Guide No.
KUANT 025.0

KUant Guides

ANOVA and Regression in R

An introductory guide to general linear model (GLM) analyses using R.

Pornprasertmanit, S. , Jorgensen, T. D., & Jia, F. (2013)

www.crmda.ku.edu

There exists already a KUant Guide that is an Introduction to R (KUant Guide #20). In this KUant Guide, we assume that you already have some exposure to R (so it is already installed on your computer) and that you are already familiar with the fundamentals of the general linear model (e.g., ANOVA, ANCOVA, and multiple regression). So we will begin with an introduction to the example data set, run some descriptive statistics, and show you how to check some basic assumptions for ANOVA and regression. We will then proceed with a one-way and a two-way factorial ANOVA, followed by single and multiple regression (including interactions between continuous and categorical variables, and between two continuous variables), regression diagnostics, and end with a demonstration of how to estimate regression parameters when there are missing data.

Get to Know Your Data

In addition to many functions included in the distribution of R, we will use some functions from five other packages:

```
library(car)
library(psych)
library(QuantPsyc)
library(phia)
library(multcomp)
```

If you need to install any of these, put the name of the package in quotes below.

```
install.packages("")
```

Attach data from the "psych" package.

```
data(sat.act)
?sat.act
```

Look at variable names, and print the first 10 observations

```
colnames(sat.act)
head(sat.act, 10)
```

Save the gender variable as an indicator for MALE (0 = female), and save the education variable as an ordered categorical factor (not numeric 0–5).

```
sat.act$male <- ifelse(sat.act$gender == 1, 1, 0)
sat.act$ed <- as.factor(sat.act$education)
```

View frequencies and cross-tabulation of gender and education.

```
tab <- table(Sex = sat.act$male, Education = sat.act$ed)
addmargins(tab)
```

```
round(addmargins(tab) / nrow(sat.act), 3)
```

View descriptive statistics of continuous variables (in columns 3–6).

```
summary(sat.act[ , 3:6])
describe(sat.act[ , 3:6]) # available in "psych" package
```

View correlations among continuous variables.

```
cor(sat.act[ , 3:6]) # SATQ has missing values, just use complete data
cor(sat.act[ , 3:6], use = "pairwise.complete.obs")
corr.test(sat.act[,3:6]) # available in "psych" package
```

Check Assumptions for ANOVA

We will now look at some descriptive statistics and graphs that help us assess whether some key assumptions have been met:

- **Independence of observations** — This is a design issue that cannot be verified statistically, although if the assumption has been violated, it has noticeable effects on statistical results.
- **Homoscedasticity (homogeneity of variances)** — Groups should have variances of similar size. ANOVA is robust to moderate heteroscedasticity when the group sample sizes are similar.
- **Normally distributed dependent variable** — The outcome should be normally distributed, although ANOVA is robust to moderate departures from normality.

Check for Normality and Outliers

Graphs to check normality include quantile–quantile (QQ) plots and histograms.

```
qqnorm(sat.act$ACT)
qqline(sat.act$ACT)
```

Do you see the ceiling effect at the maximum possible score of ACT = 36? Do you notice any outliers?

```
table(sat.act$ACT)
which(sat.act$ACT == 3)
sat.act[440, ]
```

If we have a justified reason, we can remove the outlier, or transform to the next-lowest observed score but preserve original rank.

```
sat.act[440, "ACT"] <- 14
sat.act[440, ]
```

We can also detect possible outliers using boxplots.

```
boxplot(sat.act$ACT)
```

We will use a smoothed histogram and compare the observed distribution to what we would expect if it were perfectly normal.

```
plotNormX(sat.act$ACT)
```

We found univariate skewness and kurtosis with `psych::describe()`. There are also separate functions for skew/kurtosis in the `psych` package.

```
skew(sat.act$ACT)
kurtosi(sat.act$ACT)
```

Some common transformations for non-normal data include taking the square-root, log, or inverse. Although age is not our dependent variable, it is slightly skewed, so we will use it as an example.

```
hist(sat.act$age)
plotNormX(sat.act$age)
```

Square-root transformation is for moderate skew.

```
rootAge <- sqrt(sat.act$age)
plotNormX(rootAge)
```

Log transformations are for greater skew.

```
logAge <- log(sat.act$age)
plotNormX(logAge)
```

Inverse transformations are only for very severe skew.

```
invAge <- 1 / sat.act$age
plotNormX(invAge)
```

Those are heuristics, but this is a direct Normal Translation function in the `QuantPsyc` package. However, it cannot be transformed back to the original metric.

```
normedAge <- Normalize(sat.act$age)
plotNormX(normedAge)
```

