Introduction to Programming in R
Department of Mathematics & Statistics, UNR

Paul J. Hurtado

Mathematics and Statistics
University of Nevada, Reno

Date: April 7, 2016

R scripts for all examples below (and more), and the `*.Rnw` (LaTeX/`knitr`) files for these slides can be downloaded at

http://www.pauljhurtado.com/R/

Overview

RStudio

FREE at http://www.rstudio.com

R-Studio is an IDE/GUI for R that adds a few useful features.

- Improved GUI, package management, coding tools:
  Code Completion, Syntax Highlighting, ...
- Consistency across platforms: Windows, OS X, Linux
- Integrate R code/output using `knitr` + R Markdown in HTML, LaTeX, and MS Word documents.
- Interactive Graphics with `Shiny`, `ggvis`.

Community Resources at https://www.rstudio.com/

What is R?

Language: Object-Oriented, high-level language based on **S**. Interpreted (uses scripts), similar to Python, Matlab.

Software: Modular. Packages download from CRAN (easy install from inside R). Free under Gnu GPL & other public licenses. RStudio is separate, licensed under AGPL v3

Resources:

- R Website: `http://www.r-project.org/`
- Quick R: `http://www.statmethods.net/`
- RStudio: `http://www.rstudio.com/`
- Paul's R Resources Page: `http://www.pauljhurtado.com/R/`
- **Google!**

R vs Matlab

Like Matlab, **R** is widely used as a computing tool.

Syntax is very similar between R and Matlab!

R excels at statistics, graphics, many packages available, free!
Matlab is better optimized, well supported, widely used, slightly better learning curve.

R command "cheat sheet" for Matlab users:
`http://mathesaurus.sourceforge.net/octave-r.html`

David Hiebler's Matlab/R Reference
`http://math.umaine.edu/~hiebeler/comp/matlabR.html`

For a more detailed comparison, see this book chapter.

SAS? Python? Etc?

R competes well against **SAS**, **Minitab**, **Python**, etc.

- http://r4stats.com/articles/popularity/
- http://www.analyticsvidhya.com/blog/2015/05/infographic-quick-guide-sas-python/
- http://www.burtchworks.com/2015/05/21/2015-sas-vs-r-survey-results/

Python is a strong contender! Popular in physics, engineering, web development, SAGE is python based, etc. **R** slower, but excels at statistics and graphics.

See this R vs Python comparison for details:
https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis

Packages exist to run **R** code within **Python**, and *vice versa*!

Microsoft Adopts R

Microsoft bought Revolution Analytics in spring of 2015.

Microsoft now offers an enhanced versions of **R** for commercial use (*free to academics*) called **Microsoft R Open (MRO)**.
https://mran.revolutionanalytics.com/open/

Microsoft plans to integrate **R** into SQL Server, other offerings.

This may increase demand for employees familiar with **R**!

For a list of other companies using R, see
http://www.revolutionanalytics.com/companies-using-r

R Pros?

1. **R** for **statistics**, or as a **general computing platform**
2. **Free** and widely used in academia and industry
3. Many resources to support **teaching** and **research**
4. Integrates well with other software
5. Many scientists already use **R** (but not Matlab, SAS, etc.)

Cons?

1. *Slow!* Not a low-level language
2. Symbolic tools are limited
3. Integration with C/C++ probably better in Python
4. Updates can "break" code (see MRO above)
5. **Learning Curve!**
   R is lower than specialized "point-and-click" tools

Resources

Self-tutorials:

1. Interactive **R** sessions via `swirl` @ http://swirlstats.com/ and at http://tryr.codeschool.com/
2. **R Intro** (PDF) at www.pauljhurtado.com/R/RIntro.pdf

Other Resources:

1. **Quick-R** @ www.statmethods.net
2. www.pauljhurtado.com/R
3. www.revolutionanalytics.com/r-language-resources
4. **R Style Guide:** google.github.io/styleguide/Rguide.xml
   Hadley Wickham's: adv-r.had.co.nz/Style.html
5. www.r-project.org
6. **Google!**

Examples

Overview of Examples

1. **R** Language Basics
2. Graphics
3. Data
4. Statistics
5. Networks
6. Numerical Solutions to Differential Equations
7. Optimization
8. Speeding up **R**

R Language Basics

```
y = 1 + 1  # Most R users instead write `x <- 1+1'
Y <- 3
Y + y  # R is case sensitive!

## [1] 5

# Variable names must start with a letter. Use '.' but avoid '_' in names.
# Standard objects are lists, data frames, etc. NOT vectors and matrices
# like Matlab.

long.variable.name = c(-2, -1, 0)
class(long.variable.name)

## [1] "numeric"

0.5 * long.variable.name + long.variable.name^2  # element-wise vector operatio

## [1] 3.0 0.5 0.0

long.variable.name[2]

## [1] -1
```

