What is R?

"R is a computer language that allows the user to program algorithms and use tools that have been programmed by others."

Zuur et. al. (2009)
What can you do with R?

You can ...
- do calculations
- perform statistical analysis (using available code)
- create powerful graphics
- write your own functions

What is the catch?

R has a steep learning curve:
- It requires programming...
  ... but
- the programming used in R is very similar across methods
- a lot has already been done in terms of statistical tools

Installing R on a Mac

1. Go to http://cran.r-project.org/ and select MacOS X
2. Select to download the latest version: 2.9.2 (2009-08-24)
3. Install and Open. The R window should look like this:
Comparisons in R

#: used for commenting words out
&: in logic, is used to mean AND
|: in logic, is used to mean OR
==: in logic, is used for comparison (double equal sign)
>: in logic, is used to mean GREATER THAN
<: in logic, is used to mean LESS THAN
-> OR ==: for variable assignment
?: for use with a function in R, to determine what arguments to use, examples and background information
  Example: ?mean

Creating Variables

- To use R as a calculator, type 2 + 5 and hit ENTER. (Note how R prints the result.) Your output should look like this:

[1] 7

- To create variables in R, use either − > or ==:

1  # Approach 1  2  a=5  
3  a
4  # Approach 2  5  a=5; a
6  # Approach 3  7  b<5; b

Creating Vectors

- Scalars are the most basic vectors.
- To create vectors of length greater than one, use the concatenation function c() :

1  d=c(3,4,7); d

[1] 3 4 7

- To create a null vector:

1  x=c(); x

NULL
Creating Vectors II

- Creating a vector with equal spacing, use the sequence function `seq()`:
  
  ```r
  e <- seq(from = 1, to = 3, by = 0.5); e
  ```

  ```r
  [1] 1.0 1.5 2.0 2.5 3.0
  ```

- Creating a vector of a given length, use the repeat function `rep()`:

  ```r
  f <- rep(NA, 6); f
  ```

  ```r
  [1] NA NA NA NA NA NA
  ```

Some Useful Vector Functions I

- To find the length of the vector, use `length()`:
  ```r
  length(d)
  ```

  ```r
  [1] 3
  ```

- To find the maximum value of the vector, use the maximum function `max()`:
  ```r
  max(d)
  ```

  ```r
  [1] 7
  ```

Some Useful Vector Functions II

- To find the minimum value of the vector, use the minimum function `min()`:
  ```r
  min(d)
  ```

  ```r
  [1] 3
  ```

- To find the mean of the vector, use `mean()`:
  ```r
  mean(d)
  ```

  ```r
  [1] 4.666667
  ```

Some Useful Vector Functions III

- To sort the vector, use `sort()`:
  ```r
  g <- c(2, 6, 7, 4, 5, 2, 9, 3, 6, 4, 3)
  sort(g, decreasing = TRUE)
  ```

  ```r
  [1] 9 7 6 6 5 4 4 3 3 2 2
  ```

- To find the unique elements of the vector, use `unique()`:
  ```r
  unique(g)
  ```

  ```r
  [1] 2 6 7 4 5 9 3
  ```
Some Useful Vector Functions IV

- Alternatively, to find the elements of the vector that repeat, use `duplicated(g)`

```r
duplicated(g)
```

[[1]] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE

- To sample from a vector, use `sample()`:

```r
c <- sample(g, size = 3, replace = TRUE); c
```

[1] 6 3 3

To compare elements of one vector with another, use set operations:

- `union(e, 1:5)`

```r
union(e, 1:5)
```

[1] 1.0 1.5 2.0 2.5 3.0 4.0 5.0

- `intersect(e, 1:5)`

```r
intersect(e, 1:5)
```

[1] 1 2 3

- `setdiff(e, 1:5)`

```r
setdiff(e, 1:5)
```

[1] 1.5 2.5

Sub-setting with Vectors I

- To find out what is stored in a given element of the vector, use `[ ]`:

```r
d[2]
```

[1] 4

- To see if any of the elements of a vector equal a certain number, use `==`:

```r
d == 3
```

[1] TRUE FALSE FALSE

Sub-setting with Vectors II

- To see if any of the elements of a vector do not equal a certain number, use `!=`:

```r
(d != 3) & (d < max(d))
```

[1] FALSE TRUE FALSE

- To obtain the element number of the vector when a condition is satisfied, use `which()`:

```r
which(d == 4)
```

