Zelig for R Cheat Sheet

Launch R

GUI (Windows or RAqua)	Double-click icon
ESS within XEmacs	`M-x R` (Esc, then x, then R)
Terminal	`R`

To quit, type `q()`.

Installing Zelig

Within R, type:

```
source("http://gking.harvard.edu/zelig/install.R")
```

Loading Zelig

Within R, type: `library(Zelig)`

Syntax

R is case-sensitive!

Default R prompt	`>`
Execute a command	Press Return or Enter
Comment rest of line	`#`
Store objects	`<-`
Separate arguments for functions	`,`

Saving objects to disk

To the working directory:

Save objects	`save(x1, x2, file = "object.RData")`
Save workspace	`save.image()`
Save workspace to file	`save.image(file = "May21.RData")`

Common commands

List objects in workspace	`ls()`
Remomve objects from workspacs	`remove(x1, x2)`
Length of a vector	`length(x)`
Dimensions of a matrix or array	`dim(x)`
Names for lists or data frames	`names(x)`
Type of object	`class(x)`
Summary (for most things)	`summary(x)`
Cross tab	`tabular(x)`
Loading packages	`library(PACKAGE)`
Quitting R	`q()`
Batch mode	`source("myfile.R")`

Logical operators

Exactly equals	`==`
Not equal	`!=`
Greater than	`>`
Greater than or equal	`>=`
Less than	`<`
Less than or equal	`<=`
And	`&`
Or	`|`

(Note: `=` is *not* a logical operator!)

Plots

to screen by default. Let `x` and `y` be vectors of length n

Scatterplot	`plot(x, y)`
Line plot	`plot(x, y, type = "l")`
Add points	`points(x, y)`
Add a line	`lines(x, y)`
Histogram	`hist(x)`
Kernel density plot	`plot(density(x))`
Contour plot	`contour(x, y)`

Plot options

Separate options with commas:

Title	`main = "My Title"`
X-axis label	`xlab = "Independent Var"`
Y-axis label	`ylab = "Dependent Var"`
X-axis limits	`xlim = c(0, 10)`
Y-axis limits	`ylim = c(0, 1)`
Colors	`color = "red"` or `color = c("red", "blue")`

Saving plots

Export plots as .pdf or .eps files

Call the file to which you will store the plot:

- For .eps files:

```
ps.options(family = c("Times"), pointsize = 12)
postscript(file = "mygraph.eps", horizontal = FALSE,
  paper = "special", width = 6.25, height = 4)
```

- For .pdf files:

```
pdf(file = "mygraph.pdf", width = 6.25, height = 4,
  family = "Times", pointsize = 12)
```

Draw your plot (it won't display to screen).

Close and save the file using `dev.off()`.

Math operations `+ - \ *`

For vectors or arrays of the same dimension, R performs math operations on each (i) or (i, j) or $(i, j, \ldots, n)$ element with its corresponding element in the other vector or array.

Matrix operations

Transpose	`t(P)`
Inverse	`solve(P)`
Matrix multiplication	`P %*% Q`

Data structures (or R objects)

R stores all objects in the *workspace* (or RAM). You can store all types of objects – at the same time.

Scalars

Store a scalar value using `<-` (e.g., `a <- 5`)

Numeric	`1, 3.1416, NA, NaN, -Inf, Inf`
Logical	`TRUE, FALSE`
Character	`"Alpha", "beta"`

(Character values are always enclosed in quotes.)

Arrays

An array can have one dimension (a vector), two dimensions (a matrix), or $n > 2$ dimensions. Arrays hold one type of scalar value.

Vectors:

1D arrays with undefined length.

Vector (undefined type/length)	`var <- array()`
Integer vector	`var <- 10:20`
Numerical vector	`var <- c(2, 13, 44)`
Numerical vector	`var <- seq(5, 10, by = 0.5)`
Character vector	`var <- rep("file", 15)`

Use `c()` and `rep()` with numeric, character, and logical values.

To generate a **logical vector** (`TRUE`/`FALSE`), use logical operators compare two vectors of the same length:

```
var1 <- c(1, 3, 5, 7, 9); var2 <- c(1, 2, 3, 4, 5)
logical <- var1 == var2
```

Use logical vectors to recode other vectors or matricies. Putting a logical statement in square brackets extracts only those elements for which the logical statment is `TRUE`:

`var3 <- var1[logical]` or `var3 <- var1[var1 == var2]`

A **factor vector** is a special vector that separates each unique value of the vector into either indicator variables (for unordered factors) or a Helmert contrast matrix (for ordered factors) in regression functions.