R Language Basics: Data Frames

```
# Data Frames are more like spread sheets than matrices...
x = data.frame(A = 3:1, B = long.variable.name, C = 1)
x  # class(x) is 'data.frame'

##   A  B C
## 1 3 -2 1
## 2 2 -1 1
## 3 1  0 1

x[2, 2]  # row,column addressing

## [1] -1

x[, 2]  # all rows, 2nd column

## [1] -2 -1  0

x[c(1, 3), ]  # 1st and 3rd rows, all columns

##   A  B C
## 1 3 -2 1
## 3 1  0 1

names(x)  # see also str(x)\t\t

## [1] "A" "B" "C"
```

R Language Basics: Data Frames (cont'd)

```
x$B  # access columns of data via column names

## [1] -2 -1  0

x["A"]  # class(x['A']) is data.frame

##   A
## 1 3
## 2 2
## 3 1

x[["A"]]  # class(x[['A']]) is numeric

## [1] 3 2 1

x[, c(TRUE, FALSE, TRUE)]  # Subset columns with logical vectors

##   A C
## 1 3 1
## 2 2 1
## 3 1 1

x[x$A >= 2, ]  # useful for subsetting data!

##   A  B C
## 1 3 -2 1
## 2 2 -1 1
```

R Language Basics: Matrices

```
x  # Here is our data frame. Coerce it into a proper matrix, A...
```

```
##   A  B C
## 1 3 -2 1
## 2 2 -1 1
## 3 1  0 1
```

```
A <- as.matrix(x)  # see also matrix()
A %*% t(A)  # computes A A'. See www.statmethods.net/advstats/matrix.html
```

```
##      [,1] [,2] [,3]
## [1,]   14    9    4
## [2,]    9    6    3
## [3,]    4    3    2
```

```
eigen(A, only.values = FALSE)  # eigenvectors are columns of `vectors'
```

```
## $values
## [1] 2.000000e+00 1.000000e+00 3.616722e-17
##
## $vectors
##           [,1]          [,2]       [,3]
## [1,] 0.5773503 -1.404333e-16  0.5773503
## [2,] 0.5773503 -4.472136e-01  0.5773503
## [3,] 0.5773503 -8.944272e-01 -0.5773503
```

R Language Basics: Functions

See ?sample for documentation (RStudio: type sample, F1)

```
sample  # function name alone, no '()', will often display useful code!
## function (x, size, replace = FALSE, prob = NULL)
## {
##     if (length(x) == 1L && is.numeric(x) && x >= 1) {
##         if (missing(size))
##             size <- x
##         sample.int(x, size, replace, prob)
##     }
##     else {
##         if (missing(size))
##             size <- length(x)
##         x[sample.int(length(x), size, replace, prob)]
##     }
## }
## <bytecode: 0x0000000011335c98>
## <environment: namespace:base>
sample(1:5)  # shuffles 1:5
## [1] 2 5 4 3 1
sample(1:5, replace = T, size = 10)  # 10 iid random numbers; discrete uniform
##  [1] 5 5 4 4 1 2 1 4 2 4
```

R Language Basics: Custom Functions

```
sqrt2 <- function(x) {
    retval <- x * NaN  # initialize
    for (k in 1:length(x)) {
        if (x[k] < 0) {
            retval[k] = sqrt(x[k] + (0+0i))
        } else {
            retval[k] = sqrt(x[k])
        }
    }
    return(retval)
}  # for() loops are SLOW! :-(

sqrt3 <- function(x) {
    sqrt(x + (0+0i))
}  # Faster! :-)

z <- rnorm(3, mean = 0, sd = 1)  # 3 random Normal(0,1) values
rbind(sqrt(z), sqrt2(z), sqrt3(z))  # Compare

##              [,1]         [,2]         [,3]
## [1,]           NA           NA 0.5087123+0i
## [2,] 0+0.980782i 0+0.5408565i 0.5087123+0i
## [3,] 0+0.980782i 0+0.5408565i 0.5087123+0i
```

R Language Basics: Packages

```
names(iris) # Built-in data set. See library(help = "datasets")
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```
# Download and Install packages via the menus, or in the script:
install.packages("dplyr") # THIS ONLY NEEDS TO INSTALL ONCE!
```

```
# Use without 'loading' package: packagename::functionname()
# This function subsets the 'iris' dataset.
setosa <- dplyr::filter(iris, Species=='setosa')

# load packages at the top of the script with library()
library(dplyr)  # load 'dplyr' functions into workspace

# now filter() can be called directly...
setosa <- filter(iris, Species=='setosa')
quantile(setosa$Petal.Length/setosa$Petal.Width)
```

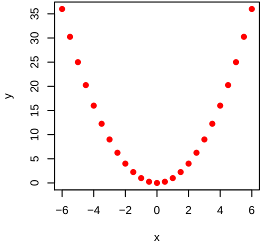
```
##        0%       25%       50%       75%      100%
##  2.666667  4.687500  7.000000  7.500000 15.000000
```

