Ben Boyd

R Coding Cheat Sheet

Table of Contents

Basic Commands

```
setwd("~/R Working Directory")
data = read.table("name.txt")
names(data)= c("subject","group","score")
```

Producing a Loop

```
#The command set.seed(123) allows you to get reproducible random numbers when using a loop.

set.seed(123)
sample.data=NA
sample.means=NA
for(counter in 1:100)
{
sample.data=rnorm(10,300,25)
sample.means[counter]=mean(sample.data)
}
#Once the loop has been run, a plot of the produced samples can be made:
plot(sample.means)

#Or any statistical measure could be computed, such as variance:
var(sample.means)
```

Computing Statistical Measures

Mean/Median

```
mean(mydata$eeg[mydata$group==1])
median(mydata$eeg[mydata$group==1])
```

Squared Errors from the Mean

```
data$V4= data$V3*data$V3
```

#If you divide this number by n-1, you have a quantity called variance - the variance is a number that describes the variability in the data.

Determining the Probability of a Mean

```
pnorm(460,500,100)
```

#Using the command pnorm allows you to determine the probability of (in this case) a mean in a distribution. The first number in () is the number in which probability is sought, the second number is the average value (i.e. the mean) of the data set and the third number is the standard deviation of the data set.

Determining Strength of Correlation

```
#Correlation ranges: weak (0.1-0.3), medium (0.3-0.5) and strong (0.5-1.0).
cor(data$V1,data$V2)

#The following command allows you to perform a formal statistical test against the null hypothesis:
cor.test(data$V1,data$V2)
```

Linear Regression

#In regression, we seek to determine whether X can predict Y.

```
model=lm(data$V1~data$V2)
summary(model)
```

Multiple Regressions

#If you wanted to see how the variables x1, x2, x3, x4, x5, and x6 predicted y you would write:

```
results=lm(y~x1+x2+x3+x4+x5+x6)
```

#You are telling R to create a linear model where y is a function of x1, x2, x3, x4, x5, and x6.

Computing Variance the Long Way

```
data$V3 = data$V1 - mean(data$V1)
data$V4= data$V3*data$V3
sum(data$V4)/99
```

Computing Variance the Short Way

```
var(data$V2)
```

Computing Standard Deviation: Long and Short Way

#Mathematically, the standard deviation is just the square root of the variance:

sqrt(var(data$V1)) → Long way

sd(data$V1) → Short way

Comparing the Means of each Condition for a Variable

```
means=aggregate(rt~condition,data3,mean)
```

Computing Difference Scores between Conditions

```
diffscore=mean1-mean2
```

Confidence Intervals

Computing Confidence Intervals

```
df=length(weightdata$weight)-1
```

#This computes the degrees of freedom of the sample, of 39 in this instance.

```
crt=qt(0.05,df)
```

#This computes the critical t value given a 95% confidence range and the afore-computed degrees of freedom.

```
std=sd(weightdata$weight)
```

#The standard deviation of the weight data.

```
sqn=sqrt(length(weightdata$weight))
```

#The square root of the number of data points.

```
ci=(crt*std)/sqn
```

#The confidence interval is equal to the critical t-value times the standard deviation of the data divided by the square root of the number of data points.

#We often use the following for ci to force a positive number.
ci=abs(crt*std/sqn)

std=sd(weightdata$weight[weightdata$time==2])
#Example of how to specify only a single group/condition.

Finding Upper and Lower Ranges of Confidence Intervals

mean(weightdata$weight)-ci
mean(weightdata$weight)+ci

Computing the Confidence Interval for the Difference Scores

#I.e. the difference in weight data between time == 2 and time == 1 for each person.

weightdiff = (weightdata$weight[weightdata$time==1])-(weightdata$weight[weightdata$time==2])

df=length(weightdiff)-1
crt=qt(0.05,df)
std=sd(weightdiff)
sqn=sqrt(length(weightdiff))
ci=abs(crt*std/sqn)

Plotting

plot(mydata$eeg)
#This generates a basic scatter plot of the values in data3$eeg with subject as the x axis value and eeg as the y axis value.
barplot(data3$eeg)
boxplot(data3$eeg
hist(data3$eeg)

plot(data3$rt~data3$condition)
#By using plot(data3$rt~data3$condition), you are telling R that you want the rt list to be plotted as the y-axis and the condition list to be plotted as the x-axis

Side by Side Plots

par(mfrow= c(1,2))
plot(data$V1,ylim=c(250,400))
plot(data$V2,ylim=c(250,400))
#Plotting data with a y limit; same process for setting x limits.

abline(a= mean(mydata$eeg[mydata$group==1]), b= 0, col= "red")
#Specifies a y intercept, a, which is equal to the mean, a slope b = 0 and makes the line colour red.

