Getting Started in R~Stata
Notes on Exploring Data
(v. 1.0)

Oscar Torres-Reyna
otorres@princeton.edu

http://dss.princeton.edu/training/

Fall 2010
What is R?

• “R is a language and environment for statistical computing and graphics”*
• R is offered as open source (i.e. free)

What is Stata?

• It is a multi-purpose statistical package to help you explore, summarize and analyze datasets.
• A dataset is a collection of several pieces of information called variables (usually arranged by columns). A variable can have one or several values (information for one or several cases).
• Other statistical packages are SPSS and SAS.

<table>
<thead>
<tr>
<th>Features</th>
<th>Stata</th>
<th>SPSS</th>
<th>SAS</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning curve</td>
<td>Steep/gradual</td>
<td>Gradual/flat</td>
<td>Pretty steep</td>
<td>Pretty steep</td>
</tr>
<tr>
<td>User interface</td>
<td>Programming/point-and-click</td>
<td>Mostly point-and-click</td>
<td>Programming</td>
<td>Programming</td>
</tr>
<tr>
<td>Data manipulation</td>
<td>Very strong</td>
<td>Moderate</td>
<td>Very strong</td>
<td>Very strong</td>
</tr>
<tr>
<td>Data analysis</td>
<td>Powerful</td>
<td>Powerful</td>
<td>Powerful/versatile</td>
<td>Powerful/versatile</td>
</tr>
<tr>
<td>Graphics</td>
<td>Very good</td>
<td>Very good</td>
<td>Good</td>
<td>Excellent</td>
</tr>
<tr>
<td>Cost</td>
<td>Affordable (perpetual licenses, renew only when upgrade)</td>
<td>Expensive (but not need to renew until upgrade, long term licenses)</td>
<td>Expensive (yearly renewal)</td>
<td>Open source</td>
</tr>
</tbody>
</table>


* [http://www.r-project.org/index.html](http://www.r-project.org/index.html)
This is the R screen in Multiple-Document Interface (MDI)…
This is the R screen in Single-Document Interface (SDI)...

R version 2.11.1 (2010-05-31)
Copyright (C) 2010 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.

>
<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Working directory</strong></td>
<td></td>
</tr>
<tr>
<td>getwd() # Shows the working directory (wd)</td>
<td>pwd /<em>Shows the working directory</em>/</td>
</tr>
<tr>
<td>setwd(&quot;C:/myfolder/data&quot;) # Changes the wd</td>
<td>cd c:\myfolder\data /<em>Changes the wd</em>/</td>
</tr>
<tr>
<td>setwd(&quot;H:\myfolder\data&quot;) # Changes the wd</td>
<td>cd &quot;c:\myfolder\stata data&quot; /<em>Notice the spaces</em>/</td>
</tr>
<tr>
<td><strong>Installing packages/user-written programs</strong></td>
<td></td>
</tr>
<tr>
<td>install.packages(&quot;ABC&quot;) # This will install the package --ABC--. A window will pop-up, select a mirror site to download from (the closest to where you are) and click ok.</td>
<td>ssc install abc /*Will install the user-defined program ‘abc’. It will be ready to run.</td>
</tr>
<tr>
<td>library(ABC) # Load the package --ABC-- to your workspace in R</td>
<td>findit abc /*Will do an online search for program ‘abc’ or programs that include ‘abc’. It also searcher your computer.</td>
</tr>
<tr>
<td><strong>Getting help</strong></td>
<td></td>
</tr>
<tr>
<td>?plot # Get help for an object, in this case for the --plot-- function. You can also type: help(plot)</td>
<td>help tab /* Get help on the command ‘tab’*/</td>
</tr>
<tr>
<td>??regression # Search the help pages for anything that has the word &quot;regression&quot;. You can also type: help.search(&quot;regression&quot;)</td>
<td>search regression /* Search the keywords for the word ‘regression’*/</td>
</tr>
<tr>
<td>apropos(&quot;age&quot;) # Search the word &quot;age&quot; in the objects available in the current R session.</td>
<td>hsearch regression /* Search the help files for the work ‘regression’. It provides more options than ‘search’*/</td>
</tr>
<tr>
<td>help(package=car) # View documentation in package ‘car’. You can also type: library(help=&quot;car&quot;)</td>
<td></td>
</tr>
<tr>
<td>help(DataABC) # Access codebook for a dataset called ‘DataABC’ in the package ABC</td>
<td></td>
</tr>
</tbody>
</table>
### Data from *.csv (copy-and-paste)

**R**

```r
# Select the table from the excel file, copy, go to the R Console and type:

mydata <- read.table("clipboard", header=TRUE, sep="\t")
```

**Stata**

```stata
/* Select the table from the excel file, copy, go to Stata, in the command line type: 

edit

/*The data editor will pop-up and paste the data (Ctrl-V). Select the link for to include variable names
```

### Data from *.csv

**R**

```r
# Reading the data directly

mydata <- read.csv("c:\mydata\mydatafile.csv", header=TRUE)
```

```r
# The will open a window to search for the *.csv file.

mydata <- read.csv(file.choose(), header = TRUE)
```

**Stata**

```stata
/* In the command line type */

insheet using "c:\mydata\mydatafile.csv"
```

```stata
/* Using the menu */

Go to File->Import->"ASCII data created by spreadsheet". Click on ‘Browse’ to find the file and then OK.
```

