UCLA Department of Statistics
Statistical Consulting Center

Basic R

Irina Kukuyeva
ikukuyeva@stat.ucla.edu

April 2, 2009

Outline

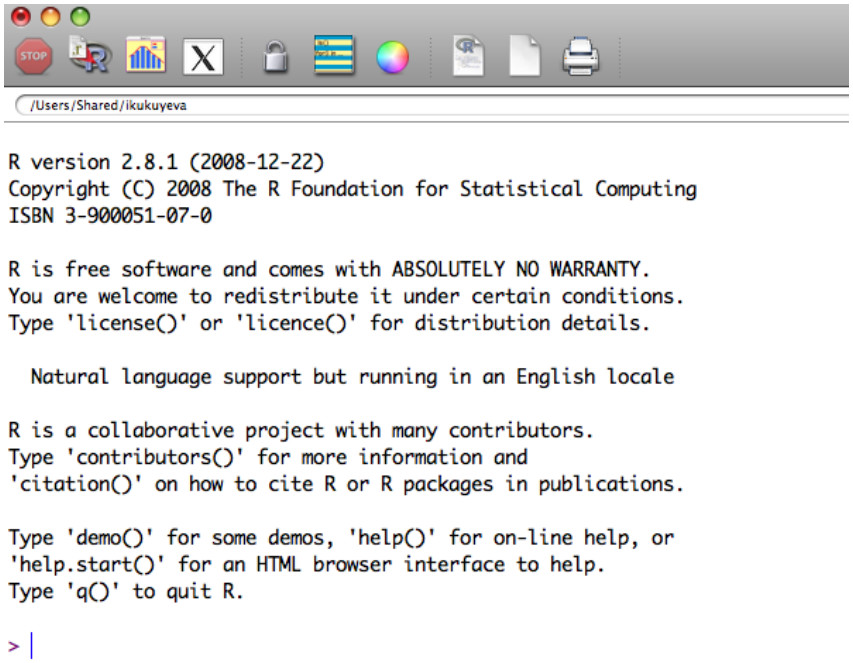
- I. Preliminaries
- II. Variable Assignment
- III. Working with Vectors
- IV. Working with Matrices
- V. From Vectors to Matrices
- VI. Handling Missing Data
- VII. Data sets in R
- VIII. Overview of Plots in R
- IX. R Environment
- X. Common Bugs and Fixes
- XI. Online Resources for R
- XII. Exercises
- XIII. Upcoming Mini-Course

Part I

Preliminaries

Installing R on a Mac

1. Go to `http://cran.r-project.org/` and select *MacOS X*
2. Select to download the latest version: 2.8.1 (2008-12-22)
3. Install and Open. The R window should look like this:



Comparisons in R

#: used for commenting words out

Comparisons in R

#: used for commenting words out

&: in logic, is used to mean AND

Comparisons in R

#: used for commenting words out

&: in logic, is used to mean AND

|: in logic, is used to mean OR

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)

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

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

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

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

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`

Part II

Variable Assignment

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 $=$:

```
# Approach 1
a=5
a
# Approach 2
a=5; a
# Approach 3
b<-5; b
```

Part III

Working with Vectors

Creating Vectors I

- 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()`:

```
1  e=seq(from=1, to=3, by=0.5); e
```

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

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

```
1  f=rep(NA, 6); f
```

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

Creating Vectors III

- To delete elements of a vector, use - and/or c():

```
1  e[-c(1,3)]; e
```

```
[1] 1.5 2.5 3.0
```

Some Useful Vector Functions I

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

```
1 length(d)
```

```
[1] 3
```

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

```
1 max(d)
```

```
[1] 7
```

Some Useful Vector Functions II

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

```
1 min(d)
```

```
[1] 3
```

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

```
1 mean(d)
```

```
[1] 4.666667
```

Some Useful Vector Functions III

- To sort the vector, use `sort()`:

```
1 g<-c(2,6,7,4,5,2,9,3,6,4,3)
2 sort(g, decreasing=T)
```

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

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

```
1 unique(g)
```

```
[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()`:

```
1 duplicated(g)
```

```
[1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE
[11] TRUE
```

Sub-setting with Vectors I

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

```
1  d[2]
```

```
[1] 4
```

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

```
1  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 `!=`:

```
1  d!=3
```

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

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

```
1  which(d==4)
```

```
[1] 2
```

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

Part IV

Working with Matrices

Creating Matrices I

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

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

- 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 II

- To multiply two matrices, use %*%.
  Note: If you use * instead, you will be performing matrix multiplication element-wise.

