```
#################################################################
### Frontiers_Code.R
###
### This file contains all code contained within the manuscript
### T.R. Prokop and M. Wininger, "A Primer on R for Numerical
### Analysis in Educational Research", Frontiers in Education.
###
### Instructions: Select any lines of code and run (Windows:
### F5 or Ctrl + R; Mac: Command + Enter). At start of session,
### be sure to set the working directory (setwd, below) by poin-
### ting to whatever directory contains the datasets. If encoun-
### tering an error in loading libraries, verify package install
### (Packages > Install Packages and follow prompts).
###
### Note: Plots are intentionally coded for plotting within the
### R console, and not for saving to file. To save figure to
### file, un-comment the line of code at top and at bottom of
### the plotting command block (png(...) and dev.off()). Fonts
### and margins are specified for best saving, not per se for
### best plotting within R. Where unsighlty graphics present in
### R environment, save image to file and inspect in .png form.
###
### Correspondence to: wininger@hartford.edu
#################################################################

### ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
### houskeeping
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# clear all variables form workspace
rm(list=ls(all=TRUE))
# close all graphical windows
graphics.off()

# load libraries
library(plotrix)      ### axis-break
library(corrplot)     ### correlation plot
library(grDevices)    ### colorRampPalette
library(lme4)         ### glmer

# set the working directory
setwd("C:/Users/Desktop/")


### ############################
### Section 2: Familiarization
### ############################

### ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 2.1: Acquisition and Operation
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# test software (should yield 16)
8*2



### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 2.2: Basic syntax
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# create simple variables
alpha = 1
beta_1 = 0.20
Score = 100

# create a vector
vect = c(1, 2, 3, 6, 7, 64, 81, 100, pi)

# create a matrix
matx = matrix(c(1,2,3,4,5,6,7,8), nrow=4)
colnames(matx)=c("col_1","col_2")
rownames(matx)=c("a","b","c","d")

# create a data frame from matrix
my_frame = data.frame(matx)
my_frame$col_1

# simple arithmetic (division)
beta_1/2

# simple arithmetic (exponent)
Score^2

# simple arithmetic (variable over-write)
gamma = 2*alpha
alpha = 10
gamma
alpha

# simple arithmetic (matrices)
matx^2

# functions (absolute value)
all_pos = abs(c(-1,0,2,3,-2))
all_pos

# functions (mean and standard deviation)
mean(vect)
sd(vect)

# functions (optional arguments)
round(mean(vect))
round(mean(vect),2)
```

```
# strings (one line compiles; the other doesn't)
all_pos / 2
"all_pos" / 2
```

```
### ############################
### Section 3: Examples
### ############################

### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 3.1: Descriptive Statistics
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# read-in the data from file
comp_data=read.csv("Frontiers_1_SubSample.csv",head=TRUE,
stringsAsFactors=TRUE)
# analyze and report
print(paste(nrow(comp_data), "non-compliant students"))

# compute average GPA across sub-group
mean_gpa = round(mean(comp_data$GPA_UG),1)
stdv_gpa = round(sd(comp_data$GPA_UG),1)
print(paste("Average GPA:",mean_gpa,"+/-",stdv_gpa))

# show GPA as box plots (break-out by cohort)
#png("Figure_1a.png")
boxplot(GPA_G~Cohort,data=comp_data)
title(main="GPA by Cohort")
title(ylab="Grade-Point Average")
#dev.off()

# modify box-plot for visualization
#png("Figure_1b.png")
boxplot(GPA_G~Cohort,data=comp_data,
names=c("Cohort I","Cohort II"), ylim=c(1.9,4.1),
cex.axis=1.5, lwd=3, col=c("grey50","grey80"))
title(main="GPA by Cohort", line=1, cex.main=2, font.main=1)
title(ylab="Grade-Point Average", line=2.5, cex.lab=2)
# from library(plotrix)
axis.break(axis=2,1.925, style="slash")
#dev.off()
```

