Problem Set 5, 10/18/2017

Group 1 Dan dbuonaiuto@g.harvard.edu
Nick nherrmann@g.harvard.edu
Zane rzwolf@g.harvard.edu
Lydia lakrasilnikova@g.harvard.edu

13.6.3

a)

The pwr library in R was used for this calculation. You enter three of the four following variables, and the fourth is calculated: effect size, sample size, significance level, and statistical power. For doses of 1, 100, and 10,000, the required sample sizes would be 7.84e8, 7.84e4, and 7.84.

```
> pwr.p.test(h=0.0001, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 1e-04
              n = 784886051
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

> pwr.p.test(h=0.01, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 0.01
              n = 78488.61
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

> pwr.p.test(h=1, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 1
              n = 7.848873
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

b)

If 0.01%, 1%, and 100% were log functions of the doses, then the actual probabilities would be 0.02%, 0.04%, 0.05%. Using the same package, we achieve:

```
> pwr.p.test(h=0.0001, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 1e-04
              n = 784886051
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

> pwr.p.test(h=0.0002, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 2e-04
              n = 196221513
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

> pwr.p.test(h=0.0005, n=NULL, sig.level=0.05, power=0.8, alternative="two.sided")

     proportion power calculation for binomial distribution (arcsine transformation)

              h = 5e-04
              n = 31395442
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

14.8.2 a-c with simulated data (d on next page with real data)

a)

```
##simulate variables
Y<-rnorm(100,50,10)
X<-rnorm(100,10,1)

ydat<-as.data.frame(Y)
xdat<-as.data.frame(X)
###make full data
full.data<-cbind(ydat,xdat)

###generate NA's
X<-ifelse(full.data$X>10,NA,full.data$X)
Xdat<-as.data.frame(X)
##make available data
avail.data<-cbind(ydat,Xdat)
```

b)

Complete-case regression `M1 lm(formula = X ~ Y, data = avail.data)`	Full dataset `M2 lm(formula = X ~ Y, data = full.data)`

`coef.est coef.se`
`(Intercept) 8.70 0.41`
`Y 0.01 0.01`
`---`
`n = 49, k = 2`
`residual sd = 0.66, R-Squared = 0.02`
	`coef.est coef.se`
	`(Intercept) 9.40 0.50`
	`Y 0.01 0.01`
	`---`
	`n = 100, k = 2`
	`residual sd = 1.06, R-Squared = 0.01`

The regression of x on y produces similar coefficients, standard errors, and R^2 values for only those samples with complete data from part a and in the complete dataset.

c)

Complete-case regression `M3 lm(formula = Y ~ X, data = avail.data)`	Full dataset `M4 lm(formula = Y ~ X, data = full.data)`

`coef.est coef.se`
`(Intercept) 26.31 23.15`
`X 2.51 2.54`
`---`
`n = 49, k = 2`
`residual sd = 11.60, R-Squared = 0.02`
	`coef.est coef.se`
	`(Intercept) 37.72 10.36`
	`X 1.22 1.03`
	`---`
	`n = 100, k = 2`
	`residual sd = 10.90, R-Squared = 0.01`

In contrast, the regression of y on x produces coefficients and standard errors that are very different: the standard error for the complete-case regression is more than twice that for the full dataset, which is not too surprising since the complete-case regression uses fewer data points, but the coefficients are very different between the two models, which could be concerning. The R^2 values, on the other hand, are not too different, but that may be because they are both so low.

d)

See below modelling with real data.

14.8.2 a-d with real data

Nick found an exciting dataset on cigarette consumption: http://koaning.io/fun-datasets.html

Here, we use **cigarette consumption** as our outcome **y variable** and **average price** as our predictor **x variable**. The full dataset contains state, year, consumer price index, state population, number of packs per capita, state personal income (total, nominal), average combined local tax, average price, and excise tax including sales tax.

```
library(arm)
library(rstanarm)

dat=read.csv("cigarette.csv",header=TRUE)
dat=subset(dat,year=="1995",select=c(state, packpc,avgprs)) #x=avgprs, y=packpc
```

a)

```
#Part A - write a program to cause about half of x values (average price) to be NA,
#dependent on y

availdat=dat

for (i in 1:length(availdat$state)){
  if (availdat$packpc[i] <= median(availdat$packpc)){
       availdat$avgprs[i]=ifelse(rbinom(1,1,0.25),NA,availdat$avgprs[i])
  }else{
       availdat$avgprs[i]=ifelse(rbinom(1,1,0.75),NA,availdat$avgprs[i])
  }
}
```

b)

```
#Part B - regression of x on y

partb1=lm(avgprs~packpc,data=availdat)
display(partb1)

partb2=lm(avgprs~packpc,data=dat)
display(partb2)
```

Complete-case regression `lm(formula = avgprs ~ packpc, data = availdat)`	Full dataset `lm(formula = avgprs ~ packpc, data = dat)`

`coef.est coef.se`
`(Intercept) 247.78 18.80`
`packpc -0.66 0.20`
`---`
`n = 23, k = 2`
`residual sd = 20.18, R-Squared = 0.34`
	`coef.est coef.se`
	`(Intercept) 244.53 11.49`
	`packpc -0.64 0.12`
	`---`
	`n = 48, k = 2`
	`residual sd = 18.90, R-Squared = 0.40`