Overlaying One Histogram on to Another

hist(data6$rt[data6$group==1],col=rgb(1,0,0,0.5),ylim=c(0,25))
hist(data6$rt[data6$group==3],col=rgb(0,0,1,0.5),ylim=c(0,25),add=T)

```
#Above, add=T is the command that overlays one histogram on to the other.
```

Plotting the Mean with Error Bars

```
means=aggregate(data4$rt,list(data4$group),mean)
means=means$x
#This removes the means from the mini data frame they are in.

#Next, find the standard deviations for each group:
sds=aggregate(data4$rt,list(data4$group),sd)
sds=sds$x
#Then, compute confidence intervals for each group:
cis=abs(qt(0.05,49)*sds/sqrt(50))

#Now, plot the barplot of the means:
bp=barplot(means,ylim=c(200,400))
#Note, we are assigning the barplot a name, bp, so we can use it below.

#Finally, add the error bars:
arrows(bp,means-cis,bp,means+cis,angle=90,code=3,length=0.5)
```

Plotting Group Means with 95% confidence Intervals as Error Bars

```
library("gplots")
plotmeans(data$rt~data$group)
```

Stylization of Plots

```
plot(data3$rt~data3$condition,main="Condition Effect",xlab="condition",ylab="reaction time(ms)",ylim=c(0,500))
#If you want to label the plot axes/include a title/set range of y-axis in R: xlab/ylab = x-axis label/y-axis label, main = title of plot/graph

abline(lm(data3$rt~data3$subject))
#Allows a regression line to be added to the scatter plot, lm=linear model.
#The command, abline, adds a straight line to the graph/plot between two points.

points(means$condition,means$rt,pch=16,col='red')
#The command, means$condition, must be before means$rt or whatever variable is being used.
#The command, pch= , is used for plotting special symbols i.e. squares, circles etc.
#E.g. for pch=16, the number 16 refers to a particular character, which for 16 is a filled circle.

#If you wanted to color subsets of the bars, you would have to specify colors for each bar. You could do something like this:
cols=data4$subject
cols[1:50]=c("red")
cols[51:100]=c("blue")
cols[101:150]=c("green")
barplot(data4$rt, col=cols)
```

Data Creation

Creating a Data Set

```
newdata=rnorm(1000,500,10)
```

#This command creates a new data set called newdata which has 1000 scores, a mean of 500, and a standard deviation of 10.

Creating a Data Set with Randomly Distributed Numbers

```
sample.data2=runif(1000,1,1000)
```

#The command, runif, can be used to make a sample of a selected size (i.e. 1000), with randomly distributed numbers between one number (i.e. 1) and another number (i.e. 1000).

Creating New Data Columns/Creating Data Sets

```
newdata1 = data.frame(newdata1)
newdata2 = data.frame(newdata2)
newdata3 = data.frame(newdata3)
```

#The above commands simply create data sets from the created data.

```
alldata=cbind(newdata1,newdata2,newdata3)
```

#The above command combines separate data into one data frame.

Computing Sum of Squared Errors

```
alldata[,4] = (alldata[,1] - mean(alldata[,1]))^2
alldata[,5] = (alldata[,2] - mean(alldata[,2]))^2
alldata[,6] = (alldata[,3] - mean(alldata[,3]))^2
```

#The above commands compute the squared errors for each newdata set. Using the command, alldata[,#], lets you create a new data column (which will by default be labelled V#), the part of the command, =alldata[,#]*alldata[,#] is saying multiply one data column by itself.

#The sum of squared errors is found by finding the mean of each value in the data column, subtracting the mean of the data column from each value and then squaring the deviation that is found for each value. The above is simply combining all these functions into a single command.

T-tests

Single Sample T-Tests

```
analysis1=t.test(caloriedata1$V1,mu=3000)
```

#This command tells R to conduct a single sample t-test of calorie.data1$V1 against a known population mean of 3000. By specifying mu, R will automatically know to treat this as a single sample t-test. To view the results of the single sample t-test:

```
print(analysis1)
```

#Note: for a single sample t-test, the degrees of freedom are: df = n -1.

Paired (Dependent) Sample T-Tests

#In R, we need to first define factors (i.e. independent variables) as factors. We do this as follows:

```
caloriedata2$time=factor(caloriedata2$time)
```

#Once the factor has been set, a paired sample t-test can be conducted:

```
analysis2=t.test(caloriedata2$calories~caloriedata2$time, paired=TRUE)
```

#This command is essentially saying, “Let's run a t-test where we examine caloriedata2$calories as a function of caloriedata2$time and oh, by the way, it’s a paired/dependent samples ttest."