### Data from/to *.txt (space, tab, comma-separated)

**R**

```r
# In the example above, variables have spaces and missing data is coded as ‘-9’

mydata <- read.table("C:/myfolder/abc.txt", header=TRUE, sep="\t", na.strings = "-9")
```

```r
# Export the data

write.table(mydata, file = "test.txt", sep = "\t")
```

**Stata**

```stata
/* See insheet above */

infile var1 var2 str7 var3 using abc.raw
```

```stata
/* Variables with embedded spaces must be enclosed in quotes */

 outsourcing using "c:\mydata\abc.csv"
```

```stata
/* Export data

outsheet using "c:\mydata\abc.csv"
```
# R

```r
library(foreign) # Load package --foreign--

mydata.spss <-
read.spss("http://dss.princeton.edu/training/mydata.sav",
  to.data.frame = TRUE,
  use.value.labels=TRUE,
  use.missings = TRUE)

# Where:
# # ‘to.data.frame’ return a data frame.
# # ‘use.value.labels’ Convert variables with value
# labels into R factors with those levels.
# # ‘use.missings’ logical: should information on
# user-defined missing values be used to set the
# corresponding values to NA.
# Source: type ?read.spss

write.foreign(mydata, codefile="test2.sps",
datafile="test2.raw", package="SPSS")

# Provides a syntax file (*.sps) to read the *.raw
data file
```

# Stata

```stata
/* Need to install the program ‘usespss’ (you do
this only once) */

ssc install usespss

/* To read the *.sav type (in one line):

usespss using
http://dss.princeton.edu/training/mydata.sav

/* For additional information type */

help usespss

Note: This does not work with SPSS portable files
(*.por)

-----------------------------------------------

/* Stata does not convert files to SPSS. You need
to save the data file as a Stata file version 9
that can be read by SPSS v15.0 or later*/

/* From Stata type: */

saveold mydata.dta /* Saves data to v.9 for SPSS
```
## Data from/to SAS

### R

```r
# To read SAS XPORT format (*.xpt)
library(foreign)  # Load package --foreign--
mydata.sas <- read.xport("http://dss.princeton.edu/training/mydata.xpt")  # Does not work for files online
mydata.sas <- read.xport("c:/myfolder/mydata.xpt")

# Using package --Hmisc--
library(Hmisc)
mydata <- sasxport.get(http://dss.princeton.edu/training/mydata.xpt)  # It works

write.foreign(mydata, codefile="test2.sas",
datafile="test2.raw", package="SAS")
# Provide a syntax file (*.sas) to read the *.raw data
```

### Stata

```stata
/*If you have a file in SAS XPORT format (*.xpt) you can use 'fdause' (or go to File->Import). */
fdause "c:/myfolder/mydata.xpt"
/* Type help fdause for more details */
/* If you have SAS installed in your computer you can use the program 'usesas', which you can install by typing: */
ssc install usesas
/* To read the *.sas7bcat type (in one line): */
usesas using "c:\mydata.sas7bdat"

/* You can export a dataset as SAS XPORT by menu (go to File->Export) or by typing */
fdasave "c:/myfolder/mydata1.xpt"
/* Type help fdasave for more details */
```

**NOTE:** As an alternative, you can use SAS Universal Viewer (freeware from SAS) to read SAS files and save them as *.csv. Saving the file as *.csv removes variable/value labels, make sure you have the codebook available.
<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td>library(foreign) # Load package --foreign--</td>
<td>/* To open a Stata file go to File -&gt; Open, or type: */</td>
</tr>
<tr>
<td>mydata &lt;- read.dta(&quot;<a href="http://dss.princeton.edu/training/students.dta">http://dss.princeton.edu/training/students.dta</a>&quot;)</td>
<td>use &quot;c:\myfolder\mydata.dta&quot;</td>
</tr>
<tr>
<td>mydata.dta &lt;- read.dta(&quot;<a href="http://dss.princeton.edu/training/mydata.dta">http://dss.princeton.edu/training/mydata.dta</a>&quot;, convert.factors=TRUE, convert.dates=TRUE, convert.underscore=TRUE, warn.missing.labels=TRUE)</td>
<td>Or</td>
</tr>
<tr>
<td># Where (source: type ?read.dta)</td>
<td>use &quot;<a href="http://dss.princeton.edu/training/mydata.dta">http://dss.princeton.edu/training/mydata.dta</a>&quot;</td>
</tr>
<tr>
<td># convert.dates. Convert Stata dates to Date class</td>
<td>/* If you need to load a subset of a Stata data file type */</td>
</tr>
<tr>
<td># convert.factors. Use Stata value labels to create factors? (version 6.0 or later).</td>
<td>use var1 var2 using &quot;c:\myfolder\mydata.dta&quot;</td>
</tr>
<tr>
<td># convert.underscore. Convert &quot;_&quot; in Stata variable names to &quot;.&quot; in R names?</td>
<td>use id city state gender using &quot;mydata.dta&quot;, clear</td>
</tr>
<tr>
<td># warn.missing.labels. Warn if a variable is specified with value labels and those value labels are not present in the file.</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>write.dta(mydata, file = &quot;test.dta&quot;) # Direct export to Stata</td>
<td>/* To save a dataset as Stata file got File -&gt; Save As, or type: */</td>
</tr>
<tr>
<td>write.foreign(mydata, codefile=&quot;test1.do&quot;, datafile=&quot;test1.raw&quot;, package=&quot;Stata&quot;) # Provide a do-file to read the *.raw data</td>
<td>save mydata, replace /<em>If the fist time</em>/</td>
</tr>
<tr>
<td>save, replace /<em>If already saved as Stata file</em>/</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Package --foreign-- can only read Stata 12 or older versions, to read Stata 13+ see slide on page 29.
<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data from/to R</strong></td>
<td>/* Stata can’t read R data files */</td>
</tr>
</tbody>
</table>
| load("mydata.RData")
load("mydata.rda") | |
| /* Add path to data if necessary */ | |
| --------------------------------- | |
| save.image("mywork.RData") # Saving all objects
to file *.RData | |
| save(object1, object2, file="mywork.rda") # Saving selected objects | |
R

