CS545: Machine Learning
Introduction to R and LaTeX

Chuck Anderson

Department of Computer Science
Colorado State University

Fall, 2009
Outline

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
Programming and Computing Environment

What do you (I) want?

R

Why?

Installing and Running R

On-line Documentation

Matrices

Variables

Matrix Operations

Defining New Functions

Lists

Graphics

Using R Source Files

Reading in Data

LaTeX

Why LaTeX?

Simple Example

Math

Sections

Makefiles

Graphics

Running R

Non-interactively
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization

R
Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX
Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization
5. Large community of users and developers.
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization
5. Large community of users and developers.
6. Free
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization
5. Large community of users and developers.
6. Free
Programming and Computing Environment

What do you (I) want?

1. Concise, intuitive programming language.
2. Ability to “play” with data and computation ideas.
   - Data persistence.
   - Matrix, vector structures and operations are easy.
   - Interpreter, using same language as programs.
   - Functional programming style
3. Rich, easy-to-use visualization
5. Large community of users and developers.
6. Free

\textbf{R} satisfies all, except perhaps “fast computation”, except see
http://www.cs.colostate.edu/~anderson/languages.
R is

- a language and environment for statistical computing and graphics;

(excerpts from http://www.r-project.org "What is R?")
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;

(excerpts from http://www.r-project.org "What is R?"

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;
- like S, designed around a true computer language, and it allows users to add additional functionality by defining new functions;
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;
- like S, designed around a true computer language, and it allows users to add additional functionality by defining new functions;
- able to link to your C, C++ and Fortran code at run time;

(excerpts from http://www.r-project.org "What is R?")
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;
- like S, designed around a true computer language, and it allows users to add additional functionality by defining new functions;
- able to link to your C, C++ and Fortran code at run time;
- is the language of choice for many researchers in statistics and machine learning.

(excerpts from http://www.r-project.org "What is R")
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;
- like S, designed around a true computer language, and it allows users to add additional functionality by defining new functions;
- able to link to your C, C++ and Fortran code at run time;
- is the language of choice for many researchers in statistics and machine learning.
R is

- a language and environment for statistical computing and graphics;
- GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues;
- available for Unix, Windows, and MacOS systems;
- like S, designed around a true computer language, and it allows users to add additional functionality by defining new functions;
- able to link to your C, C++ and Fortran code at run time;
- is the language of choice for many researchers in statistics and machine learning.

(excerpts from http://www.r-project.org “What is R?”)
Installing R

R is already installed on the Department of Computer Science network. For other machines, download from http://www.r-project.org by clicking on CRAN (for Comprehensive R Archive Network).
R Packages

You can also download browse the packages that are available at CRAN. Here are just the first few from the alphabetical list

- **ADGoFTest** Anderson-Darling GoF test
- **ADaCGH** Analysis of data from aCGH experiments
- **AER** Applied Econometrics with R
- **AIGIS** Areal Interpolation for GIS data
- **AIS** Tools to look at the data ("Ad Inidicia Spectata")
- **ALS** multivariate curve resolution alternating least squares (MCR-ALS)
- **AMORE** A MORE flexible neural network package
- **AcceptanceSampling** Creation and evaluation of Acceptance Sampling Plans
- **AdMit** Adaptive Mixture of Student-t distributions
- **AdaptFit** Adaptive Semiparametric Regression
- **AlgDesign** AlgDesign
- **Amelia** Amelia II: A Program for Missing Data
- **AnalyzeFMRI** Functions for analysis of fMRI datasets stored in the ANALYZE or NIFTI format
- **Animal** Analyze time-coded animal behavior data
- **AquaEnv** AquaEnv - an integrated development toolbox for aquatic chemical model generation
- **ArDec** Time series autoregressive-based decomposition
- **aCGH.Spline** Robust spline interpolation for dual color array comparative genomic hybridisation data
- **aaMI** Mutual information for protein sequence alignments
- **abind** Combine multi-dimensional arrays
- **accuracy** Tools for testing and improving accuracy of statistical results
- **acepack** ace() and avas() for selecting regression transformations
- **actuar** Actuarial functions
- **ada** Performs boosting algorithms for a binary response
- **adabag** Applies Adaboost.M1 and Bagging
Running R

On Windows, you can double click on the R icon. On our Linux systems, simply run the command \texttt{R}. You should see the following:

\begin{verbatim}
> R
R version 2.9.1 (2009-06-26)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
\end{verbatim}
Quitting R

To exit the R environment,

- type control-d or
** Quitting R **

To exit the R environment,

- type control-d or
- type q() then return or
Quitting R

To exit the R environment,

- type control-d or
- type q() then return or
- type quit() then return
Quitting R

To exit the R environment,

- type control-d or
- type q() then return or
- type quit() then return
Quitting R

To exit the R environment,

- type control-d or
- type q() then return or
- type quit() then return

You will see

```r
> q()
Save workspace image? [y/n/c]: n
>
```

If you answer y, all variables that have been assigned values during this R session into a file named .RData in the current working directory. If you later start R while in this same directory, your variables and their values are automatically made available by loading the .RData file.
R Has No Commands

At the R prompt you must type an expression to be evaluated. Expressions are composed of values, variables, operators, and function calls.

```r
> 42
[1] 42
> "hello"
[1] "hello"
> TRUE
[1] TRUE
> 72 * 22.3
[1] 1605.6
> "hi" + "you"
Error in "hi" + "you" : non-numeric argument to binary operator
```

Oops. Looks like + is not the operator for concatenating strings. What is the correct operator? Find out by searching at [http://www.rseek.org](http://www.rseek.org). The first hit found by searching for concatenate strings tells us to use `paste`.

```r
> paste("hi","you")
[1] " hi you"
```
Want to know more about `paste`? Use the on-line documentation by typing a question mark in front of `paste`.

```
> ?paste
```

and read the result. The examples at the end of the documentation are often very useful. Try them out by typing them into R or just selecting and pasting with the mouse.
You can even look at the implementation of the `paste` function by just evaluating its name.

```
> paste
function (..., sep = " ", collapse = NULL) {
  args <- list(...)
  if (length(args) == 0)
    if (length(collapse) == 0)
      character(0)
    else ""
  else {
    .Internal(paste(lapply(args, as.character), sep, collapse))
  }
}
<environment: namespace: base>
```

Not much here, but we do see that an “internal” function is being called to do the work. Many R functions are defined this way.
What is the value of \((100 \times 2 - 12^2)/7 \times 5 + 2\)?
What is the value of \((100 \times 2 - 12^2)/7 \times 5 + 2\)?

```r
> (100 * 2 - 12^2) / 7 * 5 + 2
[1] 42
```
What is the value of \((100 \times 2 - 12^2)/7 \times 5 + 2\)?

\[
\begin{array}{c}
> (100 \times 2 - 12^2) / 7 \times 5 + 2 \\
[1] \quad 42
\end{array}
\]

What is the value of \(\sin(\pi/2)\)?
What is the value of \((100 \times 2 - 12^2)/7 \times 5 + 2\)?

\begin{verbatim}
> (100 * 2 - 12 ^ 2) / 7 * 5 + 2
[1] 42
\end{verbatim}

What is the value of \(\sin(\frac{\pi}{2})\)?

\begin{verbatim}
> sin(pi/2)
[1] 1
\end{verbatim}

Notice that \(\pi\) is a predefined constant.
How do I find out what other trigonometric functions are available?

“How do I find out...” is the most important question for you to answer, and there are many answers. Here are four.

- Ask R to search for functions involving sin

```r
> help.search("sin")
```
How do I find out what other trigonometric functions are available?

“How do I find out...” is the most important question for you to answer, and there are many answers. Here are four.

- Ask R to search for functions involving sin
  
  ```r
  > help.search("sin")
  ```

- Ask R for R objects with names that match “sin”
  
  ```r
  > apropos("sin")
  ```
How do I find out what other trigonometric functions are available?