Check for Heteroscedasticity across Groups

Print the variance or standard deviation in each group. We will eventually see a 2 × 5 two-way factorial ANOVA, so let's check the homogeneity assumption across all 10 subgroups of Sex and Education.

```
aggregate(ACT ~ ed + male, data = sat.act, FUN = sd)
```

They look to be generally similar in size. Let's get a visual with boxplots.

```
boxplot(ACT ~ male, data = sat.act)
boxplot(ACT ~ ed, data = sat.act)
```

Use color-coding if you want both groups at once.

```
table(sat.act$male) # the order is (0, 1), i.e. (female, male)
boxplot(ACT ~ male + education, data = sat.act, col = c("pink",
"lightblue"))
```

We can test whether variances across groups are equal using Bartlett's or Levene's test.

```
bartlett.test(ACT ~ male + education, data = sat.act)
leveneTest(sat.act$ACT, group = sat.act$male) # from the "car" package
```

Now that we have tested our primary assumptions (after taking our study design into account in order to ensure independence of observations), we can confidently conduct a one-way and two-way ANOVA.

One-Way and Factorial ANOVA

One-Way ANOVA

Analysis of variance is usually used to investigate a mean difference between groups. The results will show how likely the group means are different by chance. The simplest model is to compare mean difference between two groups. For example, the difference in ACT between males and females are investigated. ANOVA is actually a submodel of a general framework, called general linear model, which also includes regression, multivariate analysis of variance, and multivariate regression. To run a general linear model, the `lm` function is used. For example, to run a simple regression, ACT is predicted by SAT-Verbal score.

```
mod <- lm(ACT ~ SATV, data = sat.act)
summary(mod)
```

You may notice that running the `lm` function does not provide any output. Actually, the result can be saved in an object with any names that we wish. In this case, the result is saved in `mod`. To ask for a result, the `summary` function is used on the saved object.

In this example, in the `lm` function, there are two arguments separated by commas. The first argument is a formula. The dependent variable is put on the left of the tilde, `~`. The independent variable is listed on right hand side of the tilde. The second argument, `data`, is the name of the target data set.

For analysis of variance, the `lm` function can be used by listing a categorical variable on the independent variable list. However, the categorical variable must be in a factor format within a target data set. For example, the gender difference in ACT score is investigated. First, the gender variable must be in a factor format by the `factor` function as mentioned above:

```
sat.act$gender <- factor(sat.act$gender, labels = c("male", "female"))
```

The `labels` argument is used for an easier interpretation. Because we will use the education variable later, the education variable is transformed into a factor format as well:

```
sat.act$education <- factor(sat.act$education, labels=paste0("L",0:5))
```

Next, the `lm` function is used to run ANOVA:

```
genderDiff <- lm(ACT ~ gender, data = sat.act)
summary(genderDiff)
```

The `genderfemale` line in the `Coefficients` section indicates the regression coefficient of the dummy variable when female is coded as 1 and male is coded as 0. This format is not similar to the ANOVA output from other statistical packages. To get a result similar to other statistical packages, there are two options. First, the `anova` function can be used on the `lm` output:

```
anova(genderDiff)
```

The second method is to use the `aov` function instead of the `lm` function:

```
genderDiff <- aov(ACT ~ gender, data = sat.act)
```

```
summary(genderDiff)
```

Note that the `aov` function and the `lm` function are basically the same thing. The `aov` function will simply adjust the output to provide results in the ANOVA framework.

Because gender has two categories, one can use independent *t*-test to analyze data:

```
t.test(ACT ~ gender, data = sat.act)
```

This *t*-test, however, does not assume homogeneity of variance as the `aov` or `lm` functions mentioned above. The result may be slightly different.

As you may expect, the `aov` or `lm` functions can be used for a factor with more than two groups. For example, the difference in ACT between education levels is investigated:

```
educDiff <- aov(ACT ~ education, data = sat.act)
summary(educDiff)
```

A Priori Contrasts and Post-Hoc Comparison

To investigate the mean differences, a priori contrasts or post hoc analysis can be used. For the post hoc analysis, there are many methods for controlling familywise error rate in a post hoc analysis. One of the popular methods is Tukey–Kramer method, which generalize the Tukey method, accounting for unequal sample size. The `TukeyHSD` function can be used:

```
TukeyHSD(educDiff, "education")
```

The first argument is the result from the `aov` function. The second argument is the name of categorical variable being used for pairwise comparison. The result provides both adjusted *p*-value and simultaneous 95% confidence interval. Note that the `TukeyHSD` function works with the object created from the `aov` function only (not with the object from `lm` function).

