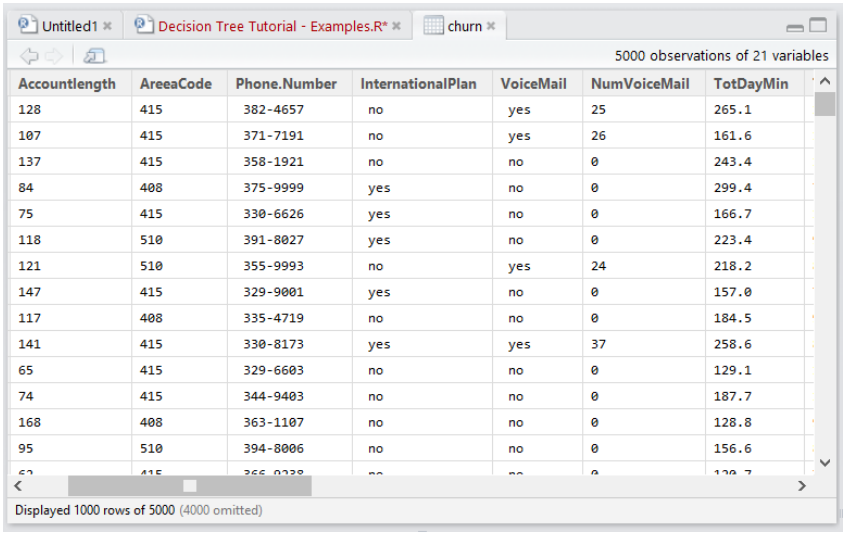
- The top right section shows information related to the objects (data frames, matrices, vectors, values, etc…) which are being used in R.



R Studio Environment – Pane 2

- One very helpful feature of the "Environment" section is the ability to review datasets (R calls them data-frames) after they have been loaded in R.
- After clicking on a data-frame, the table will open up in the top left window.

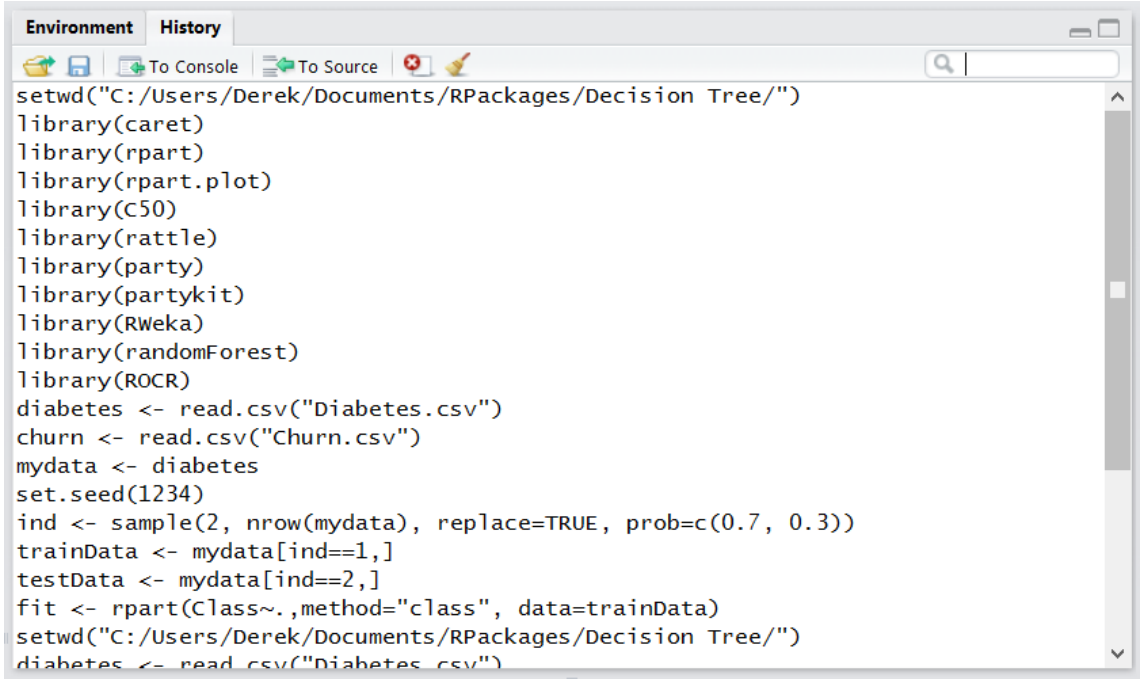




Accountlength	AreeaCode	Phone.Number	InternationalPlan	VoiceMail	NumVoiceMail	TotDayMin
128	415	382-4657	no	yes	25	265.1
107	415	371-7191	no	yes	26	161.6
137	415	358-1921	no	no	0	243.4
84	408	375-9999	yes	no	0	299.4
75	415	330-6626	yes	no	0	166.7
118	510	391-8027	yes	no	0	223.4
121	510	355-9993	no	yes	24	218.2
147	415	329-9001	yes	no	0	157.0
117	408	335-4719	no	no	0	184.5
141	415	330-8173	yes	yes	37	258.6
65	415	329-6603	no	no	0	129.1
74	415	344-9403	no	no	0	187.7
168	408	363-1107	no	no	0	128.8
95	510	394-8006	no	no	0	156.6