“How do I find out...” is the most important question for you to answer, and there are many answers. Here are four.

- Ask R to search for functions involving sin

  > help.search("sin")

- Ask R for R objects with names that match “sin”

  > apropos("sin")

- start a web browser on the on-line documentation,

  > help.start()
How do I find out what other trigonometric functions are available?

“How do I find out...” is the most important question for you to answer, and there are many answers. Here are four.

- Ask R to search for functions involving sin

  > help.search("sin")

- Ask R for R objects with names that match “sin”

  > apropos("sin")

- start a web browser on the on-line documentation,

  > help.start()

- or search the web. For example, use http://www.rseek.org to search using the words “trigonometric functions”.
How do I apply trigonometric functions to arguments in degrees?

You can use the methods suggested above to try to find out. At some point, hopefully soon, you will realize that you are not finding any functions that do this for you. So you must do it yourself...duh.

\[
\sin(90 \times \frac{\pi}{180})
\]

[1] 1
Matrices

- Can I work with vectors and matrices in R?
Matrices

- Can I work with vectors and matrices in R?
- Of course! No data analysis tool is worth the bytes it burns if it doesn’t. To create a matrix, use the `matrix()` function. Try typing `?matrix`.

```r
> matrix(1, 3, 2)
```

Result is a matrix with 3 rows and 2 columns. The first argument provides the contents of the matrix. Values are repeated until matrix is full. Use the concatenate function `c()` to provide a list of numbers.
Matrices

- Can I work with vectors and matrices in R?
- Of course! No data analysis tool is worth the bytes it burns if it doesn’t. To create a matrix, use the `matrix()` function. Try typing `?matrix`.
- How do I construct the matrix

\[
\begin{pmatrix}
10 & 10 \\
10 & 10 \\
10 & 10
\end{pmatrix}
\]
Matrices

- Can I work with vectors and matrices in R?
- Of course! No data analysis tool is worth the bytes it burns if it doesn’t. To create a matrix, use the `matrix()` function. Try typing `?matrix`.
- How do I construct the matrix

\[
\begin{pmatrix}
10 & 10 \\
10 & 10 \\
10 & 10
\end{pmatrix}
\]

\>
`matrix(10,3,2)`

\[
\begin{array}{cc}
[1 , ] & 10 & 10 \\
[2 , ] & 10 & 10 \\
[3 , ] & 10 & 10
\end{array}
\]

Result is a matrix with 3 rows and 2 columns. The first argument provides the contents of the matrix. Values are repeated until matrix is full. Use the concatenate function `c()` to provide a list of numbers.
What is the value of \texttt{matrix(c(1,2,3,4,5,6),3,2)}?
What is the value of `matrix(c(1,2,3,4,5,6),3,2)`?

```
> matrix(c(1,2,3,4,5,6),3,2)
[,1] [,2]
[1,] 1  4
[2,] 2  5
[3,] 3  6
```

The matrix is filled column by column.
What is the value of \texttt{matrix(c(1,2,3),3,2)}?
What is the value of \texttt{matrix(c(1,2,3),3,2)}?

\begin{verbatim}
> matrix(c(1,2,3),3,2)
   [,1] [,2]
[1,] 1 1
[2,] 2 2
[3,] 3 3
\end{verbatim}
How do I learn about other `matrix()` arguments?
How do I learn about other `matrix()` arguments?

As an R programmer showing tremendous potential, you decide to learn for yourself. Try `?matrix`. For a short summary of the arguments try `args(matrix).`
How do I learn about other `matrix()` arguments?

As an R programmer showing tremendous potential, you decide to learn for yourself. Try `?matrix`. For a short summary of the arguments try `args(matrix)`.

```
> args(matrix)
function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
NULL
```

The returned value is `NULL`. Every expression must return some value.
How do I fill a matrix row by row?

```r
> matrix(c(1, 2, 3, 4, 5, 6), 3, 2, byrow = TRUE)
```

The matrix is filled row by row. As you see, arguments can be assigned values by position or by name. You also see that boolean constants in R are `TRUE` and `FALSE`. 

---

**LaTeX**

- Why LaTeX?
- Simple Example
- Math
- Sections
- Makefiles
- Graphics
- Running R
- Non-interactively
How do I fill a matrix row by row?

```r
> matrix(c(1,2,3,4,5,6),3,2,byrow=TRUE)
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

The matrix is filled row by row. As you see, arguments can be assigned values by position or by name. You also see that boolean constants in R are `TRUE` and `FALSE`.
Are there easy ways of specifying sequences, rather than typing `c(1,2,3,4,5,6)`?
Are there easy ways of specifying sequences, rather than typing `c(1,2,3,4,5,6)`?

```r
> c(1,2,3,4,5,6)
[1] 1 2 3 4 5 6
> 1:6
[1] 1 2 3 4 5 6
> seq(1,6,by=1)
[1] 1 2 3 4 5 6
> seq(1,6,by=2)
[1] 1 3 5
> seq(1,by=2,length=6)
[1] 1 3 5 7 9 11
```

Want to know more? Try `?seq`. 

Variables

Okay, I get expressions. Now, how do I assign the result of an expression to a variable?
Variables

Okay, I get expressions. Now, how do I assign the result of an expression to a variable?

```r
> a <- 4
> a
[1] 4
> a <- 4
> b <- 6
> c <- b * b + 2 * a - 2
> c
[1] 42
```
Now, watch this!

> 42 -> humm
> humm
[1] 42

Assignments can be made at the end of an expression, too. This is not used very often as it leads to unreadable code. It is probably allowed to let you decide at the last minute while typing a line to assign the result to a variable. This is not very useful with command line editing. For example, type control-a to give the cursor back to beginning of line and type in a variable name and <- then press return.
Matrix Operations

What is the value of the last expression here?

```
a <- matrix(2, 3, 3)
b <- matrix(1:9, 3, 3)
a * b
```
Matrix Operations

What is the value of the last expression here?

```r
a <- matrix(2,3,3)
b <- matrix(1:9,3,3)
a * b
```

```r
> a <- matrix(2,3,3)
> b <- matrix(1:9,3,3)
> a * b

[,1] [,2] [,3]
[1,]  2   8  14
[2,]  4  10  16
[3,]  6  12  18
```

The * operator does a component-wise multiplication. Use %*% to do matrix multiplication.
What is $a \%\% b$?

```r
> a
  [,1] [,2] [,3]
[1,]  2  2  2
[2,]  2  2  2
[3,]  2  2  2
> b
  [,1] [,2] [,3]
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9
> a %*% b
  [,1] [,2] [,3]
[1,] 12 30 48
[2,] 12 30 48
[3,] 12 30 48
```
A matrix is transposed by the `t()` function.

\[
\begin{bmatrix}
1 & 4 & 7 \\
2 & 5 & 8 \\
3 & 6 & 9 \\
\end{bmatrix}
> b
\]

\[
\begin{bmatrix}
1 & 2 \\
4 & 5 \\
7 & 6 \\
\end{bmatrix}
> t(b)
\]
- A matrix is transposed by the `t()` function.

```r
> b
> t(b)

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>[2,]</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>[3,]</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>
```

- What is `a %*% t(b)`?

```r
> a %*% t(b)

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>24</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>[2,]</td>
<td>24</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>[3,]</td>
<td>24</td>
<td>30</td>
<td>36</td>
</tr>
</tbody>
</table>
```
A matrix is transposed by the `t()` function.