There are many more routines in R available for multiple comparisons. The `multcomp` package provides a framework for multiple comparisons in not only analysis of variance but other statistical models too, such as logistic regression.

```
library(multcomp)
pairwise <- glht(educDiff, linfct = mcp(education = "Tukey"))
summary(pairwise, test = adjusted(type = "bonferroni"))
```

The `glht` function is used to create a test for multiple contrasts. The contrasts can be all-possible pairwise comparisons, which is referred to as `"Tukey"` in this package. Then, the result of multiple contrasts can be summarized and adjusted for familywise error rate. The Bonferroni method is used in this example. Note that if the test argument is not specified, the single-step approach is used. The adjusted *p* values is computed from the joint normal or *t* distribution of the *z* statistics such that the *p* value represents the probability of getting at least one significant result by chance if all *z* or *t* values are the same in all contrasts. The Tukey method and the single-step approach will provide the same results if the group sizes are equal. The simultaneous confidence interval can be computed by the `confint` function:

```
confint(pairwise)
```

I believe that the confidence interval here is adjusted using the single-step approach because the results are congruent with the result from the `summary` function with the single-step approach. The confidence interval can be plotted:

```
plot(confint(pairwise))
```

Please check the help pages for the type of contrasts and type of familywise error rate adjustments:

```
?glht
?summary.glht
?confint.glht
```

Instead of post hoc comparison, researchers may have a priori contrasts from their research hypotheses. For example, researchers expect a linear trend in the impact of education on ACT score. The contrast coefficient of the linear trend for six groups can be created:

```
ctr <- matrix(c(-2.5, -1.5, -0.5, 0.5, 1.5, 2.5), 1, 6)
```

The polynomial contrast coefficient is based on Table A10 in Maxwell and Delaney (2004). The contrast is specified in a matrix where rows represent different contrasts and columns represent the coefficient of each group. The contrast can be evaluated using the `glht` function from the `multcomp` package:

```
linear <- glht(educDiff, linfct = mcp(education = ctr))
summary(linear)
```

The contrast is provided in the `linfct` argument. Because only one contrast is tested, the familywise error rate correction is not needed.

Next, all polynomial contrasts up to the fifth order are investigated for the difference in ACT between education levels:

```
linear <- c(-5, -3, -1, 1, 3, 5)
quadratic <- c(5, -1, -4, -4, -1, 5)
cubic <- c(-5, 7, 4, -4, -7, 5)
quartic <- c(1, -3, 2, 2, -3, 1)
quintic <- c(-1, 5, -10, 10, -5, 1)
mctr <- rbind(linear, quadratic, cubic, quartic, quintic)
```

As mentioned above, the row of the contrast matrix represents different contrasts. The matrix of multiple contrasts can be used in the `glht` function:

```
polynomial <- glht(educDiff, linfct = mcp(education = mctr))
summary(polynomial, test=adjusted(type = "bonferroni"))
```

Factorial ANOVA

In factorial ANOVA, more than one categorical variable is used as independent variables. Each categorical variable will have their main effect on the target dependent variable. In addition, the interaction effect between categorical variables can be used to predict the dependent variable. In R, two signs can be used to represent interaction between independent variables: "`:`" or "`*`". The colon represents the interaction effect only whereas the asterisk represents the interaction effect and the main effect of each variable (and all lower-order interactions). Thus, the following lines are equivalent:

```
twoway <- aov(ACT ~ education + gender + education:gender, data = sat.act)
twoway <- aov(ACT ~ education*gender, data = sat.act)
```

The result can be summarized:

```
summary(twoway)
```

When group sample sizes are not equal, different sum of squares can be computed. Note that the type of sum of squares will not change the sum of square of the highest interaction (two-way interaction in this case). The type of sum of squares will affect the sums of squares of main effects (and lower-order interactions). If the `summary` function is used, the sum of squares type 1 is provided, which means the effect of the current effect after the effects listed above are controlled. Thus, the education effect is not controlled by anything. The effect of gender is controlled by education. The interaction effect is controlled by the main effects of education and gender. If the gender effect is listed first, the resulting sums of squares will be different.

Researchers may request the sum of squares type 2 and 3 by using the `Anova` function from the `car` package:

```
library(car)
Anova(twoway, type = 2)
Anova(twoway, type = 3)
```

Note that the sum of squares type 2 represents the effect of controlling other variables at the same level. Thus, the education effect is controlled by gender. The effect of gender is controlled by education. The interaction effect is controlled by the main effects of education and gender. If the gender effect is listed first, the resulting sums of squares will not be different. The sum of squares type 3 represents the effect of controlling all other variables in the model. Thus, the education effect is controlled by gender and the interaction. The effect of gender is controlled by education and the interaction. The interaction effect is controlled for the main effects of education and gender. If the gender effect is listed first, the resulting sums of squares will not be different.[1] Anyway, the interpretation of the main effects should be more careful when the interaction effect exists because the difference between one categorical variable depends on the level of another categorical variable.

