R cheat sheet (Version 4)

Contents

Preamble

This document is to provide generic syntax for many of the more common R functions and analyses you are likely to use. You should note that this document is a very simple crash course. Where possible, I have stuck with 'Vanilla R' (i.e. you shouldn't need any special libraries). However, when we get to the more advanced models I have had to use syntax from specific third-party libraries.

In terms of style, idenitfier (name) prefixed with 'my" is generic and is not to be taken literately (i.e. Your code shouldn't have these names in it), instead you should adapt the code for your own purposes. Specifically:

- **my.y**, **my.x1** and **my.x2** are three continuous variables with `my.y` assumed to be the outome (endpoint) variable and `my.x1` and `my.x2` assumed to be predictors.
- **my.a** and **my.b** are two categorical variables that are initially just number coded (i.e. R itself doesn't yet know they are categorical)
- **my.a.fac** and **my.b.fac** are the "Factors" correponding to `my.a` and `my.b`. In other words, we have now let R know that these variables are categorical (see code below).
- **my.c.fac.within** is a within-subject effect (e.g. for longitudinal data)
- **my.pat.id** is a random-effect variable that idenitfies the patient (for longitudinal data)
- **myfoo.csv** is some comma delimited text file (can be created by excel) that stores our data
- **mydata.df** is an R data frame that holds all of the above variables

1 Data I/O and manipulation

1.1 Reading in data

There are MANY ways to input data in R. Personally, I like to keep it simple. I tend to just use comma delimited text file, or 'csv' files that I have created in Excel.

READING IN DATA

```
##Read in data from a comma delimited file##

#1. Set working directory (note forwardslash, NOT backslash)
setwd("C:/myrdata")

#2. Read data into mydata.df (dataframe)
mydata.df<-read.csv("somedatafile.csv")
```

Also, see the `foreign` library help to see how to read in files from other statistics packages like Stata or SPSS.

1.2 (very) Basic manipulation of a data frame

Now let's do some very simple manipulation of the data. Note that R is VERY powerful when it comes to data manipulation (subsetting etc). I will only touch on it VERY briefly here. Look up some of the free texts available for R for MUCH more in this area.

Note that R uses the [ROW, COLUMN] that is standard in matrix algebra. i.e. $A_{3,2}$ is a 3 row by 2 column matrix.

BASIC SUBSETTING

```
#1a. New data frame with subset of variables (cols 1 to 3 only)
mynew.df<-mydata.df[,c(1:3)]

#1b. New df with subset of variables (excluding cols 1 to 3)
mynew.df<-mydata.df[,-c(1:3)]

#1c. Alternatively, use the variable names
mynew.df<-mydata.df[,c("age", "sex", "bmi")]

#2a. Now a subset of row (e.g. males only)
mynew.df<-mydata.df[mydata.df$sex==0,]

#2b. Or only patients less than 60 years old
mynew.df<-mydata.df[mydata.df$age<60,]
```

1.3 Creating factors

Factors are useful because they are our way of telling R that we want a variable to be considered as categorical. If a variable is a text variable (e.g. "Male" and "Female"), R automatically makes it a factor. BUT if a variable is coded numerically (e.g. Sex = 0 and 1), we have to let R know that this is not a continuous variable We do this using **factors**. This is also a useful way of providing labels for our variables. Now how do we create these factors.

CREATING FACTORS

```
#1. Create a factor for gender
# (currently coded as 0[males] and 1[females])
sex.fac<-factor(sex, labels=c("Males", "Females"))

#2. Now see if it works
table(sex.fac)
```

2 Data summary

2.1 Summary statistics

2.1.1 Summarizing categorical variables

To tabulate a single categorical variable (frequency table):

FREQUENCY TABLES

```
#1. Generate feq table for A factor
my.tab<-table(mydata.df$mya.fac)
my.tab
```

To cross tabulate two categorical variable

CROSS TABULATIONS

```
#1. Generate X-tab of A by B factor variables
my.tab<-table(mydata.df$mya.fac, mydata.df$myb.fac)
my.tab
```

2.1.2 Desriptive statistics for a continuous variable

Many ways to do this, but I think it is easiest to use the R library `psych` (You may need to download this from CRAN first).

```
#Load psych library into memory
library(psych)

#Generate descriptive stats for the continuous variable 'myx'
describe(mydata.df$myx)

#Generate descriptive stats for 'myx' by groups (e.g.by gender)
describeBy(mydata.df$myx, group=mydata.df$mya.fac)
```

Note the describe function gives you most summary stats (mean, median, sd, IQR, range, min, max, n, nmiss etc)

2.2 Basic graphics

Now for continuous variable (univariate). To get a histogram and a boxplot for a single variable

UNIVARIATE GRAPHICS FOR A CONTINUOUS VARIABLE

```
#1. Histogram of continuous variable
hist(mydata.df$my.y)

#2. Boxplot of continuous variable
boxplot(mydata.df$my.y)
```

and now the bivariate relationship between a continuous outcome in terms of a categorical explanatory (factor)

BIVARIATE CONTINUOUS Y AND CATEGORICAL X

```
#1. Side-by-side boxplot
boxplot(my.y~my.a, data=mydata.df)
```

Now for two continuous variables.