```r
> b
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
> t(b)
[,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

What is `a %*% t(b)`?

```r
> a %*% t(b)
[,1] [,2] [,3]
[1,] 24 30 36
[2,] 24 30 36
[3,] 24 30 36
```
Elements and sub-matrices are easily extracted. Given the previous assignment of \( b \), what are the values of the following expressions?

\[
\begin{align*}
  b[1,1] &> b \\
  b[2,3] &> [1\,,1] [1\,,2] [1\,,3] \\
  b[1\,,] &> [1\,,] 1\ 4\ 7 \\
  b[\,,2] &> [2\,,] 2\ 5\ 8 \\
  b[1:2\,,] &> [3\,,] 3\ 6\ 9
\end{align*}
\]
Elements and sub-matrices are easily extracted. Given the previous assignment of \( b \), what are the values of the following expressions?

\[
\begin{align*}
&b[1,1] > b \\
&b[2,3] \quad \quad \quad [1,] [2,] [3,] \\
&b[1,] \quad \quad \quad [1,] 1 4 7 \\
&b[,] \quad \quad \quad [2,] 2 5 8 \\
&b[1:2,] \quad \quad \quad [3,] 3 6 9
\end{align*}
\]

\[
\begin{align*}
&> b[1,1] \\
&[1] 1 \\
&> b[2,3] \\
&[1] 8 \\
&> b[1,] \\
&[1] 1 4 7 \\
&> b[,] \\
&[1] 4 5 6 \\
&> b[1:2,] \\
&[1,] [2,] [3,] \\
&[1,] 1 4 7 \\
&[2,] 2 5 8
\end{align*}
\]
What are the values of the following expressions?

\begin{verbatim}
b[, 2:3] > b
b[, c(1, 2)] > [ , 1] [ , 2] [ , 3]
[1 ,] 1 4 7
[2 ,] 2 5 8
[3 ,] 3 6 9
\end{verbatim}
What are the values of the following expressions?

\begin{verbatim}
b[, 2:3] > b
b[, c(1,2)]
\begin{bmatrix}
[1,] & 1 & 4 & 7 \\
[2,] & 2 & 5 & 8 \\
[3,] & 3 & 6 & 9
\end{bmatrix}
\end{verbatim}

\begin{verbatim}
b[, 2:3]
\begin{bmatrix}
[1,] & 4 & 7 \\
[2,] & 5 & 8 \\
[3,] & 6 & 9
\end{bmatrix}
\end{verbatim}

\begin{verbatim}
b[, c(1,2)]
\begin{bmatrix}
[1,] & 1 & 4 \\
[2,] & 2 & 5 \\
[3,] & 3 & 6
\end{bmatrix}
\end{verbatim}

\begin{verbatim}
b[c(1,3), c(1,2)]
\begin{bmatrix}
[1,] & 1 & 4 \\
[2,] & 3 & 6
\end{bmatrix}
\end{verbatim}
Negative integers as indices removes the specified portions of a matrix. What are

\[
\begin{align*}
&\text{\(b[-1,]\)} & \text{\(> b\)} \\
&\text{\(b[c(-2,-3),]\)} & \text{\([,1] [,2] [,3]\)} \\
&\text{\(b[,]-2\)} & \text{\([1,] 1 4 7\)} \\
&\text{\(b[1:2,-3]\)} & \text{\([2,] 2 5 8\)} \\
& & \text{\([3,] 3 6 9\)}
\end{align*}
\]
Negative integers as indices removes the specified portions of a matrix. What are

\[
\begin{align*}
& b[-1,] > b \\
& b[c(-2,-3),] \\
& b[1:2,-3]
\end{align*}
\]

\[
\begin{align*}
& b \begin{bmatrix}
& [1,] & [2,] & [3,] \\
& 1 & 4 & 7 \\
& 2 & 5 & 8 \\
& 3 & 6 & 9 \\
\end{bmatrix} > b[-2,] \\
& b \begin{bmatrix}
& [1,] & [2,] & [3,] \\
& 1 & 4 & 7 \\
& 2 & 5 & 8 \\
& 3 & 6 & 9 \\
\end{bmatrix} > b[c(-2,-3),] \\
& b \begin{bmatrix}
& 1 & 4 & 7 \\
& 2 & 5 & 8 \\
& 3 & 6 & 9 \\
\end{bmatrix}
\end{align*}
\]
What is \( a \times b[,1] \)?

```r
> a
     [,1] [,2] [,3]
[1,]   2   2   2
[2,]   2   2   2
[3,]   2   2   2
> b
     [,1] [,2] [,3]
[1,]   1   4   7
[2,]   2   5   8
[3,]   3   6   9
```
What is \( a \times b[,1] \)?

\[
\begin{array}{ccc}
\hline
& [1,] & [2,] & [3,] \\
[1,] & 2 & 2 & 2 \\
[2,] & 2 & 2 & 2 \\
[3,] & 2 & 2 & 2 \\
\hline
\end{array}
\]

\[
\begin{array}{ccc}
\hline
& [1,] & [2,] & [3,] \\
[1,] & 1 & 4 & 7 \\
[2,] & 2 & 5 & 8 \\
[3,] & 3 & 6 & 9 \\
\hline
\end{array}
\]

\[
\begin{array}{ccc}
\hline
& [1,] \\
[1,] & 12 \\
[2,] & 12 \\
[3,] & 12 \\
\hline
\end{array}
\]
What is \( a[1,] \%*\% b \)?

\[
\begin{array}{ccc}
\mathbf{a}
\begin{bmatrix}
2 & 2 & 2 \\
2 & 2 & 2 \\
2 & 2 & 2 \\
\end{bmatrix}
\end{array}
\begin{array}{ccc}
\mathbf{b}
\begin{bmatrix}
1 & 4 & 7 \\
2 & 5 & 8 \\
3 & 6 & 9 \\
\end{bmatrix}
\end{array}
\]

\[
\text{a}[1,] \%*\% b
\begin{bmatrix}
12 & 30 & 48 \\
\end{bmatrix}
\]
What is \code{a[1,] %*% b}?

\begin{verbatim}
> a
   [,1] [,2] [,3]
[1,]  2  2  2
[2,]  2  2  2
[3,]  2  2  2

> b
   [,1] [,2] [,3]
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9

> a[1,] %*% b
   [,1] [,2] [,3]
[1,] 12  30  48
\end{verbatim}
This and the previous answer are a little tricky. In the previous question, \( b[,1] \) was treated as a column vector. In this answer, \( a[1,] \) was treated as a row vector. However, if you evaluate each one, you see

\[
> b[,1] \\
[1] 1 2 3 \\
> a[1,] \\
[1] 2 2 2
\]
This and the previous answer are a little tricky. In the previous question, `b[,1]` was treated as a column vector. In this answer, `a[1,]` was treated as a row vector. However, if you evaluate each one, you see

```r
> b[,1]
[1] 1 2 3
> a[1,]
[1] 2 2 2
```

They look like they are the same shape. The trickiness here is that R will change the shape of the arguments to the matrix multiplication operator to try to make the operation succeed.
Sometimes I don’t want this. What is the value of \( b[,1] \%*\% b \)?
• Sometimes I don’t want this. What is the value of \( b[,1] \times b \) ?

\[
\begin{array}{ccc}
\text{[,1]} & \text{[,2]} & \text{[,3]} \\
[1,] & 14 & 32 & 50
\end{array}
\]

\( b[,1] \) was treated as a row vector here, even though it looks like \( b[,1] \) is a column vector.
But I really want $b[,1]$ to be a column vector.
But I really want \( b[,1] \) to be a column vector.

This can be accomplished in two ways. You can tell R not to drop the single-valued dimension by using the \texttt{drop} argument.

\begin{verbatim}
> b[,1]
[1] 1 2 3
> b[,1, drop=FALSE]
     [,1]
[1,] 1
[2,] 2
[3,] 3
\end{verbatim}

Now

\begin{verbatim}
> b[,1, drop=FALSE] %*% b[1,]
     [,1]    [,2]    [,3]
[1,]  1   4    7
[2,]  2   8   14
[3,]  3  12  21
\end{verbatim}
The second way of doing this is to replace the usual \%*\% operator by \%o\%, the outer product operator.

\begin{verbatim}
> b[,1] %o% b[1,]
   [,1] [,2] [,3]
[1,]  1   4   7
[2,]  2   8  14
[3,]  3  12  21
\end{verbatim}
How do I calculate the inverse of a matrix?