Data from ACII Record form

```r
mydata.dat <-
read.fwf(file="http://dss.princeton.edu/training/mydata.dat",
    width=c(7, -16, 2, 2, -4, 2, -10, 2, -110,
            3, -6, 2),
    col.names=c("w","y","x1","x2","x3", "age", "sex"),
    n=1090)
```

# Reading ASCII record form, numbers represent the width of variables, negative sign excludes variables not wanted (you must include these).

# To get the width of the variables you must have a codebook for the data set available (see an example below).

# To get the widths for unwanted spaces use the formula:

```
Start of var(t+1) - End of var(t) - 1
```

*Thank you to Scott Kostyshak for useful advice/code.

Stata

```stata
/* Using infix */
infix var1 1-7 var2 24-25 str2 var3 26-27 var4 32-33 str2 var5 44-45 var6 156-158 var7 165-166 using "http://dss.princeton.edu/training/mydata.dat"
```

```stata
/* Using infile */
dictionary using c:\data\mydata.dat {
    _column(1) var1 %7.2f " Label for var1 "
    _column(24)  var2 %2f " Label for var2 "
    _column(26)  str2 var3 %2s " Label for var3 "
    _column(32)  var4 %2f " Label for var4 "
    _column(44)  str2 var5 %2s " Label for var5 "
    _column(156) str3 var6 %3s " Label for var6 "
    _column(165) str2 var5 %2s " Label for var7 "
}
```

/*Do not forget to close the brackets and press enter after the last bracket*/

Save it as mydata.dct

With infile we run the dictionary by typing:

```
infile using c:\data\mydata
```


Data locations usually available in codebooks

<table>
<thead>
<tr>
<th>Var</th>
<th>Rec</th>
<th>Start</th>
<th>End</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>F7.2</td>
</tr>
<tr>
<td>var2</td>
<td>1</td>
<td>24</td>
<td>25</td>
<td>F2.0</td>
</tr>
<tr>
<td>var3</td>
<td>1</td>
<td>26</td>
<td>27</td>
<td>A2</td>
</tr>
<tr>
<td>var4</td>
<td>1</td>
<td>32</td>
<td>33</td>
<td>F2.0</td>
</tr>
<tr>
<td>var5</td>
<td>1</td>
<td>44</td>
<td>45</td>
<td>A2</td>
</tr>
<tr>
<td>var6</td>
<td>1</td>
<td>156</td>
<td>158</td>
<td>A3</td>
</tr>
<tr>
<td>var7</td>
<td>1</td>
<td>165</td>
<td>166</td>
<td>A2</td>
</tr>
</tbody>
</table>
## R

**Exploring data**

- `str(mydata)`  
  # Provides the structure of the dataset
- `summary(mydata)`  
  # Provides basic descriptive statistics and frequencies
- `names(mydata)`  
  # Lists variables in the dataset
- `head(mydata)`  
  # First 6 rows of dataset
- `head(mydata, n=10)`  
  # First 10 rows of dataset
- `head(mydata, n=-10)`  
  # All rows but the last 10
- `tail(mydata)`  
  # Last 6 rows
- `tail(mydata, n=10)`  
  # Last 10 rows
- `tail(mydata, n=-10)`  
  # All rows but the first 10
- `mydata[1:10,]`  
  # First 10 rows of data
- `mydata[1:10,1:3]`  
  # First 10 rows of data of the first 3 variables
- `edit(mydata)`  
  # Open data editor

**Missing data**

- `sum(is.na(mydata))`  
  # Number of missing in dataset
- `rowSums(is.na(mydata))`  
  # Number of missing per variable
- `rowMeans(is.na(mydata)) * length(mydata)`  
  # No. of missing per row
- `mydata[mydata$age=="&","age"] <- NA`  
  # NOTE: Notice hidden spaces.
- `mydata[mydata$age==999,"age"] <- NA`

The function `complete.cases()` returns a logical vector indicating which cases are complete.

# list rows of data that have missing values
`mydata[!complete.cases(mydata),]`

The function `na.omit()` returns the object with listwise deletion of missing values.