```
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 3.2: Conditions
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# test for students exceeding undergrad GPA threshold
comp_data$GPA_UG>3.5

# return rows corresponding to hi undergrad GPAs
which(comp_data$GPA_UG>3.5)

# find which students are in Cohort I
which(comp_data$Cohort=="I")

# find students with both undergrad- and graduate GPA above 3.4
intersect(which(comp_data$GPA_UG>3.4), which(comp_data$GPA_G>3.4))



### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 3.3: Dataset Manipulations
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# convert skill from 20-points to 100% scale in Cohort I
inds_coh1 = which(comp_data$Cohort== "I")
comp_data$Skill[inds_coh1] = comp_data$Skill[inds_coh1] * 5

# convert skill to leveled variable
comp_data$SkillLevel=cut(comp_data$Skill,c(0,85,90,100),
labels=c("Fail","Retest","Pass"))

# re-level factor variable
comp_data$Category=rep("",nrow(comp_data))
comp_data$Category[comp_data$SkillLevel %in%
     c("Fail","Retest")]="REDFLAG"
comp_data$SkillLevel %in% c("Fail","Retest")

# sort data-frame based on Red Flag status
comp_data=comp_data[order(comp_data$Category,decreasing=TRUE),]

# sort on Red Flag; sub-sort on Skill Level
comp_data=comp_data[order(comp_data$Category,comp_data$SkillLevel,
     decreasing=TRUE),]



### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 3.4: Diagnostics
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# read-in alc dataset
alc_data =
read.csv("Frontiers_2_ALCData.csv",head=TRUE,stringsAsFactors=TRUE)

# identify outliers
boxplot.stats(alc_data$GPA_G)$out
boxplot.stats(alc_data$GPA_UG)$out

# double-check outliers via standardization
range(scale(alc_data$GPA_G))
range(scale(alc_data$GPA_UG))

# generate a scatter plot of GPA datasets
#png("Figure_2a.png")
plot(alc_data$GPA_UG, alc_data$GPA_G,main="",xlab="",ylab="")
title(main="Scatterplot of GPA")
title(xlab="Graduate GPA")
title(ylab="Undergraduate GPA")
#dev.off()

# modify scatterplot for visualization
#png("Figure_2b.png")
plot(alc_data$GPA_UG, alc_data$GPA_G,
     main="",xlab="",ylab="",pch=21,lwd=2,cex=2,
     col="darkblue",bg="lightgreen",ylim=c(1.9,4.1))
title(main="Scatterplot of GPA", line=1, cex.main=2, font.main=1)
title(xlab="Graduate GPA", line=3, cex.lab=2, font.main=1)
title(ylab="Undergraduate GPA", line=2.5, cex.lab=2, font.main=1)
linmod=lm(GPA_G~GPA_UG,data=alc_data)
abline(linmod)
axis.break(axis=2,1.925, style="slash")
rho_val=round(cor(alc_data$GPA_UG, alc_data$GPA_G),2)
text(3.1,4,paste("Correlation = ",rho_val),adj=c(0,0),cex=1.4)
#dev.off()

# create two more GPA variables (for exploration)
set.seed(311)
alc_data$GPA_R1=3.0 + runif(nrow(alc_data))
set.seed(311)
alc_data$GPA_R2=2.5 + runif(nrow(alc_data))
set.seed(331)
alc_data$GPA_R3=3.0 + runif(nrow(alc_data))
set.seed(156)
alc_data$GPA_R4=1.0 + runif(nrow(alc_data))

# take as only those columns containing "GPA" in their column names
gpa_data = alc_data[,grep("GPA",names(alc_data))]

# show scatter
#png("Figure_3a.png")
plot(gpa_data,main="Collinearity Review Scatterplot")
```

```
#dev.off()

# modify scatterplot for visualization
#png("Figure_3b.png")
plot(gpa_data,pch=21,lwd=2,cex=1.5,
     col="darkred",bg="pink",xaxt="n",yaxt="n")
title(main="Collinearity Review Scatterplot", line=2, cex.main=2,
font.main=1)
#dev.off()

# show correlation matrices
### from library(corrplot)
#png("Figure_4a.png")
corrplot(cor(gpa_data))
title(main="Correlation Heat Map")
#dev.off()

# modify correlation plot for visualization
### from library(grDevices)
#png("Figure_4b.png")
clrs=colorRampPalette(c("red","white","blue"))
corrplot(cor(gpa_data), col=clrs(10),
     addCoef.col="white",number.digits=2,
     number.cex=0.75, tl.pos="d", tl.col="black", type="upper")
title(main="Correlation Heat Map", line=1, cex.main=2, font.main=1)
#dev.off()
```

```
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 3.5: Hypothesis Testing
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# read-in agq dataset
agq_data=read.csv("Frontiers_3_AGQData.csv",head=TRUE,