Go ask http://www.rseek.org. Use keywords "inverse matrix". You will soon discover that the function `solve()` is used for matrix inversion.
How do I calculate the inverse of a matrix?

Go ask http://www.rseek.org. Use keywords “inverse matrix”. You will soon discover that the function `solve()` is used for matrix inversion.
Using \texttt{runif()} to produce random values uniformly distributed between 0 and 1, I should get matrix \( b \) back from the last expression in

\begin{verbatim}
a <- matrix(runif(9), 3, 3)
b <- matrix(runif(9), 3, 3)
c <- a %*% b
bnew <- solve(a) %*% c
\end{verbatim}
Using `runif()` to produce random values uniformly distributed between 0 and 1, I should get matrix b back from the last expression in

```r
a <- matrix(runif(9),3,3)
b <- matrix(runif(9),3,3)
c <- a %*% b
bnew <- solve(a) %*% c
```

```r
da <- matrix(runif(9),3,3)
b <- matrix(runif(9),3,3)
c <- a %*% b
bnew <- solve(a) %*% c
bnew
```

```r
> a <- matrix(runif(9),3,3)
> b <- matrix(runif(9),3,3)
> c <- a %*% b
> bnew <- solve(a) %*% c
> bnew
```

```
> b
```

```
[,1]       [,2]       [,3]
[1,] 0.8475927 0.1675809 0.27199269
[2,] 0.1293552 0.5486751 0.57194672
[3,] 0.4011202 0.5970193 0.07890513
```

```
> b
```

```
[,1]       [,2]       [,3]
[1,] 0.8475927 0.1675809 0.27199269
[2,] 0.1293552 0.5486751 0.57194672
[3,] 0.4011202 0.5970193 0.07890513
```
How can I verify that the above answer is the same as b?
How can I verify that the above answer is the same as b?

Many operators can be applied to matrices as well as scalars. Try the equality operator ==.
What is `b == bnew`?
What is \texttt{b == bnew}?

```
> b == bnew
     [,1] [,2] [,3]
[1,] FALSE FALSE FALSE
[2,] FALSE FALSE TRUE
[3,] FALSE FALSE FALSE
```

To see if all elements are \texttt{TRUE} use the function \texttt{all}.

```
> all(b == bnew)
[1] FALSE
```
What is \( b == bnew \)?

\[
> b == bnew \\
\begin{array}{ccc}
[ ,1] & [ ,2] & [ ,3] \\
[1 ,] & FALSE & FALSE & FALSE \\
[2 ,] & FALSE & FALSE & TRUE \\
[3 ,] & FALSE & FALSE & FALSE \\
\end{array}
\]

To see if all elements are TRUE use the function \texttt{all}.

\[
> \texttt{all}(b == bnew) \\
[1] \text{FALSE}
\]
b and bnew are not equal. I’m sure they are equal within some tolerance. How can I check this?

Use all.equal().

> all.equal(b, bnew)

[1] TRUE

Using ?all.equal you will see that the optional argument tolerance can be set and that by default it is the precision of the machine. On my machine, I see

> Machine$double.eps

[1] 2.220446e-16

94 / 189
b and \texttt{bnew} are not equal. I’m sure they are equal within some tolerance. How can I check this?

Use \texttt{all.equal()}.

\begin{verbatim}
> all.equal(b, bnew)
[1] TRUE
\end{verbatim}

Using \texttt{?all.equal} you will see that the optional argument \texttt{tolerance} can be set and that by default it is the precision of the machine. On my machine, I see

\begin{verbatim}
> .Machine$double.eps
[1] 2.220446e-16
\end{verbatim}
I want to do this tolerance test myself.
I want to do this tolerance test myself.

Good for you. That’s the spirit. You can use the `abs()` function applied to the difference matrix.

```r
> all(abs(b-bnew) < 0.001)
[1] TRUE
```
How do I do the inner product between vectors $[1,3]$ and $[2,4]$?
How do I do the inner product between vectors \([1,3]\) and \([2,4]\)?

One way is

\[
\begin{array}{c}
\text{> matrix(c(1,3),1,2) \%*\% matrix(c(2,4),2,1)} \\
\text{[ ,1]} \\
\text{[1 ,] 14}
\end{array}
\]
How do I do the inner product between vectors \([1,3]\) and \([2,4]\)?

One way is

```r
> matrix(c(1,3),1,2) %*% matrix(c(2,4),2,1)
     [,1]
[1,]   14
```

Or, you can rely on R doing the right thing and simply type

```r
> c(1,3) %*% c(2,4)
Error in c(1, 3) %*% c(2, 4) : non-conformable arguments
```

Whoops. Was I wrong?
How do I do the inner product between vectors [1,3] and [2,4]?

One way is

```r
> matrix(c(1,3),1,2) %*% matrix(c(2,4),2,1)
[,1]
[1,] 14
```

Or, you can rely on R doing the right thing and simply type

```r
> c(1,3) %*% c(2,4)
Error in c(1, 3) %*% c(2.4) : non-conformable arguments
```

Whoops. Was I wrong?


```r
> c(1,3) %*% c(2,4)
[,1]
[1,] 14
```
How do I do the inner product between vectors [1,3] and [2,4]?

One way is

```r
> matrix(c(1, 3), 1, 2) %*% matrix(c(2, 4), 2, 1)
[ , 1]
[1 ,] 14
```

Or, you can rely on R doing the right thing and simply type

```r
> c(1, 3) %*% c(2, 4)
Error in c(1, 3) %*% c(2, 4) : non-conformable arguments
```

Whoops. Was I wrong?


```r
> c(1, 3) %*% c(2, 4)
[ , 1]
[1 ,] 14
```

Which way is better? I prefer to carefully define all arguments as matrices of the correct shape. This allows R to catch any errors I have in my math logic.
Let

```r
> x <- matrix(c(1,3),1,2)
> y <- matrix(c(2,4),1,2)
> x
 [,1] [,2]
[1,]  1  3
> y
 [,1] [,2]
[1,]  2  4
```

be two row vectors. How do I perform an outer product to produce matrix

\[
\begin{bmatrix}
2 & 4 \\
6 & 12
\end{bmatrix}
\]
Let

\[
\begin{align*}
> x & \leftarrow \text{matrix}(c(1,3),1,2) \\
> y & \leftarrow \text{matrix}(c(2,4),1,2) \\
> x & \\
\begin{bmatrix}
[1,] & [2] \\
1 & 3
\end{bmatrix} \\
> y & \\
\begin{bmatrix}
[1,] & [2] \\
2 & 4
\end{bmatrix}
\end{align*}
\]

be two row vectors. How do I perform an outer product to produce matrix

\[
\begin{bmatrix}
2 & 4 \\
6 & 12
\end{bmatrix}
\]?

You can do this by

\[
\begin{align*}
> \text{t}(x) \%\%*\% y \\
\begin{bmatrix}
[1,] & [2] \\
[1,] & 2 & 4 \\
[2,] & 6 & 12
\end{bmatrix}
\end{align*}
\]
Can also do like

\> \texttt{drop(x \%o\% y)}

\[
\begin{array}{ccc}
[ ,1 ] & [ ,2 ] \\
[1 ,] & 2 & 4 \\
[2 ,] & 6 & 12 \\
\end{array}
\]

The outer product results in a four-dimensional structure, because you are taking an outer product of two two-dimensional matrices. Drop the single-valued first and third dimensions using the function \texttt{drop()}.
Can also do like

\[
\text{\texttt{drop(x \%o\% y)}} \\
\text{\texttt{[,] [,2]}} \\
\text{\texttt{[1 ,] 2 4}} \\
\text{\texttt{[2 ,] 6 12}}
\]

The outer product results in a four-dimensional structure, because you are taking an outer product of two two-dimensional matrices. Drop the single-valued first and third dimensions using the function \texttt{drop()}. This can also be accomplished by calling the function \texttt{outer()}

\[
\text{\texttt{drop(outer(x, y))}} \\
\text{\texttt{[,] [,2]}} \\
\text{\texttt{[1 ,] 2 4}} \\
\text{\texttt{[2 ,] 6 12}}
\]

The function \texttt{outer()} takes other arguments, including a function to apply to each pair of elements.
How would I use vector operations to calculate the root, mean, square error (RMSE) between vectors of actual and predicted values? Say the actual values are 1, 5, 10, and 20 and their predicted values are 1.1, 4.8, 10.5, and 18.6.

```r
> actual <- c(1, 5, 10, 20)
> predicted <- c(1.1, 4.8, 10.5, 18.6)

Hint: `mean()` and `sqrt()` exist.
```
How would I use vector operations to calculate the root, mean, square error (RMSE) between vectors of actual and predicted values? Say the actual values are 1, 5, 10, and 20 and their predicted values are 1.1, 4.8, 10.5, and 18.6.

