Lab 2: HW #2 Preparation

Wednesday Bushong

2/15/2017

1 Preliminaries

In this lab we will perform linear regression on a dataset with information about adult attachment styles and their relationship to accommodation behavior.

Predictor variables:

- Three attachment styles:
  - Secure
  - Avoidant
  - Anxious-ambivalent
- Masculinity
- Femininity

Outcome variables:

- Four accommodation behaviors:
  - Voice
  - Exit
  - Loyalty
  - Neglect

Important: We are going to use a custom summary function that extracts semi-partial correlations from lm() models (this information is not provided in R by default). To load this custom summary function run the following command in R:

```
library(foreign)
library(pwr)
source("showmelm_WB.R")
```

Now, we'll load in the data for today's lab, and we'll also remove rows that have NA values:

```
d <- read.spss("attachdiag.sav", to.data.frame = TRUE)
```

```
## Warning in read.spss("attachdiag.sav", to.data.frame = TRUE):
## attachdiag.sav: Unrecognized record type 7, subtype 18 encountered in
## system file
```

```
## re-encoding from CP1252
```

```
head(d)
```

```
##      sex age secure avoid     ambiv  masc   fem    voice     exit    loyal
## 1   <NA>  NA    4.6     5  6.000000 3.500 3.750 5.666667 7.333333 1.500000
## 2 female  19    4.0    20 20.000000 2.875 4.000 5.666667 3.000000 3.000000
## 3 female  24    2.2     3  2.400000 3.000 4.125 3.333333 5.333333 2.666667
## 4 female  NA    4.6     5  3.000000 3.500 4.250 6.333333 5.666667 3.333333
## 5 female  25    5.4     3  2.600000 3.875 4.125 6.333333 2.000000 2.333333
## 6 female  22    3.4     2  2.333333 3.750 4.375 7.333333 1.666667 2.666667
##    neglect
## 1 4.666667
## 2 3.666667
## 3 2.666667
## 4 4.000000
## 5 2.000000
## 6 1.333333
```

```
nrow(d) # get number of observations
```

```
## [1] 84
```

```
d <- d[complete.cases(d), ]
nrow(d)
```

```
## [1] 68
```

2 Obtaining Correlations

The correlation function in R is cor(). Let's say, for example, that we're interested in the correlation between all the predictor variables and one of our outcome variables, neglect:

```
cors <- cor(d[, c("neglect", "secure", "avoid", "ambiv", "masc", "fem")], use = "pairwise.complete")
cors
```

```
##            neglect      secure      avoid       ambiv       masc
## neglect  1.0000000 -0.24010977  0.2749096  0.21916070 -0.1474272
## secure  -0.2401098  1.00000000 -0.3712361 -0.03707411  0.2372318
## avoid    0.2749096 -0.37123611  1.0000000  0.76776224 -0.2213604
## ambiv    0.2191607 -0.03707411  0.7677622  1.00000000 -0.2277632
## masc    -0.1474272  0.23723179 -0.2213604 -0.22776323  1.0000000
## fem     -0.3743650  0.54486448 -0.2516153 -0.01971816  0.1345563
##                fem
## neglect -0.37436505
## secure   0.54486448
## avoid   -0.25161529
## ambiv   -0.01971816
## masc     0.13455625
## fem      1.00000000
```

3 Univariate Regression

```
m <- lm(neglect ~ secure, data = d)
```

m is a special kind of object called a model object. Let's see what it contains:

```
names(m)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

We can then index into things that might be of interest, for example coefficients:

```
m$coefficients
```

```
## (Intercept)      secure
##    4.175144   -0.350456
```

Like with data.frames, you can also call summary() on model objects:

```
summary(m)
```

```
##
## Call:
## lm(formula = neglect ~ secure, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0537 -1.1203 -0.1152  1.0378  3.8233
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.1751     0.7864   5.309 1.39e-06 ***
## secure       -0.3505     0.1744  -2.009   0.0486 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.573 on 66 degrees of freedom
## Multiple R-squared:  0.05765,    Adjusted R-squared:  0.04337
## F-statistic: 4.038 on 1 and 66 DF,  p-value: 0.04858
```

It turns out that summaries themselves are also objects! We can investigate particular aspect of the model summary like so:

```
names(summary(m))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
summary(m)$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  4.175144  0.7863915  5.309244 1.385798e-06
## secure      -0.350456  0.1744043 -2.009446 4.858191e-02
```

Question: How can we tell whether the coefficients are Bs or βs?

4 Hierarchichal Regression

To conduct hierarchical regression, we'll use the same linear model function as above in the univariate case, except we add more predictors.

Now, let's see how much more variance in neglect we can explain by additionally adding another variable of interest, "avoid":