Note: for a dependent samples t-test, the degrees of freedom are: df = n -1.

Independent T-test

```
analysis3=t.test(caloriedata3$calories~caloriedata3$group)
```

#Note: for an independent samples t-test the degrees of freedom are: df = n1 - 1 + n2 - 1.

Finding Means of the Data (or Other Descriptives) for the Actual Conditions

```
mean(caloriedata2$calories[caloriedata2$time==1])
```

#This command is use to find the mean calories for the subjects in the first condition.

```
mean(caloriedata2$calories[caloriedata2$time==2])
```

#Conversely, this command allows you to find the mean calories for the subjects in the second condition.

Analysis of Variance (ANOVA) Assumptions to be considered:

Assumption One: Between Group Independence.

#essentially, your groups cannot be related.

Assumption Two: Within Group Sampling and Independence.

#the members of each groups are sampled randomly and are independent of each other

Assumption Three: Normality.

#the sampling distribution of the mean of the population from which the data is drawn is normally distributed, which approximately means the data for each group are drawn from a normally distributed population. There are three ways to do this:

1 - Plot Histograms for each group and assess this through visual inspection.

```
par(mfcol=c(1, 3))
hist(data$rt[data$group==1])
hist(data$rt[data$group==2])
hist(data$rt[data$group==3])
```

2 - Examine the skew and kurtosis of the data for each group. Skewness assesses how much to the left or right the distribution is pulled (0 is perfectly normal) and kurtosis assesses how flat or peaked the distribution is (3 is perfectly normal).

```
library("moments")
skewness(data$rt[data$group==1])
kurtosis(data$rt[data$group==1])
skewness(data$rt[data$group==2])
kurtosis(data$rt[data$group==2])
```

3 - Use QQ plots to assess normality. On a QQ plot the data is normal if it all falls on a line.

```
qqnorm(data$rt[data$group==1])
qqline(data$rt[data$group==1])
qqnorm(data$rt[data$group==2])
qqline(data$rt[data$group==2])
```

Assumption Four: Homogeneity of Variance.

#ANOVA assumes that the variances of all groups are equivalent. There are three ways to test this:

1 - Plot Histograms and visually ensure a normal distribution.
2 - Compute the variances for each group:

```
var(data$rt[data$group==1])
var(data$rt[data$group==2])
var(data$rt[data$group==3])
```

#A general rule of thumb is that if the smallest variance is within 4 magnitudes of the largest variance then the assumption is met.

3 - Use a statistical test of the assumption i.e. Hartley's rule of thumb or Bartlett's test

```
bartlett.test(data$rt~data$group)
```

Computing an ANOVA

```
data$condition=factor(data$condition)
analysis=aov(data$score~data$condition)
```

#This command tells R to run an analysis of variance (aov) on data$rt with data$group as a factor.

```
Summary(analysis)
```

#Typically, you would report an ANOVA as follows: Our analysis revealed that there was a difference between groups, F(2,147) = 17.58, p < 0.001.

#NOTE: an ANOVA simply tells you whether or not there is a difference between groups, it does not tell you where the difference is. To find where the difference is, a test such as the TukeyHSD can be computed.

Methods for Performing a Post-Hoc Analysis

TukeyHSD

#The TukeyHSD can be used to statistically test whether or not there are differences between specific groups

```
analysis=aov(data$score~data$condition)
TukeyHSD(analysis)
```

Bonferonni Analysis

#Similar to the TukeyHSD test, you could also do pairwise t-tests with a Bonferonni correction by using:

```
pairwise.t.test(data$rt,data$group,p.adjust="bonf")
```

Contrast Analysis

```
summary.lm(analysis)
```

#This command shows you the contrasts of groups 2 and 3 relative to group 1 - the default contrast.

```
#To use a different contrast, you can define it through the contrast command and then re-execute the model.
contrasts(data$group)=contr.treatment(3,base=3)
modelnew=aov(data$rt~data$group)
summary.lm(modelnew)
#The above code would conduct a contrast comparing the first two groups to the third.

contrasts(data$group) = contr.poly(3)
modelnew = aov(data$rt~data$group)
summary.lm(modelnew)

#The above code would conduct a contrast examining the polynomial trends between the groups (linear, quadratic etc).
```