> actual <- c(1, 5, 10, 20)
> predicted <- c(1.1, 4.8, 10.5, 18.6)

Hint: mean() and sqrt() exist.

Most R functions can be applied to vectors and matrices as well as scalars. Type an expression you know works for scalars, then see if it works for vectors or matrices. Oh, the wonders of R.

> sqrt(mean((actual - predicted)^2))
[1] 0.7516648
I want to do this often. How do I define a new function to do this?

```r
function (a, b) {
  sqrt(mean((a - b)^2))
}
```

But, to actually refer to it by name, you must assign the result of `function()` to a variable.

```r
calcRMSE <- function (a, b) {
  sqrt(mean((a - b)^2))
}
```

```r
calcRMSE (actual, predicted)
```

[1] 0.7516648
Writing Functions

I want to do this often. How do I define a new function to do this?

```r
> function (a, b) { sqrt( mean((a - b)^2) ) }
function (a, b) { sqrt( mean((a - b)^2) ) }
```

But, to actually refer to it by name, you must assign the result of `function()` to a variable.

```r
calcRMSE <- function (a, b) {
  sqrt( mean((a - b)^2) )
}
```

```r
calcRMSE (actual, predicted)
[1] 0.7516648
```
Writing Functions

- I want to do this often. How do I define a new function to do this?

  ```r
  function (a, b) {
    sqrt(mean((a - b)^2))
  }
  ```

- But, to actually refer to it by name, you must assign the result of `function()` to a variable.

  ```r
  > calcRMSE <- function (a,b) {sqrt(mean((a-b)^2))}
  > calcRMSE(actual, predicted)
  [1] 0.7516648
  ```
You can write functions, or any expression, on multiple lines.

```r
> calcRMSE <- function (a, b) {
+    sqrt(mean((a-b)^2))
+}
> calcRMSE(actual, predicted)
[1] 0.7516648
```

The first expression is continued on multiple lines, shown by the prompt changing to `+`. 
Say I define the vectors \( ax = 0.1, 0.2, \ldots, 0.5 \) and \( ay = -1, -0.8, \ldots, 1 \) as the values along the \( x \) and \( y \) axis of a graph for which you want to calculate the values of the function \( f(x, y) = \frac{(x-y)}{(x+y)} \). How do I build the matrix of values of \( f \)?
Say I define the vectors \( ax = 0.1, 0.2, \ldots, 0.5 \) and \( ay = -1, -0.8, \ldots, 1 \) as the values along the \( x \) and \( y \) axis of a graph for which you want to calculate the values of the function \( f(x, y) = \frac{(x-y)}{(x+y)} \). How do I build the matrix of values of \( f \)?

```r
> ax <- (1:5) * 0.1
> ay <- seq(-1,1,by=0.2)
> options(digits=3, width=40)
> f <- outer(ax, ay, function(x, y) { (x-y)/(x+y) })
> f
```

```
[1,] -1.22000 -1.28600 -1.4000 -1.670000
[2,] -1.50000 -1.66800 -2.0000 -3.000000
[3,] -1.86000 -2.20000 -3.0000 -7.000000
[4,] -2.33000 -3.00000 -5.0000  7.210000
[5,] -3.00000 -4.33000 -11.0000  9.000000
[1,] -3.000000 1 -3.330000 -6.000000
... more ... (abbreviated for slide)
```

This shows that anonymous functions can be defined and passed as arguments. The function `options` may be used to control how values are printed. See `?options`. 

---

**R**
- Why?
- Installing and Running R
- On-line Documentation
- Matrices
- Variables
- Matrix Operations
- Defining New Functions
- Lists
- Graphics
- Using R Source Files
- Reading in Data

**LaTeX**
- Why LaTeX?
- Simple Example
- Math
- Sections
- Makefiles
- Graphics
- Running R
- Non-interactively
In C, a data structure containing items of different types is built as a `struct`. In C++ and Java, it is a `class`. In R, the functions `list` and `c` are used to build and concatenate lists. How do I make a list of the names of the first three months of the year and their number of days and assign it to `months`?

```r
> months <- list("January", 31, "February", 28, "March", 31)
```
Lists

- In C, a data structure containing items of different types is built as a `struct`. In C++ and Java, it is a `class`. In R, the functions `list` and `c` are used to build and concatenate lists. How do I make a list of the names of the first three months of the year and their number of days and assign it to `months`?

```r
> months <- list("January",31,"February",28,
+ "March",31)
```
How do I access "January", the pair "January" and 31, or all three month names?
How do I access "January", the pair "January" and 31, or all three month names?

```r
> months[1]
[[1]]
[1] "January"

> months[1:2]
[[1]]
[1] "January"
[[2]]
[1] 31

> months[c(1,3,5)]
[[1]]
[1] "January"
[[2]]
[1] "February"
[[3]]
[1] "March"
```
Now what if I want to change the number of days in February to 29? A list access like `months[1]` actually returns a list of one item. To return the actual element, use `months[[1]]`. Therefore, to change one element, use

```r
> months[[4]] <- 29
> months
[[1]]
[1] "January"
[[2]]
[1] 31
[[3]]
[1] "February"
[[4]]
[1] 29
[[5]]
[1] "March"
[[6]]
[1] 31
```
How do I add the "April" and 30 to my list?
How do I add the "April" and 30 to my list?

Either

```r
> c(months,"April", 30)
```

or

```r
> c(months, list("April", 30))
```
A common use of lists is for returning multiple values from a function. Suppose I have a vector of targets which are \((1, 0, 0, 1, 0)\) and predictions of \((0.6, 0.2, 0.5, 0.8, 0.2)\). How do I calculate the root mean square error value and assemble all three things in a list called \texttt{results}?

\begin{verbatim}
A common use of lists is for returning multiple values from a function. Suppose I have a vector of targets which are \((1, 0, 0, 1, 0)\) and predictions of \((0.6, 0.2, 0.5, 0.8, 0.2)\). How do I calculate the root mean square error value and assemble all three things in a list called \texttt{results}?