[1] 2

To store the result, type: `a=which(d==4); a`
Creating Matrices I

- To create a matrix, use the `matrix()` function:

```
1 mat <- matrix(10:15, nrow=3, ncol=2, byrow=F); mat
```

```
[,1] [,2]
[1,] 10 13
[2,] 11 14
[3,] 12 15
```

Some Useful Matrix Functions I

- To add two matrices, use `+`:

```
1 mat + mat
```

```
[,1] [,2]
[1,] 20 26
[2,] 22 28
[3,] 24 30
```

Some Useful Matrix Functions II

- To find the transpose of a matrix, use `t()`:

```
1 t(mat)
```

```
[,1] [,2] [,3]
[1,] 10 11 12
[2,] 13 14 15
```

Some Useful Matrix Functions III

- To multiply two matrices, use `%*%`.

```
1 mat %*% t(mat)
```

```
[,1] [,2] [,3]
[1,] 269 292 315
[2,] 292 317 342
[3,] 315 342 369
```
Some Useful Matrix Functions IV

- To find the dimensions of a matrix, use `dim()`:
  ```r
  dim(mat)
  ```
  [1] 3 2

- Alternatively, we can find the rows and columns of the matrix, by `nrow()` and `ncol()`:
  ```r
  nrow(mat); ncol(mat)
  ```
  [1] 3
  [1] 2

Subsetting with Matrices I

- To see what is stored in the first element of the matrix, use `[ ]`:
  ```r
  mat[1,1]
  ```
  [1] 10

- To see what is stored in the first row of the matrix:
  ```r
  mat[1,]
  ```
  [1] 10 13

Subsetting with Matrices II

- To see what is stored in the second column of the matrix:
  ```r
  mat[,2]
  ```
  [1] 13 14 15

- To extract elements 1 and 3 from the second column, use `c()` and `[ ]`:
  ```r
  mat[c(1,3), 2]
  ```
  [1] 13 15

Subsetting with Matrices III

- To extract *everything but* elements 1 and 3 from the second column, use `-c()` and `[ ]`:
  ```r
  mat[-c(1,3), 2]
  ```
  [1] 14

- To extract the observation containing the maximum value, use `which.max()` and `[ ]`:
  ```r
  mat[which.max(mat)]
  ```
  [1] 15
Subsetting with Matrices IV

● To extract observations matching a certain criteria, use `which()` and `[ ]`:

**Example 1:** List observations of `mat` that are greater than 12.

```r
mat [which (mat > 12)]
```

```
[1] 13 14 15
```

**Example 2:** Sum all the even rows of column 2 of the 6*6 matrix that contains squares of the first 36 numbers.

```r
mat.res <- matrix((1:36)^2, ncol=6); mat.res
ans <- sum(mat.res [seq(from=2, to=6, by=2)], 2); ans
```

Matrix:

```
[1,]  1  49 169 361 625 961
[2,]  4  64 196 400 676 1024
[3,]  9  81 225 441 729 1089
[4,] 16 100 256 484 784 1156
[5,] 25 121 289 529 841 1225
[6,] 36 144 324 576 900 1296
```

Answer:

```
[1] 308
```

Creating Matrices from Vectors I

● To stack two vectors, one below the other, use `rbind()`:

```r
mat1 <- rbind(d, d); mat1

[,1] [,2] [,3]
  d  3  4  7
  d  3  4  7
```

Creating Matrices from Vectors II

● To stack two vectors, one next to the other, use `cbind()`:

```r
mat2 <- cbind(d, d); mat2

[,1] [,2]
  d  d
[1,] 3 3
[2,] 4 4
[3,] 7 7
```
Missing Data in Vectors

- Start by creating a vector with missing data:
  
  ```r
  d[2] <- NA; d
  ```

  [1] 3 NA 7

- To see if any of the elements of a vector are missing use `is.na()`:
  
  ```r
  is.na(d)
  ```

  [1] FALSE TRUE FALSE

Missing Data in Vectors II

- To obtain the element number of the vector of the missing value(s), use `which()` and `is.na()`:
  
  ```r
  which(is.na(d))
  ```

  [1] 2

- To calculate the mean in presence of missing value(s), use `mean()`:
  
  ```r
  mean(d, na.rm=TRUE)
  ```

  [1] 5

Missing Data in Matrices

- Start by creating a matrix with missing data:
  
  ```r
  h <- matrix(c(NA,3,1,7,-8,NA), nrow=3, ncol=2, byrow=TRUE); h
  ```

  ```r
  [,1] [,2]
  [1,] NA  3
  [2,]  1  7
  [3,] -8 NA
  ```
Missing Data in Matrices