# create new dataset without missing data
`newdata <- na.omit(mydata)`

## Stata

**Exploring data**

- `describe`  
  /* Provides the structure of the dataset*/
- `summarize`  
  /* Provides basic descriptive statistics for numeric data*/
- `ds`  
  /* Lists variables in the dataset */
- `list in 1/6`  
  /* First 6 rows */
- `edit`  
  /* Open data editor (double-click to edit*/
- `browse`  
  /* Browse data */

**Missing data**

- `tabmiss`  
  /* # of missing. Need to install, type scc install tabmiss. Also try findit tabmiss and follow instructions */

/* For missing values per observation see the function `rowmiss` and the `egen` command*/
### R

#### Renaming variables

```r
#With reference number
as.matrix(colnames(mydata))
names(mydata)[7] <- "new_name"

#Multiple
names(mydata)[c(7,8)] <- c("new_name1", "new_name2")

#Without reference numbers
names(mydata)[names(mydata) == "old_name"] <- "new_name"

#Using the `plyr` package
library(plyr)
mydata <- rename(mydata, c("old_name"="new_name"))

#Multiple
mydata <- rename(mydata, c("old_name1"="new_name1", "old_name2"="new_name2"))
```

### Stata

```stata
edit  /* Open data editor (double-click to edit)
rename oldname newname
rename lastname last
rename firstname first
rename studentstatus status
rename averagescoregrade score
rename heightin height
rename newspaperreadershiptimeswk read
```

### Variable labels

Use variable names as variable labels

```stata
/* Adding labels to variables */
label variable w "Weight"
label variable y "Output"
label variable x1 "Predictor 1"
label variable x2 "Predictor 2"
label variable x3 "Predictor 3"
label variable age "Age"
label variable sex "Gender"
```
### R

#### Value labels

```r
# Use factor() for nominal data
mydata$sex <- factor(mydata$sex, levels = c(1,2), labels = c("male", "female"))

# Use ordered() for ordinal data
mydata$var2 <- ordered(mydata$var2, levels = c(1,2,3,4), labels = c("Strongly agree", "Somewhat agree", "Somewhat disagree", "Strongly disagree"))

mydata$var8 <- ordered(mydata$var2, levels = c(1,2,3,4), labels = c("Strongly agree", "Somewhat agree", "Somewhat disagree", "Strongly disagree"))  # Making a copy of the same variable
```

### Stata

#### Value labels

```stata
/* Step 1 defining labels */
label define approve 1 "Approve strongly" 2 "Approve somewhat" 3 "Disapprove somewhat" 4 "Disapprove strongly" 5 "Not sure" 6 "Refused"

label define well 1 "Very well" 2 "Fairly well" 3 "Fairly badly" 4 "Very badly" 5 "Not sure" 6 "Refused"

label define partyid 1 "Party A" 2 "Party B" 3 "Equally party A/B" 4 "Third party candidates" 5 "Not sure" 6 "Refused"

label define gender 1 "Male" 2 "Female"

/* Step 2 applying labels */
label values y approve
label values x1 approve
tab x1
d x1
destring x1, replace
label values x1 approve
label values x2 well
label values x3 partyid
tab x3
destring x3, replace ignore(&)
label values x3 partyid
tab x3
label values sex gender

tab1 y x1 x2 x3 age sex
```
### Creating ids/sequence of numbers

#### R

```r
# Creating a variable with a sequence of numbers or to index

# Creating a variable with a sequence of numbers from 1 to n (where ‘n’ is the total number of observations)
mydata$id <- seq(dim(mydata)[1])

# Creating a variable with the total number of observations
mydata$total <- dim(mydata)[1]

/* Creating a variable with a sequence of numbers from 1 to n per category (where ‘n’ is the total number of observations in each category) */
mydata <- mydata[order(mydata$group),]
idgroup <- tapply(mydata$group, mydata$group, function(x) seq(1,length(x),1))
mydata$idgroup <- unlist(idgroup)
```

(1) Thanks to Alex Acs for the code

#### Stata

```stata
/* Creating a variable with a sequence of numbers or to index */

/* Creating a variable with a sequence of numbers from 1 to n (where ‘n’ is the total number of observations) */
gen id = _n

/* Creating a variable with the total number of observations */
gen total = _N

/* Creating a variable with a sequence of numbers from 1 to n per category (where ‘n’ is the total number of observations in each category) */
bysort group: gen id = _n

For more info see:

http://www.stata.com/help.cgi?_n
http://dss.princeton.edu/training/StataTutorial.pdf
```
<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recoding variables</strong></td>
<td><strong>recode age (18 19 = 1 &quot;18 to 19&quot;) ///</strong></td>
</tr>
<tr>
<td><strong>library(car)</strong></td>
<td><strong>(20/28 = 2 &quot;20 to 29&quot;) ///</strong></td>
</tr>
<tr>
<td>`mydata$Age.rec &lt;- recode(mydata$Age,</td>
<td><strong>(30/39 = 3 &quot;30 to 39&quot;) (else=.),</strong></td>
</tr>
<tr>
<td>&quot;18:19='18to19';</td>
<td><strong>generate(agegroups) label(agegroups)</strong></td>
</tr>
<tr>
<td>20:29='20to29';</td>
<td></td>
</tr>
<tr>
<td>30:39='30to39'&quot;)</td>
<td></td>
</tr>
<tr>
<td><code>mydata$Age.rec &lt;- as.factor(mydata$Age.rec)</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dropping variables</strong></th>
<th><strong>drop var1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mydata$Age.rec &lt;- NULL</code></td>
<td><strong>drop var1-var10</strong></td>
</tr>
<tr>
<td><code>mydata$var1 &lt;- mydata$var2 &lt;- NULL</code></td>
<td></td>
</tr>
</tbody>
</table>

| **Keeping track of your work**          | ***/A log file helps you save commands and output** |
|----------------------------------------|**to a text file (*.log) or to a Stata read-only** |
| # Save the commands used during the session | **file (*.smcl). The best way is to save it as a** |
| `savehistory(file="mylog.Rhistory")` | **text file (*.log)***/          |
| # Load the commands used in a previous session | **log using mylog.log   */** |
| `loadhistory(file="mylog.Rhistory")` | **log close */**                |
| # Display the last 25 commands          | **log using mylog.log, append  */** |
| `history()`                             | **log using mylog.log, replace */**   |
| # You can read mylog.Rhistory with any word processor. Notice that the file has to have the extension *.Rhistory | /*You can read mylog.log using any word processor*/ |
Categorical data: Frequencies/Crosstabs