\end{verbatim}
A common use of lists is for returning multiple values from a function. Suppose I have a vector of targets which are \((1, 0, 0, 1, 0)\) and predictions of \((0.6, 0.2, 0.5, 0.8, 0.2)\). How do I calculate the root mean square error value and assemble all three things in a list called `results`?

```r
> targets <- c(1,0,0,1,0)
> predictions <- c(0.6,0.2,0.5,0.8,0.2)
> rmse <- sqrt(mean((targets - predictions)^2))
> rmse
[1] 0.3255764
> results <- list(targets, predictions, rmse)
> results
[[1]]
[1] 1 0 0 1 0
[[2]]
[1] 0.6 0.2 0.5 0.8 0.2
[[3]]
[1] 0.3255764
```
I don’t want to remember that RMSE is at index 3. Some or all elements of a list can be given names using name = value syntax. How do I do this for the variable results? The R function names results in a vector of strings showing the names of the elements.
I don’t want to remember that RMSE is at index 3. Some or all elements of a list can be given names using `name = value` syntax. How do I do this for the variable `results`? The R function `names` results in a vector of strings showing the names of the elements.

```r
> results <- list("correct"=targets,
+                  "predictions"=predictions,"RMSE"=rmse)
> results[[3]]
[1] 0.3255764
> results[["RMSE"]]
[1] 0.3255764
> names(results)
[1] "correct" "predictions" "RMSE"
```
A list can be converted to a vector with `unlist`. I’ll try this with `results`, but what will the type of the elements be? And if I do this for `months`, what will those elements be?
A list can be converted to a vector with `unlist`. I’ll try this with `results`, but what will the type of the elements be? And if I do this for `months`, what will those elements be?

```r
> unlist(results)
correct1 correct2 correct3
   1.000  0.000  0.000
correct4 correct5 predictions1
   1.000  0.000  0.600
predictions2 predictions3 predictions4
   0.200  0.500  0.800
predictions5 RMSE
   0.200  0.326
```

```r
> unlist(months)
[1] "January" "31"     "February"
[4] "28"      "March"  "31"
```
The type of the elements of `unlist(results)` is numeric, while the elements of `unlist(months)` are of type character. The `mode` function will tell you this.

```r
> mode(unlist(results))
[1] "numeric"
> mode(unlist(months))
[1] "character"
```
For practice, I want to define a function `sortedMonths` that takes a list of months and days as an argument, sorts the months by the number of days, and returns a list containing a vector of sorted month names and a vector of sorted days.

```r
+ "Nov",30,"Dec",31)
> sortedMonths <- function(months) {
+  
Now what?
```
Here is a complete function.

```r
> sortedMonths <- function (months) {
+   monthNames <- unlist(months[seq(1,24,by=2)])
+   days <- unlist(months[seq(2,24,by=2)])
+   indexOrder <- order(days)
+   list(names=monthNames[indexOrder], days=days[indexOrder])
+ }
> m <- sortedMonths(months)
> names(m)
[1] "names" "days"
```
Here is a complete function.

```r
> sortedMonths <- function (months) {
+   monthNames <- unlist(months[seq(1,24,by=2)])
+   days <- unlist(months[seq(2,24,by=2)])
+   indexOrder <- order(days)
+   list(names=monthNames[indexOrder], days=days[indexOrder])
+ }
> m <- sortedMonths(months)
> names(m)
[1] "names"  "days"
```

What do the expressions `m[['names']]` and `m[['days']]` return?
Here is a complete function.

```r
> sortedMonths <- function (months) {
+   monthNames <- unlist(months[seq(1,24,by=2)])
+   days <- unlist(months[seq(2,24,by=2)])
+   indexOrder <- order(days)
+   list(names=monthNames[indexOrder], days=days[indexOrder])
+ }
> m <- sortedMonths(months)
> names(m)
[1] "names" "days"

What do the expressions m[['names']] and m[['days']] return?

```r
> m[['names']]
[1] "Feb" "Apr" "Jun" "Sep" "Nov" "Jan"
[7] "Mar" "May" "Jul" "Aug" "Oct" "Dec"
> m[['days']]
R has a fairly complete graphics capability that can produce gorgeous, publication quality, visualizations of data. Do

```r
> demo(graphics)
```

in R to see sample displays of many of R’s graphics functions.
Guess what...R has a function named `plot`. Take a look at its documentation by doing

> `plot`

A fun thing to do is to scroll down through a function’s documentation to the examples near the end and run an example by using your mouse to select part of the example text and paste it into your R prompt. Then use `control-p`, `control-n`, `control-f`, `control-b` and `backspace` to repeat a previous command with modifications.
Let’s start with something simple. How do I generate data from a parabola and graph it?

\( x = \text{seq}(-10,10,\text{by}=1) \)

\[ y = x^2 \]

\[
\begin{array}{cccccccccc}
-10 & -9 & -8 & -7 & -6 & -5 & -4 & -3 & -2 & 0 \\
100 & 81 & 64 & 49 & 36 & 25 & 16 & 9 & 4 & 1 \\
\end{array}
\]

\( \text{plot}(x,y) \)
Let’s start with something simple. How do I generate data from a parabola and graph it?

```r
> x <- seq(-10,10,by=1)
> x
[ 1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0
     1   2   3   4   5   6   7   8
> y <- x^2
> y
[ 1] 100  81  64  49  36  25  16   9   4   1   0
     1   4   9  16  25  36  49  64
> plot(x, y)
```

![Graph of Parabola](image)
I only see points, but I want the curve. Try

```r
> plot(x, y, type="lines")
```

or to see lines and points, do

```r
> plot(x, y, type="both")
```

Arguments and values can be abbreviated as long as they are not ambiguous with other arguments and values:

```r
> plot(x, y, t="b")
```

but you would never do this in an R source file (see next section) because it is unreadable.
I’m sure I can specify an x and y axis label and a title. How do I do that? Read through results of `?plot` again.
I’m sure I can specify an x and y axis label and a title. How do I do that? Read through results of `?plot` again.

Yep, you found it. Here is an example:

```r
> plot(x, y, type="b", 
     xlab="Cleverly Generated Sequence", 
     ylab="Cleverly Squared", main="Top Title")
```

**Top Title**

```
0 20 40 60 80 100
-10 -5 0 5 10
```

```
Cleverly Squared
```

```
Cleverly Generated Sequence
```

```
Top Title
```

```
rnorm can be used to produce a number of values drawn from a Normal distribution. How can I use this to produce data from a noisy quadratic function? Let’s use a Normal distribution with a variance of 10, using variable y from before.
\textbf{rnorm} can be used to produce a number of values drawn from a Normal distribution. How can I use this to produce data from a noisy quadratic function? Let’s use a Normal distribution with a variance of 10, using variable \texttt{y} from before.

\begin{verbatim}
> ynoisy <- y + rnorm(length(y), sd=10)
\end{verbatim}
Now I want to plot both the quadratic values and the noisy quadratic values on same plot, but a call to plot always erases the old plot and creates a new one. The functions `lines` and `points` will add lines and points to an existing plot, so let's use those. First, read `?lines` and `?points`.

```r
plot(x, y, type="b")
points(x, y_noisy, col="red", pch=4)
lines(x, y_noisy, col="green")
```
Now I want to plot both the quadratic values and the noisy quadratic values on same plot, but a call to plot always erases the old plot and creates a new one. The functions lines and points will add lines and points to an existing plot, so let's use those. First, read ?lines and ?points.

```r
> plot(x, y, type="b")
> points(x, ynoisy, col="red", pch=4)
> lines(x, ynoisy, col="green")
```
I can imagine cases where I have constructed a matrix where each column contains samples of a particular variable. For example, column 1 might be temperatures and column 2 might be pressures. How might I plot the columns on the same plot without calling `plot` then `lines` or `points`? Try `matplot`.

```r
temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
presures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
data <- cbind(temps, pressures)
data temps pressures
[1,] 60 30.1
[2,] 65 30.5
... 
[9,] 88 28.5
[10,] 91 29.1
matplot(data, type="b", pch=c("T", "P"), bty="n", xlab="Day", ylab="Measurements")
```

Notice the automatically named columns!
I can imagine cases where I have constructed a matrix where each column contains samples of a particular variable. For example, column 1 might be temperatures and column 2 might be pressures. How might I plot the columns on the same plot without calling plot then lines or points? Try `matplot`.

```r
> temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
> pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, + 30.8, 28.9, 28.5, 29.1)
> data <- cbind(temps, pressures)
> data
   temps pressures
   [1,] 60     30.1
   [2,] 65     30.5
   [9,] 88     28.5
   [10,] 91     29.1
> matplot(data, type="b", pch=c("T","P"), bty="n", + xlab="Day", ylab="Measurements")
```

Notice the automatically named columns!
Notice less “chart junk” (Tufte).
R Source Files

So far I have been typing directly into the R command line. I’d really like to save a sequence of commands in an R source file to be run anytime I choose. Given that the file is a “source” file, maybe I can figure out the name of the function to use to run the file.
So far I have been typing directly into the R command line. I’d really like to save a sequence of commands in an R source file to be run anytime I choose. Given that the file is a “source” file, maybe I can figure out the name of the function to use to run the file.

Did you look up `?source`
What will the source file look like that produces the temperature and pressure plot? How do I run it?