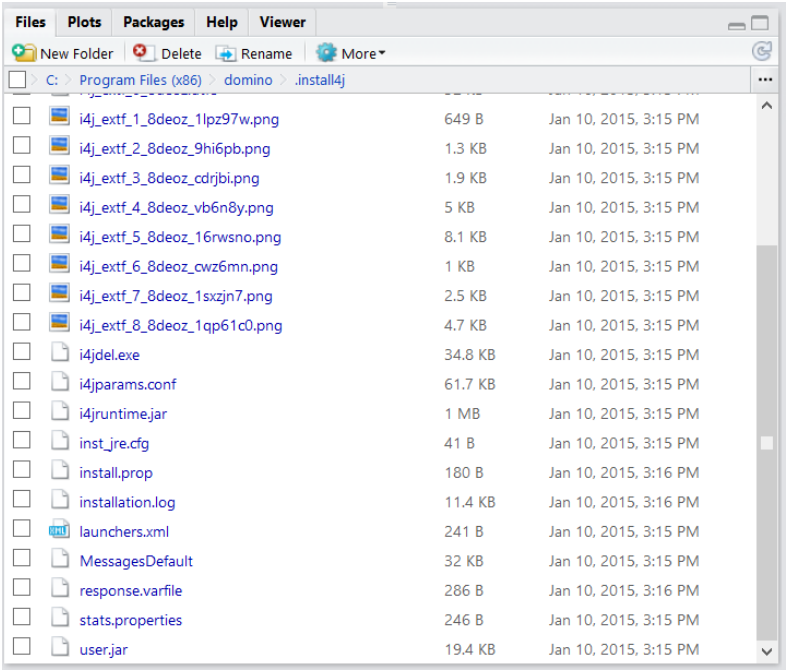
R Studio Environment – Pane 2

- This pane also shows the history of the R code execution which has been run in the current section. This is more useful for programmers who are reviewing production level code. I don't really use this often and prefer to follow the execution of the code in the console on Pane 1.



R Studio Environment – Pane 3

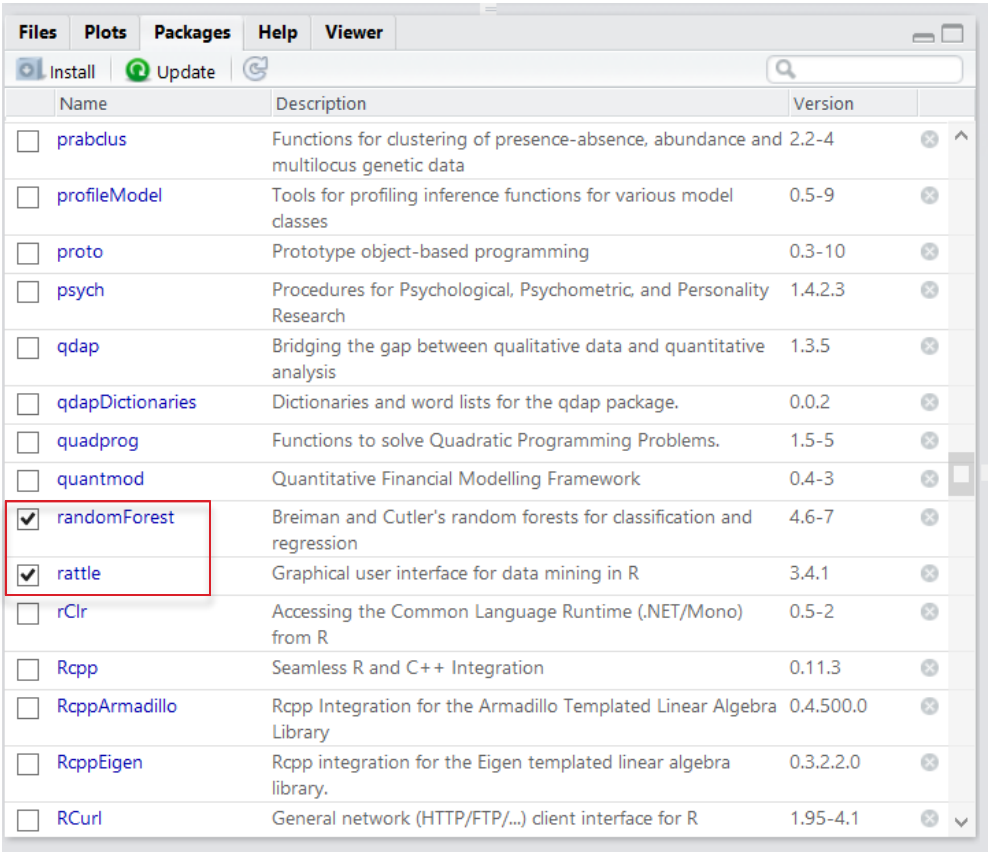
- The pane at the bottom right hand side of the R Management Studio contains a lot of useful features.
  - **Files –** Allows a review of exported files and ease of access to the working directory.
  - **Package Viewer –** Allows for a review of all available downloaded packages and easy activation of packages.
  - **Plot Viewer –** This section contains any plots that have been generated in the R session.
  - **Help –** This contains a menu for specific questions related to base R or any of the installed packages.