R

table(mydata$Gender)
table(mydata$Read)
readgender <- table(mydata$Read, mydata$Gender)
prop.table(readgender, 1)  # Row proportions
prop.table(readgender, 2)  # Col proportions
prop.table(readgender)     # Tot proportions
chisq.test(readgender)     # Do chisq test Ho: no relationship
fisher.test(readgender)    # Do fisher'exact test Ho: no relationship
round(prop.table(readgender, 2), 2)  # Round col prop to 2 digits
round(prop.table(readgender, 2), 2)  # Round col prop to 2 digits
round(100* prop.table(readgender, 2), 2)  # Round col % to 2 digits
round(100* prop.table(readgender, 2))  # Round col % to whole numbers
addmargins(readgender)     # Adding row/col margins

#install.packages("vcd")
library(vcd)
assocstats(majorgender)

# NOTE: Chi-sqr = sum (obs-exp)^2/exp
Degrees of freedom for Chi-sqr are (r-1)*(c-1)

# NOTE: Chi-sqr contribution = (obs-exp)^2/exp

# Cramer's V = sqrt(Chi-sqr/N*min)
Where N is sample size and min is a the minimun of (r-1) or (c-1)

Stata

tab gender /*Frequencies*/
tab read

tab read gender, col row /*Crosstabs*/
tab read gender, col row chi2 V

/*Crosstabs where chi2 (The null hypothesis (Ho) is that there is no relationship) and V (measure of association goes from 0 to 1)*/
bysort studentstatus: tab read gender, colum row
## Categorical data: Frequencies/Crosstabs

### R

```r
install.packages("gmodels")
library(gmodels)
mydata$ones <- 1  # Create a new variable of ones
CrossTable(mydata$Major,digits=2)
CrossTable(mydata$Major,mydata$ones, digits=2)
CrossTable(mydata$Gender,mydata$ones, digits=2)
CrossTable(mydata$Major,mydata$Gender,digits=2, expected=TRUE,dnn=c("Major","Gender"))
CrossTable(mydata$Major,mydata$Gender,digits=2, dnn=c("Major","Gender"))
chisq.test(mydata$Major,mydata$Gender)  # Null hypothesis: no association
# 3-way crosstabs
test <- xtabs(~Read+Major+Gender, data=mydata)
```

### Stata

```stata
tab gender /*Frequencies*/
tab major

tab major gender, col row  /*Crosstabs*/
tab major gender, col row chi2 V

/*Crosstabs which chi2 (The null hypothesis (Ho) is that there is no relationship) and V (measure of association goes from 0 to 1)*/
bysort studentstatus: tab gender major, colum row
```
### R

```r
install.packages("pastecs")
library(pastecs)
stat.desc(mydata)
stat.desc(mydata[,c("Age","SAT","Score","Height", "Read")])
stat.desc(mydata[,c("Age","SAT","Score")],
basic=TRUE, desc=TRUE, norm=TRUE, p=0.95)
stat.desc(mydata[10:14], basic=TRUE, desc=TRUE, norm=TRUE, p=0.95)

# Selecting the first 30 observations and first 14 variables
mydata2 <- mydata2[1:30,1:14]

# Selection using the --subset--
mydata3 <- subset(mydata2, Age >= 20 & Age <= 30)
mydata4 <- subset(mydata2, Age >= 20 & Age <= 30, select=c(ID, First, Last, Age))
mydata5 <- subset(mydata2, Gender=="Female" & Status=="Graduate" & Age >= 30)
mydata6 <- subset(mydata2, Gender=="Female" & Status=="Graduate" & Age == 30)
```

### Stata

```stata
summarize /*N, mean, sd, min, max*/
summarize, detail /*N, mean, sd, min, max, variance, skewness, kurtosis, percentiles*/
summarize age, detail
summarize sat, detail

------------------------------
tabstat age sat score heightin read /*Gives the mean only*/
tabstat age sat score heightin read, statistics(n, mean, median, sd, var, min, max)
tabstat age sat score heightin read, by(gender)
tabstat age sat score heightin read, statistics(mean, median) by(gender)
/*Type help tabstat for a list of all statistics*/

------------------------------
table gender, contents(freq mean age mean score)
tab gender major, sum(sat) /*Categorical and continuous*/
bysort studentstatus: tab gender major, sum(sat)
```
### R