Simple Main Effects

To investigate interactions, the examination of the means of each condition would be helpful. The `aggregate` function can be useful:

```
aggregate(ACT ~ education*gender, data = sat.act, FUN = mean)
```

[1] The result of sum of square type 3 is different from the SAS output, which the sums of squares were 511, 20, and 230. The author of the Anova function, John Fox, realized the difference. He mentioned in the R-help mailing list, http://tolstoy.newcastle.edu.au/R/help/05/11/16368.html, that

> "The Anova() function in car calculates "Type-III" (and "Type-II") tests differently from SAS. (The difference originates in the fact that SAS uses a deficient-rank parameterization of the model while R uses a full-rank parametrization; it would be possible to mimic SAS's behaviour more closely, but I think that there are problems with it.) As a consequence, you have to use a contrast-generating function, such as contr.helmert or contr.sum (but not contr.SAS), that provides contrasts that are orthogonal in the row-basis of the model matrix."

To get the same result, researchers must change the options of the car package by the following codes:

```
options(contrasts=c("contr.sum","contr.poly"))

Anova(twoway, type=3)
```

Alternatively, the `phia` package provides the `interactionMeans` function to investigate the condition means from the result from the `aov` function directly:

```
interactionMeans(twoway)
```

In this case, the `aggregate` and `interactionMeans` functions provide the same results. When covariates are included in the ANOVA model, the `interactionMeans` function will provide adjusted means, which will be described in the ANCOVA section.

The results from the `interactionMeans` function can be plotted:

```
plot(interactionMeans(twoway))
```

If the interaction effect between education and gender was significant, researchers would wish to see the ACT score difference between gender groups at each education level or the ACT score difference between education levels within each gender group. This comparison is also referred to as simple main effects. The `testInteractions` function in the `phia` package can be used:

```
testInteractions(twoway, fixed = "education", across = "gender")
```

The result showed the gender difference at each level of education. The `across` argument is the factor to be investigated. The `fixed` argument is the factor to be classified into subgroups (or moderators). As a default, the *p*-value is adjusted by the Holm's method, which is similar to Bonferroni method (see Howell, 2007, for further details). The ACT difference between education levels within each gender group can be examined:

```
testInteractions(twoway, fixed = "gender", across = "education", adjustment
                 = "bonferroni")
```

The `adjustment` argument represents the method of controlling for familywise error rate, which is Bonferroni's method in this example. The post hoc pairwise comparison among education levels within gender group can be implemented by using the `pairwise` argument:

```
testInteractions(twoway, pairwise = "education", fixed = "gender")
```

A priori contrast can be implemented in the `testInteractions` function as well. However, the implementation are a little complicated. First, a matrix of contrast is constructed. Unlike the `multcomp` package, the `phia` package use columns to represent different contrasts and rows represent different groups:

```
ctr <- cbind(linear, quadratic)
```

Next, a list of a contrast is built where the name of the contrast is the factor the contrast applies:

```
total.ctr <- list(education = ctr)
```

Finally, the list of contrasts is specified in the custom argument:

```
testInteractions(twoway, custom = total.ctr, fixed = "gender")
```

Analysis of Covariance

Researchers may have covariates that they wish to control for their influences in the mean comparisons. For example, the ACT scores difference across education levels are investigated controlling for SAT-Q and SAT-V

scores. The `aov` or `lm` functions can still be used for running ANCOVA. The order of variables putting in the list of independent variables influences the result of the `summary` function.

```
ancova1 <- aov(ACT ~ SATV + SATQ + education, data = sat.act)
ancova2 <- aov(ACT ~ SATV + SATQ + education, data = sat.act)
summary(ancova1)
summary(ancova2)
```

As mentioned in the factorial ANOVA section, the output from the `summary` function is sum of square type 1, which indicates the effect of the current variables controlling for the effect of other variables listed above. Therefore, as a good practice, covariates should be listed first in the right hand side of the tilde.

Alternatively, the `Anova` function with sum of square type 2 or 3 (having the same meaning in this case) provide the same result for any order of independent variables:

```
Anova(ancova1, type = 2)
Anova(ancova2, type = 2)
```

The adjusted means, which is the expected group means when the covariates equal to their means, can be computed by the `interactionMeans` function:

```
interactionMeans(ancova1, factors = "education")
```

Actually, the `factors` argument is not necessary because the education variable is the only factor variable in the model. In factorial ANCOVA, the adjusted marginal means can be computed by specifying the `factors` argument.