R Studio Environment – Pane 3

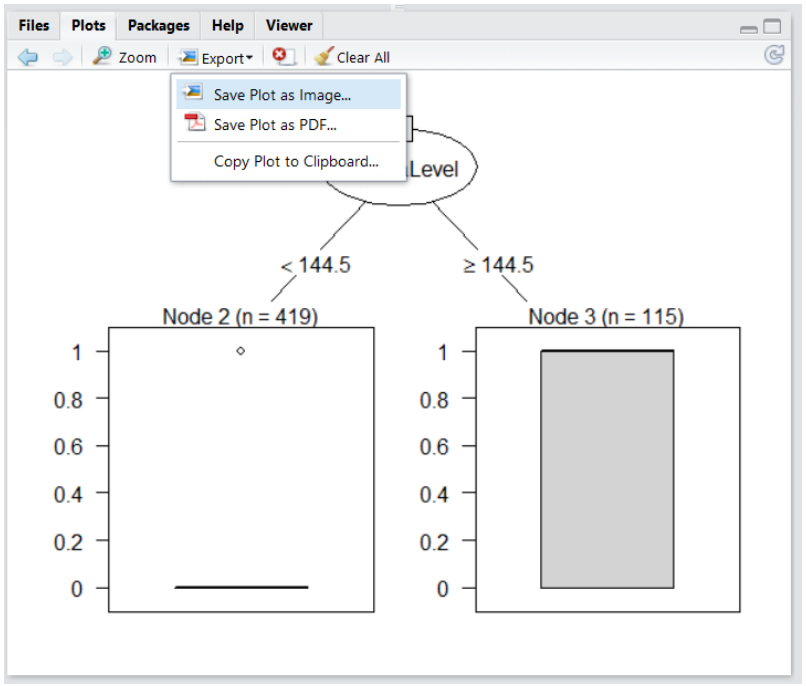
The Package Viewer

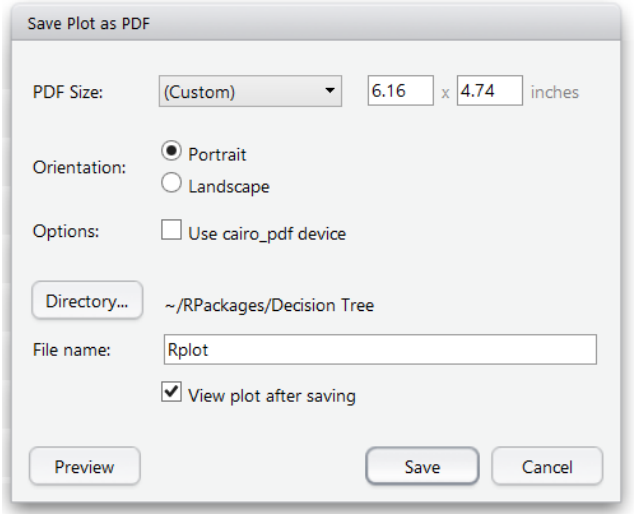
Activated Packages

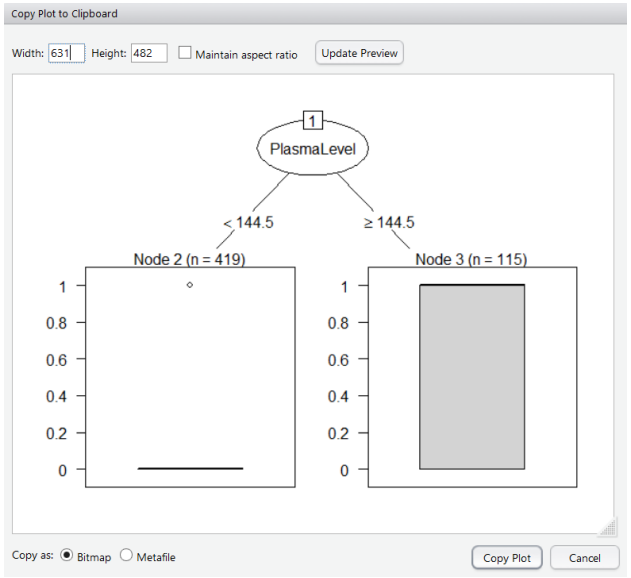


R Studio Environment – Pane 3

Plot Viewer

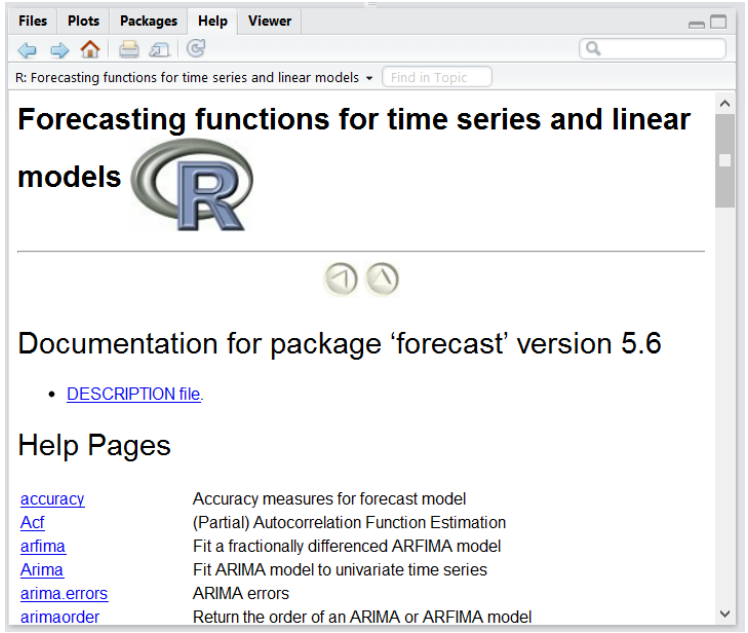






R Studio Environment – Pane 3

Help Menu



forecast.Arima {forecast} R Documentation

Forecasting using ARIMA or ARFIMA models

Description

Returns forecasts and other information for univariate ARIMA models.

Usage

```
## S3 method for class 'Arima'
forecast(object, h=ifelse(object$arma[5]>1,2*object$arma[5],10),
    level=c(80,95), fan=FALSE, xreg=NULL, lambda=object$lambda,
    bootstrap=FALSE, npaths=5000, ...)
## S3 method for class 'ar'
forecast(object, h=10, level=c(80,95), fan=FALSE, lambda=NULL,
    bootstrap=FALSE, npaths=5000, ...)
## S3 method for class 'fracdiff'
forecast(object, h=10, level=c(80,95), fan=FALSE, lambda=object$lamb
```

Arguments