Matricies and arrays

2+D arrays have fixed dimensions.

Create a matrix	`mat <- matrix(NA, nrow = 5, ncol = 5)`
	`mat <- cbind(v1, v2)`
	`mat <- rbind(v1, v2)`
Create an array	`arr <- array(NA, dim = c(3, 2, 1), dimnames = list(NULL, NULL, "x1"))`

Extracting or recoding elements in a matrix or array:

Row in a matrix	`mat[5, ]`
Column in a matrix	`mat[ , 5]`
3rd dimension in a 4D array	`arr[ , , 5, ]`

Lists

A list can contain scalars, matricies, and arrays of different types (numeric, logical, factor, and character) at the same time. Lists have a flexible number of elements and can be enlarged on the fly.

Create a list	`ll <- list(a = 5, b = c("in", "out"))`
	`ll <- list(); ll$a <- 5`
Extract list elements	`ll[[6]]`
	`ll$a`
Remove list elements	`ll[[6]] <- NULL`
	`ll$a <- NULL`

Data Frames

A data frame is a list in which every list element has the same length or number of observations. Like a list, each element can be of a different class. Use list *or* matrix operations on a data frame. For a data frame `data`:

View the 5th row	`data[5,]`
Extract the 7th variable	`data[[7]]`
Extract the age variable	`data$age`
Insert a new variable	`data$new <- new.var`

Delimiters

Functions you use	`( )`
Functions you write	`{ }`
N-dimensional arrays	N = 1: `[ ]`
	N = 2: `[ , ]`
	N = 3: `[ , , ]`
Lists	`$` or `[[ ]]`

Loading data

Change directories (using `setwd()`) to the directory that contains your data files *before* attempting to read data into R!

Space- or tab-delimited	`data <- read.table("data.tab")`
Comma-separated values	`data <- read.csv("data.csv")`

Stata `.dta` file	`library(foreign)`
	`data <- read.dta("data.dta")`

SPSS `.sav` file	`library(foreign)`
	`data <- read.spss("data.sav",`
	`to.data.frame = TRUE)`

Options for loading data

- ○ First row = variable names:
  `data <- read.table("data.tab", header = TRUE)`
- ○ Missing values = −9 (for example)
  `data <- read.table("data.tab", na.strings = "-9")`
  (Recodes missing values as R `NA` values.)

(You can combine both options.)

Verifying data integrity

The `data` object is an R `data.frame`, with special properties:

- ○ Each variable (column) has a name:
  - To view names: `names(data)`
  - If names are missing or incorrect, assign correct names
    `names(data) <- c("Y", "X")`
- ○ Observations (rows) may have a name:
  - To view names: `rownames(data)`
  - If names are missing or incorrect, assign correct names
    `rownames(data) <- 1:nrow(data)`
- ○ Display the 5th row: `data[5,]`
- ○ Display the variable `Y`: `data$Y`
- ○ Display a summary of the entire data frame: `summary(data)`

Distributions

For all distribtions, let

`x`, `q` be vectors of quantiles
`p` be a vector of probabilities
`n` be a scalar (the number of random draws).

Uniform

CDF	dunif(x, min = 0, max = 1)
PDF	punif(q, min = 0, max = 1)
Quantiles	qunif(p, min = 0, max = 1)
Random Draws	runif(n, min = 0, max = 1)

Bernoulli

Same as Binomial with `size = 1`.

Binomial

CDF	dbinom(x, size, prob)
PDF	pbinom(q, size, prob)
Quantiles	qbinom(p, size, prob)
Random Draws	rbinom(n, size, prob)

Beta

CDF	dbeta(x, shape1, shape2)
PDF	pbeta(q, shape1, shape2)
Quantiles	qbeta(p, shape1, shape2)
Random Draws	rbeta(n, shape1, shape2)

Poisson

CDF	dpois(x, size, prob)
PDF	ppois(q, size, prob)
Quantiles	qpois(p, size, prob)
Random Draws	rpois(n, size, prob)

Gamma

CDF	dgamma(x, shape, rate = 1, scale = 1/rate)
PDF	pgamma(q, shape, rate = 1, scale = 1/rate)
Quantiles	qgamma(p, shape, rate = 1, scale = 1/rate)
Random Draws	rgamma(n, shape, rate = 1, scale = 1/rate)

Normal

CDF	dnorm(x, mean = 0, sd = 1)
PDF	pnorm(q, mean = 0, sd = 1)
Quantiles	qnorm(p, mean = 0, sd = 1)
Random Draws	rnorm(n, mean = 0, sd = 1)