**Descriptive Statistics**

<table>
<thead>
<tr>
<th>R Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean(mydata)</td>
<td>Mean of all numeric variables, same using --sapply--('s' for simplify)</td>
</tr>
<tr>
<td>mean(mydata$SAT)</td>
<td>Mean with simplification</td>
</tr>
<tr>
<td>median(mydata$SAT)</td>
<td>Median of SAT scores</td>
</tr>
<tr>
<td>table(mydata$Country)</td>
<td>Mode by frequencies -&gt; max(table(mydata$Country)) / names(sort(-table(mydata$Country)))[1]</td>
</tr>
<tr>
<td>var(mydata$SAT)</td>
<td>Variance</td>
</tr>
<tr>
<td>sd(mydata$SAT)</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>max(mydata$SAT)</td>
<td>Max value</td>
</tr>
<tr>
<td>min(mydata$SAT)</td>
<td>Min value</td>
</tr>
<tr>
<td>range(mydata$SAT)</td>
<td>Range</td>
</tr>
<tr>
<td>quantile(mydata$SAT)</td>
<td>Quantile for SAT scores</td>
</tr>
<tr>
<td>fivenum(mydata$SAT)</td>
<td>Boxplot elements. From help: &quot;Returns Tukey's five number summary (minimum, lower-hinge, median, upper-hinge, maximum) for the input data ~ boxplot&quot;</td>
</tr>
<tr>
<td>length(mydata$SAT)</td>
<td>Num of observations when a variable is specify</td>
</tr>
<tr>
<td>length(mydata)</td>
<td>Number of variables when a dataset is specify</td>
</tr>
<tr>
<td>which.max(mydata$SAT)</td>
<td>From help: &quot;Determines the location, i.e., index of the (first) minimum or maximum of a numeric vector&quot;</td>
</tr>
<tr>
<td>which.min(mydata$SAT)</td>
<td>From help: &quot;Determines the location, i.e., index of the (first) minimum or maximum of a numeric vector&quot;</td>
</tr>
<tr>
<td>stderr &lt;- function(x) sqrt(var(x)/length(x))</td>
<td>Standard error of the mean</td>
</tr>
<tr>
<td>incster &lt;- tapply(incomes, statef, stderr)</td>
<td>Incentiles for SAT scores</td>
</tr>
</tbody>
</table>

### Stata

<table>
<thead>
<tr>
<th>Stata Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>summarize</td>
<td>/<em>N, mean, sd, min, max</em>/</td>
</tr>
<tr>
<td>summarize, detail</td>
<td>/<em>N, mean, sd, min, max, variance, skewness, kurtosis, percentiles</em>/</td>
</tr>
<tr>
<td>summarize age, detail</td>
<td></td>
</tr>
<tr>
<td>summarize sat, detail</td>
<td></td>
</tr>
<tr>
<td>tabstat age sat score heightin read</td>
<td>/<em>Gives the mean only</em>/</td>
</tr>
<tr>
<td>tabstat age sat score heightin read, statistics(n, mean, median, sd, var, min, max)</td>
<td></td>
</tr>
<tr>
<td>/<em>Type help tabstat for a list of all statistics</em>/</td>
<td></td>
</tr>
<tr>
<td>tabstat age sat score heightin read, by(gender)</td>
<td></td>
</tr>
<tr>
<td>tabstat age sat score heightin read, statistics(mean, median) by(gender)</td>
<td></td>
</tr>
<tr>
<td>table gender, contents(freq mean age mean score)</td>
<td></td>
</tr>
<tr>
<td>tab gender major, sum(sat)</td>
<td>/<em>Categorical and continuous</em>/</td>
</tr>
<tr>
<td>bysort studentstatus: tab gender major, sum(sat)</td>
<td></td>
</tr>
</tbody>
</table>
### R

```r
# Descriptive statistics by groups using --tapply--
mean <- tapply(mydata$SAT, mydata$Gender, mean)
# Add na.rm=TRUE to remove missing values in the estimation
sd <- tapply(mydata$SAT, mydata$Gender, sd)
median <- tapply(mydata$SAT, mydata$Gender, median)
max <- tapply(mydata$SAT, mydata$Gender, max)
cbind(mean, median, sd, max)
round(cbind(mean, median, sd, max), digits=1)
t1 <- round(cbind(mean, median, sd, max), digits=1)

t1
```

### Stata

```stata
summarize /*N, mean, sd, min, max*/
summarize, detail /*N, mean, sd, min, max, variance, skewness, kurtosis, percentiles*/
summarize age, detail
summarize sat, detail
tabstat age sat score heightin read /*Gives the mean only*/
tabstat age sat score heightin read, statistics(mean, median) by(gender)
/table gender, contents(freq mean age mean score)
tab gender major, sum(sat) /*Categorical and continuous*/
bysort studentstatus: tab gender major, sum(sat)
```

**Descriptive Statistics**

- **R**
  - `mean <- tapply(mydata$SAT, mydata$Gender, mean)`
  - `sd <- tapply(mydata$SAT, mydata$Gender, sd)`
  - `median <- tapply(mydata$SAT, mydata$Gender, median)`
  - `max <- tapply(mydata$SAT, mydata$Gender, max)`
  - `cbind(mean, median, sd, max)`
  - `round(cbind(mean, median, sd, max), digits=1)`
  - `t1 <- round(cbind(mean, median, sd, max), digits=1)`