`object`	An object of class "`Arima`", "`ar`" or "`fracdiff`". Usually the result of a call to `arima`, `auto.arima`, `ar`, `arfima` or `fracdiff`.

Author(s)

Rob J Hyndman

References

Peiris, M. & Perera, B. (1988), On prediction with fractionally differenced ARIMA models, *Journal of Time Series Analysis*, **9**(3), 215-220.

See Also

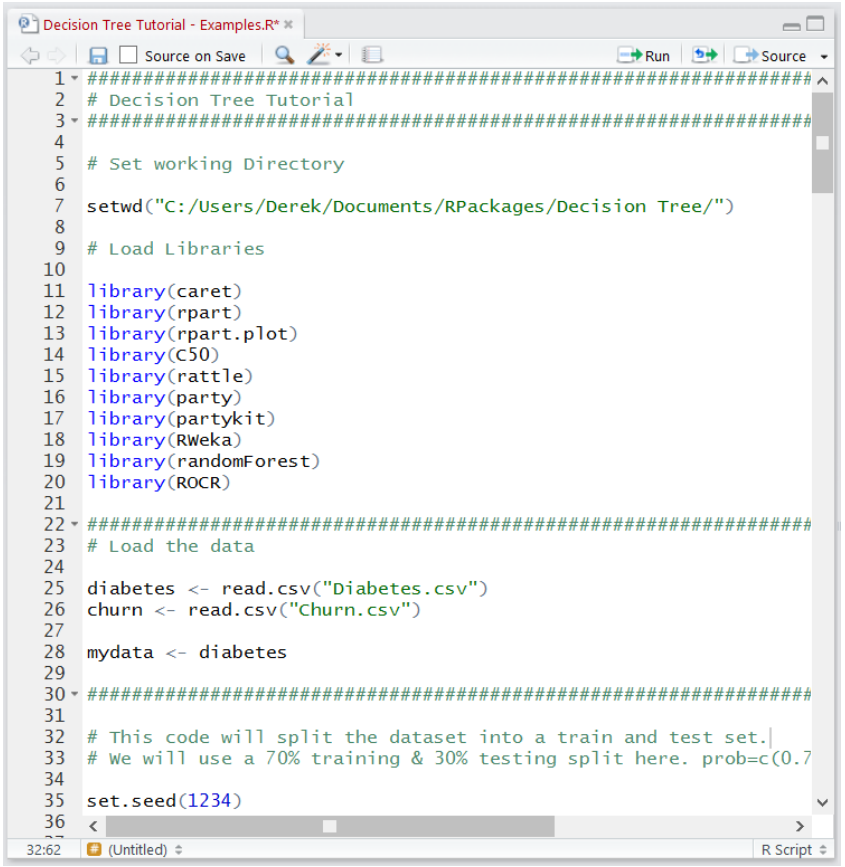
`predict.Arima`, `predict.ar`, `auto.arima`, `Arima`, `arima`, `ar`, `arfima`.

Examples

```
fit <- Arima(WWWusage,c(3,1,0))
plot(forecast(fit))

library(fracdiff)
x <- fracdiff.sim( 100, ma=-.4, d=.3)$series
fit <- arfima(x)
plot(forecast(fit,h=30))
```

R Studio Environment – Pane 4

- This is the main section of working within R Studio and R as a whole.
- All of our work will be constructed here and R Studio has designed the interface to be as user friendly as possible.
- This section allows for us to develop the R code freely without executing the code sequentially as in the R console.