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

```
     [,1] [,2] [,3]
[1,]  269  292  315
[2,]  292  317  342
[3,]  315  342  369
```

- To find the dimensions of a matrix, use dim():

```
1  dim(mat)
```

```
[1] 3 2
```

Some Useful Matrix Functions III

- Alternatively, we can find the rows and columns of the matrix, by `nrow()` and `ncol()`:

```
1  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 `[ ]`:

```
1  mat[1,1]
```

```
[1] 10
```

- To see what is stored in the first row of the matrix:

```
1  mat[1,]
```

```
[1] 10 13
```

Subsetting with Matrices II

- To see what is stored in the second column of the matrix:

```
1  mat[, 2]
```

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

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

```
1  mat[c(1,3), 2]
```

```
[1] 13 15
```

Part V

From Vectors to Matrices

Creating Matrices from Vectors I

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

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

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

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

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

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

Part VI

Handling Missing Data

Missing Data in Vectors I

- Start by creating a vector with missing data:

```
1  d[2]<-NA; d
```

```
[1] 3 NA 7
```

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

```
1  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()`:

```
1  which(is.na(d))
```

```
[1] 2
```

- To calculate the mean in presence of missing value(s), use `mean()`:

```
1  mean(d, na.rm=TRUE)
```

```
[1] 5
```

Missing Data in Matrices I

- Start by creating a matrix with missing data:

```
1 h=matrix(c(NA,3,1,7,-8,NA), nrow=3,ncol=2,
2 byrow=T); h
```

```
     [,1] [,2]
[1,]   NA    3
[2,]    1    7
[3,]   -8   NA
```

Missing Data in Matrices II

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

```
1  is.na(h)
```

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

Missing Data in Matrices III

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

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

```
[1] 2
```

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

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

```
[1] 1 6
```

Missing Data in Matrices IV

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

```
1  na.omit(h)
```

```
     [,1] [,2]
[1,]    1    7
attr(,"na.action")
[1] 1 3
attr(,"class")
[1] "omit"
```

Part VII

Data sets in R

Data from the Internet I

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 entires 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

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

Importing Data from Your Computer I

1. Check what folder R is working with now:

```
getwd()
```

2. Tell R in what folder the data set is stored (if different from (1)). Suppose your data set is on your desktop:

```
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 I

1. To use a data set available in one of the R packages, install that package (if needed).
2. Load the package into R, using the `library()` function.

```
1 library(alr3)
```

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

```
1 data(UN2)
```

Working with Data sets in R I

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

```
1 data(UN2)
2 attach(UN2)
```

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

```
1 names(UN2)
```

- To see the descriptions of the variables, use `?`:

```
1 ?UN2
```

Working with Data sets in R II

- After modifying variables, use `detach()` and `attach()` to save the results:

```
1 # Make a copy of the data set
2 UN2.copy<-UN2
3 detach(UN2)
4 attach(UN2.copy)
5 # Change the 10th observation for
      logFertility
6 UN2.copy[10, 2]<-999
```

Working with Data sets in R III

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

```
1 # Check that the change has been made
2 summary(UN2)
3 summary(UN2.copy)
```

Working with Data sets in R IV

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

```
1  mean(UN2)
```

```
   logPPgdp  logFertility       Purban     Locality
  10.993094      1.018016    55.538860           NA
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])
```

