Top 5 x 5 R Tricks

Shaun S. Wulff

04/21/17

This talk will present 5 of the top R tricks in 5 areas as selected by an R *convert* (as well as a few added surprises). These areas include: (1) the R environment, (2) data functions, (3) math functions, (4) statistics functions, (5) graphics. The talk is designed for all (statistical) R programming levels However, many of the tricks are neither novel nor likely to be unknown to R fanatics and students in R based courses. Nevertheless, it is hoped that the presentation and resulting discussions will benefit all in attendance and allow us to express our affections for use R!

# R Console 3.0.2

**1. Environment**  
  
***(a) Online Help***

Search "R Tricks"

# \_rseek.org\_ # search engine for R #   
# http://stackoverflow.com/questions/1295955/what-is-the-most-useful-r-trick  
# http://adv-r.had.co.nz/ #  
# https://www.quora.com/What-are-some-good-hacks-at-using-R #

***(b) Reproducible Research***

# https://yihui.name/knitr/demo/minimal/  
# http://www.stat.cmu.edu/~cshalizi/rmarkdown/#math-in-r-markdown #   
# https://github.com/rstudio/rmarkdown/issues/103  
# library(sweave)  
# library(knitr)  
# library(rmarkdown) # html\_document,pdf\_document #   
# rmarkdown::render("F:/R/r\_tricks2.R",output\_format="word\_document")

***(c) Source***

source("F:/R/r\_tricks.R") # read trick functions #

***(d) Flexbile Variable Assignments***

# http://stackoverflow.com/questions/2271575/whats-the-difference-between-and-in-r #

x <- 5; w = y = 6; 2 -> z;  
w+x+y+z

## [1] 19

***(e) loops***

u = 0; start = proc.time()  
for (i in 1:10000) {  
 u = u+1 # \*\* cat \*\* #  
 if (u/1000==floor(u/1000)) { cat(i,sep="\n"); u = 0; } }

## 1000  
## 2000  
## 3000  
## 4000  
## 5000  
## 6000  
## 7000  
## 8000  
## 9000  
## 10000

proc.time() - start # \*\* proc.time \*\* #

## user system elapsed   
## 0.04 0.00 0.04

**2. Data Functions**  
  
***(a) scan***

# scan with tab delimiter ? #  
# http://stats.idre.ucla.edu/r/faq/how-to-input-data-into-r/ #  
# http://grokbase.com/t/r/r-help/14as7n1x0p/r-r-markdown-and-scan #

data.frame(scan(text = "  
12 bobby  
24 kate  
35 david  
20 michael",what=list(age=0,name="")))

## age name  
## 1 12 bobby  
## 2 24 kate  
## 3 35 david  
## 4 20 michael

***(b) read other data files formats***

# http://www.statmethods.net/input/importingdata.html #

library(xlsx)  
file <- system.file("tests","test\_import.xlsx",package="xlsx")  
res <- read.xlsx(file,1) # read first sheet #  
head(res[,-8]) # \*\* head \*\* #

## NA. Population Income Illiteracy Life.Exp Murder HS.Grad Area  
## 1 Alabama 3615 3624 2.1 69.05 15.1 41.3 50708  
## 2 Alaska 365 6315 1.5 69.31 11.3 66.7 566432  
## 3 Arizona 2212 4530 1.8 70.55 7.8 58.1 113417  
## 4 Arkansas 2110 3378 1.9 70.66 10.1 39.9 51945  
## 5 California 21198 5114 1.1 71.71 10.3 62.6 156361  
## 6 Colorado 2541 4884 0.7 72.06 6.8 63.9 103766

***(c) apply***

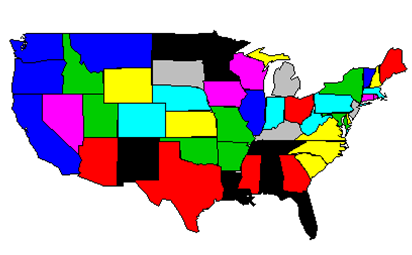
# require (2)(b) #  
n.res = res[,-1]  
sum.o = cbind(means=apply(n.res,2,mean),medians=apply(n.res,2,median),

sds=apply(n.res,2,sd))  
options(scipen=99) # \*\* scipen \*\* #   
print(sum.o,digits=4) # \*\* print \*\* #