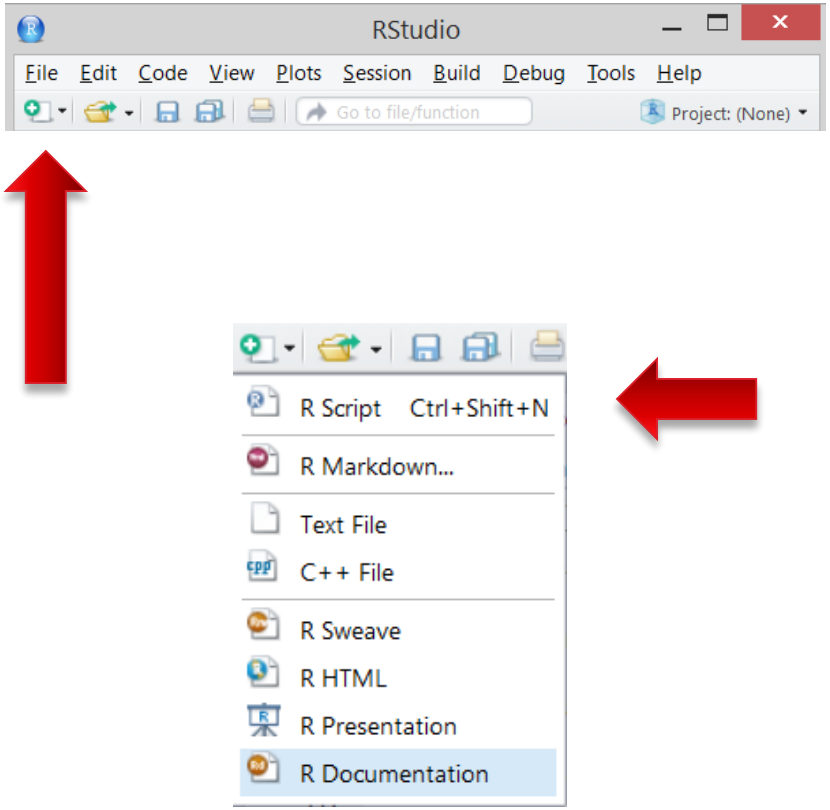


R Studio Environment – Pane 4

- The R language allows the use of txt and C++ files to be run, however, to use R Studio's incredible features we need to make sure that we are working in R Scripts exclusively.

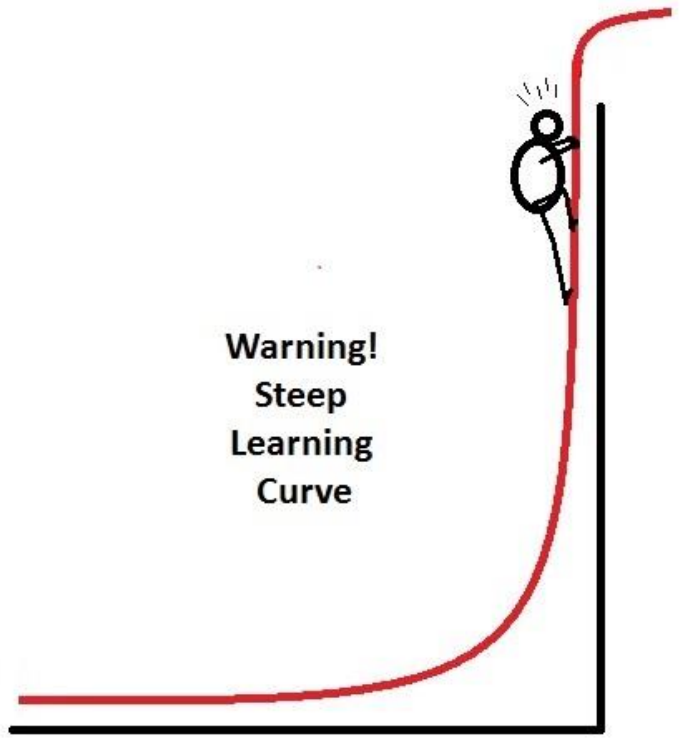
- Ease of Use Tips:
  - A comment can be created by entering the # sign at the beginning of the line. Helpful comments are your best friend as a R developer.

  - Highlight the section of the code you would like to run and press cntl + enter or select the "run" command in the pane. This allows you to develop your models in an effective stepwise manner that cannot be done in the base R.



R Script Development

- There are many different ways that the R language can be used to perform statistical calculations, deploying machine learning algorithms, and manipulating data.
- The learning curve for mastering the R language is fairly substantial and can be rather daunting for inexperienced programmers.
- Rather than going through all of the mechanics and nuances of the language, I will focus on the practical application of R scripts and give an overview designed to get you up and running quickly.
- There will be a number of questions that you will come up with and I would recommend drawing from your googling skills, seek out the R community, and fork some code that's tailored to your problem.