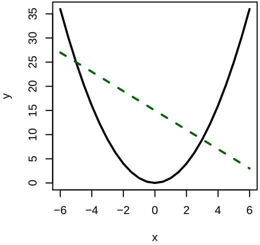
More about **R** packages at

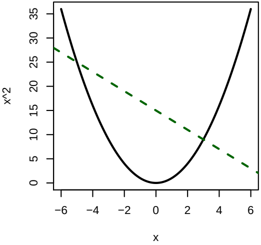
http://www.statmethods.net/interface/packages.html

Base Graphics

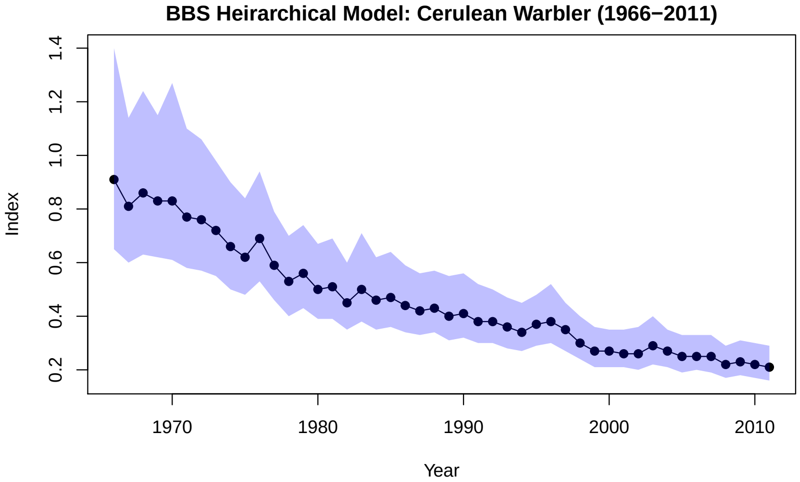
```
x = seq(-6, 6, length = 25)
y = x^2
# 1st plot:
plot(x, y, col = "red", pch = 19)  # default: open circles, pch=type
# 2nd plot:
plot(x, y, type = "l", lwd = 2)  # specify a line, not points
# add a line
points(x, 15 - 2 * x, type = "l", lwd = 2, col = "darkgreen", lty = 2)
# Alternative (3rd plot):
curve(x^2, from = -6, to = 6, lwd = 2)  # draw a function of x
abline(15, -2, col = "darkgreen", lwd = 2, lty = 2)  # give intercept, slope
```



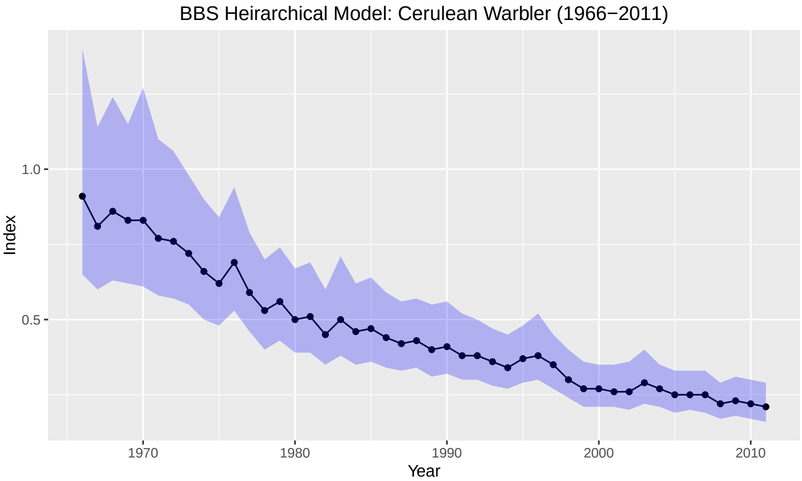




Base Graphics



Extended Graphics: ggplot2, lattice, ...



Compare examples in base graphics vs ggplot2 at:

flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics

Extended Graphics: rgl

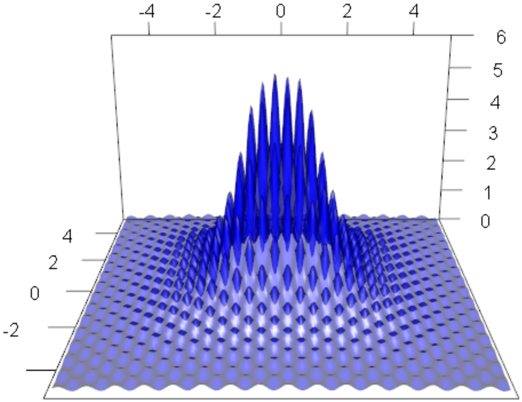
Here's a 3D surface plot using `rgl`:

$$z = 2\frac{(\sin(10x)\ \cos(10y) + 2)}{\sqrt{x^4 + y^4 + 1}}$$

```
library(rgl)
fun = function(x,y) { 2*(sin(10*x)*cos(10*y)+2)/sqrt(x^4+y^4+1) }

# Plot the surface
x=seq(-5,5,length=200) # tick marks on x axis
y=seq(-5,5,length=200) # tick marks on y axis; defines grid for...
z=outer(x,y,fun) # matrix for plotting -- z vals / height of surface
surface3d(x,y,z,col="blue",alpha=0.5)
axes3d()
rgl.viewpoint(theta=0, phi=-70, fov=50, zoom=0.7)
```