The post hoc pairwise comparisons using Tukey procedure can be implemented:

```
TukeyHSD(ancova1, "education")
```

As you may expect, the `glht` function from the `multcomp` package can be used for multiple comparisons or a priori contrast:

```
pairwisec <- glht(ancova1, linfct = mcp(education = "Tukey"))
summary(pairwisec, test = adjusted(type = "bonferroni"))
confint(pairwisec)
ctr <- rbind(linear)
linearc <- glht(ancova1, linfct = mcp(education = ctr))
summary(linearc)
mctr <- rbind(linear, quadratic)
polynomialc <- glht(ancova1, linfct = mcp(education = mctr))
summary(polynomialc, test = adjusted(type = "bonferroni"))
```

Actually, the `testInteractions` function can be used to perform a priori contrasts as well:

```
testInteractions(ancova1, pairwise = "education")
testInteractions(ancova1, custom = total.ctr)
```

If gender is also of interest, two-way ANCOVA can be performed. The `aov` function can be used and, as a good practice, the covariates should be listed first:

```
twoancova <- aov(ACT ~ SATV + SATQ + education*gender, data = sat.act)
```

The sum of square type 1, 2, and 3 can be calculated[2]:

```
summary(twoancova)
Anova(twoancova, type = 2)
Anova(twoancova, type = 3)
```

All functions listed in the factorial ANOVA section are still valid in two-way factorial ANCOVA:

```
interactionMeans(twoancova)
plot(interactionMeans(twoancova))
testInteractions(twoancova, fixed = "education", across = "gender")
testInteractions(twoancova, fixed = "gender", across = "education")
testInteractions(twoancova, pairwise = "education", fixed = "gender")
testInteractions(twoancova, custom = total.ctr, fixed = "gender")
```

Simple and Multiple Regression

Simple Regression

Simple regression model is used to describe the relationship between two variables, an independent variable (IV) and a dependent variable (DV). The research question answered by simple regression is whether IV has effect on DV, and based on a significant effect people can use the model to make predictions for DV values for any new values of IV. When assuming a linear relationship between the two variables, the mathematical expression of a simple regression model is $y_i = b_0 + b_1 x_i + \varepsilon_i$, where b_0 and b_1 are regression coefficients and ε is an error term. The DV must be a continuous variable, but the IV can be continuous or categorical. One-way ANOVA is a special case of simple regression, where we have a categorical IV. In this section we discuss the cases where the IV is continuous.

R provides the `lm()` function for linear model. With the `sat.act` data, suppose we are interested in whether `SATV` has effect on `ACT` scores, the model can be specified in the following way.

```
lm(ACT ~ SATV, data = sat.act)  # ACT(DV)is explained by age(IV).
```

The output only provides the estimates of the two regression coefficients: intercept (b0) and slope (b1).

The `lm()` function generates more information than we can directly see in the above output. There are a couple of extractor functions that can be used to obtain more information from the modeling function. Let us first save the modeling function into an R object `sReg`, then use the `summary()` function on it.

```
sReg <- lm(ACT ~ age, data = sat.act)
summary(sReg)
```

The `coef()` returns the coefficients only.

```
coef(sReg)
```

The variance -covariance matrix of the estimated intercept and slope can be obtained by the `vcov()` function.

```
vcov(sReg)
```

[2] The sum of squares type 3 will not match with SAS. The sums of squares for SATV, SATQ, education, gender, and education*gender from R are 828, 1361, 188, 41, 147, but from SAS are 828, 1361, 306, 1, 147. The same trick mentioned in the factorial ANOVA section can be used to get the matched result.

The standard error for each coefficient can be computed by taking square root of its covariance.

```
seInt <- sqrt(vcov(sReg)[1,1])
seInt
seSlope <- sqrt(vcov(sReg)[2,2])
seSlope
```

The `confint()` function generates the confidence intervals for the two coefficients.

```
confint(sReg)
```

The `anova()` function extracts the various sums of squares and F test result from the linear model.

```
anova(sReg)
```

After drawing a scatterplot of IV and DV, we can superimpose a best-fit line using the `abline()` function.

```
plot(ACT ~ SATV, data = sat.act)
abline(sReg)
```

Alternatively, we can directly specify the intercept (a) and slope (b) of the predict regression line in the `abline()` function.

```
plot(ACT ~ SATV, data = sat.act)
abline(13.872, 0.024)
```

Multiple Linear Regression

Multiple linear regression allows more than one IVs to predict the dependent variable (DV). To examine effects of multiple IVs on the DV, we can simply add IVs in the linear model, $y_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + ...+ b_p x_{pi} + \varepsilon_i$. Take the `sat.act` data for example, suppose we are interested in how well `SATV` and `SATQ` predict `ACT`.

```
mReg1 <- lm(ACT ~ SATV + SATQ, data = sat.act)
summary(mReg1)
```

Based on the result, we find that `SATV` and `SATQ` both significantly predict `ACT` scores. Both predictors account for a significant amount of variation in `ACT`.