```
              logPPgdp logFertility     Purban
logPPgdp     5.6408387   -0.8647205  44.555873
logFertility -0.8647205    0.2887060  -7.630714
Purban       44.5558730   -7.6307145 579.197701
```

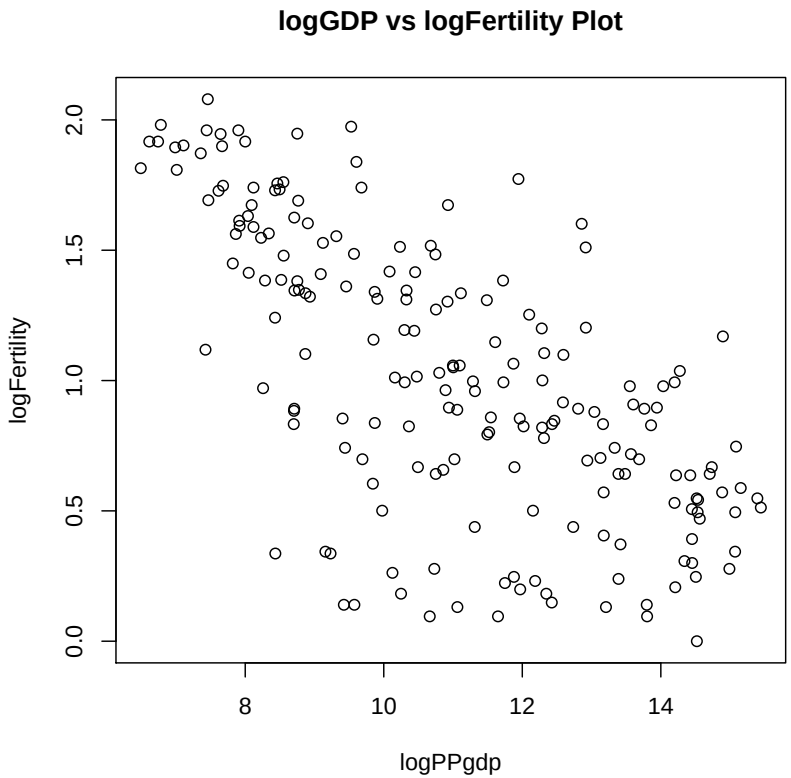
Part VIII

Overview of Plots in R

Creating Plots in R I

- To make a plot in R, you can use `plot()`:

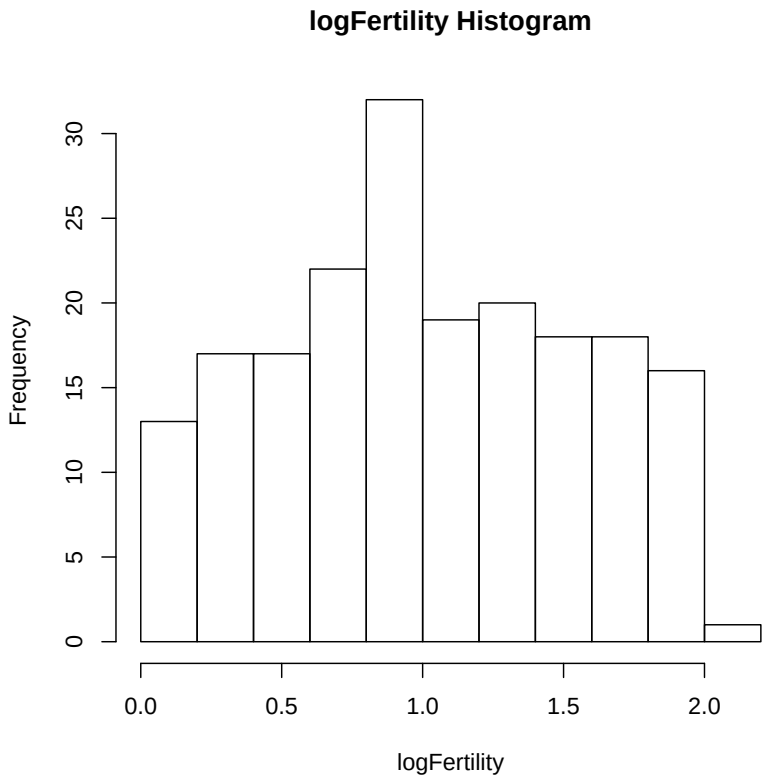
```
1 plot(logFertility~logPPgdp, xlab="logPPgdp",
      ylab="logFertility",
2 main="logGDP vs logFertility Plot")
```



Creating Plots in R II

- To make a histogram in R, you can use `hist()`:

