General and Generalized Linear Models

http://datascience.tntlab.org

Module 9

Today's Agenda

- Some new *tidyverse*
- Correlations and covariances
- Review of **lm**() and **glm**()
  - More plotting
  - Stepwise and subsets
  - Model comparisons
  - Relative importance
- Some new *magrittr*

First, Some New *tidyverse*

- If you want to create new mean scores by row, you should do it with a pipe of **mutate()**; however, you'll need a new command, **rowwise()**

- Compare
  - **my_df <- bfi %>%**
    transmute(Conscientiousness = mean(c(C1, C2, C3, C4, C5), na.rm=T))

  - **my_df <- bfi %>%**
    rowwise() %>%
    transmute(Conscientiousness = mean(c(C1, C2, C3, C4, C5), na.rm=T)) %>%
    ungroup()

Correlations and Covariances

- You can create a correlation matrix in several different ways
  - To compute covariances, replace **cor**() with **cov**()

- Simplest: **cor(df)** # but only if all columns are numeric
- Next simplest: **cor(var1, var2)** # but you don't get a full matrix

- Note that any correlation with any NAs won't be calculated unless:
  - use=complete.obs # listwise deletion
  - use=pairwise.complete.obs # pairwise deletion

- Also note alternative correlation types:
  - **method="kendall"** or **method="spearman"**

- In *tidyverse*, very easy: **df %>% select(x1, x2, x3) %>% cor()**
- With *apaTables*, even better: **df %>% select(x1, x2, x3) %>% apa.cor.table()**

The General Linear Model

- Functions common to all **lm**() models
  - **coef**(lm) # returns predictor coefficients
  - **summary**(lm) # we've been using this already
  - **fitted**(lm) # shows fitted values for model (same as **fitted.values(lm)**)
  - **predict**(lm, df) # shows predicted values given new inputs, which will be
    # same as fitted values given model predictor inputs in general
    # linear modeling but may change otherwise

- Useful functions from *broom*
  - **tidy**(lm) # same as with ANOVA; cleans up the summary *df*
  - **augment**(lm) # provides a bit more info than **tidy**(), including residuals,
    # leverage/Cook's distances, fitted values, and SE`

- A potentially useful function from *QuantPsyc*
  - **lm.beta**(lm)

Diagnostic Plots and Assumption Tests

- **plot(lm)** calls the four major diagnostic plots
  - Residuals vs. Fitted: linearity and homoscedasticity (hamburger plot)
  - Scale-Location: homoscedasticity (hamburger plot)
  - Normal Q-Q: normality
  - Residuals vs. Leverage: influential cases
  - http://data.library.virginia.edu/diagnostic-plots/ if you've forgotten interpretation
  - Try calling after **par(mfrow=c(2,2))**

- A variety of more powerful plots and tests available in *car*, including:
  - **leveragePlots(lm)** # leverage plots
  - **avPlots(lm)** # added-variable (partial regression) plots for influential cases

  - **outlierTest(lm)** # Bonferonni p-value test on most extreme residual
  - **ncvTest(lm)** # non-constant variance test for homoscedasticity
  - **vif(lm)** # multicollinearity (variance inflation factors)

Parallel Slopes vs Interactive Models

- Parallel slopes is analogous to a ANCOVA without a defined interaction term
  - **lm**() will automatically recode categorical predictors as dummy codes **as long as it is correctly specified as factor**, so you do not need to add these variables to your df
  - For plotting: **ggplot(augment(mod), aes(x = x1, y = y, color = x2)) + geom_line(aes(y=.fitted))**
- Interactive models can be defined two ways, with either explicit or implicit interaction terms
  - **lm(y ~ x1 + x2 + x1:x2) == lm(y ~ x1 * x2)**
  - For plotting: **ggplot(mydf, aes(x = x1, y = y, color = x2, group = x2))) + geom_smooth(method="lm")**
- In both cases, giving **ggplot()** the correct aesthetic will create all specified regression lines

Stepwise Regression with *MASS*

- You can use stepwise regressions in R (but you probably shouldn't)
  - **model <- lm(y ~ x1 + x2 + x3 + x4 + x5, data=my_df)**
  - **step <- stepAIC(model, direction="forward")**
  - **step <- stepAIC(model, direction="backward")**
  - **step <- stepAIC(model, direction="both")**

- All-subsets regression is a better choice if you need to do this using *leaps* and *car* together
  - **model <- regsubsets(y ~ x1 + x2 + x3 + x4 +x5, nbest=10, data=my_df)**
  - **summary(model)**
  - **plot(model, scale="r2")**
  - **subsets(model, statistic="rsq")**

Comparison of Incremental/Nested Models

- **anova()** will compare any number of models

- **model1 <- lm(y ~ x1 + x2)**
- **model2 <- lm(y ~ x1 * x2)**
- **anova(model1, model2)**

- To find the change in R^2, you'll need to explore the model summary objects (**tidy()** will not find it) and compare them directly

Relative Importance

- From the *relaimpo* library

- **calc.relimp(lm, type="type", rela=T)**
  - **rela=T** to sum to 100%, **rela=F** to sum to R^2
  - Several types
    - **lmg**: R^2 contribution averaged over orderings (Chevan & Sutherland, 1991)
    - **pmvd**: proportional marginal variance (Feldman, 2005)
    - **pratt**: product of standardized coefficient and correlation
    - **genizi**: R^2 decomposition according to Genizi (1993)
    - **car:** R^2 decomposition according to Zuber and Strimmer (2010)

Generalized Linear Model

- Mostly uses **glm**(formula, family="type")
- Same format of formula from **lm**(), but incorporates a link function
  - binomial (logit), gaussian (normal), poisson (natural log), Gamma (chi)
  - See **?family** for complete list
- Logistic regression Nagelkerke pseudo-R^2 in *fmsb*: **NagelkerkeR2**(glm)

- Visualization is a bit more complex
  - **model <- glm(y ~ x, family=binomial, data=my_df)**
  - **ggplot(my_df, aes(x = x, y = y) +**
    geom_smooth(method="glm", method.args = list(family = "binomial"))

- Fitted values will default to those incorporating the link function
  - If you want the original scale, use **predict**() or
    augment(model, newdata=newdf, type.predict="response")

- For more advanced modeling, use **glm4**() from *MatrixModels*

Some New *magrittr* (not in *tidyverse*)

- Sometimes it is useful to pass the end-result of pipes into non-tidyverse functions
- For this, you can use "explode": **%$%**

- Example
  - **sat.act %>% filter(age==18) %>% cor(SATV, SATQ, use="complete")** # doesn't work
  - **sat.act %>% filter(age==18) %$% cor(SATV, SATQ, use="complete")** # does work

  - **sat.act %>% filter(age==18) %>% select(SATV, SATQ) %>% cor(use="complete")** # also works

Summary of Libraries and Functions

- Base-R: **summary()**
- Base-R/*graphics*: **plot()**, **par()**
- Base-R/*stats*: **lm()**, **glm()**, **cor()**, **cov()**, **fitted()**, **coef()**, **predict()**, **anova()**
- tidyverse/*ggplot2*: **ggplot()**
- tidyverse/dplyr: **rowwise()**
- *broom:* **tidy()**, **augment()**
- *apaTables*: **apa.cor.table()**
- *car:* **leveragePlots()**, **avPlots()**, **outlierTest()**, **ncvTest()**, **vif()**, **subsets()**
- *fmsb:* **NagelkerkeR2()**
- *magrittr:* **%$%**
- *QuantPsyc*: **lm.beta()**
- *leaps:* **regsubsets()**
- *relaimpo:* **calc.relimp()**