Repeated Measures ANOVA

```
data$subject = factor(data$subject)
data$condition = factor(data$condition)
model = aov(data$data~data$condition+Error(data$subject/data$condition))
summary(model)

#Given the theory behind RM ANOVA, you are removing between participant variance from the overall error variance. The Error term accomplishes that.
```

Post-Hoc Analysis of RM ANOVA

```
#To perform a post-hoc on data from a RM ANOVA, all you need to do is run a series of pairwise comparisons:

pairwise.t.test(data$data,data$condition,paired=TRUE)
```

Computing a Factorial ANOVA

```
#The linear model for a factorial ANOVA has to include both main effects (age, gender) and the interaction between age and gender.

rt=data$rt
age=factor(data$age)
gender=factor(data$gender)
model=aov(rt~age+gender+gender*age)
summary(model)

plot(rt ~ age)
plot(rt ~ gender)
interaction.plot(age,gender,rt)

#To verify the results of the above, you need to do a posthoc analysis of the main effects of age and gender and the interaction.
TukeyHSD(model)

#An even simpler way to post-hoc the ANOVA would be to do the following:
```

Pick a direction: Age or Gender. If you pick Age, then you can run an independent samples t-test at each level of age to see where the difference lies.

```
age1 = subset(data,age==1)
t.test(age1$rt~age1$gender)
age2 = subset(data,age==2)
t.test(age2$rt~age2$gender)
age3 = subset(data,age==3)
t.test(age3$rt~age3$gender)
```

#Then, you can look to see whether the tests are significant, and therefore, whether there is similarity between groups.

Testing the Assumptions of Multiple Regression

Assumption of Independence of Errors

```
library(car)
durbinWatsonTest(results)
```

#You want the D-W value to be between 1 and 3 and as close to 2 as possible.

Assumption of Normality, Linearity, & Homoscedasticity of Residuals

#One simple test of the assumptions in R is to examine the residuals. If the residuals are normally distributed then one can generally assume that all of the assumptions have been met.

```
res=resid(results)
```

#This command puts all of the residuals (the difference between the actual and predicted y values) into a variable called res.

```
hist(res)
```

#If the histogram is normally distributed, then one can assume the assumptions were met.

```
plot(results)
```

#Input the return(results) command until you see Q-Q plot. The Q-Q plot in a multiple regression shows deviations from normality, thus, you want it to be straight.

Assumption of Multi-Collinearity

#To test the Assumption of Multi-Collinearity, you typically examine variance inflation factors(vif).
Three criteria must be met:
1. No VIF above 10 - check with vif(results).
2. The average VIF should be close to 1- check with mean(vif(results))
3. Ideally, the tolerance (1/vif) should be not be less than 0.1, and less than 0.2 may be a problem - check with 1/vif(results)

```
vif(results)
mean(vif(results))
1/vif(results)
```

Testing for Outliers and Influential Cases

#A simple way to examine for multivariate outliers is to compute a Cook's Distance:

```
cooks=cooks.distance(results)
plot(cooks)
```

#If the Cook's Distance test reveals and outlier, you want to remove the outlier and redo the whole test.

Testing the Assumption of Homogeneity of Variance (Independent T-test)

#For an independent samples t-test, the assumption of homogeneity of variance needs to be tested. Hartley's rule of thumb is the easiest way to do this. The Bartlett and Levene's tests are other ways to test this assumption.

Hartley's Rule of Thumb

```
vars=aggregate(caloriedata3$calories,list(caloriedata3$group),var)
vars
```

#Essentially, you are looking to ensure that the variance of one group is no more than four times the variance of the other group (or vice versa).

Bartlett Test

```
bartlett.test(caloriedata3$calories~caloriedata3$group)
```

Levene's Test

```
library(car)
leveneTest(caloriedata3$calories~caloriedata3$group)
```

#The Levene Test is generally considered to be more reliable than the Bartlett test, especially for smaller sample sizes.

The Assumption of Sphericity

#The assumption of sphericity is that the variances of the difference scores are equal. The reality is that in R the test of this assumption is not as easy as one might think. Indeed, it is easier to just run the RM ANOVA using the ezANOVA package.

```
library("ez")
model = ezANOVA(data=data, dv = .(data), wid =.(subject), within =.(condition), detailed = TRUE, type =
3)
model
```

#This will give you a full RM ANOVA table plus the test of sphericity. Note, if you fail the test of sphericity you should use one of the corrected p-values and correct the degrees of freedom. In general, the closer epsilon is to 1 then the more closely the assumption is met. For this design epsilon should have a range between 0.5 and 1. The lower bound of epsilon is defined as 1 / (number of levels - 1). If epsilon is closer to the lower bound than 1 then the assumption is definitely not met.