R Script Development

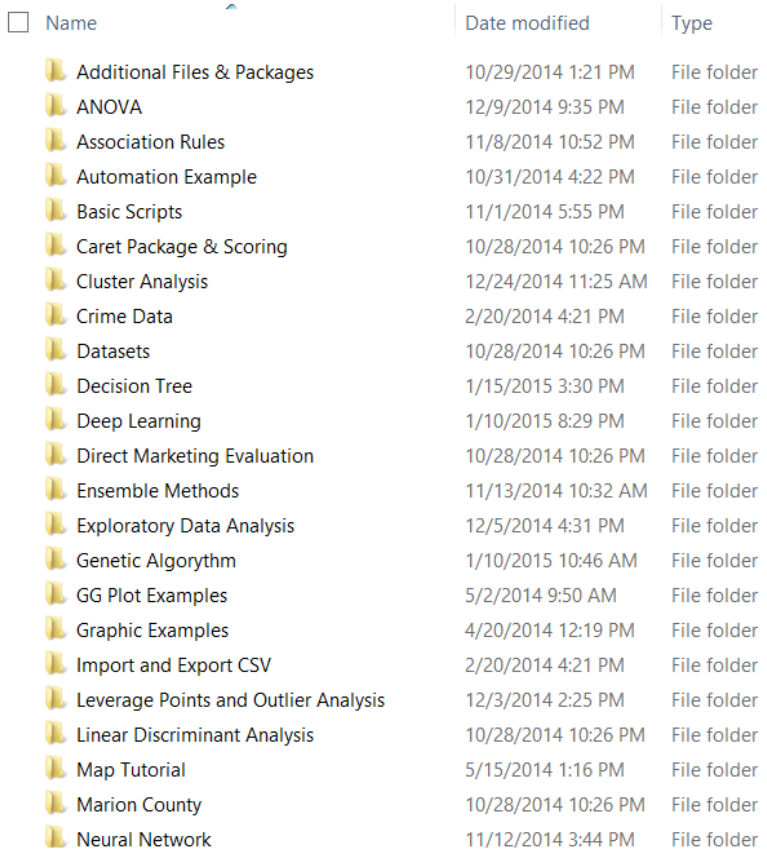
The basic format for model building I use in R is the following:

- Set the working directory
- Load the libraries
- Load the data
  - Transformation of variables
- EDA
  - Summary Statistics
  - Train and Test Split
- Build and Evaluate Model
  - Develop model on Training data
  - Tweak tuning parameters on model
  - Evaluate model performance on Test data
- Apply Model to Data
- Export Results

R Script Development

- We should create a folder that contains the R model, import dataset (if applicable), and export the result dataset into this folder.
- This will keep our projects nice and organized and allow for easy retrieval of our code.
- This code sets the working directory:

```
#############################################################
# Decision Tree Tutorial
#############################################################

# Set Working Directory

setwd("C:/Users/Derek/Documents/RPackages/Decision Tree/")
```

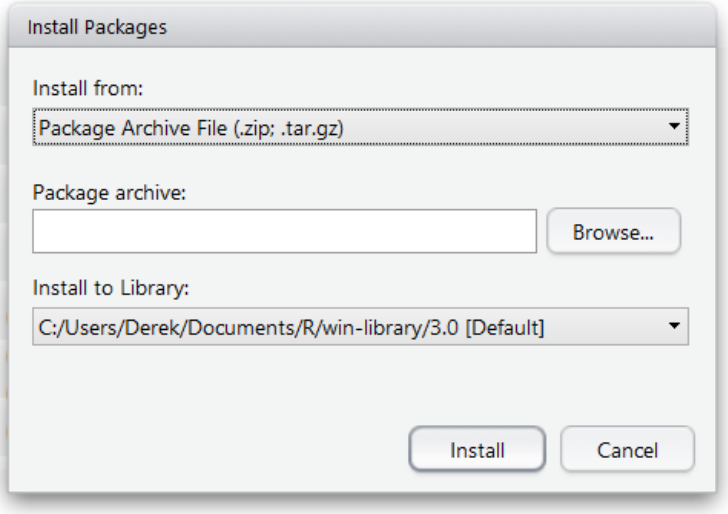


Name	Date modified	Type
Additional Files & Packages	10/29/2014 1:21 PM	File folder
ANOVA	12/9/2014 9:35 PM	File folder
Association Rules	11/8/2014 10:52 PM	File folder
Automation Example	10/31/2014 4:22 PM	File folder
Basic Scripts	11/1/2014 5:55 PM	File folder
Caret Package & Scoring	10/28/2014 10:26 PM	File folder
Cluster Analysis	12/24/2014 11:25 AM	File folder
Crime Data	2/20/2014 4:21 PM	File folder
Datasets	10/28/2014 10:26 PM	File folder
Decision Tree	1/15/2015 3:30 PM	File folder
Deep Learning	1/10/2015 8:29 PM	File folder
Direct Marketing Evaluation	10/28/2014 10:26 PM	File folder
Ensemble Methods	11/13/2014 10:32 AM	File folder
Exploratory Data Analysis	12/5/2014 4:31 PM	File folder
Genetic Algorythm	1/10/2015 10:46 AM	File folder
GG Plot Examples	5/2/2014 9:50 AM	File folder
Graphic Examples	4/20/2014 12:19 PM	File folder
Import and Export CSV	2/20/2014 4:21 PM	File folder
Leverage Points and Outlier Analysis	12/3/2014 2:25 PM	File folder
Linear Discriminant Analysis	10/28/2014 10:26 PM	File folder
Map Tutorial	5/15/2014 1:16 PM	File folder
Marion County	10/28/2014 10:26 PM	File folder
Neural Network	11/12/2014 3:44 PM	File folder