## means medians sds  
## Population 4246.420 2838.50 4464.4914  
## Income 4435.800 4519.00 614.4699  
## Illiteracy 1.170 0.95 0.6095  
## Life.Exp 70.879 70.67 1.3424  
## Murder 7.378 6.85 3.6915  
## HS.Grad 53.108 53.25 8.0770  
## Frost 104.460 114.50 51.9808  
## Area 70735.880 54277.00 85327.2996

***(d) by***

# require (2)(b) #  
library(maps)  
map("state",fill=TRUE,col=palette())



west = res[,1] %in% c("California","Oregon","Washington","Idaho","Nevada","Arizona","Utah","Montana","Wyoming","Colorado","New Mexico")  
apply(n.res[,-7],2,function(x) by(x,as.factor(west),mean))

## Population Income Illiteracy Life.Exp Murder HS.Grad Area  
## FALSE 4503.974 4408.564 1.246154 70.78949 7.502564 50.71795 60554.67  
## TRUE 3333.273 4532.364 0.900000 71.19455 6.936364 61.58182 106832.91

***(e) paste***

qux = c("awesome","R","is","!","totally"); qux

## [1] "awesome" "R" "is" "!" "totally"

qux = qux[c(2,3,5,1,4)];   
paste(qux,collapse="\*\*\*")

## [1] "R\*\*\*is\*\*\*totally\*\*\*awesome\*\*\*!"

noquote(paste(qux,collapse=" ")) # \*\*noquote\*\* #

## [1] R is totally awesome !

**3. Math Functions**  
  
**(a) %x%**  
  
The kronecker product operator .

# https://www.zoology.ubc.ca/~schluter/R/data/ #  
# http://www.cookbook-r.com/Manipulating\_data/Renaming\_levels\_of\_a\_factor/ #

A = rep(1,4)%x%c(1,2,3)  
B = rep(1,3)%x%c(1,2)%x%c(1,1)  
dat = cbind(A,B)  
A = as.factor(A) # \*\* as.factor \*\* #   
levels(A)[levels(A)==c(1,2,3)] = c("low","med","high")  
dat = list(f1=A,f2=B)  
dat = data.frame(f1=A,f2=B)  
dat

## f1 f2  
## 1 low 1  
## 2 med 1  
## 3 high 2  
## 4 low 2  
## 5 med 1  
## 6 high 1  
## 7 low 2  
## 8 med 2  
## 9 high 1  
## 10 low 1  
## 11 med 2  
## 12 high 2

***(b) all.equal***  
  
Can be used to easily check equality of expressions or matrices.

d45 <- pi\*(1/4 + 1:10)  
test <- rep(1,10)  
cbind(tan(d45),test)

## test  
## [1,] 1 1  
## [2,] 1 1  
## [3,] 1 1  
## [4,] 1 1  
## [5,] 1 1  
## [6,] 1 1  
## [7,] 1 1  
## [8,] 1 1  
## [9,] 1 1  
## [10,] 1 1

all.equal(tan(d45),test) # TRUE #

## [1] TRUE

all(tan(d45) == test) # FALSE #

## [1] FALSE

all.equal(tan(d45),test,tol=0) # Difference #

## [1] "Mean relative difference: 0.00000000000000129526"

***(c) array***  
  
Generalization of a matrix, say .

vector1 <- c(5,9,3)  
vector2 <- c(10,11,12,13,14,15)  
column.names <- c("COL1","COL2","COL3") # \*\* column.names \*\* #  
row.names <- c("ROW1","ROW2","ROW3") # \*\* row.names \*\* #  
matrix.names <- c("Matrix1","Matrix2") # \*\* matrix.names \*\* #  
array(c(vector1,vector2),dim=c(3,3,2),dimnames=list(row.names,column.names,matrix.names))

## , , Matrix1  
##   
## COL1 COL2 COL3  
## ROW1 5 10 13  
## ROW2 9 11 14  
## ROW3 3 12 15  
##

## , , Matrix2  
##   
## COL1 COL2 COL3  
## ROW1 5 10 13  
## ROW2 9 11 14  
## ROW3 3 12 15

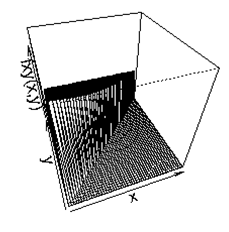
***(d) Integration***  
  
Numerically evaluate the bivariate integral .

fxy = function(x,y) exp(-y)\*(x<y)\*(x>0)  
lower = function(x) x; upper = function(x) 1+x; upper2 = Inf;   
fx = function(x) {sapply(x,function(x) {integrate(function(y) fxy(x,y),lower(x),upper(x))$value }) }   
integrate(fx,0,upper2)$value

## [1] 0.6321206

r = seq(0,1,length=50)

z = outer(r,r,fxy) # \*\* persp \*\* #   
persp(r,r,z,xlab="x",ylab="y",zlab="fxy(x,y)",theta=-20,phi=40)



***(e) flexible operations/operators***  
  
The trace operator and the horizontal direct product .

