Key Rcmdr commands: Basics

Below are the key commands (using the pull down menus) used in the R lab. You should note the "script" R generates from these point-and-click actions. Using the script allows you to replicate everything you did as well as document the steps taken in an analysis. The point and click actions allow you to begin learning the R code/script. Once you begin to get familiar with the code/script you can write your own files to do a lot of work and then simply run all the code at once. Note that the script used within Rcmdr may differ slightly from what you might run in an R window because Rcmdr loads several packages with it. Thus some syntax may actually be calling a command from one of these packages, rather calling a native R command. Rcmdr is a good tool for using with undergrads, and for learning R syntax yourself, but it should not be relied upon for more than very basic analyses. Also note that just because Rcmdr doesn't include a function, does not mean R can't do it; you just need to find out the R syntax to run more complex tasks. You can include this syntax or code within the Rcmdr script window, and highlight and submit it yourself.

Key facts to note before using R:

- R is case sensitive: Test, test, TEST are all treated as different variable names
- R uses <- as the assign operator. It can go either direction (i.e. x <- 4+5 or 4+5 -> x)
- R stored data in active memory, so large datasets require large amounts of RAM. If you add a lot of objects in the course of your analysis, each additional object takes more memory, thus more RAM. Results for all models run are stored as objects, not just temporary output on the screen. The command **ls()** will display all objects in memory.
- There are many, many R tutorials for beginners but this one gives a good overview of what R is, how it works and makes clear how it is a bit different from SPSS, Stata and some other statistical software packages
    - http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- Another very helpful resource for learning R is the website Quick R : http://www.statmethods.net/ which is geared to introduce R to SAS, SPSS, and Stata users.
- Google is an R learners best friend.

Key tasks	Key actions
Data Management	
Import dataset from file	**Data—Import Data**
Identify cases by name or ID variable	**Data—Active Data Set—Set Case names**
Convert categorical variables whose categories identified by numbers into factors (not numeric variables)	**Data—Manage variables…—Convert numeric variables to factors**
Make a quantitative variable into a categorical one (basic)	**Data—Manage variables…—Bin numeric variable**
Make a quantitative variable into a categorical one (more complex)	**Data—Manage variables…—Recode…** *NB: You need to know a little about R syntax for recoding here to use in the recode instructions. QuickR website, section in managing data is good for this. E.g. may want to recode a quant. Variable into quintiles, so you need to know how to tell R this.*
Recoding a variable (in general)	**Data—Manage variables…—Recode…** *See NB above*
Generating a new variable	**Data—Manage variables…—Compute new variable…** *Again, nee dot know R syntax if doing complex manipulation to generate new variable.*
Remove cases with missing data	**Data—Active Data set—Remove cases with missing data** *NB: Consider whether to check or uncheck box about using all variables. Using all is listwise deletion, or will delete all rows with missing on ANY variable. Dave E. has script for how to do regressions with an analysis sample that has no missing in the most complex model.*
	See more data management under Data—Active dataset… or Manage…
UNIVARIATE Analysis	
Get 5-number summary plus mean (or freq. counts for factors) for all variables in dataset	**Statistics—Summaries—Active data set**
Get more detailed descriptive statistics for specific variables	**Statistics—Summaries—Numerical summaries** *NB: can summarize by groups if there is a group variable defined as a factor.*