```r
# Declare and store values for temperatures and pressures
temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
presures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
data <- cbind(temps, pressures)

# Plot data using matplot
matplot(data, type="b", pch=c("T", "P"), xlab="Day", ylab="Measurements")
```

Notice the contents of the R source file look just like the lines as they would be typed directly at the R prompt. Run this with:

```r
source("plotTempPress.R")
```
What will the source file look like that produces the temperature and pressure plot? How do I run it?

Call the source file `plotTempPress.R`. The `system` function can be used to run any Unix command, like more to show the contents of the file.

```r
> system("more plotTempPress.R")
temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
data <- cbind(temps, pressures)
matplot(data, type="b", pch=c("T","P"),
        xlab="Day", ylab="Measurements")
```

Notice the contents of the R source file look just like the lines as they would be typed directly at the R prompt.
What will the source file look like that produces the temperature and pressure plot? How do I run it?

Call the source file plotTempPress.R. The system function can be used to run any Unix command, like more to show the contents of the file.

```r
> system("more plotTempPress.R")
```

```r
temps <- c(60, 65, 72, 66, 78, 84, 83, 88, 91)
pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
data <- cbind(temps, pressures)
matplot(data, type="b", pch=c("T","P"),
        xlab="Day", ylab="Measurements")
```

Notice the contents of the R source file look just like the lines as they would be typed directly at the R prompt.

Run this with

```r
> source("plotTempPress.R")
```
Usually an R source file first defines a set of functions to be used. The above commands can be encapsulated into a function named `plotTempPress` in a file. How?
Usually an R source file first defines a set of functions to be used. The above commands can be encapsulated into a function named `plotTempPress` in a file. How?

```r
> system("more plotTempPress.R")
plotTempPress <- function (){
  temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
  pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
  data <- cbind(temps, pressures)
  matplot(data, type="b", pch=c("T", "P"),
          xlab="Day", ylab="Measurements")
}
```

Now,

```r
> source("plotTempPress.R")
```

does not produce a plot, it just defines the function that can now be called. How?
Usually an R source file first defines a set of functions to be used. The above commands can be encapsulated into a function named `plotTempPress` in a file. How?

```r
> system("more plotTempPress.R")
plotTempPress <- function () {
  temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
  pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
  data <- cbind(temps, pressures)
  matplot(data, type="b", pch=c("T", "P"),
           xlab="Day", ylab="Measurements")
}
```

Now,

```r
> source("plotTempPress.R")
```

does not produce a plot, it just defines the function that can now be called. How?

```r
> plotTempPress()
```
If you want the `source` function to also run the function, just add the line that evaluates the function to the end of your file.
If you want the `source` function to also run the function, just add the line that evaluates the function to the end of your file.

```r
> system("more plotTempPress.R")
plotTempPress <- function () {
  temps <- c(60, 65, 72, 66, 78, 74, 84, 83, 88, 91)
  pressures <- c(30.1, 30.5, 29.3, 29.5, 32.3, 31.2, 30.8, 28.9, 28.5, 29.1)
  data <- cbind(temps, pressures)
  matplot(data, type="b", pch=c("T","P"),
           xlab="Day", ylab="Measurements")
}

plotTempPress()

Now,

```r
> source("plotTempPress.R")
```

does produce the plot.
Reading Data

Say the temperature and pressure data already exists in a file, named `temppress.data`, that contains

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>30.1</td>
</tr>
<tr>
<td>65</td>
<td>30.5</td>
</tr>
<tr>
<td>72</td>
<td>29.3</td>
</tr>
<tr>
<td>66</td>
<td>29.5</td>
</tr>
<tr>
<td>78</td>
<td>32.3</td>
</tr>
<tr>
<td>74</td>
<td>31.2</td>
</tr>
<tr>
<td>84</td>
<td>30.8</td>
</tr>
<tr>
<td>83</td>
<td>28.9</td>
</tr>
<tr>
<td>88</td>
<td>28.5</td>
</tr>
<tr>
<td>91</td>
<td>29.1</td>
</tr>
</tbody>
</table>

The function `read.table` can be used to read this data into R. How?
Reading Data

- Say the temperature and pressure data already exists in a file, named `temppress.data`, that contains

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>30.1</td>
</tr>
<tr>
<td>2</td>
<td>65</td>
<td>30.5</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
<td>29.3</td>
</tr>
<tr>
<td>4</td>
<td>66</td>
<td>29.5</td>
</tr>
<tr>
<td>5</td>
<td>78</td>
<td>32.3</td>
</tr>
<tr>
<td>6</td>
<td>74</td>
<td>31.2</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
<td>30.8</td>
</tr>
<tr>
<td>8</td>
<td>83</td>
<td>28.9</td>
</tr>
<tr>
<td>9</td>
<td>88</td>
<td>28.5</td>
</tr>
<tr>
<td>10</td>
<td>91</td>
<td>29.1</td>
</tr>
</tbody>
</table>

The function `read.table` can be used to read this data into R. How?

```
> data <- read.table("temppress.data")
> data
     V1  V2
 1  60 30.1
 2  65 30.5
   ...  
 9  88 28.5
10 91 29.1
```
I like keeping those column labels with the data. To do so, I hope R can read in a data file that looks like

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>30.1</td>
</tr>
<tr>
<td>65</td>
<td>30.5</td>
</tr>
<tr>
<td>72</td>
<td>29.3</td>
</tr>
<tr>
<td>66</td>
<td>29.5</td>
</tr>
<tr>
<td>78</td>
<td>32.3</td>
</tr>
<tr>
<td>74</td>
<td>31.2</td>
</tr>
<tr>
<td>84</td>
<td>30.8</td>
</tr>
<tr>
<td>83</td>
<td>28.9</td>
</tr>
<tr>
<td>88</td>
<td>28.5</td>
</tr>
<tr>
<td>91</td>
<td>29.1</td>
</tr>
</tbody>
</table>

read.table can also be used for this file. The documentation shows how.
I like keeping those column labels with the data. To do so, I hope R can read in a data file that looks like

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>30.1</td>
</tr>
<tr>
<td>65</td>
<td>30.5</td>
</tr>
<tr>
<td>72</td>
<td>29.3</td>
</tr>
<tr>
<td>66</td>
<td>29.5</td>
</tr>
<tr>
<td>78</td>
<td>32.3</td>
</tr>
<tr>
<td>74</td>
<td>31.2</td>
</tr>
<tr>
<td>84</td>
<td>30.8</td>
</tr>
<tr>
<td>83</td>
<td>28.9</td>
</tr>
<tr>
<td>88</td>
<td>28.5</td>
</tr>
<tr>
<td>91</td>
<td>29.1</td>
</tr>
</tbody>
</table>

read.table can also be used for this file. The documentation shows how.

```r
> data <- read.table("tempPress.data", head=TRUE)
> data
     Temperature Pressure
1         60       30.1
2         65       30.5
...  
9         88       28.5
10        91       29.1
```
This looks like a matrix, but it is actually a data.frame, an R data structure that can contain columns of different types. Because it is a data.frame, it cannot be treated as a matrix:

```r
> t(data) %%*% data
Error in t(data) %%*% data :
  requires numeric matrix/vector arguments
```
This looks like a matrix, but it is actually a data.frame, an R data structure that can contain columns of different types. Because it is a data.frame, it cannot be treated as a matrix:

```r
> t(data) %*% data
Error in t(data) %*% data :
  requires numeric matrix/vector arguments
```

R includes functions for type conversion. I can use `as.matrix` to convert the data.frame to a matrix.

```r
> data <- as.matrix(read.table("temppress.data", head=TRUE))
> t(data) %*% data
```

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>58895.0</td>
<td>22815.30</td>
</tr>
<tr>
<td>22815.3</td>
<td>9024.64</td>
</tr>
</tbody>
</table>
Outline

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics

CS545: Machine Learning
Introduction to R and LaTeX

Chuck Anderson
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.a

- Advantages:
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- Advantages:
  - It’s free.
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- Advantages:
  - It’s free.
  - It’s available for many machines.
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.a

- Advantages:
  - It’s free.
  - It’s available for many machines.
  - LaTeX files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
Why \LaTeX{}?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- **Advantages:**
  - It’s free.
  - It’s available for many machines.
  - \LaTeX{} files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
  - You can use the editor of your choice (even Word).
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- Advantages:
  - It’s free.
  - It’s available for many machines.
  - LaTeX files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
  - You can use the editor of your choice (even Word).
  - The typesetting’s better, especially the maths.
Why \LaTeX? 