stringsAsFactors=TRUE)

# paired t-test
pre=agq_data$AGQ_1
post=agq_data$AGQ_2
t.test(pre,post,paired=TRUE)

# report differences
print(paste("AGQ Pre = ",round(mean(pre),1),
     "+/-",round(sd(pre),1)))
print(paste("AGQ Post = ",round(mean(post),1),
     "+/-",round(sd(post),1)))
```

```
# assess control group separately
pre=agq_data$AGQ_1[which(agq_data$Group=="C")]
post=agq_data$AGQ_2[which(agq_data$Group=="C")]
t.test(pre,post,paired=TRUE)

# assess treatment group separately
pre=agq_data$AGQ_1[which(agq_data$Group=="T")]
post=agq_data$AGQ_2[which(agq_data$Group=="T")]
t.test(pre,post,paired=TRUE)

# convert dataset for multi-variate analysis
agq_data$Change=agq_data$AGQ_2-agq_data$AGQ_1

# demonstrate that t-test is the same as paired test
t.test(agq_data$Change)

# report p-value
aov_summ=summary(aov(Change~Group,data=agq_data))
print(paste("P-value, for AGQ change=",round(aov_summ[[1]][1,5],3)))

# merge AGQ dataset onto ALC dataset for co-variates
agq_data=merge(agq_data,alc_data)

# test with full anova
aov(Change~Cohort+Group+GPA_UG+GPA_G,data=agq_data)

# summarize anova
summary(aov(Change~Cohort+Group+GPA_UG+GPA_G,data=agq_data))

# test with anova with an interaction term
summary(aov(Change~Cohort+GPA_UG+Group*GPA_G,data=agq_data))

# identify the students in each stratum
cohort_1=which(agq_data$Cohort=="I")
cohort_2=which(agq_data$Cohort=="II")
agq_incr=which(agq_data$Change<0)
agq_decr=which(agq_data$Change>=0)

# determine cell values for a contingency table
agq_cell1=length(intersect(cohort_1,agq_incr))
agq_cell2=length(intersect(cohort_2,agq_incr))
agq_cell3=length(intersect(cohort_1,agq_decr))
agq_cell4=length(intersect(cohort_2,agq_decr))

# create the contingency table, test two ways
agq_table=matrix(c(agq_cell1,agq_cell2,agq_cell3,agq_cell4),nrow=2)
rownames(agq_table)=c("Cohort_I","Cohort_II")
colnames(agq_table)=c("Increase","Decrease")
chisq.test(agq_table)
fisher.test(agq_table)
```

```
# display contingency table
agq_table


# perform linear regression on alc dataset
linmod = lm(GPA_UG~GPA_G,data=alc_data)
# summarize the linear regression
summary(linmod)

# verify that correlation yields same p-value
cor.test(alc_data$GPA_G, alc_data$GPA_UG)

# merge compliant versus non-compliant
data_nonc=read.csv("Frontiers_1_SubSample.csv",head=TRUE,
stringsAsFactors=TRUE)
data_comp=read.csv("Frontiers_2_ALCData.csv",head=TRUE,
stringsAsFactors=TRUE)

# verify that there is no overlap between datasets
olap=intersect(data_nonc$ID,data_comp$ID)
if (length(olap)==0){
     print("Data Quality Check OK")
}else{
     print("Data Error: Non-distinct datasets")
}

# bind an indicator for compliant versus non-compliant
data_nonc$Status="Noncompliant"
data_comp$Status="Compliant"
data_nonc=subset(data_nonc,
select=c("ID","Cohort","Group","GPA_UG","GPA_G","Status"))
data_comp=subset(data_comp,
select=c("ID","Cohort","Group","GPA_UG","GPA_G","Status"))

# merge datasets
data_stat=rbind(data_nonc,data_comp)

# enforce classes
data_stat$Status=as.factor(data_stat$Status)

# set up a logistic regression
logmod=glm(Status~GPA_G+GPA_UG,data=data_stat,family=binomial)
summary(logmod)

### from library(lme4)
# set up a mixed effects logistic regression
logmod2=glmer(Status~GPA_G+(1|GPA_UG),data=data_stat,family=binomial)
```

```
### ############################
### Section 4: Notes
### ############################

### ~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Section 4.1: Control Structures
### ~~~~~~~~~~~~~~~~~~~~~~~~~~~

# simple conditional control structure
gpa_val=3.5
if (gpa_val<3.6){
      print("Med")
}else if (gpa_val<3.0){
      print("Low")
}else{
      print("Hi")
}

# conditional control with non-numeric item
"g" %in% c("a","b","c","d","e","f","g","h")

# simple for-loop
for (i in 1:5){
      print(paste("This is loop number ", i, sep=""))
}

# scroll through each record and report red-flag students
for (i in 1:nrow(comp_data)){
      if (comp_data$Category[i]=="REDFLAG"){
            print(paste("Student ",comp_data$ID[i],
            ": Red Flag", sep=""))
      }
}
```