Get more detailed freq. info for factors	**Statistics—Summaries—Frequency distributions**
Identify #missing on each variable	**Statistics—Summaries—Count missing observations**
Generate a histogram for quant. Variable	**Graphs—Histogram**
Generate stem and leaf display for quant. variable	**Graphs—Stem and leaf**
Generate boxplot (box and whisker plot) for quant. variable	**Graphs—Boxplot** *NB: Can generate side-by-side boxplots by a factor here, see also bi/multivariate section*
Generate single variable bar graph for factor	**Graphs—Bar graph**
Generate pie chart for factor	**Graphs—Pie chart**
	While graph window is open, can select Graphs—Save graphs to save to a file. Or can right-click on graphics window and copy or save, as either metafile for Windows paste, or save as .eps for LaTeX inclusion
	See additional tutorials on R language to augment script generated by Rcmdr to adjust for axis labels and graph titles. R graphs are very powerful, but also very complex to move beyond those simple graphs available in Rcmdr.
BI-/MULTI-VARIATE Analysis	
Get full correlation matrix for desired set of variables	**Statistics—Summaries—Correlation matrix**
Generate a two-way crosstab	**Statistics—Contingency tables—Two-way**
Generate a two-way crosstab with 3rd control	**Statistics—Contingency tables—Multi-way**
Graph a grouped bar chart to visualize contingency table results *(this must be hard coded by hand, example code provided)*	**Code for grouped bar plot using Chile data available in the Chile dataset [Data—Data in packages—read data—look for Chile set]:** counts <- table(Chile$vote, Chile$sex) barplot(counts, main="Grouped bar plot: Voting Outcomes by Sex", xlab="Vote", ylab="Frequency", col=c("darkblue", "red"),legend=rownames(counts),beside=TRUE)
Generate boxplot (box and whisker plot) for quant. variable	**Graphs—Boxplot** *NB: Can generate side-by-side boxplots by a factor here,simply select by group option and choose factor*
Generate scatterplot/matrix	**Graphs—Scatterplot/ Graphs—Scatterplot Matrix** *NB: can alter many aspects-but I recommend removing tick for smooth line. Can add tick to Identify points if you have associated case names or ID with observations so as to ID outliers. Can also plot by groups if you have a relevant factor. This uses different symbols and plots separate fit lines by group.*
Run simple regressions with only quant DV and IV	**Statistics—Fit Models—Linear Regression**
Run regressions with a factor variable as IV	**Statistics—Fit Models—Linear Model**
POST-ESTIMATION	
Retrieve results from a previously run model	**Models—Select Active model** *Select the model you want form the list then* **Models—Summarize model**
Get model results with coef. Estimate and CI	**Models—Confidence intervals**
Generate residuals, fitted values and other post-estimation measures	**Models—Add Observation Statisitcs**
SAVING AND EXPORTING	
Save script, output, workspace	**File—Save "X" as** *NB: Saving the R workspace allows you to pick back up exactly where you left off in terms of the datasets, but it doesn't image the command history. Save your scripts*
Re-open a saved R workspace in	**Reopen R, re-load Rcmdr, Data—Load dataset and browse to the R workspace file you**

Rcmdr	**saved. If you have multiple datasets in use, select the one you want to use.** **Re-open the script file saved to get the past history of commands, if useful.** *NB: There are very likely better ways to do this but I don't use this myself so I'm not sure.*
Export a dataset for use in other programs	**Data—Active dataset—Export data** *NB: can also save dataset from this step, just select Save rather than Export*

Some misc. scripts that I use in teaching stats:

More detailed histograms

Say I wanted a histogram with bins of width 20. I know I have values from min=55 to max=397. SO I need to first create a variable I'll call 'bins' and give it the values I want. Then I'll tell R to break the data for the histogram by the variable 'bins'.

First, do this (you can type it on the line in the script window, select that command then hit the "Submit" arrow just at the bottom right of the script window to send a command to R). This command below will generate a sequence of numbers btw 50 and 400, by 20s—these will be the breaks for our histogram bins.

```
bins=seq(50,400,by=20)
```

NOW, submit the following command-assuming you named your jail dataset 'jail':

```
hist(jail$inc, breaks=bins)
```

AND if you want to get super fancy (and learn about the ways to tweak/modify your histograms…)

```
hist(jail$inc, breaks=bins, col="lightblue", xlab="Incarceration rate, per
100,000", ylab="Count", main="Distribution of Incarceration Rates by State,
1985")
```

If I wanted to have separate histograms by South:

```
hist(jail$inc[jail$south==1], breaks=bins, col="lightblue", xlab="Incarceration
rate, per 100,000", ylab="Count", main="Distribution of Incarceration Rates by
State, 1985: Southern states")

hist(jail$inc[jail$south==0], breaks=bins, col="lightblue", xlab="Incarceration
rate, per 100,000", ylab="Count", main="Distribution of Incarceration Rates by
State, 1985: Non-southern states")
```

using the Lattice library (need to load it first) to make a conditional histogram (this needs some additional code to properly:

```
library(lattice)

histogram(~inc|south, data=jail, type="count", breaks=bins, col="blue",
xlab="Incarceration rate, per 100,000", ylab="Count", main="Distribution of
Incarceration Rates by State, 1985")
```

Adding a normal curve to a histogram

This is a little more difficult, but below is some code that produces the graph that follows.

```
x <- jail$inc
h<-hist(x, breaks=10, col="red", xlab="Incarceration Rate",
   main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```