- **Stata**
  - `summarize /*N, mean, sd, min, max*/`
  - `summarize, detail /*N, mean, sd, min, max, variance, skewness, kurtosis, percentiles*/`
  - `summarize age, detail`
  - `summarize sat, detail`
  - `tabstat age sat score heightin read /*Gives the mean only*/`
  - `tabstat age sat score heightin read, statistics(mean, median) by(gender)`
  - `table gender, contents(freq mean age mean score)`
  - `tab gender major, sum(sat) /*Categorical and continuous*/`
## R

```r
library(car)
head(Prestige)
hist(Prestige$income)
hist(Prestige$income, col="green")
with(Prestige, hist(income)) # Histogram of income with a nicer title.
with(Prestige, hist(income, breaks="FD", col="green")) # Applying Freedman/Diaconis rule
p.120 ("Algorithm that chooses bin widths and locations automatically, based on the sample size and the spread of the data"
box()
hist(Prestige$income, breaks="FD")
# Conditional histograms
par(mfrow=c(1, 2))
hist(mydata$SAT[mydata$Gender=="Female"],
    breaks="FD", main="Female",
    xlab="SAT", col="green")
hist(mydata$SAT[mydata$Gender=="Male"],
    breaks="FD", main="Male",
    xlab="SAT", col="green")
    # Braces indicate a compound command allowing several commands with 'with' command
par(mfrow=c(1, 1))
with(Prestige, {
    hist(income, breaks="FD",
    freq=FALSE, col="green")
lines(density(income), lwd=2)
lines(density(income, adjust=0.5), lwd=1)
rug(income)
})
```

## Stata

```
hist sat
hist sat, normal
hist sat, by(gender)
```
<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td># Histograms overlaid</td>
<td></td>
</tr>
<tr>
<td>hist(mydata$SAT, breaks=&quot;FD&quot;, col=&quot;green&quot;)</td>
<td>hist sat</td>
</tr>
<tr>
<td>hist(mydata$SAT[mydata$Gender==&quot;Male&quot;],</td>
<td>hist sat, normal</td>
</tr>
<tr>
<td>breaks=&quot;FD&quot;, col=&quot;gray&quot;,</td>
<td>hist sat, by(gender)</td>
</tr>
<tr>
<td>add=TRUE)</td>
<td></td>
</tr>
<tr>
<td>legend(&quot;topright&quot;, c(&quot;Female&quot;,&quot;Male&quot;),</td>
<td></td>
</tr>
<tr>
<td>fill=c(&quot;green&quot;,&quot;gray&quot;))</td>
<td></td>
</tr>
</tbody>
</table>
Scatterplots

# Scatterplots. Useful to 1) study the mean and variance functions in the regression of y on x p.128; 2) to identify outliers and leverage points.

# plot(x,y)

```r
plot(mydata$SAT) # Index plot
plot(mydata$Age, mydata$SAT)
plot(mydata$Age, mydata$SAT, main="Age/SAT",
     xlab="Age", ylab="SAT", col="red")
abline(lm(mydata$SAT~mydata$Age), col="blue")
# regression line (y~x)
lines(lowess(mydata$Age, mydata$SAT), col="green")
# lowess line (x,y)
identify(mydata$Age, mydata$SAT, row.names(mydata))
```

# On row.names to identify. "All data frames have a row names attribute, a character vector of length the number of rows with no duplicates nor missing values." (source link below).


```r
mydata$Names <- paste(mydata$Last, mydata$First)
row.names(mydata) <- mydata$Names
plot(mydata$SAT, mydata$Age)
identify(mydata$SAT, mydata$Age, 
     row.names(mydata))
```

twoway scatter y x
twoway scatter sat age, title("Figure 1. SAT/Age")
twoway scatter sat age, mlabel(last)
twoway scatter sat age, mlabel(last) || lfit sat age
twoway scatter sat age, mlabel(last) || lfit sat age || lowess sat age /* locally weighted scatterplot smoothing */
twoway scatter sat age, mlabel(last) || lfit sat age, yline(1800) xline(30)
twoway scatter sat age, mlabel(last) by(major, total)
twoway scatter sat age, mlabel(last) by(major, total) || lfit sat age

/* Adding confidence intervals */
twoway (lfitci sat age) || (scatter sat age)
    /*Reverse order shaded area cover dots*/
twoway (lfitci sat age) || (scatter sat age, mlabel(last))
twoway (lfitci sat age) || (scatter sat age, mlabel(last)), title("SAT scores by age") ytitle("SAT")
twoway scatter sat age, mlabel(last) by(gender, total)
## Scatterplots