# rmarkdown cannot recognize these quotes #   
# "+" = function(A) sum(diag(A))

tr = function(A) sum(diag(A))  
A = matrix(c(3,4,3,5),2,2)  
# "+"(A)  
tr(A)

## [1] 8

"%\*~%" <- function(x,y){  
 n <- nrow(x)  
 out <- matrix(0,n,ncol(x)\*ncol(y))  
 for(i in 1:n) out[i, ] <- c(y[i, ] %o% x[i, ])  
 out }  
  
x <- matrix(1:4,2,2,TRUE)  
y <- matrix(5:8,2,2,TRUE)  
x %\*~% y

## [,1] [,2] [,3] [,4]  
## [1,] 5 6 10 12  
## [2,] 21 24 28 32

**4. Statistics Functions**  
  
***(a) shut the stars up***

library(car)   
options(show.signif.stars=FALSE)  
mod = lm(conformity ~ fcategory\*partner.status,data=Moore,   
 contrasts=list(fcategory=contr.sum, partner.status=contr.sum))  
Anova(mod) # \*\* Anova \*\* #

## Anova Table (Type II tests)  
##   
## Response: conformity  
## Sum Sq Df F value Pr(>F)  
## fcategory 11.61 2 0.2770 0.759564  
## partner.status 212.21 1 10.1207 0.002874  
## fcategory:partner.status 175.49 2 4.1846 0.022572  
## Residuals 817.76 39

***(b) model selection***

Use the birthweight data to select a best logistic regression model based upon the model selection criteria AIC, BIC, and AREA under ROC.

library(MASS) # birthwt data #   
library(verification) # \*\* roc.area \*\* #   
  
data = birthwt # [HL, 1980, p 92, 125] #   
head(data)

## low age lwt race smoke ptl ht ui ftv bwt  
## 85 0 19 182 2 0 0 0 1 0 2523  
## 86 0 33 155 3 0 0 0 0 3 2551  
## 87 0 20 105 1 1 0 0 0 1 2557  
## 88 0 21 108 1 1 0 0 1 2 2594  
## 89 0 18 107 1 1 0 0 1 0 2600  
## 91 0 21 124 3 0 0 0 0 0 2622

data$race = as.numeric(data$race==1)   
fit.f = glm(low~., binomial,data[,-10]) # full model #   
X = model.matrix(fit.f)[1:nrow(data),] # \*\* model.matrix \*\* #  
p = ncol(X)-1; num\_model = 2^(ncol(X)-1);   
d.g = as.data.frame(t(matrix(1:0,p,2,byrow=T)))  
X.i = cbind(matrix(1,num\_model),expand.grid(d.g));

# \*\* expand.grid \*\* #

var\_names = c("intercept",names(data[-c(1,10)]));   
colnames(X.i) = var\_names  
print(c(n=nrow(data),p=p,models=num\_model))

## n p models   
## 189 8 256

m.out = matrix(NA,num\_model,5) # output for criteria #   
for (select in 1:num\_model) { # loop through models #   
 # select model #   
X.m = as.matrix(X[,X.i[select,]==1]); cand.names = var\_names[X.i[select,]==1]; colnames(X.m) = cand.names;   
model = glm(low~X.m,binomial,data) # model fit #   
phat = model$fitted.values; param = ncol(X.m);   
   
AIC = AIC(model,k=2); # criteria #   
BIC = AIC(model,k=log(nrow(data))) # \*\* AIC \*\* #  
AREA = -roc.area(data$low,phat)$A # store criteria #   
m.out[select,] = cbind(select,param,AIC,BIC,AREA)   
} # end of loop through models #   
  
rank.m = apply(m.out[,3:5],2,rank) # \*\* apply \*\* #   
score = apply(rank.m,1,mean)   
mod.out = cbind(m.out[,2],X.i[,-1],rank.m,score)[order(score),]   
colnames(mod.out)[c(1,10,11,12)] = c("# parms","AIC.r","BIC.r","AREA.r")  
top.mod = head(mod.out,10);  
print(top.mod,digits=1,row.names=F)