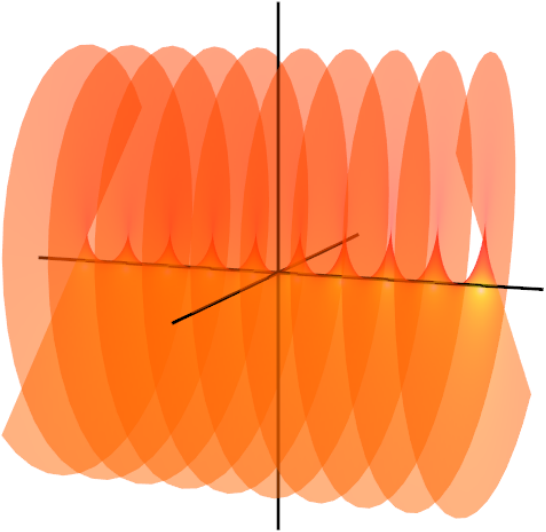
In R: Use your cursor/mouse to rotate in 3D!

Extended Graphics: rgl



Extended Graphics: rgl

$$x = u\cos(v),\ y = u\sin(v),\ z = v$$



Data Manipulation

Merge and reshape data: dplyr, tidyr, reshape2, ...

```
head(dat1,4)

##   ID Species  Weight1  Weight2
## 1  1       A 10.82000 11.27014
## 2  2       B 12.11148 12.44722
## 3  3       C 13.00420 13.23085
## 4  4       A 11.04800 10.89227

dat2

##   Species Avg.Weight
## 1       A         11
## 2       B         12
## 3       C         13

dat3 = merge(dat1,dat2,by="Species",sort=FALSE)
head(dat3,3)

##   ID Species Avg.Weight  Weight1  Weight2
## 1  1       A         11 10.82000 11.27014
## 4  2       B         12 12.11148 12.44722
## 8  3       C         13 13.00420 13.23085
```

Data Manipulation

Convert from Wide to Long format with `tidyr::gather()`

```
head(dat3,3)

##   ID Species Avg.Weight  Weight1  Weight2
## 1  1       A         11 10.82000 11.27014
## 4  2       B         12 12.11148 12.44722
## 8  3       C         13 13.00420 13.23085

dat <- gather(dat3,Replicate,Weight,Weight1:Weight2)
dat$Replicate <- type.convert(gsub('Weight','',dat$Replicate))
dat <- dat[order(dat$ID),];  # sort by ID
rownames(dat) <- c() # remove old row numbers
head(dat,5)

##   ID Species Avg.Weight Replicate   Weight
## 1  1       A         11         1 10.82000
## 2  1       A         11         2 11.27014
## 3  2       B         12         1 12.11148
## 4  2       B         12         2 12.44722
## 5  3       C         13         1 13.00420
```

More at `www.statmethods.net` and RStudio's *Data Wrangling* cheatsheet:
www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Statistics: Built-in data sets, Diagnostics, etc

```
head(trees, 1)  # look at the trees data set
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
```

```
# Regression models with and without interaction term
fit1 = lm(Volume ~ Girth + Height, data = trees)
fit2 = lm(Volume ~ Girth * Height, data = trees)

# Compare models via AIC, BIC, ANCOVA
cbind(AIC(fit1, fit2), BIC = BIC(fit1, fit2)[, 2])
```

```
##      df      AIC      BIC
## fit1  4 176.9100 182.6459
## fit2  5 155.4692 162.6391
```

```
anova(fit1, fit2)
```

```
## Analysis of Variance Table
##
## Model 1: Volume ~ Girth + Height
## Model 2: Volume ~ Girth * Height
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     28 421.92
## 2     27 198.08  1    223.84 30.512 7.484e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

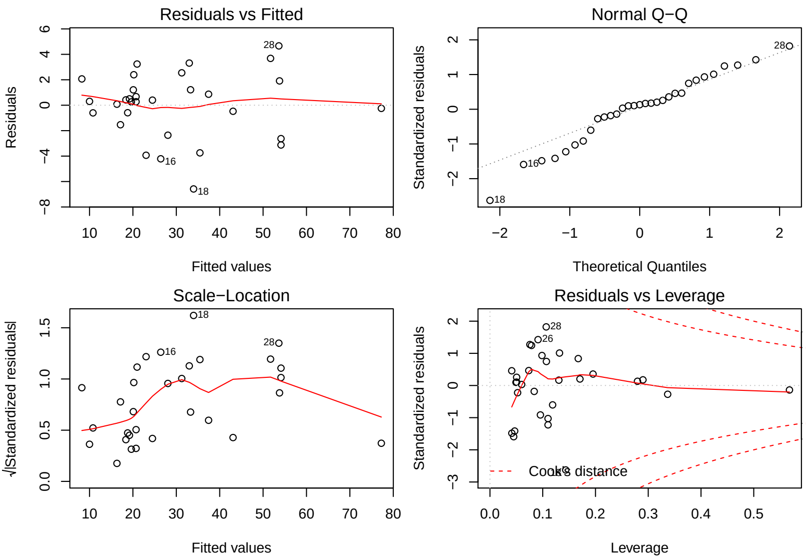
```
summary(fit2)
```