# Rule on span for lowess, big sample smaller (~0.3), small sample bigger (~0.7)

```r
library(car)
scatterplot(SAT~Age, data=mydata)
scatterplot(SAT~Age, id.method="identify", data=mydata)
scatterplot(SAT~Age, id.method="identify", boxplots= FALSE, data=mydata)
scatterplot(prestige~income, span=0.6, lwd=3, id.n=4, data=Prestige)
# By groups
scatterplot(SAT~Age|Gender, data=mydata)
scatterplot(SAT~Age|Gender, id.method="identify", data=mydata)
scatterplot(prestige~income|type, boxplots=FALSE, span=0.75, data=Prestige)
scatterplot(prestige~income|type, boxplots=FALSE, span=0.75, col=gray(c(0,0.5,0.7)), data=Prestige)
```

```stata
twoway scatter y x
twoway scatter sat age, title("Figure 1. SAT/Age")
twoway scatter sat age, mlabel(last)
twoway scatter sat age, mlabel(last) || lfit sat age
twoway scatter sat age, mlabel(last) || lfit sat age || lowess sat age /* locally weighted scatterplot smoothing */
twoway scatter sat age, mlabel(last) || lfit sat age, yline(1800) xline(30)
twoway scatter sat age, mlabel(last) by(major, total)
twoway scatter sat age, mlabel(last) by(major, total) || lfit sat age */
/* Adding confidence intervals */
twoway (lfitci sat age) || (scatter sat age) /*Reverse order shaded area cover dots*/
twoway (lfitci sat age) || (scatter sat age, mlabel(last))
twoway (lfitci sat age) || (scatter sat age, mlabel(last)), title("SAT scores by age") ytitle("SAT")
twoway scatter sat age, mlabel(last) by(gender, total)
```
### Scatterplots (multiple)

```r
scatterplotMatrix(~ prestige + income + education + women, span=0.7, id.n=0, data=Prestige)
pairs(Prestige)  # Pariwise plots. Scatterplots of all variables in the dataset
pairs(Prestige, gap=0, cex.labels=0.9)  # gap controls the space between subplot and cex.labels the font size (Dalgaard:186)
```

```stata
graph matrix  sat age score heightin read
graph matrix  sat age score heightin read, half
```

### 3D Scatterplots

```r
library(car)

scatter3d(prestige ~ income + education, id.n=3, data=Duncan)
```
### Scatterplots (for categorical data)

<table>
<thead>
<tr>
<th>R</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot(vocabulary ~ education, data=Vocab)</code></td>
<td>/<em>Categorical data using mydata.dat and the jitter option</em>/</td>
</tr>
<tr>
<td><code>plot(jitter(vocabulary) ~ jitter(education), data=Vocab)</code></td>
<td>/<em>&quot;scatter will add spherical random noise to your data before plotting if you specify jitter(#), where # represents the size of the noise as a percentage of the graphical area. This can be useful for creating graphs of categorical data when the data not jittered, many of the points would be on top of each other, making it impossible to tell whether the plotted point represented one or 1,000 observations.” Source: Stata’s help page, type: help scatter</em>/</td>
</tr>
<tr>
<td><code>plot(jitter(vocabulary, factor=2) ~ jitter(education, factor=2), data=Vocab)</code></td>
<td>/<em>Use mydata.dat</em>/</td>
</tr>
<tr>
<td><code># cex makes the point half the size, p. 134</code></td>
<td></td>
</tr>
<tr>
<td><code>plot(jitter(vocabulary, factor=2) ~ jitter(education, factor=2), col=&quot;gray&quot;, cex=0.5, data=Vocab)</code></td>
<td></td>
</tr>
<tr>
<td><code>with(Vocab, {</code></td>
<td></td>
</tr>
<tr>
<td><code>  abline(lm(vocabulary ~ education), lwd=3, lty=&quot;dashed&quot;)</code></td>
<td></td>
</tr>
<tr>
<td><code>  lines(lowess(education, vocabulary, f=0.2), lwd=3)</code></td>
<td></td>
</tr>
<tr>
<td><code>})</code></td>
<td></td>
</tr>
<tr>
<td><code>graph matrix  y x1 x2 x3</code></td>
<td></td>
</tr>
<tr>
<td><code>scatter y x1, jitter(7) title(xyz)</code></td>
<td></td>
</tr>
<tr>
<td><code>scatter y x1, jitter(7) msize(0.5)</code></td>
<td></td>
</tr>
<tr>
<td><code>scatter y x1, jitter(13) msize(0.5)</code></td>
<td></td>
</tr>
<tr>
<td>`twoway scatter y x1, jitter(13) msize(0.5)</td>
<td></td>
</tr>
<tr>
<td><code>graph matrix  x1 x1 x2 x3, jitter(5)</code></td>
<td></td>
</tr>
<tr>
<td><code>graph matrix  y x1 x2 x3, jitter(13) msize(0.5)</code></td>
<td></td>
</tr>
<tr>
<td><code>graph matrix  y x1 x2 x3, jitter(13) msize(0.5) half</code></td>
<td></td>
</tr>
</tbody>
</table>
**From Stata 13+ to R**

Library `foreign` can only read Stata 12 or older.

To read Stata 13 or newer versions into R, one option is to use package `-readstata13-` (see also package `-haven-`). To install it type:

```
install.packages("readstata13")
```

Then load it, type:

```
library(readstata13)
```

**Use the function** `read.dta13()`:

```
mydata <- read.dta13("stata13file.dta")
```

You can convert the file back to Stata (a version that Stata 9-12 can read) by using the function `write.dta()` in package `-foreign-`:

```
library(foreign)
write.dta(mydata, file="mydata.dta")
```
References/Useful links

- DSS Online Training Section http://dss.princeton.edu/training/
- Princeton DSS Libguides http://libguides.princeton.edu/dss
- John Fox’s site http://socserv.mcmaster.ca/jfox/
- Quick-R http://www.statmethods.net/
- UCLA Resources to learn and use R http://www.ats.ucla.edu/stat/R/
- UCLA Resources to learn and use Stata http://www.ats.ucla.edu/stat/stata/
- DSS - Stata http://dss/online_help/stats_packages/stata/
- DSS - R http://dss.princeton.edu/online_help/stats_packages/r
References/Recommended books

- Data Manipulation with R / Phil Spector, Springer, 2008
- Statistics with Stata (updated for version 9) / Lawrence Hamilton, Thomson Books/Cole, 2006