We can also add in `age` and `education` build up a second model, which examines whether `age` and `education` have any effect on `ACT` scores, controlling for `SATV` and `SATQ`.

```
mReg2 <- lm(ACT ~ SATV +  SATQ + age + education, data = sat.act)
summary(mReg2)
```

All of the four predictors are significant in the second model. And the second model accounts for a significant amount of variation in the `ACT`. But, is the gain in prediction by adding two more predictors significant? The `anova()` function can be used to test the gain in prediction between the two nested models.

```
anova(mReg1, mReg2, test = "F")
```

Based on the result of model comparison, we conclude that age and education explained a substantial amount of variation in `ACT` above and beyond `SATV` and `SATQ`. The four-predictor model prevides better information.

Regression with Categorical Predictor(s)

As we have discussed, when all the predictors (IVs) are categorical, regression is equivalent to ANOVA. But regression is more flexible because it can handle both continuous and categorical predictors in a same model.

Before including a categorical predictor in a regression model, we have to make sure that it is interpretable as a "factor" in R. In the `sat.act` data, the variable `gender` is coded as 1 = males and 2 = females. If we want to examine the effect of `gender` on the `ACT` score, the values of 1 and 2 should not be directly used in the regression model. We need to first convert `gender` into a factor.

```
sat.act$gender <- factor(sat.act$gender, labels = c("male","female"))
```

The above command converts `gender` into a factor and also creates labels `"male"` and `"female"` for the two categories (levels) of the variable. Note that the order of the labels matters. Since the category `males` corresponds to a smaller value (1 = males) in the original data, the label `"male"` should go first. To check the levels of a categorical variable, use the `levels()` function.

```
levels(sat.act$gender)
```

By default, factors are treated as dummy code variables, and the "lowest" value (first) level is treated as the "baseline" category. Let us take a look at the current coding scheme of `gender` using the `contrasts()` function. And we can see that the baseline category is defaulted to `male`.

```
contrasts(sat.act$gender)
```

Now try a simple regression with one predictor `gender`.

```
cReg1 <- lm(SATQ ~ gender, data = sat.act)
summary(cReg1)
```

The slope estimate (-39.878) represents the difference in mean `SATQ` scores between females and males. The negative number means that on average females have low `SATQ` scores than males.

Sometimes it is desirable to have a different baseline category. The `relevel()` function can easily change the baseline category.

```
sat.act$gender.new <- relevel(sat.act$gender, "female")
contrasts(sat.act$gender.new)
```

Now `female` becomes the baseline. Let us run the model again.

```
cReg2 <- lm(SATQ ~ gender.new, data = sat.act)
summary(cReg2)
```

Besides the categorical predictor, we can also add continuous predictors into the model. For example, we are interested in whether `education` affects `SATQ`, controlling for `gender`.

```
cReg3 <- lm(SATQ ~ gender.new + education, data = sat.act)
summary(cReg3)
```

Another way to test whether it is worthwhile to add `education` into the model is to perform the `anova` function on the two nested models.

```
anova(cReg2, cReg3, test = "F")
```

Interactions

If the effect of x_1 on y depends on the specific values of a third variable x_2, we say that x_1 interacts with x_2. x_2 is called a moderator of the relationship of x_1 and y. To model interaction, we need to take product of the two predictors and add the product term in the regression equation. For example, we hypothesize that `SATQ` is the moderator of the relationship of `SATV` and `ACT`, controlling for other predictors. In the `lm()` function, we specify the interaction and the two main effects by simply using `*` to connect `SATV` and `ACT`.

```
iReg1 <- lm(ACT ~  SATV * SATQ + age + education, data = sat.act)
```

There are two equivalent ways to test the interaction effect, (1) t-test of the regression coefficient for the product term,

```
summary(iReg1)
```

and (2) *F* test of the gain in prediction by adding the product term in the model.

```
anova(mReg2, iReg1)
```

Based on the *t* or *F* test, we conclude that the effect of `SATV` on `ACT` depends on the specific values of `SATQ`. The `rockchalk` package provides tools to plot and test the effect of `SATV` on `ACT` at different values of `SATQ`. Here we discuss two functions: `plotSlopes()` and `testSlopes()`. Before making the plots, we need to choose some representative values in `SATQ`. Without specification, the `plotSplopes()` function will choose three values, the 1st quartile, median, and the 3rd quartile.

```
plotSlopes(iReg1, plotx = "SATV", modx = "SATQ")
```