```
##
## Call:
## lm(formula = Volume ~ Girth * Height, data = trees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5821 -1.0673  0.3026  1.5641  4.6649
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.39632   23.83575   2.911  0.00713 **
## Girth        -5.85585    1.92134  -3.048  0.00511 **
## Height       -1.29708    0.30984  -4.186  0.00027 ***
## Girth:Height  0.13465    0.02438   5.524 7.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.709 on 27 degrees of freedom
## Multiple R-squared:  0.9756, Adjusted R-squared:  0.9728
## F-statistic: 359.3 on 3 and 27 DF,  p-value: < 2.2e-16
```

```
fit2$coefficients
```

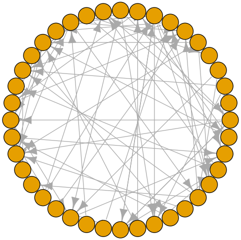
```
##  (Intercept)        Girth       Height Girth:Height
##   69.3963156   -5.8558479   -1.2970834    0.1346544
```

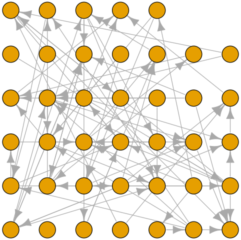
```r
plot(fit2)  # plots 4 diagnostic plots (not the regression line!)
```



Networks

```
library(igraph)
# Generate random adjacency matrix (directed, unweighted graph)
adjM = matrix(rbinom(40^2, size = 1, prob = 0.05), nrow = 40, ncol = 40)
GraphAdjM = graph.adjacency(adjM, mode = "directed", diag = FALSE)
par(mfrow = c(1, 2))
plot.igraph(GraphAdjM, vertex.label = NA, layout = layout_in_circle)
plot.igraph(GraphAdjM, vertex.label = NA, layout = layout.grid)
```





See also the `network` and `sna` packages.

Numerical Solutions to ODEs, PDEs, DDEs

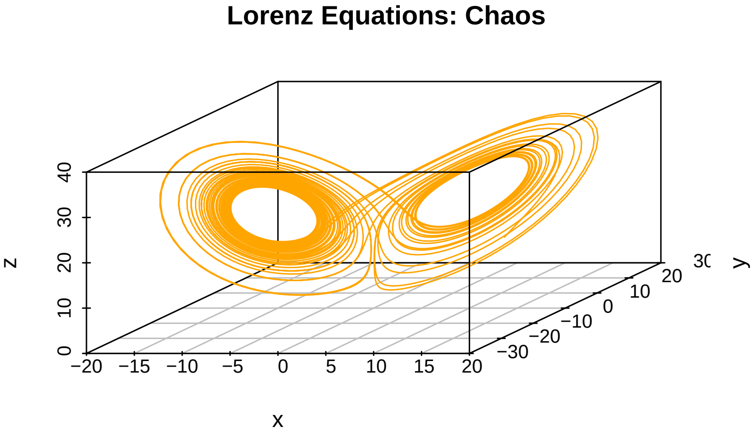
`deSolve` provides Fortran and C implementations of solvers from ODEPACK (LLNL), R-K solvers, and ODE solvers for finite difference approximations of PDEs up to 3D.

```
library(deSolve)
params <- c(sigma=10, r=24.5, b=8/3)
lorenz <- function(t,Y,p) { # ODE Example
        x=Y[1]; y=Y[2]; z=Y[3]; # unpack state variables
        sigma=p[["sigma"]]; r=p[["r"]]; b=p[["b"]] # parameters
        dx=sigma*(y-x) # Model equations
        dy=r*x - y - x*z
        dz=x*y - b*z
        return(list(c(dx,dy,dz))) # Return derivative values
}
Y0=c(x=10, y=11, z=12)  # initial conditions
tvals=seq(0,40,by=0.01) # time points
soln = ode(Y0, func=lorenz, parms=params, times=tvals,
       method="lsoda", rtol = 1e-12, atol = 1e-12)
head(soln,1) # 1st colum = tvals
```