As with our simulated data, the two models are consistent: they have very similar coefficients, standard errors, and R^2 values.

c)

```
#Part C - regression of y on x

partc1=lm(packpc~avgprs,data=availdat)
display(partc1)

partc2=lm(packpc~avgprs,data=dat)
display(partc2)
```

Complete-case regression `lm(formula = packpc ~ avgprs, data = availdat)`	Full dataset `lm(formula = packpc ~ avgprs, data = dat)`

`coef.est coef.se`
`(Intercept) 187.27 29.65`
`avgprs -0.51 0.16`
`---`
`n = 23, k = 2`
`residual sd = 17.80, R-Squared = 0.34`
	`coef.est coef.se`
	`(Intercept) 210.33 20.94`
	`avgprs -0.62 0.11`
	`---`
	`n = 48, k = 2`
	`residual sd = 18.69, R-Squared = 0.40`

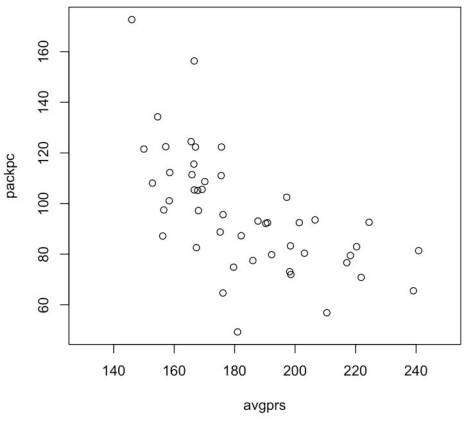
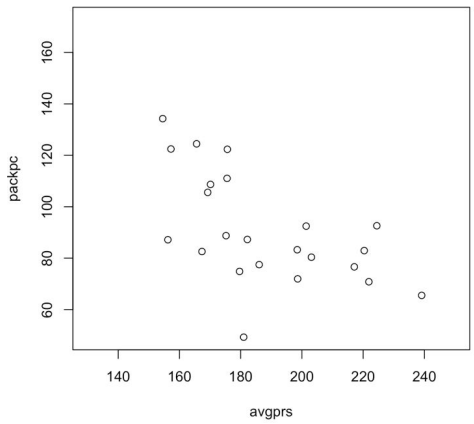
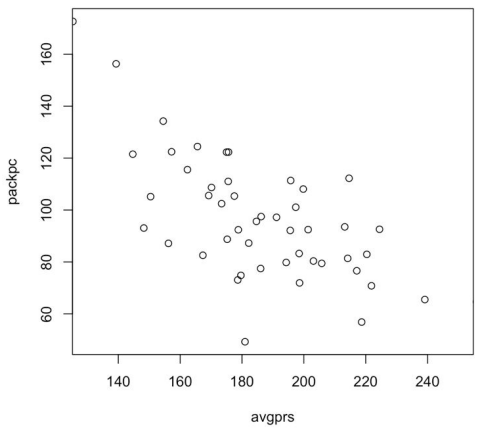
The differences between the complete-case regression on the dataset with missingness and the regression with the full dataset are not as pronounced as they were for our simulated data, but the two models we produced here are nonetheless not nearly as similar to each other as the

models we produced in part b. The coefficients are different but at least this time they are in the same ballpark. We again see that in part c, the complete-case regression has higher standard errors, which makes sense since the complete-case regression uses fewer datapoints. The model produced using the full dataset also has a greater R^2 value.

d)

```
#Part D - randomly imputing missing x data

# fits model using available data
fitimp = stan_glm(avgprs ~ packpc, data = availdat)

# selects only predictor variable (packpc)
predictors = availdat[2]

# random imputation using predictor variable values and model from available data
pred = posterior_predict(fitimp, newdata = predictors)
imps = impute(availdat$avgprs, pred[1,])
impdat = as.data.frame(cbind(availdat$packpc, availdat$avgprs, imps))
colnames(impdat) = c("packpc", "old.avgprs", "avgprs")

# plots of imputed data, data with missingness, and original data
plot(packpc~avgprs,data=impdat, xlim=c(130,250))
plot(packpc~avgprs,data=availdat, xlim=c(130,250))
plot(packpc~avgprs,data=dat, xlim=c(130,250))
```

Here is what our data looks like:

Original data	Data with missingness	Imputed data
		

```
# models
partd1=lm(packpc~avgprs,data=impdat)
display(partd1)

partd2=lm(packpc~avgprs,data=dat)
display(partd2)
```

We repeat the regression of y on x from part c, now using the imputed data.

```
lm(formula = packpc ~ avgprs, data = impdat)
            coef.est coef.se
(Intercept) 206.75    18.42
avgprs       -0.59     0.10
---
n = 48, k = 2
residual sd = 17.93, R-Squared = 0.44
```

Comparing this model to those produced in part c, the coefficients are similar to those for the complete-case regression and the full dataset. The standard errors are close to those for the full dataset and smaller than those produced by complete-case regression—in fact, in this run, the standard errors are smaller and the R^2 value is larger than produced from the full dataset. (Running the code a few times, I get R^2 values of from 0.24 to 0.52.)