- To see if any of the elements of a vector are missing use `is.na()`:

  ```r
  is.na(h)
  ```

  
  ```
  [,1] [,2]
  [1,] TRUE FALSE
  [2,] FALSE FALSE
  [3,] FALSE TRUE
  ```

Missing Data in Matrices II

- To see how many missing values there are, use `sum()` and `is.na()` (TRUE=1, FALSE=0):

  ```r
  sum(is.na(h))
  ```

  
  ```
  [1] 2
  ```

- To obtain the element number of the matrix of the missing value(s), use `which()` and `is.na()`:

  ```r
  which(is.na(h))
  ```

  
  ```
  [1] 1 6
  ```

Missing Data in Matrices III

- To keep only the rows without missing value(s), use `na.omit()`:

  ```r
  na.omit(h)
  ```

  
  ```
  [,1] [,2]
  [1,] 1 7
  ```

  ```r
  attr("na.action")
  ```

  
  ```
  [1] 1 3
  ```

  ```r
  attr("class")
  ```

  
  ```
  [1] "omit"
  ```
Data from the Internet

When downloading data from the internet, use `read.table()`.

In the arguments of the function:
- `header`: if TRUE, tells R to include variables names when importing
- `sep`: tells R how the entries in the data set are separated
  - `sep="",` when entries are separated by COMMAS
  - `sep="\t"` when entries are separated by TAB
  - `sep=" "` when entries are separated by SPACE

```
data <- read.table("http://www.stat.ucla.edu/~vlew/stat130a/datasets/twins.csv", header=TRUE, sep="",)
```

Importing Data from Your Computer

1. Check what folder R is working with now:
   ```r
   getwd()
   ```

2. Tell R in what folder the data set is stored (if different from (1)). Suppose your data set is on your desktop:
   ```r
   setwd("~/Desktop")
   ```

3. Now use the `read.table()` command to read in the data, substituting the name of the file for the website.

Using Data Available in R

To use a data set available in one of the R packages, install that package (if needed).
- Go to Packages & Data and R Package Manager to see the list of packages already installed.
- Go to Packages & Data and R Package Installer and type in the package you wish to install into the search box.

Load the package into R, using the `library()` function.
```r
library(alr3)
```

Extract the data set you want from that package, using the `data()` function. In our case, the data set is called `UN2`.
```r
data(UN2)
```

Working with Data sets in R I

After the variable names have been "attached", to see the variable names, use `names()`:
```r
names(UN2)
```

```
[1] "logPPgdp" "logFertility" "Purban" "Locality"
```

To use the variable names when working with data, use `attach()`:
```r
attach(UN2)
```

To see the descriptions of the variables, use `?`:
```r
?UN2
```
Working with Data sets in R II

- After modifying variables, use `detach()` and `attach()` to save the results:
  1. `UN2.copy <- UN2`
  2. `detach(UN2)`
  3. `UN2.copy[10,2] <- -0.5`

- To get an overview of the data sets and its variables, use the `summary()` function:
  1. `summary(UN2.copy)`

Working with Data sets in R III

<table>
<thead>
<tr>
<th>logPPgdp</th>
<th>logFertility</th>
<th>Purban</th>
<th>Locality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Min : 6.492</td>
<td>Min : -0.5000</td>
<td>Min : 6.00</td>
</tr>
<tr>
<td>2</td>
<td>1st Qu. : 8.867</td>
<td>1st Qu. : 0.6043</td>
<td>1st Qu. : 35.00</td>
</tr>
<tr>
<td>3</td>
<td>Median : 10.920</td>
<td>Median : 0.9783</td>
<td>Median : 57.00</td>
</tr>
<tr>
<td>4</td>
<td>Mean : 10.993</td>
<td>Mean : 1.0111</td>
<td>Mean : 55.54</td>
</tr>
<tr>
<td>5</td>
<td>3rd Qu. : 12.938</td>
<td>3rd Qu. : 1.4493</td>
<td>3rd Qu. : 75.00</td>
</tr>
<tr>
<td>6</td>
<td>Max : 15.444</td>
<td>Max : 2.0794</td>
<td>Max : 100.00</td>
</tr>
<tr>
<td>7</td>
<td>(Other) : 187</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Working with Data sets in R IV

- To get the mean of all the variables in the data set, use `mean()`:
  1. `mean(UN2)`

