

Using the R Statistical Computer Program

R is a powerful statistical computer program that is freely available under the General Public License (GPL). It runs under Unix and Linux, Microsoft Windows and Macintosh OS 9 and X operating systems. Download your own copy here: <http://www.r-project.org/>. To get started, go here: <http://www.statmethods.net/>. Below are R commands (in **bold**) used to solve some of the math skills questions that are on the syllabus. The hash marks (#) are comments to clarify what is going on.

```
# Question 3:
x <- c(10, 9, 12, 11, 8.5, 13, 8, 10, 7, 11.5) # create data vector
mean(x) # compute the mean of the numbers in x
sd(x) # compute the standard deviation
# -----
# Question 4:
obs <- c(174, 172, 104, 92, 41, 8) # observed data
prd <- c(175.5, 167.8, 106.5, 90.4, 44.3, 6.5) # predicted data
p <- prd/sum(prd) # predicted frequencies to probabilities
chisq.test(obs, p=p) # do the chi-square test
# -----
# Question 5:
# There are two ways to test the hypothesis: t-test or ANOVA
# They will give exactly the same results:  $t^2 = F$ 
# Step 1: make a data frame with three columns
# Make the subject, levels, and dependent variable vectors and
# Now assemble the factors and data into a data frame
sj <- factor(c("S01", "S02", "S03", "S04", "S05", "S06", "S07", "S08", "S09", "S10"))
iv <- factor(rep(1:2, each = 5)) # indep factor with 2 levels
dv <- c(8.0, 9.0, 7.5, 7.0, 8.5, 10.0, 9.5, 11.0, 9.0, 10.5) # dep var
df <- data.frame(sj, iv, dv)
# Step 2: compute a t-test for two independent groups
with(df, t.test(dv[iv == 1], dv[iv == 2], paired = FALSE))
# Step 3: compute and print the ANOVA comparing the two levels of
# the independent variable (iv)
summary(aov(dv ~ iv, data = df))
# Step 4: print summary table in nice format
xbar <- tapply(dv, iv, mean) # holds the means
sdev <- tapply(dv, iv, sd) # holds the standard deviations
numb <- tapply(dv, iv, length) # holds the number of samples
cbind(mean=xbar, std.dev=sdev, n=numb)
# -----
# Question 6:
qnorm(0.8413447) # convert probability to z-score
1 - pnorm(1.959964) # convert z-score to probability
pnorm(1.959964, lower.tail = FALSE) # another way to get upper tail
# -----
# Questions 7 & 8:
x <- c(1.0, 3.0, 5.0, 7.0, 9.0) # x data vector
y <- c(4.1, 9.9, 16.1, 22, 27.9) # y data vector
df <- data.frame(x, y) # put x and y vectors into a data frame
reg <- lm(y ~ x, data = df) # compute the regression
summary(reg) # prints summary of the regression
plot(df) # plots graph of data
abline(reg) # plots the regression line
# -----
```