We can also manually specify some values, for example, 200, 400, 600 and 800.

```
plotSlopes(iReg1, plotx = "SATV", modx = "SATQ", modxVals =
c(200,400,600,800))
```

Some people prefer the mean, one standard deviation above mean, and one standard deviation below mean.

```
plotSlopes(iReg1, plotx = "SATV", modx = "SATQ", modxVals = "std.dev.")
```

The slopes of `ACT` on `SATV` at specific values of `SATQ` are called simple slopes. To testing the simple slopes, we need to save the `plotSlopes()` object first and then apply the `testSlopes()` function on it.

```
ss <- plotSlopes(iReg1, plotx = "SATV", modx = "SATQ")
testSlopes(ss)
```

Same functions apply to the situation where we have a categorical moderator. Since categorical variables usually have limited numbers of levels, we can plot and test simple slopes on all levels rather than choosing some representative values.

```
iReg2 <- lm(ACT ~  SATV * gender + age + education, data = sat.act)
plotSlopes(iReg2, plotx = "SATV", modx = "gender")
```

Rescaling and Standardized Regression

Sometimes people want to rescale variables to make results more interpretable. If we feel that a 1-unit change in `SATV` and `SATQ` is not useful for our interpretation, we can change it to 100-unit increments.

```
sat.act$SATV.T <- sat.act$SATV / 100
sat.act$SATQ.T <- sat.act$SATQ / 100
tReg1 <- lm(ACT ~ SATV.T + SATQ.T, data = sat.act)
```

```
summary(tReg1)
```

The `scale()` function can linearly transforms a numeric variable to have a mean of 0 and a *SD* of 1 (standardization). We can obtain standardized regression coefficients by using the standardized DV and IVs in the model.

```
tReg2 <- lm(scale(ACT) ~ scale(SATV) + scale(SATQ), data = sat.act)
summary(tReg2)
```

Nonlinear Regression

R also allows us to fit nonlinear models to our data. To include a squared term in the model, we have to use the "identity" protection function using the `I()` syntax.

```
nReg1 <- lm(ACT ~  SATQ + I(SATQ^2), data = sat.act)
summary(nReg1)
```

An identical alternative is to create a squared term outside of the model:

```
sat.act$SATQ2 <- sat.act$SATQ^2
nReg2 <- lm(ACT ~  SATQ + SATQ2, data = sat.act)
summary(nReg2)
```

We can also specify an exponential relationship between our variables. To do that, we take the natural log of either or both sides of the regression equation.

```
sat.act$logSATQ <- log(sat.act$SATQ)
sat.act$logACT <- log(sat.act$ACT)
nReg3 <- lm(logACT ~ logSATQ, data = sat.act)
summary(nReg3)
```

Check Assumptions of Linear Regression (Post Hoc)

We will now look at some descriptive statistics and graphs that help us assess whether some key assumptions of regression have been met:

- Residuals are independent and identically distributed — Independence is (again) a design issue that cannot be verified statistically, although if the assumption has been violated, it has noticeable effects on statistical results. The “identical distribution” part essentially refers to normality and homoscedasticity across all levels of the predictors.
- Homoscedasticity (homogeneity of variance) — Residuals should have similar variance across all levels of the predictor(s).
- Normally distributed residuals — The residuals should be normally distributed.
- Linearity — The relationship between predictors and outcome is a linear one. This can include curvilinear (e.g., quadratic) relationships, but linearity means that there is nothing more complex than multiplying the predictors by some number (not other parameters) and adding them together.
- Multicollinearity — The predictors are not so highly correlated that they are redundant. There is no assumption that the predictors must be independent of each other, but if the predictors are highly

correlated, then they are both explaining the same information in the outcome. This has undesired side effects, such as very high standard errors and inconsistency of estimates across independent samples (these are related to each other because *SE*s estimate uncertainty).

Although these assumptions seem different than those of ANOVA because they refer to the residuals, they are really the same assumptions. In ANOVA, the residuals are merely differences from group means, so assessing the normality and variances of these residuals (which are mean-centered) is identical to assessing normality and variances of the raw scores (which are not mean-centered).