<table>
<thead>
<tr>
<th>logPPgdp</th>
<th>logFertility</th>
<th>Purban</th>
<th>Locality</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.993094</td>
<td>1.018016</td>
<td>55.538860</td>
<td>NA</td>
</tr>
</tbody>
</table>

Warning message:
In `mean.default(X[[4L]], ...)`: argument is not numeric or logical: returning NA

Working with Data sets in R V

- To get the variance-covariance matrix of all the (numerical) variables in the data set, use `var()`:
  1. `var(UN2[, -4])`

<table>
<thead>
<tr>
<th>logPPgdp</th>
<th>logFertility</th>
<th>Purban</th>
<th>Locality</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6408387</td>
<td>-0.8647205</td>
<td>44.555873</td>
<td></td>
</tr>
<tr>
<td>logFertility</td>
<td>-0.8647205</td>
<td>0.2887060</td>
<td>-7.630714</td>
</tr>
<tr>
<td>Purban</td>
<td>44.5558730</td>
<td>-7.6307145</td>
<td>579.197701</td>
</tr>
</tbody>
</table>
Creating Plots in R I

- To make a plot in R, you can use `plot()`:
  1. `plot(logFertility ~ logPPgdp, xlab = "logPPgdp", ylab="logFertility", main="logGDP vs logFertility Plot")`

Creating Plots in R II

- To make a histogram in R, you can use `hist()`:
  1. `hist(logFertility, xlab="logFertility", main="logFertility Histogram")`

Creating Plots in R III

- To add information to the histogram you can use `abline()`:
  1. `hist(logFertility, xlab="logFertility", main="logFertility Histogram")`
  2. `abline(v=mean(logFertility), col="red", lwd =3)`
Creating Plots in R IV

- To make a boxplot in R, you can use `boxplot()`:  
  ```
  boxplot(UN2[, -4], 
  xlab="Variable Names", main="Boxplot of the Data")
  ```

Creating Plots in R V

- To make a scatterplot for all the (numerical) variables, you can use `pairs()`:  
  ```
  pairs(UN2[, -4])
  ```

Creating Plots in R VI

- To add one point to an existing plot, use `points()`:  
  ```
  plot(logFertility ~ logPPgdp, xlab="logPPgdp", ylab="logFertility", 
  main="logGDP vs logFertility Plot")
  points(9, 0.3, col="red", pch=19, cex =2)
  ```

Creating Plots in R VII

- To add more than one point to an existing plot, use `points()`:  
  ```
  ind <- which(Purban>50)
  plot(logFertility ~ logPPgdp, xlab="logPPgdp", ylab="logFertility", 
  main="logGDP vs logFertility Plot")
  points(logFertility[ind], logPPgdp[ind], col="red", pch =19)
  ```
Creating Plots in R VIII

- To add a legend to an existing plot, use `legend()`:

```r
ind <- which(Purban > 50)
plot(logFertility ~ logPPgdp, xlab="logPPgdp", ylab="logFertility", main="logGDP vs logFertility Plot")
points(logFertility[ind], col="red", pch=19)
legend("topright", pch=c(1,19), col=1:2, c("Purban<50", "Purban>=50"))
```

![LogGDP vs LogFertility Plot](image)

Saving Plots as a PDF

*Note:* The files will be saved in the folder specified with `setwd()`. To save a plot in R as a PDF, you can use `pdf()`:

```r
pdf("UN2pairs.pdf", width=6, height=6, onefile =FALSE)
pairs(UN2[, -4])
dev.off()
```

Exploring R Objects I

- To see the names of the objects available to be saved (in your current workspace), use `ls()`.

```r
ls()
```

```
[1] "UN2" "a" "b" "d" "data" "e" "f" "h" "mat1" "mat2"
```

- To remove objects from your workspace, use `rm()`.

```r
rm(d)
ls()
```

```
[1] "UN2" "a" "b" "data" "e" "f" "h" "mat1" "mat2"
```
Exploring R Objects

To remove all the objects from your workspace, type:

```r
1 rm(list=ls())
2 ls()
```

character(0)

---

Saving and Loading R Objects

To save (to the current directory) all the objects in the workspace, use `save.image()`.

```r
1 save.image("basicR.RData")
```

To load (from the current directory), use `load()`.

```r
1 load("basicR.RData")
```

---

Exporting R Objects to LaTeX

To export certain R objects to be used in LaTeX, use `xtable()`.

```r
1 library(xtable)
2 xtable(summary(UN2))
```