## # parms age lwt race smoke ptl ht ui ftv AIC.r BIC.r AREA.r score  
## 6 0 1 1 1 1 1 0 0 3 7 5 5  
## 6 0 1 1 1 0 1 1 0 2 3 13 6  
## 7 0 1 1 1 1 1 1 0 1 26 1 9  
## 5 0 1 1 1 0 1 0 0 8 1 54 21  
## 5 0 1 1 1 1 0 0 0 23 15 36 25  
## 7 1 1 1 1 1 1 0 0 6 62 7 25  
## 5 0 1 0 0 1 1 1 0 35 25 19 26  
## 7 1 1 1 1 0 1 1 0 5 61 16 27  
## 7 0 1 1 1 0 1 1 1 9 65 11 28  
## 6 0 0 1 1 1 1 1 0 12 30 43 28

***(c) optim***

Find the MLE for for where is known. In this simple case, which can also be verified numerically.

set.seed(23); n = 5; sigma = 1; # \*\* set.seed \*\* #  
xx = rnorm(n,0,sigma); xbar = mean(xx) # \*\* cat \*\* #  
cat("mome is",c(xbar,sigma/sqrt(n))," ",fill=TRUE)

## mome is 0.6923581 0.4472136

# log likelihood #

llike = function(theta,x) sum(log(dnorm(x,theta,sigma)))

mle = optim(0,llike,method="BFGS",hessian=TRUE,x=xx,control=list(fnscale=-1))  
mle.out = c(mle$par,sqrt(-1/mle$hessian))   
cat("mle is",t(mle.out)," ")

## mle is 0.6923581 0.4472136

***(d) Bayesian Statistics***

Identify the posterior . Here and .

# library(R2WinBugs)  
# library(rjags)  
# library(stan)  
library(MCMCpack)

set.seed(23); n = 5; sigma = 1;   
xx = rnorm(n,0,sigma); xbar = mean(xx)  
mu = 1; tau = 1000   
lpost = function(theta,x) sum(log(dnorm(x,theta,sigma))) + log(dnorm(theta,mu,tau))   
post.samp = MCMCmetrop1R(lpost,theta.init=0,x=xx,thin=1,mcmc=100000,burnin=1,logfun=TRUE)

## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.70342  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

summary(post.samp) # summaries from MCMC samples #

## Iterations = 2:100001  
## Thinning interval = 1   
## Number of chains = 1   
## Sample size per chain = 100000   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE   
## 0.691069 0.447233 0.001414 0.004186   
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%   
## -0.1861 0.3911 0.6922 0.9926 1.5622

***(e) Machine Learning***

Classify the brand of cereal based upon ingredients.

library(MASS) # USCereal, lda, qda #   
library(e1071) # svm #   
library(rpart) # rpart #  
library(randomForest) # randomForest #   
  
data = UScereal; gg = factor(data$mfr)   
kk = (gg=="G")+(gg=="K") # select Manufacturer and subset #   
Y.s = subset(data[,-c(1,9,11)],(rep(1,nrow(data))==kk),drop=T)   
gg.s = factor(subset(gg,(rep(1,nrow(data))==kk),drop=T))  
data.s = data.frame(gg.s,Y.s) # \*\* subset \*\* #  
print(colnames(data.s))

## [1] "gg.s" "calories" "protein" "fat" "sodium" "fibre"   
## [7] "carbo" "sugars" "potassium"

print(table(gg.s))

## gg.s  
## G K   
## 22 21

p.da = p.cart = p.svm = p.rf = NULL  
for(i in 1:nrow(data.s)) { # n fold cv #   
 data.s.i = data.s[i,]; data.s.xi = data.s[-i,];  
 f.da = qda(gg.s~.,data=data.s.xi,CV=F)  
 f.cart = rpart(gg.s~.,data=data.s.xi,method="class")  
 f.svm = svm(gg.s~.,data=data.s.xi,kernel="radial",type="C-

classification",probability=T)

f.rf =

randomForest(gg.s~.,data=data.s.xi,method="class",sampsize=nrow(data.s.xi))  
   
 p.da = c(p.da,as.character(predict(f.da,data.s.i,type="class")[[1]]))  
 p.cart = c(p.cart,as.character(predict(f.cart,data.s.i,type="class")[[1]]))  
 p.svm = c(p.svm,as.character(predict(f.svm,data.s.i,type="class")[[1]]))  
 p.rf = c(p.rf,as.character(predict(f.rf,data.s.i,type="class")[[1]]))  
}  
  