```
m2 <- lm(neglect ~ secure + avoid, d)
summary(m2)
```

```
##
## Call:
## lm(formula = neglect ~ secure + avoid, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.846 -1.203 -0.045  1.077  3.734
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.14136    0.98584   3.186  0.00221 **
## secure      -0.23371    0.18520  -1.262  0.21150
## avoid        0.14149    0.08332   1.698  0.09428 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.552 on 65 degrees of freedom
## Multiple R-squared:  0.09768,    Adjusted R-squared:  0.06992
## F-statistic: 3.518 on 2 and 65 DF,  p-value: 0.03542
```

```
r.sq.dif <- summary(m2)$r.squared - summary(m)$r.squared
```

Comparing Models

We've seen that our R^2 increased from m to m2, but was this increase actually significant? We can test this using the anova() function on our two model objects:

```
anova(m, m2)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: neglect ~ secure
## Model 2: neglect ~ secure + avoid
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     66 163.41
## 2     65 156.47  1     6.941 2.8835 0.09428 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obtaining Partial Correlations: showme.lm() function

You'll notice that the summary() function doesn't give you all of the information you might want from your model. For example, coefficient estimates don't give us the whole story about the actual correlations between the dependent and independent variables! (see point above about Bs vs. βs.) So, a lovely student in this class years ago developed a function called showme.lm() which will helpfully provide us with additional information, like semipartial correlations in an easy-to-digest format like summary():

```
showme.lm(m2, verbose = TRUE)
```

```
## ### Model Summaries ###
##
## Call:
## lm(formula = neglect ~ secure + avoid, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.846 -1.203 -0.045  1.077  3.734
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)   sr2
## (Intercept)   3.1414     0.9858  3.1865   0.0022    NA
## secure       -0.2337     0.1852 -1.2619   0.2115 0.022
## avoid         0.1415     0.0833  1.6981   0.0943 0.040
##
## Residual standard error: 1.552 on 65 degrees of freedom
## Multiple R-squared:  0.09768,    Adjusted R-squared:  0.06992
## F-statistic: 3.518 on 2 and 65 DF,  p-value: 0.03542
```

Now, we can see how much each independent variable uniquely contributes to variance explained in the dependent variable! In this example, secure accounts for 1% of the variance and avoid accounts for 4.9% of the variance. You'll notice that this is smaller than the total amount of variance explained by the model, 8.68%. Think back to the ballantines – this suggests that the shared variance between secure and avoid in predicting neglect is ~2.7% (but remember that suppression means this isn't always the case!)

5 Stepwise Regression

Now, we'll get into doing stepwise regression. You can do this using the step() function in R. First, you fit two kinds of models. A "base" model, which includes none of your independent variables of interest, and one that contains all of them. Then, you call the step() function and specify whether you want to do backward or forward stepwise regression.

```r
m.base <- lm(neglect ~ 1, d) # base model only fits an intercept
m.full <- lm(neglect ~ secure + avoid + ambiv + masc + fem, d) # full model uses all variables of inter

step(m.base, scope = list(lower = m.base, upper = m.full), direction = "forward")
```

```
## Start:  AIC=65.66
## neglect ~ 1
##
##          Df Sum of Sq    RSS    AIC
## + fem     1   24.3024 149.10 57.388
## + avoid   1   13.1050 160.30 62.312
## + secure  1    9.9972 163.41 63.618
## + ambiv   1    8.3288 165.07 64.309
## <none>                 173.40 65.656
## + masc    1    3.7689 169.63 66.161
##
## Step:  AIC=57.39
## neglect ~ fem
##
##          Df Sum of Sq    RSS    AIC
## + ambiv   1    7.7802 141.32 55.744
## + avoid   1    6.0457 143.06 56.573
## <none>                 149.10 57.388
## + masc    1    1.6635 147.44 58.625
## + secure  1    0.3220 148.78 59.241
##
## Step:  AIC=55.74
## neglect ~ fem + ambiv
##
##          Df Sum of Sq    RSS    AIC
## <none>                 141.32 55.744
## + masc    1   0.45373 140.87 57.525
## + secure  1   0.23045 141.09 57.633
## + avoid   1   0.17850 141.14 57.658

##
## Call:
## lm(formula = neglect ~ fem + ambiv, data = d)
##
## Coefficients:
## (Intercept)          fem        ambiv
##      5.6235      -0.8894       0.1404
```