---

Exporting R Objects to LaTeX II

```latex
\begin{table}
\begin{center}
\begin{tabular}{rllll}
\hline
& logPPgdp & logFertility & Purban & Locality \\
\hline
1 & Min. : 6.492 & Min. :0.0000 & Min. : 6.00 & Afghanistan: 1 \\
2 & 1st Qu.: 8.867 & 1st Qu.:0.6366 & 1st Qu.: 35.00 & Albania : 1 \\
3 & Median :10.920 & Median :0.9783 & Median : 57.00 & Algeria : 1 \\
4 & Mean :10.993 & Mean :1.0180 & Mean : 55.54 & Angola : 1 \\
5 & 3rd Qu.:12.938 & 3rd Qu.:1.4493 & 3rd Qu.: 75.00 & Argentina : 1 \\
6 & Max. :15.444 & Max. :2.0794 & Max. :100.00 & Armenia : 1 \\
7 & & & & (Other) :187 \\
\hline
\end{tabular}
\end{center}
\end{table}
```
Exporting R Objects to LaTeX

Exporting R Objects to LaTeX III

<table>
<thead>
<tr>
<th></th>
<th>logPPgdp</th>
<th>logFertility</th>
<th>Purban</th>
<th>Locality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Min.: 6.492</td>
<td>Min. : 0.0000</td>
<td>Min. : 6.00</td>
<td>Afghanistan : 1</td>
</tr>
<tr>
<td>2</td>
<td>1st Qu.: 8.867</td>
<td>1st Qu.: 0.6366</td>
<td>1st Qu.: 35.00</td>
<td>Albania : 1</td>
</tr>
<tr>
<td>3</td>
<td>Median : 10.920</td>
<td>Median : 0.9783</td>
<td>Median : 57.00</td>
<td>Algeria : 1</td>
</tr>
<tr>
<td>4</td>
<td>Mean : 10.953</td>
<td>Mean : 1.0180</td>
<td>Mean : 55.54</td>
<td>Angola : 1</td>
</tr>
<tr>
<td>5</td>
<td>3rd Qu.: 12.938</td>
<td>3rd Qu.: 1.4493</td>
<td>3rd Qu.: 75.00</td>
<td>Argentina : 1</td>
</tr>
<tr>
<td>6</td>
<td>Max. : 15.444</td>
<td>Max. : 1.0000</td>
<td>Max. : 100.00</td>
<td>Armenia : 1</td>
</tr>
<tr>
<td>7</td>
<td>(Other) : 187</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To save (to the current directory) certain objects in the workspace to be used in Excel, use `write.csv()`.

```r
write.csv(mat, "mat.csv")
```

Saving R Commands

Saving R Commands I

To see all of the commands you typed in an R session, click on the Yellow and Green Tablet

To save all of the commands you typed in an R session, use `savehistory()`.

```r
savehistory(file="history.log")
```

Saving R Commands II

Alternatively, use a .r file to store your commands.

1. Go to: File -> New Document
2. Type your commands
3. Save the file as "code.r"
4. Go back to the R Console
5. To run all the commands, use:

```r
source("code.r")
```
Error: syntax error

Possible causes:
- Incorrect spelling (of the function, variable, etc.)
- Including a ”+” when copying code from the Console
- Having an extra parenthesis at the end of a function
- Having an extra bracket when subsetting

Error in ... : requires numeric matrix/vector arguments

Possible causes:
- Not closing a function call with a parenthesis
- Not closing brackets when subsetting
- Not closing a function you wrote with a squiggly brace

Possible solutions:
- Coerce (a copy of) the data set to be a matrix, with the \texttt{as.matrix()} command
- Coerce (a copy of) the vector to have numeric entries, with the \texttt{as.numeric()} command
Exercise in R

Using the data set UN2, please recreate the following plot. Hint: The five colors in the plot are numbered 1-5.

Online Resources for R

- Download R: http://cran.stat.ucla.edu/
- Search Engine for R: http://rseek.org
- R Reference Card: http://cran.r-project.org/doc/contrib/Short-refcard.pdf
- R Graph Gallery: http://addictedtor.free.fr/graphiques/
- Statistics with R: http://zoonek2.free.fr/UNIX/48R/all.html
Online Resources for R II

- UCLA Statistical Consulting Center: http://scc.stat.ucla.edu

Upcoming Mini-Course
This Wednesday

4:30PM: Basic LaTeX

Thank you.
Any questions?