```
##      time  x  y  z
## [1,]    0 10 11 12
```

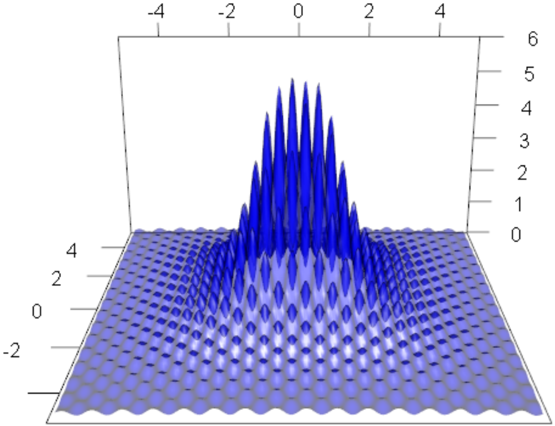
Numerical Solutions to ODEs

```
library(scatterplot3d)
scatterplot3d(soln[,2],soln[,3],soln[,4], type="l", color="orange",
  main="Lorenz Equations: Chaos",xlab="x",ylab="y",zlab="z", angle=30)
```



Optimization

Find the maximum of...



Optimization

Use various methods via `optim()` or `optimx()`.
Here, we use Generalized Simulated Annealing:

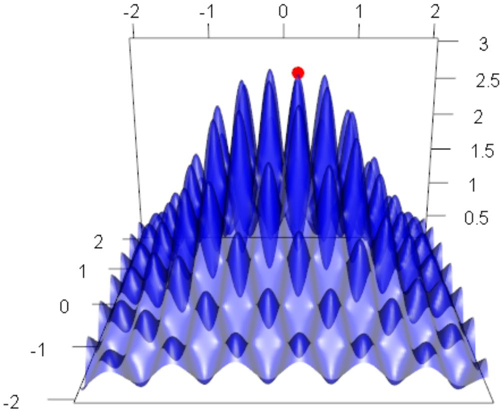
```
fun = function(x,y) { (sin(10*x)*cos(10*y)+2)/sqrt(x^4+y^4+1) }
obj = function(z) { -fun(z[1],z[2])}

## "global" optimization with GenSA
library(GenSA)
fit <- GenSA(par=c(2,2),fn=obj,lower=c(-3,-3),upper=c(3,3))
fit[c('par','value')] # or fit£par; fit£value
```

```
## $par
## [1] 1.568483e-01 1.710852e-12
##
## $value
## [1] -2.99909
```

More info: **CRAN Task View: Optimization**

https://cran.r-project.org/web/views/Optimization.html

Optimum Found!



Constrained Optimization

Feasible region defined by $\mathbf{u_i}\,\theta - \mathbf{c_i} >= 0$. Ex: $\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \geq 3$

```
sumsq = function(vec, n, x, y){
  a = vec[1]; b = vec[2]
  sum(y^2)-2*a*sum(y)-2*b*sum(x*y)+n*a^2+2*a*b*sum(x)+b^2*sum(x^2)
}
n =15
x = 1:n
y = 3 + 1.5*x + rnorm(n, 0, 1)
ui = c(1, -1)
ci = 3
constrOptim(theta=c(4, -1), sumsq, grad=NULL, ui, ci, n=n, x=x, y=y)[1:2]
```

```
## $par
## [1] 4.382542 1.382542
##
## $value
## [1] 12.833
```

Speeding up R: Coding tricks

R can be slow, but there are a few tricks to speed it up!

1. Avoid `for()` and `apply()` functions
2. Vectorize!
3. Use fast functions in C/fortran based packages
4. Link to C/Fortran code via `Rcpp`
5. Use `compiler::cmpfun()`,
6. Multiple cores? Use the `parallel` package
7. Compile R yourself

Resources:

http://www.noamross.net/blog/2013/4/25/faster-talk.html
http://www.r-bloggers.com/how-to-go-parallel-in-r-basics-tips/

Documents with Integrated **R**

RStudio

FREE at http://www.rstudio.com

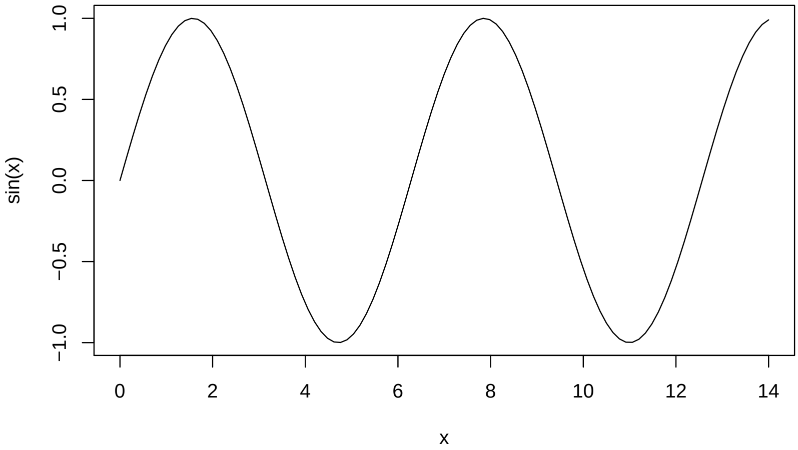
R-Studio is an IDE/GUI for R that adds a few useful features.

- Improved GUI, package management, coding tools:
  Code Completion, Syntax Highlighting, ...
- Consistency across platforms: Windows, OS X, Linux
- **Integrate R code/output using `knitr` + R Markdown in HTML, LaTeX, and MS Word documents.**
- Interactive Graphics with `Shiny`, `ggvis`.

Minimal Example

Here's how to plot the curve $\sin(x)$ in **R**:

```
curve(sin(x),from=0,to=14);
```



LaTeX + R using the knitr package

Here's the LaTeX+**R** that created the previous slide:

```
\documentclass{beamer}
\begin{document}
\setbeamertemplate{navigation symbols}{}
% The usual minimal beamer slide...
\begin{frame} \frametitle{Minimal Example}
    Here's how to plot the curve $\sin(x)$ in \textbf{R}:
    % The next three lines are parsed by R's knitr package!
    <<curve-example, tidy=FALSE, fig.height=4, fig.width=7, echo=-1>>=
        par(oma=c(0,0,0,0), mar=c(4,4,1,1)) # Parsed, but omitted by "echo=-1"
        curve(sin(x),from=0,to=14);
    @
\end{frame}

\end{document}
```

To configure TeXstudio to compile *.Rnw files:
http://www.pauljhurtado.com/latex/texstudio.html

Fancy `knitr` tables with `kable`

```
# standard data frame output:
head(iris,3)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa

# kable() output.
knitr::kable(head(iris,4), caption="The iris data set.",
             booktabs=TRUE, align="c")
```

Table: The iris data set.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa

Fancy tables with `stargazer`

```
library(stargazer)
fit1 <- lm(mpg ~ wt, mtcars)
fit2 <- lm(mpg ~ wt + hp, mtcars)
stargazer(fit1, fit2, title="Cars Data Set", single.row=TRUE,
covariate.labels=c("Weight (lb/1000)","Gross Horsepower"))
```

Table: Cars Data Set

	Dependent variable:	
	mpg	
	(1)	(2)
Weight (lb/1000)	−5.344*** (0.559)	−3.878*** (0.633)
Gross Horsepower		−0.032*** (0.009)
Constant	37.285*** (1.878)	37.227*** (1.599)
Observations	32	32
R^2	0.753	0.827
Adjusted R^2	0.745	0.815
Residual Std. Error	3.046 (df = 30)	2.593 (df = 29)
F Statistic	91.375*** (df = 1; 30)	69.211*** (df = 2; 29)
Note:		*p<0.1; **p<0.05; ***p<0.01

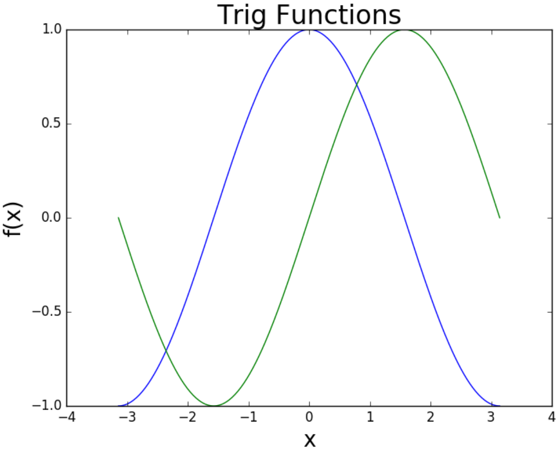
See also: http://jakeruss.com/cheatsheets/stargazer.html and these examples.

LaTeX + R + Python?!

Python output via **R** and `knitr`:

```
import numpy as np
import matplotlib.pyplot as plt

x = 'hello, python world!'
print(x)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)
plt.plot(X, C)
plt.plot(X, S)
plt.ylabel('f(x)',size=20)
plt.xlabel('x',size=20)
plt.title('Trig Functions',size=24)

plt.savefig("pyplotexample.png")