BIVARIATE CONTINUOUS Y AND CONTINUOUS X

```
#1. Scatter plot
plot(x=mydata.df$my.x, y=mydata.df$my.y)
```

3 Classical bivariate tests

Now for the classical bivariate tests. I will consider three:

1. χ^2 test of independance
2. t-tests (both independant and paired)
3. Correlation (Pearson's and Spearman's)

3.1 χ^2 test of independance

χ^2 TEST OF INDEPENDANCE

```
#1. Generate X-tab of A by B factor variables
my.tab<-table(mydata.df$mya.fac, mydata.df$myb.fac)

#2. Input table into the chi.sq test
chisq.test(my.tab)
```

3.2 Independant and paired t-tests

T-TESTS

```
#1. Indpendant t-test
t.test(my.y ~ my.a, data = mydata.df)

#2. Paired t-test
t.test(y1, y2, paired=TRUE, data = mynewdata.df)
#Note data needs to be in wide format for this type of test
```

3.3 Correlation

CORRELATION

```
#1. Pearson correlation
cor(mydata.df$my.x1, mydata.df$my.x2 ,    method ="pearson")
#Note Pearson's is default so you could just write
cor(mydata.df$my.x1,mydata.df$my.x2)

#1b. Above only generates coeffeicent, need to tests using
cor.test(mydata.df$my.x1,mydata.df$my.x2)

2. Spearmans
cor(mydata.df$my.x1, mydata.df$my.x2 ,    method ="spearman")
```

4 Modelling

One of the things you will notice about R is that modelling is very simple. Once you understand the basics of modeling in one situation (e.g. a continous outcome), extending to other types and outcomes and situations is very easy. Modelling in R is based on the concept of a *formula*:

$$y \sim x1 + x2$$

in a linear regression (assumeing $y, x1$ and $x2$ are continuous) implies:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

If this same formula is used (for example) in a Poisson regression (for count data) using a log link then

$$y \sim x1 + x2$$

implies

$$y = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

Regardless, the same basic form of the formula is used throughout all R modelling

4.1 Linear regression and the general linear model

Linear regression and the general linear model use exactly the same model (`lm()`, they only differ in that factors (categorical predictors) will occur in the general linear model

LINEAR REGRESSION AND THE GENERAL LINEAR MODEL

```
#1. A multivariable linear regression
my.model<-lm(my.y~my.x1+my.x2, data=mydata.df)

#1b. Get summary (R-sq and coeffcients) and ANOVA table
summary(my.model)
```

```
anova(my.model)

2. A general linear model (using a temporary factors)
my.model<-lm(my.y~my.x1+as.factor(my.a), data=mydata.df)

2b. ...or defining factor (permanantly) first
mya.fac<-factor(my.a, labels=c("low", "med", "high")
my.model<-lm(my.y~my.x1+mya.fac, data=mydata.df)

2c. Now get summary and ANOVA table
summary(my.model)
anova(my.model)
```

4.2 Logistic regression and other GLMs

The Generalized Linear Model (GLM) in R is a simple extension of the command used for the standard Linear Model (Linear Regression, ANOVA and the General Linear Model). Note that the only REAL difference (as far as coding goes), is the addition of the `family` argument. Here we will use Binary Logistic regression as an example, but we can also fit any of the other GLMs (e.g. Possion regression) using the same command (just change the 'family' argument).

Assume that the outcome variable, `my.a` is a binary outcome variable:

```
#1. A binary logistic regression
my.model<-lm(my.a~my.x1+my.x2, data=mydata.df, family=binomial)

#1b. Get summary (R-sq and coeffcients) and ANOVA table
summary(my.model)
anova(my.model)

#Note I wrote the below function myself-get the R code from me
print.ORCIs.glmm.wald(my.model)
```

4.3 Longitudinal and other correlated outcomes

Longitudinal data (and correlated data) is where we have several *observations* associated with each *observational unit*. For example:

- Longitudianl data: We have several observations (over time) for each patient
- For clustered data (e.g. a multicentre study), we might have many patients associated with each clinic or hospital (and to not account for this in our analysis would render our results invalid).

Here, I will just give an example of a longitudunal analysis (using the patient identifier **my.pat.id**), but the approach used for clustered data (e.g. Hospital ID) is very similar.

4.3.1 Continuous longitudinal or otherwise correlated outcomes

For a continuous outcome variable, we would use a Linear Mixed Model (LMM) to analysis our data. Typically we will have a 'within-subject effect' (e.g. Time), and one or more 'between level effect(s)' (e.g. Treatment, Gender). Between-subject effects are the ones we are used to.

Note that there are several R libraries for LMMs. My prefered library is called `lme4`. Again, the first time you use this, you may have to download from CRAN.

```
library(lme4)

#1. A linear mixed model (Random intercept model)
# Note: As long as you specify the patient ID, R can workout
# the between- and within-subject effects itself
my.model<-lmer(my.y~my.x1+my.c.fac.within + (1|my.pat.id), data=mydata.df)

#1b. Summaries model and coeffcients
summary(my.model)
anova(my.model)
#Note I wrote the below function myself-get the R code from me
print.BetaCI.lmm(my.model)

#2. A linear mixed model (Random coeffcients model)
my.model<-lmer(my.y~my.x1+my.c.fac.within + (my.c.fac.within|my.pat.id), data=mydata.df)

#2b. Summaries model and coeffcients
summary(my.model)
anova(my.model)
print.BetaCI.lmm(my.model)
```

4.3.2 Categorical longitudinal or otherwise correlated outcomes

Just like a standard linear regession (General Linear Model) can be extended (generalized) to a Generalized Linear model, a LMM correspondingly extended to a Generalized Linear Mixed Model. In terms of R, the only real difference is the addition of the `family` argument.

```
library(lme4)

#1. A Generalized linear mixed model (Random intercept model)
# Note: This is a binary logistic mixed effect regression
# Assume that my.a is a binary avriable
my.model<-glmer(my.a~my.x1+my.c.fac.within+(1|my.pat.id), data=mydata.df, family=binomial)

#1b. Summaries model and coeffcients
```

```
summary(my.model)
anova(my.model)
#Note I wrote the below function myself-get the R code from me
print.ORCIs.glmm.wald(my.model)

#Note: As with the LMM we can also fit a Random coeffcients
#version of the GLMM
```