R Script Development

- Most everything that we will want to do will require a 3rd party package.
- A package can be installed and the library loaded through executing the following script.
- In rare instances, packages cannot be installed this way. (Ex. tm package for text mining)
- There is a workaround in R-Studio: Tools – Install Packages.
- You may have to download the .zip file from the developers website.

```
#########################################
# Install a Package and load the library
#########################################

install.packages("ROCR")
library(ROCR)
```



R Script Development

- Here is the code we can use to load libraries.
- Notice that we are loading multiple libraries for this decision tree model project.
- By setting the working directory, we are able to load the csv file through shorthand notation.
- The dataset is called mydata and the "<-" tells R to apply this as a dataframe.
- If you want to refer to the full directory path for loading the .csv file, this can also be accomplished.

```
# Load Libraries

library(caret)
library(rpart)
library(rpart.plot)
library(C50)
library(rattle)
library(party)
library(partykit)
library(RWeka)
library(randomForest)
library(ROCR)
library(xlsx)
```

```
#################################
# Load the data
#################################

mydata <- read.csv("Diabetes.csv")
summary(mydata)
```

```
mydata<- read.csv("C:/Users/Derek/Documents
                  /RPackages/Deep Learning/mydata.csv")
```

R Script Development

- Data can also be imported/exported from a variety sources instead of a .csv (ODBC, HTML, XML, HDFS, .txt, .xlsx, etc…)

```
# Open a connection to the MS SQL Server

library(RODBC)

MSSQLServer <- odbcConnect(dsn="test", uid="myself",
                           pwd="Risgreat")

# This is how to import a table dbo.Customer into R as a
# Data Frame R.Customer

R.Customer <- sqlFetch(MSSQLServer, "Customer")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# This is sample code to append data into a SQL Table

sqlUpdate(channel=MSSQLServer, dat=R.Customer,
          tablename="Customer", index="ID")


# Close the connection to the SQL server

odbcClose(MSSQLServer)
```

R Script Development

- Once we have the data loaded in R, we can begin to transform the variables.
- This is where we would address data transformations, changing variable types, removing variables, dummy coding, etc…
- In our example, we mathematically applied a log transformation to the Pedigree variable and created a new variable called "logPedigree".
- Then we will delete the variables Pedigree and BloodPressure from the dataset.
- Also, the Class variable has been changed from a numeric integer data type into a factor data type.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Recoding and Transforming variables
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mydata$logPedigree <- log(mydata$Pedigree)

mydata$Pedigree <- NULL
mydata$BloodPressure <- NULL


mydata$Class <- as.factor(mydata$Class)
```

R Script Development

```
################################
# EDA
################################

# Correlation Matrix

mcor<-cor(mydata)
round(mcor, digits=2)

# Scatterplot for all variables

plot(mydata)


# This code will split the dataset into a
# train and test set. We will use a 70%
# training & 30% testing split here.

set.seed(1234)

ind <- sample(2, nrow(mydata), replace=TRUE,
              prob=c(0.7, 0.3))

trainData <- mydata[ind==1,]
testData <- mydata[ind==2,]
```

- The EDA section is devoted to assessing the variables for the model building and will discussed at length in the other tutorials.
- An important aspect of model building in R is the splitting of the data into a training and test dataset.
- The seed (1234) is important and allows for us to recreate the results exactly as R produced them. Use seeds before random data splitting.
- The mechanics and intricacies of the R language for manipulating data can be seen here when creating the data split.

R Script Development

- Depending on the predictive model that we are using, we need to run some diagnostics that assess the performance.
- This code will change depending on the model we are using.
- An important function in R is the predict function which applies the model to data.
- This function (first line of code) is creating a new variable on the testData called Yhat and applying the pruned decision tree model (pfit) to the testData.
- The resulting prediction is now in the Yhat variable.

```
##########################################
# Evaluate Model Performance
##########################################

# ROC and AUC for pruned model

testData$YHat <- predict(pfit, testData,
                         type="prob")

fit.scores <- prediction(testData$YHat[,2],
                         testData$Class)
fit.perf <- performance(fit.scores,
                        "tpr", "fpr")

# Plot the ROC curve

plot(fit.perf, col = "green", lwd = 1.5)
abline(0,1,col="Red")

# AUC for the decision tree

fit.auc <- performance(fit.scores, "auc")
fit.auc
```

R Script Development

- The final dataset (testData) now contains the predictive analytics results.
- We will now export these results from the R program into a file that can be used in other software interfaces. (Ex. MS SQL Server, Oracle, Tableau, MS Excel, etc…)

```
#################################################################
# Export Results to MS Excel
#################################################################

write.xlsx(testData, "C:/Documents/Decision Tree/ModelExport.xlsx",
           sheetName="Sheet1", col.names=TRUE, row.names=TRUE,
           append=FALSE, showNA=TRUE)
```

Additional R Tips

- We should create an R Script that contains snippets of R code that you have used for various applications and models.
- This recipe book will be extremely helpful when you need to perform a specific task and cannot remember exactly how to write the code.
- With the rich variety of notation available from 3rd party developers, this will make the difference between being a good data scientist and a great one.