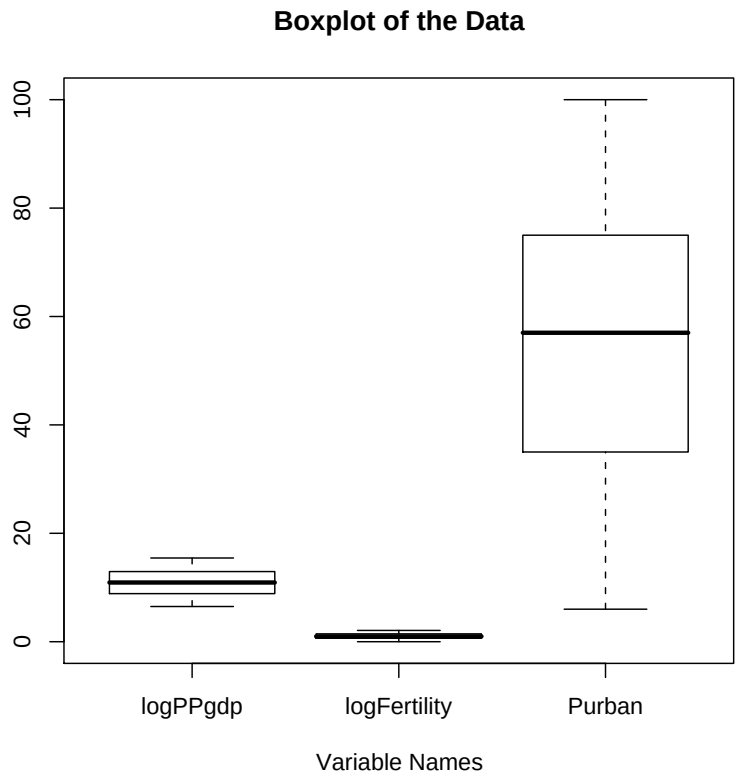
```
1  hist(logFertility,
       xlab="
       logFertility",
       main="
       logFertility
       Histogram")
```



Creating Plots in R III

- To make a boxplot in R, you can use `boxplot()`:

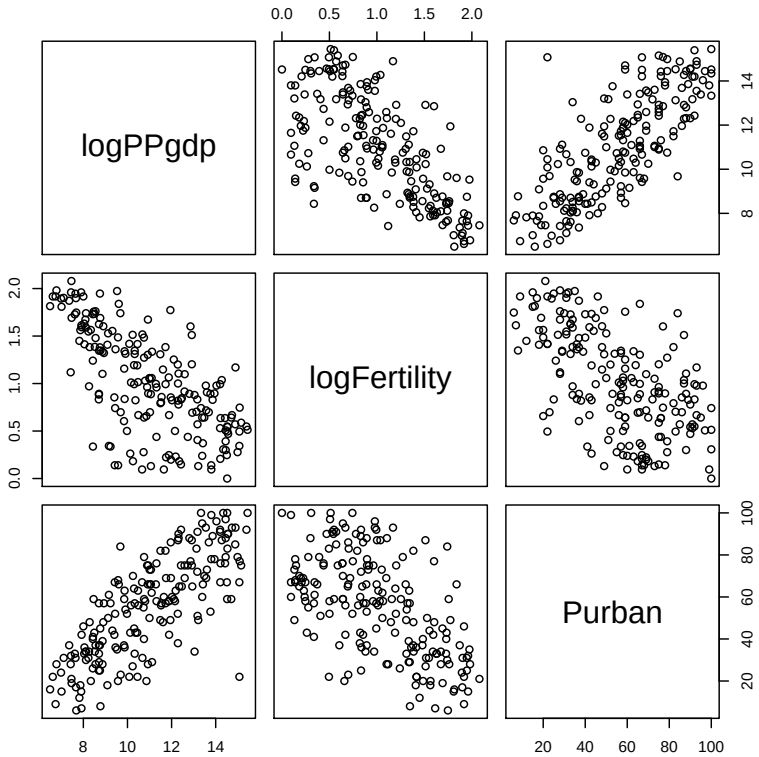
```
1  boxplot(UN2[, -4],
       xlab="Variable
       Names", main="
     Boxplot of the
     Data")
```



Creating Plots in R IV

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

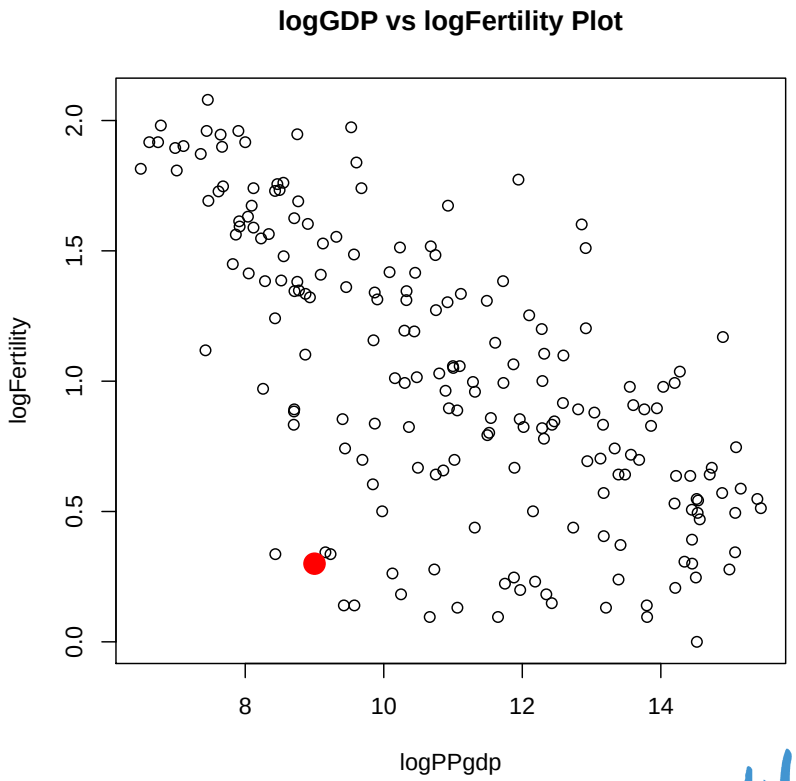
```
1  pairs(UN2[, -4])
```



Creating Plots in R V

- To add more points to an existing plot, use `points()`:

```
1 plot(logFertility~logPPgdp, xlab="logPPgdp",
      ylab="logFertility",
2 main="logGDP vs logFertility Plot")
3 points(9,0.3, col="red", pch=19, cex=2)
```



Saving Plots as a PDF I

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()`:

```
1 pdf("UN2pairs.pdf", width=6, height=6, onefile
      =F)
2 pairs(UN2[, -4])
3 dev.off()
```

Part IX

R Environment

Exploring R Objects I

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

```
1 ls()
```

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

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

```
1 rm(d)
2 ls()
```

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

Exploring R Objects II

- To remove *all* the objects from your workspace, type:

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

```
character(0)
```

Saving and Loading R Objects I

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

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

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

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

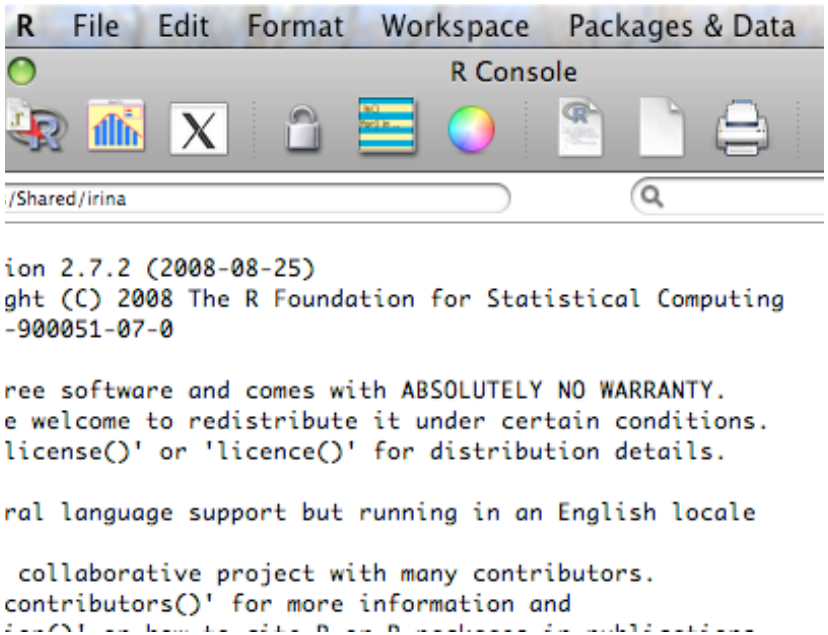
Exporting R Objects to Other Formats I

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

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

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:

```
1 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:

```
1  source("code.r")
```

Part X

Common Bugs and Fixes

`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

Trailing +

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

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

Possible causes:

1. Objects are data frames, not matrices
2. Elements of the vectors are characters

Possible solutions:

1. Coerce (a copy of) the data set to be a matrix, with the `as.matrix()` command
2. Coerce (a copy of) the vector to have numeric entries, with the `as.numeric()` command

Part XI

Online Resources for R

Online Resources for R

Download R: http://cran.stat.ucla.edu/

Online Resources for R

Download R: http://cran.stat.ucla.edu/

Search Engine for R: rseek.org

Online Resources for R

Download R: http://cran.stat.ucla.edu/

Search Engine for R: rseek.org

R Reference Card: http://cran.r-project.org/doc/contrib/Short-refcard.pdf

Online Resources for R

Download R: http://cran.stat.ucla.edu/

Search Engine for R: rseek.org

R Reference Card: http://cran.r-project.org/doc/contrib/Short-refcard.pdf

UCLA Statistics Information Portal: http://info.stat.ucla.edu/grad/

Online Resources for R

Download R: http://cran.stat.ucla.edu/

Search Engine for R: rseek.org

R Reference Card: http://cran.r-project.org/doc/contrib/Short-refcard.pdf

UCLA Statistics Information Portal: http://info.stat.ucla.edu/grad/

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

Part XII

Exercises

Exercises in R

1. Sum all the even rows of column 2 of the 6*6 matrix that contains squares of the first 36 numbers.
2. Using the `UN2` data set, plot `logFertility` vs `logPPgdp` , highlighting in red those observations that have a percentage of the urban population greater than 50%.

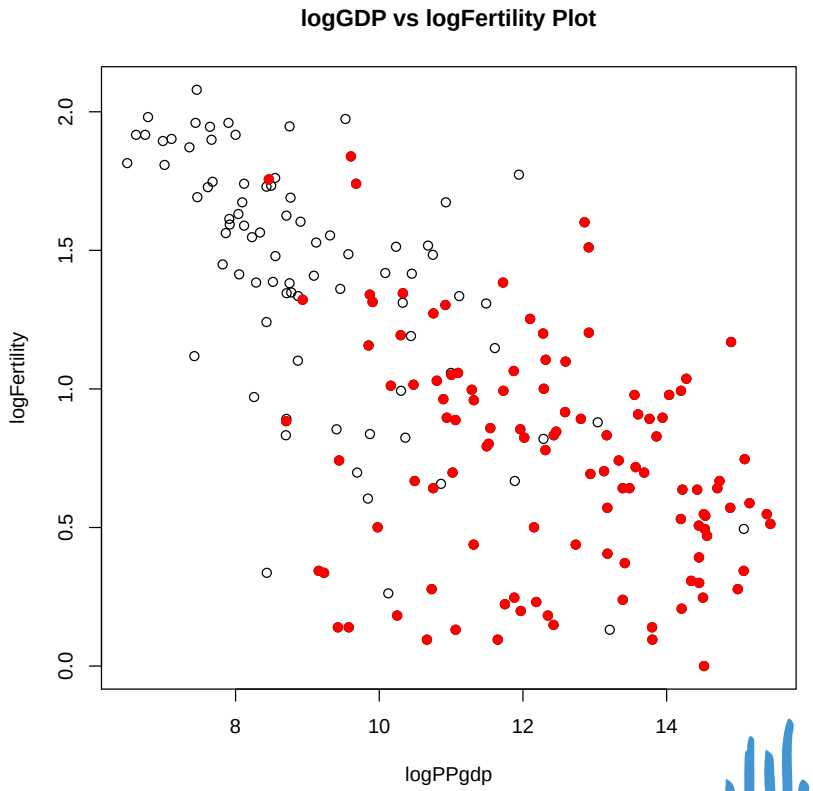
Solution Exercise 1

```
1  s<-seq(from=1, to=36, by=1)
2  mat.res<-matrix(s^2,6,6) ; mat.res
3  ind<-seq(from=2, to=6, by=2)
4  ans<-sum(mat.res[ind, 2]); ans
```

Solution to the Exercises

Exercise 2 Solution

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



Part XIII

Upcoming Mini-Course

Upcoming Mini-Course

- This Thursday (April 2):
  - Introductory Statistics with R
- For a schedule of all mini-courses offered please visit *http://scc.stat.ucla.edu/mini-courses*.

Thank you
Any questions?