Check for Normality and Homoscedasticity

Using this model, we will check the distribution of variances. The model object includes a vector of residuals.

```
mod1 <- lm(ACT ~ SATV * SATQ + age, data = sat.act)
mod1$residuals
```

Check the normality assumption.

```
qqnorm(mod1$residuals)
qqline(mod1$residuals)
plotNormX(mod1$residuals)
```

We check the heteroscedasticity assumption by plotting the residuals against (a) the predicted values of the outcome and (b) each of the predictors.

```
## check that the residuals are random across predicted values
plot(mod1$residuals ~ mod1$fitted.values)

## check that the residuals are random across predictors
plot(mod1$residuals ~ mod1$model$SATQ)
plot(mod1$residuals ~ mod1$model$SATV)
plot(mod1$residuals ~ mod1$model$age)
```

Check for Linearity and Multicollinearity

Check linearity with a scatterplot matrix as a quick way to see several scatterplots at once.

```
pairs(sat.act[ , 3:6])
```

Add a loess curve (and 95% confidence lines) to check linearity using the "car" package

```
scatterplot(sat.act$SATQ, sat.act$SATV)

## "car" also provides a matrix with more information
scatterplotMatrix(sat.act[ , 3:6])

## as does "psych"
pairs.panels(sat.act[ , 3:6])
```

Check the R^2 values (which assume a linear relationship) to see whether there is multicollinearity.

```
round(cor(sat.act[ , 3:6], use = "pairwise.complete")^2, 3)
```

Squaring the matrix doesn't do matrix algebra. It just squares each element.

Check tolerance/VIF using "car" package. First, regress the DV and all IVs on an unrelated variable, like ID (or any variable that is in your `data.frame` but not in your model). That way, you can include all continuous predictors AND the outcome as "predictors" in this first step.

```
mod2 <- lm(gender ~ ACT + age + SATQ + SATV, data = sat.act)
```

Then you can check multicollinearity among all items, using the variance inflation factor (VIF) or tolerance (which is the inverse/reciprocal of VIF).

```
vif(mod2) # any values > 10 are problematic
## tolerance is the reciprocal of the variance inflation factor
1 / vif(mod2) # any values < 0.1 are problematic
```

Estimating Regression Parameters with Missing Data

The `SATQ` variable has some missing observations, so the regression parameters were estimated using a subsample $N = 687$ instead of the full sample $N = 700$. This is a small percentage of missing values:

```
mean(is.na(sat.act$SATQ))
```

but missing values can still bias parameter estimates. Here, we introduce the two preferred methods for handling missing data: Multiple Imputation (MI) and Full-Information Maximum Likelihood (FIML).

Multiple Imputation (MI)

Do not include redundant variables in your imputation model.

```
colnames(sat.act)
dat <- sat.act[ , 3:8]
colnames(dat)
```

Which rows have missing values?

```
myNAs <- which(is.na(dat$SATQ))
dat[myNAs, ]
```

Impute missing values using the `Amelia` package (others available: `mi` and `mice`). Indicate which variables are nominal (`noms`) or ordinal (`ords`), along with how many imputations you want.

```
library(Amelia)
 myImps <- amelia(dat, m = 20, ords = "ed", noms = "male")
```

Compare imputed values across imputations.

```
myImps$imputations$imp1[myNAs, ]
myImps$imputations$imp2[myNAs, ]
```

Analyze the entire list of imputed data sets with the same model. The `amelia:mi.meld()` function automatically combines the results across imputations, and there is an example of how to use it on the help page:

```
?mi.meld

betas <- NULL
```

```
SEs <- NULL
for (i in 1:20) {
  output <- lm(ACT ~ SATV*SATQ + age, data = myImps$imputations[[i]] )
  betas <- rbind(betas, output$coef)
  SEs <- rbind(SEs, coef(summary(output))[ , 2])
}
combined <- mi.meld(q = betas, se = SEs)
combined
```

Compare with the listwise deletion results.

```
MI <- data.frame(Estimate = t(combined$q.mi), Std.Error = t(combined$se.mi),
t.value = t(combineResults$q.mi / combineResults$se.mi))
round(MI, 4)
round(coef(summary(mod1)), 4)
```

Full-Information Maximum Likelihood (FIML)

Use FIML by specifying a regression model as a path analysis in the SEM software package `lavaan`. (Note that there is also a KUant Guide available about the software package `lavaan`.)

```
library(lavaan)
myModel <- "ACT ~ SATV + SATQ + age + SAT.interact"
```

The downside is that SEM does not automatically create product terms for interactions, but we can create this interaction variable ourselves.

```
sat.act$SAT.interact <- sat.act$SATV * sat.act$SATQ
```

Fit the model to the data (with the missing values) using listwise deletion, which is the default in `lavaan::sem()`, and should match the results from `lm()`.

```
listwise <- sem(myModel, data = sat.act, meanstructure = TRUE)
summary(listwise)
## compare to lm() results
round(coef(summary(mod1)), 3)
```

Fit the model to the data (with the missing values) using FIML.

```
FIML <- sem(myModel, data = sat.act, meanstructure = TRUE, missing = "fiml")
summary(FIML)
```

Compare to MI results, which should match FIML if we had infinite imputations.

```
round(MI, 3)
```