```
################################################
# R Code Snippets, Functions, and Tips
################################################

# Update a column in a dataset with parameters.
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

testData$class[testData$class == "12"] <-"1"

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add a variable to a string of text -
# Note that %s describes the variable.
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

x <- 2349
sprintf("Substitute in a string or number: %s", x)

sprintf("Can have multiple %s
        occurrences %s", x, "- got it?")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Declare a variable and pass that
# variable to .csv filesave
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Var1 <- '#ChicagoBulls'

write.csv(mydata, file=sprintf('C:/Users/Derek/
          Documents/%s.csv', Var1), row.names=F)
```

Additional R Tips

- With so many different packages available it is hard to find an exhaustive list. Here are some packages which I utilize on a regular basis.

```
install.packages('ggplot2')
install.packages('sqldf')
install.packages('forecast')
install.packages('plyr')
install.packages('RODBC')
install.packages('lubridate')
install.packages('reshape2')
install.packages('randomForest')
install.packages('XLConnect')
install.packages('xlsx')
install.packages('survival')
install.packages('shiny')
install.packages('ggmap')
install.packages('rJava')
install.packages('ROCR')
install.packages('DAAG')
```

```
install.packages('party')
install.packages('rpart')
install.packages('partykit')
install.packages('RWeka')
install.packages('evtree')
install.packages('C50')
install.packages('caret')
install.packages('e1071')
install.packages('tm')
install.packages('wordcloud')
install.packages('SnowballC')
install.packages('GGally')
install.packages('car')
install.packages('corrplot')
install.packages('languageR')
install.packages('Design')
```

```
install.packages('TTR')
install.packages('rattle')
install.packages('arules')
install.packages('doParallel')
install.packages('foreach')
install.packages('foreign')
install.packages('gclus')
install.packages('lattice')
install.packages('MASS')
install.packages('nnet')
install.packages('parallel')
install.packages('pROC')
install.packages('Rfacebook')
install.packages('RgoogleMaps')
install.packages('ROCR')
install.packages('survival')
```

Additional R Tips

- R allows for the use of parallel processing. This is particularly useful because of the manner in which models are processed in- memory.
- If you have complex models that are having trouble being processed in a timely manner or want to take advantage of your Big Data / Hadoop HDFS system, the "parallel" package is certainly worth taking a look at.

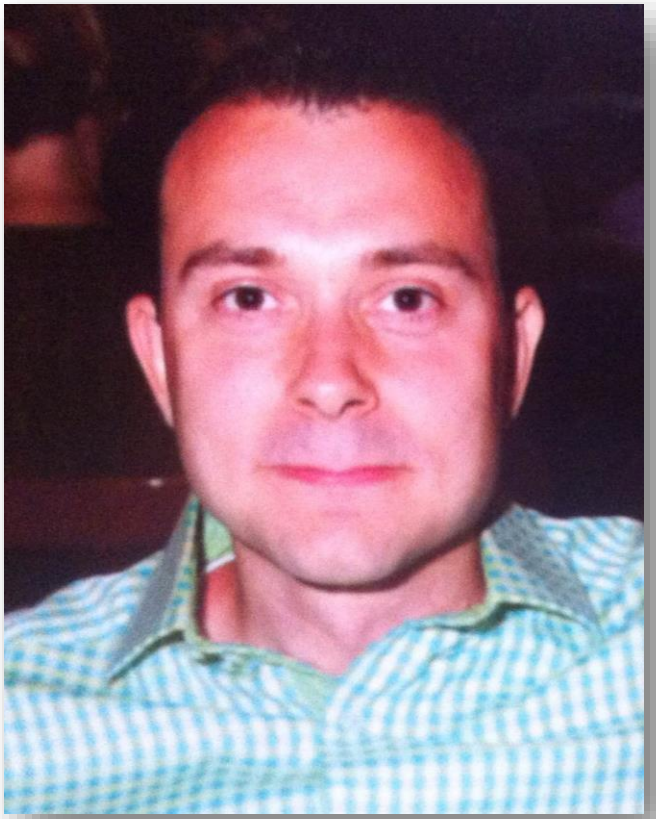
```
# This code allows the user to
# specify the number of cores to use for
# parallel processing.

library(parallel)
myfun <- function(i) { Sys.sleep(1); i }
mclapply(1:8, myfun, mc.cores=4)

# 4 Cores will be used.
```

Additional R Tips

- This tutorial is only intended on showing some of the functionality of the R language through example. Please seek out the numerous tutorials available on the internet to build up your skills.
- Don't worry if you are struggling with getting your models to process at first; the language is difficult.
- However, the payoff for learning the R language is great and opens up the full spectrum of statistical methods, web scrapping, big data, and machine learning.
- Some of the techniques are at the cutting edge of human knowledge and unavailable in SAS / SPSS software platforms.
- Good luck and welcome to R!!!

About Me

- Reside in Wayne, Illinois
- Active Semi-Professional Classical Musician (Bassoon).
- Married my wife on 10/10/10 and been together for 10 years.
- Pet Yorkshire Terrier / Toy Poodle named Brunzie.
- Pet Maine Coons' named Maximus Power and Nemesis Gul du Cat.
- Enjoy Cooking, Hiking, Cycling, Kayaking, and Astronomy.
- Self proclaimed Data Nerd and Technology Lover.



Fine