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- Advantages:
  - It's free.
  - It’s available for many machines.
  - \LaTeX\ files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
  - You can use the editor of your choice (even Word).
  - The typesetting’s better, especially the maths.
  - Style changes are neater in \LaTeX.
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.aaa

- Advantages:
  - It’s free.
  - It’s available for many machines.
  - LaTeX files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
  - You can use the editor of your choice (even Word).
  - The typesetting’s better, especially the maths.
  - Style changes are neater in LaTeX.
  - LaTeX is extensible.
Why LaTeX?

The following lists are from http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

- Advantages:
  - It’s free.
  - It’s available for many machines.
  - LaTeX files are ASCII, so are portable and easy to read to discover how someone achieved an effect.
  - You can use the editor of your choice (even Word).
  - The typesetting’s better, especially the maths.
  - Style changes are neater in LaTeX.
  - LaTeX is extensible.
  - LaTeX’s floats are no more awkward than Word’s anchored frames, and the latter don’t always work well.
Disadvantages:

- Font selection is difficult compared to selection in Word and others.
Disadvantages:

- Font selection is difficult compared to selection in Word and others.
- LaTeX’s not good at flowing text around pictures.
Disadvantages:

- Font selection is difficult compared to selection in Word and others.
- LaTeX’s not good at flowing text around pictures.
- LaTeX encourages (almost insists on) structured writing and the separation of style from content. This is not the way that many people (especially non-programmers) are used to working.
Disadvantages:

- Font selection is difficult compared to selection in Word and others.
- LaTeX's not good at flowing text around pictures.
- LaTeX encourages (almost insists on) structured writing and the separation of style from content. This is not the way that many people (especially non-programmers) are used to working.
- Without a WYSIWYG front end, it's not always easy to find out how to do things. But, WYSIWYG's, such as LyX, are becoming better.
Your First \LaTeX\ Document

Enter this \LaTeX\ code into a file named `simple.tex`

```
\documentclass{article}
\begin{document}
Hello.
\end{document}
```

and on our \texttt{linux} systems run the commands

```
latex simple
dvips -t letter -Ppdf -o simple.ps simple.dvi
gv simple.ps &
```

or

```
latex simple
dvips -t letter -Ppdf -o simple.ps simple.dvi
ps2pdf simple.ps
acroread simple.pdf
```
Introduction to R and LaTeX

R

Why?
Installing and Running R
On-line Documentation
Matrices
Variables
Matrix Operations
Defining New Functions
Lists
Graphics
Using R Source Files
Reading in Data

LaTeX

Why LaTeX?
Simple Example
Math
Sections
Makefiles
Graphics
Running R
Non-interactively
You’ve heard LaTeX is excellent at formatting math. Type these lines right after the `Hello.` line

```
I am so smart, I do math like
\begin{displaymath}
RMSE = \sqrt{ \frac{1}{n} \sum_{i=1}^{n} \left( t_i - y_i \right)^2 }
\end{displaymath}
```

and then produce the formatted document again before. What do you see?
Hello. I am so smart, I do math like

\[ \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (t_i - y_i)^2} \]
Sections

If \section{Section Name} and \subsection{SubSection Name} create section and subsection headings, how would you make

1 Introduction
Good stuff here, introducing more good stuff.

1.1 Details
Further good stuff, in detail.

1.2 Limitations
Good stuff that I didn’t cover.
We added some lines after `\documentclass{article}` to use the `geometry` package to specify margins and that we want to use letter-sized paper (rather than A4 that the rest of the world uses).
Using a Makefile

Getting tired of typing all of those commands to format the document? How would you write a makefile that would handle make commands like these?

```
make simple.pdf
make simple.ps
make simple.dvi
```
Using a Makefile

Getting tired of typing all of those commands to format the document? How would you write a makefile that would handle make commands like these?

```
make simple.pdf
make simple.ps
make simple.dvi
```

Put these lines into the file named Makefile

```
simple.pdf:  simple.ps
    ps2pdf simple.ps

simple.ps:  simple.dvi
    dvips -t letter -Ppdf -o simple.ps simple.dvi

simple.dvi: simple.tex
    latex simple
    latex simple
```

Be careful. The indented lines all start with a single character, a tab. `make` requires this. `latex` is run twice to correctly produce figure references, the table of contents, and other things.
Graphics

To include graphic figures, we must tell LaTeX to also use the graphicx package at the top of the file.

```latex
\documentclass{article}
\usepackage{geometry}
\geometry{letterpaper, tmargin=1in, bmargin=1in, lmargin=1in, rmargin=1in}
\usepackage{graphicx}  %%% NEW LINE
\begin{document}

\section{Introduction}
Good stuff here, introducing more good stuff.
\subsection{Details}
Further good stuff, in detail.
\subsection{Limitations}
Good stuff that I didn’t cover.

\end{document}
```

If you just add that one new line and \texttt{latex} the document again, you will see no change, because we haven’t included any graphic objects yet.
Now let’s try including a figure into the document. The graphicx packages provides a new latex command \includegraphics. Blank lines start new paragraphs.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{geometry}
\geometry{letterpaper, tmargin=1in, bmargin=1in, lmargin=1in, rmargin=1in}
\begin{document}
Hi everyone. I'm learning good stuff here. I can make documents with figures from R inserted in them. Someday I am sure I will appreciate the true power of R and LaTeX.

My beautiful results are shown in Figure~\ref{fig:sinethingy}.
\begin{figure}
\begin{center}
\includegraphics[width=5in]{sine.ps}
\caption{A sine curve, produced in R using \texttt{plot(sin(1:10), type="b"); dev.copy2eps(file="sine.ps")}}
\end{center}
\label{fig:sinethingy}
\end{figure}
\end{document}
```
Hi everyone. I'm learning good stuff here. I can make documents with figures from R inserted in them. Someday I am sure I will appreciate the true power of R and LaTeX.

My beautiful results are shown in Figure 1.

Figure 1: A sine curve, produced in R using `plot(sin(1:10), type="b")`; `dev.copy2pdf(file="sine.pdf")`
Watch this! I can use the same file Makefile to produce my figure files, if I have already written my R code in an R source file named simplecode.R. R can be applied non-interactively to an R source file as explained in the manual page shown by doing the unix command `man R`

```
simple.pdf: simple.ps
    ps2pdf simple.ps

simple.ps: simple.dvi sine.ps
    dvips -t letter -Ppdf -o simple.ps simple.dvi

simple.dvi: simple.tex sine.ps
    latex simple
    latex simple

sine.ps: simplecode.R
    R --quiet --no-save < simplecode.R > output
```
To Run R as a Batch Job

One change must be made to the R file. When run noninteractively, we must tell R how to format the graphics before generating them. Here is the changed contents of simplecode.R:

```r
postscript(file="sine.ps")
plot(sin(1:10), type="b")
```

Now when you do the unix command `make simple.pdf` you will see these commands run:

```
R --quiet --no-save < simplecode.R > simplecode.output
latex simple
latex simple
ps2pdf simple.ps
```

with a lot of text in between.