## hello, python world!
```

Python Example Continued...

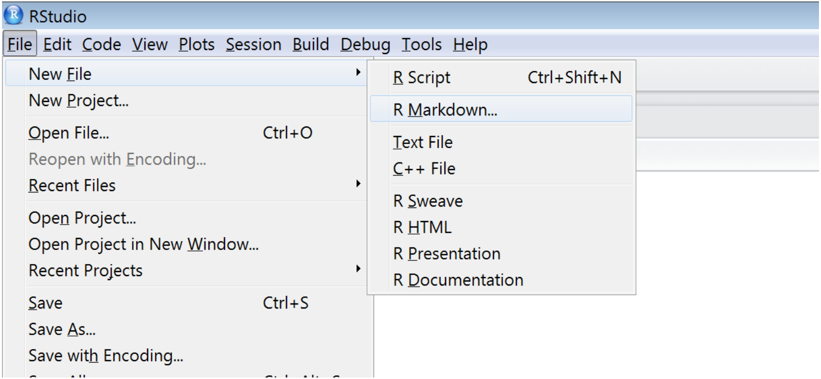


The LaTeX...

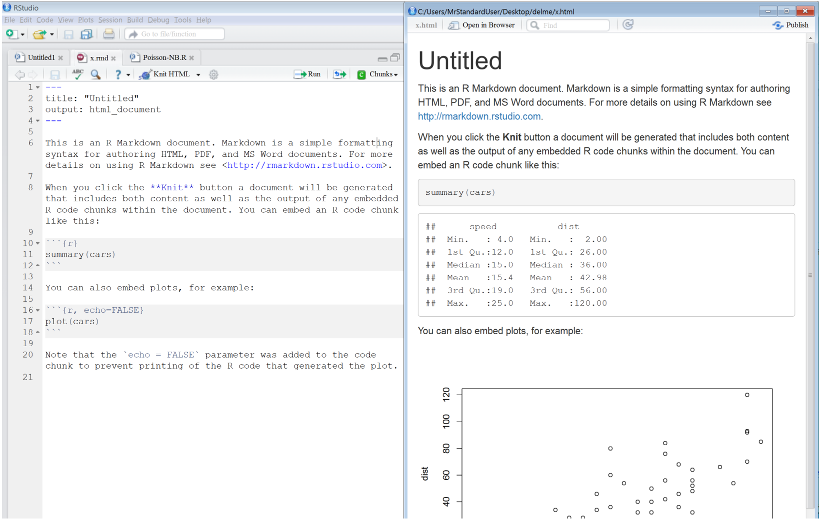
```
% Python 3 example
\begin{frame}[fragile,t]\frametitle{LaTeX + R + Python?!}\large
Python output via \textbf{R} and \texttt{knitr}:
<<hi-python, engine='python'>>=
x = 'hello, python world!'
:
plt.savefig("pyplotexample.png")
@
\end{frame}

\begin{frame}[fragile,t]\frametitle{Python Example Continued...}\large
    \centerline{\includegraphics[height=.7\textheight]{pyplotexample.png}}
\end{frame}
```

R Markdown

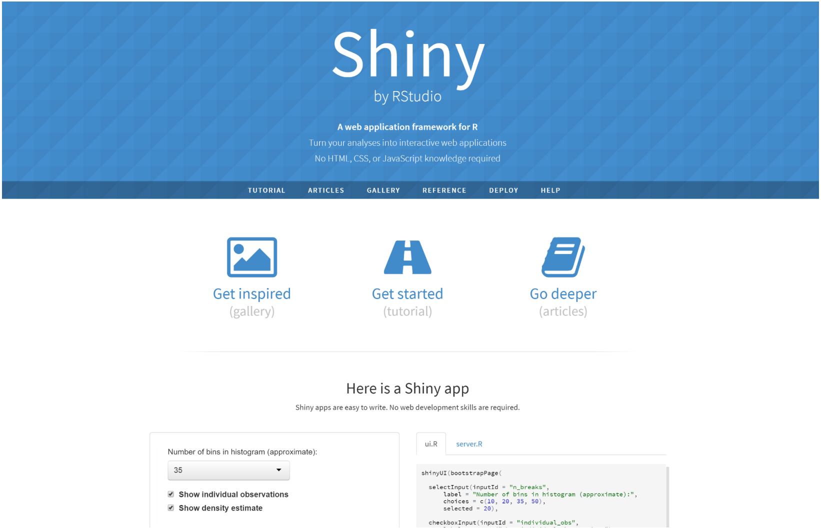
Create an R Markdown document in RStudio...



R Markdown


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Untitled1
x.rmd
Poisson-NB.R
Knit HTML
Run
Chunks
---
title: "Untitled"
output: html_document

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r}
summary(cars)
```
You can also embed plots, for example:
```{r, echo=FALSE}
plot(cars)
```
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
C:/Users/MrStandardUser/Desktop/delme/x.html
x.html
Open in Browser
Find
Publish
Untitled
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.
When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
summary(cars)
speed dist
Min. : 4.0 Min. : 2.00
1st Qu.:12.0 1st Qu.: 26.00
Median :15.0 Median : 36.00
Mean :15.4 Mean : 42.98
3rd Qu.:19.0 3rd Qu.: 56.00
Max. :25.0 Max. :120.00
You can also embed plots, for example:
dist
40
60
80
100
120



Examples and tutorials at http://shiny.rstudio.com/

Introduction to Programming in **R**

Getting Started

Guided, interactive **R** sessions are a great way to begin!

1. Work through Paul's Intro to R (PDF) at `http://pauljhurtado.com/R/RIntro.pdf`
2. RStudio's page: Getting Started with R
3. The **Try R** website at `http://tryr.codeschool.com`
4. Interactive sessions in **R** with `swirl` at `http://swirlstats.com/students.html`
5. **R scripts for the examples above**, and other resources, can be found at `http://pauljhurtado.com/R/`

Go Play! ☺