# Discriminant Analysis (DA) #  
table(gg.s,p.da)

## p.da  
## gg.s G K  
## G 14 8  
## K 7 14

1-sum(diag(prop.table(table(gg.s,p.da))))

## [1] 0.3488372

# Classification and Regression Tree (CART) #   
table(gg.s,p.cart)

## p.cart  
## gg.s G K  
## G 17 5  
## K 6 15

1-sum(diag(prop.table(table(gg.s,p.cart))))

## [1] 0.255814

# Support Vector Machine (SVM) #   
table(gg.s,p.svm)

## p.svm  
## gg.s G K  
## G 17 5  
## K 7 14

1-sum(diag(prop.table(table(gg.s,p.svm))))

## [1] 0.2790698

# Random Forest (RF) #   
table(gg.s,p.rf)

## p.rf  
## gg.s G K  
## G 18 4  
## K 6 15

1-sum(diag(prop.table(table(gg.s,p.rf))))

## [1] 0.2325581

**5. Graphics**  
  
***(a) Images***

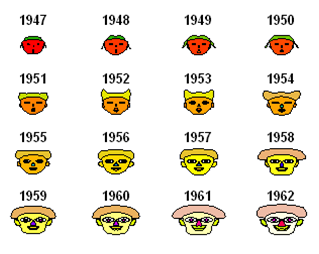
library(jpeg)  
image = readJPEG("F:/R/muir.jpg")  
plot(0:1,0:1,type="n",ann=FALSE,axes=FALSE)  
rasterImage(image,0,0,1,1)



***(b) Faces***

Data with 7 highly correlated economical variables observed yearly from 1947 to 1962.

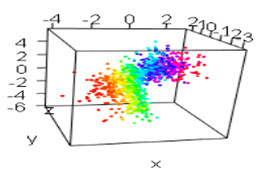
library(aplpack)  
# cor(longley)  
faces(longley,face.type=1,print.info=T)



## modified item Var   
## "height of face " "GNP.deflator"  
## "width of face " "GNP"   
## "structure of face" "Unemployed"   
## "height of mouth " "Armed.Forces"  
## "width of mouth " "Population"   
## "smiling " "Year"   
## "height of eyes " "Employed"   
## "width of eyes " "GNP.deflator"  
## "height of hair " "GNP"   
## "width of hair " "Unemployed"   
## "style of hair " "Armed.Forces"  
## "height of nose " "Population"   
## "width of nose " "Year"   
## "width of ear " "Employed"   
## "height of ear " "GNP.deflator"

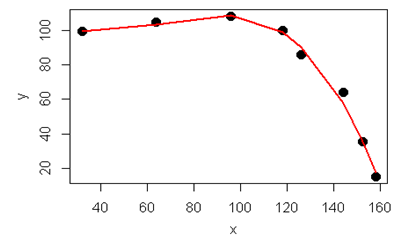
***(c) 3D Plots***

library(rgl)  
library(png)  
x <- sort(rnorm(1000))  
y <- rnorm(1000)  
z <- rnorm(1000) + atan2(x,y)  
# plot3d(x,y,z,col=rainbow(1000),axes=T,aspect=T)  
# rgl.snapshot("F:/R/plot3d.png",fmt="png",top=TRUE)  
image = readPNG("F:/R/plot3d.png",native=T)  
plot(0:1,0:1,type="n",ann=FALSE,axes=FALSE)  
rasterImage(image,0,0,1,1)



***(d) Plot Functions***

x <- c(32,64,96,118,126,144,152.5,158)   
y <- c(99.5,104.8,108.5,100,86,64,35.3,15)  
plot.f = function(s=16,ssize=1.5,formx=poly(x,3),color='red',line=1,ww=2) {  
 plot(x,y,pch=s,cex=ssize)  
 pred = predict(lm(y~formx))  
 lines(x,pred,col=color,lty=line,lwd=ww) }  
plot.f()



***(e) Sounds***

# http://stackoverflow.com/questions/3365657/is-there-a-way-to-make-r-beep-play-a-sound-at-the-end-of-a-script #

library(beepr) # \*\* wait \*\* #  
options(error = function() for (i in 1:3) wait(beep(7),2),warn=2)   
log(-1000)

## Warning in log(-1000): NaNs produced

options(op)   
options(error = function() {})