We can see that the stepwise regression starts at the base model and then fits 5 other models that each contain one additional variable. It chooses the variable that gives the best improvement and moves on, fitting more models, until it gets to the "best" model for the data. We can also do the reverse using the backward direction in step():

```r
m.final <- step(m.full, scope = list(lower = m.base, upper = m.full), direction = "backward")
```

```
## Start:  AIC=61.43
## neglect ~ secure + avoid + ambiv + masc + fem
```

```
##
##          Df Sum of Sq    RSS    AIC
## - secure  1    0.0401 140.70 59.446
## - avoid   1    0.0773 140.74 59.464
## - masc    1    0.3693 141.03 59.605
## - ambiv   1    1.6356 142.30 60.213
## <none>                140.66 61.427
## - fem     1   14.1988 154.86 65.966
##
## Step:  AIC=59.45
## neglect ~ avoid + ambiv + masc + fem
##
##         Df Sum of Sq    RSS    AIC
## - avoid  1    0.1634 140.87 57.525
## - masc   1    0.4386 141.14 57.658
## - ambiv  1    1.6711 142.38 58.249
## <none>               140.70 59.446
## - fem    1   18.2247 158.93 65.728
##
## Step:  AIC=57.53
## neglect ~ ambiv + masc + fem
##
##         Df Sum of Sq    RSS    AIC
## - masc   1    0.4537 141.32 55.744
## <none>               140.87 57.525
## - ambiv  1    6.5705 147.44 58.625
## - fem    1   22.4685 163.34 65.588
##
## Step:  AIC=55.74
## neglect ~ ambiv + fem
##
##         Df Sum of Sq    RSS    AIC
## <none>               141.32 55.744
## - ambiv  1    7.7802 149.10 57.388
## - fem    1   23.7538 165.07 64.309
```

Now, the algorithm is starting from the full model. For the first step, it fits 5 different models that each contain all variables but one. Whichever variable results in the least penalty to the model is removed from consideration. This continues until the algorithm has reached a point where removing any single variable results in a significantly worse-fitting model.

5 Sets

In R, there's no straightforward way to encode sets into regression formulas. Rather, we'll just add in all of the variables that belong in the set we've defined.

Let's say that we want to know whether "attachment style" accounts for neglect better than "masculinity/femininity". We will fit three different models:

- one that contains all of the predictors of interest
- one that contains only the set of attachment variables
- one that contains only the set of masculinity/femininity

```
m.full <- lm(neglect ~ secure + avoid + ambiv + masc + fem, d)
m.attach <- lm(neglect ~ secure + avoid + ambiv, d)
m.mascfem <- lm(neglect ~ masc + fem, d)
```

Then, we can compare the full model with each of the sub-models to determine how much each set of variable adds over & above the other set:

```
## Comparing m.full and m.attach gives us the effect of adding masculinity/femininity to the model that
anova(m.full, m.attach)
```

```
## Analysis of Variance Table
##
## Model 1: neglect ~ secure + avoid + ambiv + masc + fem
## Model 2: neglect ~ secure + avoid + ambiv
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     62 140.66
## 2     64 155.30 -2   -14.634 3.2251 0.04651 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(m.full)$r.squared - summary(m.attach)$r.squared
```

```
## [1] 0.08439243
```

```
## Comparing m.full and m.mascfem gives us the effect of adding attachment variables to the model that
anova(m.full, m.mascfem)
```

```
## Analysis of Variance Table
##
## Model 1: neglect ~ secure + avoid + ambiv + masc + fem
## Model 2: neglect ~ masc + fem
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     62 140.66
## 2     65 147.44 -3    -6.774 0.9953 0.4011
```

```
summary(m.full)$r.squared - summary(m.mascfem)$r.squared
```

```
## [1] 0.03906492
```

(Notice that this method is similar to how R does stepwise regression, except we do it manually!)

6 Power Analyses (Post-Hoc)

Finally, let's consider . The SPSS lab will be using an interface called PiFace, but this requires extra downloading of external software, calculating values in R which you then transfer to a separate interface, etc. In my mind, this is (a) too much work, and (b) prone to human error. Instead, we will be using the pwr.f2.test() function from the pwr library in R to compute power. We will need the following ingredients:

- Degrees of freedom in our model

$$f^2 = \frac{R^2_{AB} - R^2_A}{1 - R^2_{AB}}$$

where R^2_A = variance accounted for in the population by variable set A
R^2_{AB} = variance accounted for in the population by variable set A and B together

Figure 1: effect size calculation

- The effect size of the variable we're interested in, via this equation:

- α: 0.05

Let's say we were interested in the power we had in detecting an effect of fem in our final model.

```
## Step 1: get df of model
df1 <- summary(m.final)$fstatistic["numdf"]
df2 <- summary(m.final)$fstatistic["dendf"]
## Step 2: get effect size of fem
r2.overall <- summary(m.final)$r.squared
r2.fem <- showme.lm(m.final)$RegressionTables$coefficients["fem", "sr2"]
f2.fem <- (r2.overall - r2.fem)/(1 - r2.overall)

power <- pwr.f2.test(u = df1, v = df2, f2 = f2.fem, sig.level = 0.05)
power
```

```
##
##      Multiple regression power calculation
##
##               u = 2
##               v = 65
##              f2 = 0.05891772
##       sig.level = 0.05
##           power = 0.3993918
```