# Module 4: Regression Methods: Concepts and Applications

## *Example Analysis Code*

*Rebecca Hubbard, Mary Lou Thompson*

*July 11-13, 2018*

---

## Install R

- Go to http://cran.rstudio.com/ (http://cran.rstudio.com/)
- Click on the "Download R for [operating system]" link that is appropriate for your operating system and follow the instructions.
- Open R and make sure it works (i.e. that no error messages come up)

## Install RStudio

- Go to http://www.rstudio.com/products/rstudio/download/ (http://www.rstudio.com/products/rstudio/download/)
- Select the installer that is appropriate for your operating system under "Installers for Supported Platforms" and follow the instructions.
- Open RStudio and make sure it works.

## Install R packages

- For this module we will be using the *gee*, *multcomp* and *lmtest* packages
- To use these packages you first need to install them using install.packages()

```
install.packages("gee")
install.packages("multcomp")
install.packages("lmtest")
```

## Load libraries

- From within R Studio you will need to load the following libraries:

```
library(gee)
library(multcomp)
library(lmtest)
```

- After the first time you install the packages on your computer, you will only need to load the libraries in the future

# Class Github repository

- Materials related to the labs for this module can be found in our Github repository: https://github.com/rhubb/SISG2018 (https://github.com/rhubb/SISG2018)
- This sample code can be found there in the *exercises* directory
- You will submit your solutions to the labs to in the *submit* directory

---

# Creating a script file

You will want to save your work in R using a script file. The script file saves all the commands that you write so that you can run them again at a later time. To create a new script file Choose File -> New File -> R Script from the RStudio menu bar. Make sure that you type all of your commands into the script file and save it often. **Do not type commands directly into the console**. Commands typed into the console are executed but not saved.

# Loading a data set into R

Throughout the lab sessions we will be using data on the relationship between serum cholesterol and genotype included in the SISG-Data-cholesterol data set. This data set is available for download from the module Github repository and contains the following variables:

ID: Subject ID

sex: Sex: 0 = male, 1 = female

age: Age in years

chol: Serum total cholesterol, mg/dl

BMI: Body-mass index, kg/m2

TG: Serum triglycerides, mg/dl

APOE: Apolipoprotein E genotype, with six genotypes coded 1-6: 1 = e2/e2, 2 = e2/e3, 3 = e2/e4, 4 = e3/e3, 5 = e3/e4, 6 = e4/e4

rs174548: Candidate SNP 1 genotype, chromosome 11, physical position 61,327,924. Coded as the number of minor alleles: 0 = C/C, 1 = C/G, 2 = G/G.

rs4775401: Candidate SNP 2 genotype, chromosome 15, physical position 59,476,915. Coded as the number of minor alleles: 0 = C/C, 1 = C/T, 2 = T/T.

HTN: diagnosed hypertension: 0 = no, 1 = yes

chd: diagnosis of coronary heart disease: 0 = no, 1 = yes

You can download the data file and read it into R as follows:

```
cholesterol = read.csv("https://raw.githubusercontent.com/rhubb/SISG2018/master/data/SISG-Dat
  a-cholesterol.csv", header=T)
```

# Simple Linear Regression

We will start by exploring the data using descriptive statistics, as illustrated in the lectures. We first take a look at the structure of the data.

```
str(cholesterol)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sex      : int  1 1 0 0 1 1 0 0 0 0 ...
##  $ age      : int  74 51 64 34 52 39 79 38 52 58 ...
##  $ chol     : int  215 204 205 182 175 176 159 169 175 189 ...
##  $ BMI      : num  26.2 24.7 24.2 23.8 34.1 22.7 22.9 24.9 20.4 22 ...
##  $ TG       : int  367 150 213 111 328 53 274 137 125 209 ...
##  $ APOE     : int  4 4 4 2 2 4 2 2 4 5 ...
##  $ rs174548 : int  1 2 0 1 0 0 2 1 0 0 ...
##  $ rs4775401: int  2 1 1 1 0 2 1 1 1 1 ...
##  $ HTN      : int  1 1 1 1 1 0 1 0 0 1 ...
##  $ chd      : int  1 1 0 0 0 0 0 0 0 0 ...
```

```
head(cholesterol)
```

```
##   ID sex age chol  BMI  TG APOE rs174548 rs4775401 HTN chd
## 1  1   1  74  215 26.2 367    4        1         2   1   1
## 2  2   1  51  204 24.7 150    4        2         1   1   1
## 3  3   0  64  205 24.2 213    4        0         1   1   0
## 4  4   0  34  182 23.8 111    2        1         1   1   0
## 5  5   1  52  175 34.1 328    2        0         0   1   0
## 6  6   1  39  176 22.7  53    4        0         2   0   0
```

Next we can take a look at some descriptive statistics and plots that begin to help us see the relationship between cholesterol and age.

```
attach(cholesterol)
summary(chol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   117.0   168.0   184.0   183.9   199.2   247.0
```

```
summary(age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   30.00   43.00   55.00   54.82   67.00   80.00
```

```
group = 1*(age > 55)
group=factor(group,levels=c(0,1), labels=c("30-55","56-80"))
table(group)
```

```
## group
## 30-55 56-80
##   201   199
```

```
by(chol, group, mean)
```

```
## group: 30-55
## [1] 179.9751
## ------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------
## group: 56-80
## [1] 187.8945
```

```
by(chol, group, sd)
```

```
## group: 30-55
## [1] 23.02117
## ------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------
## group: 56-80
## [1] 20.46465
```

```r
mean(chol[group=="30-55"]) # identify a subset of the data using []
```

```
## [1] 179.9751
```

```r
mean(chol[group=="56-80"])
```

```
## [1] 187.8945
```

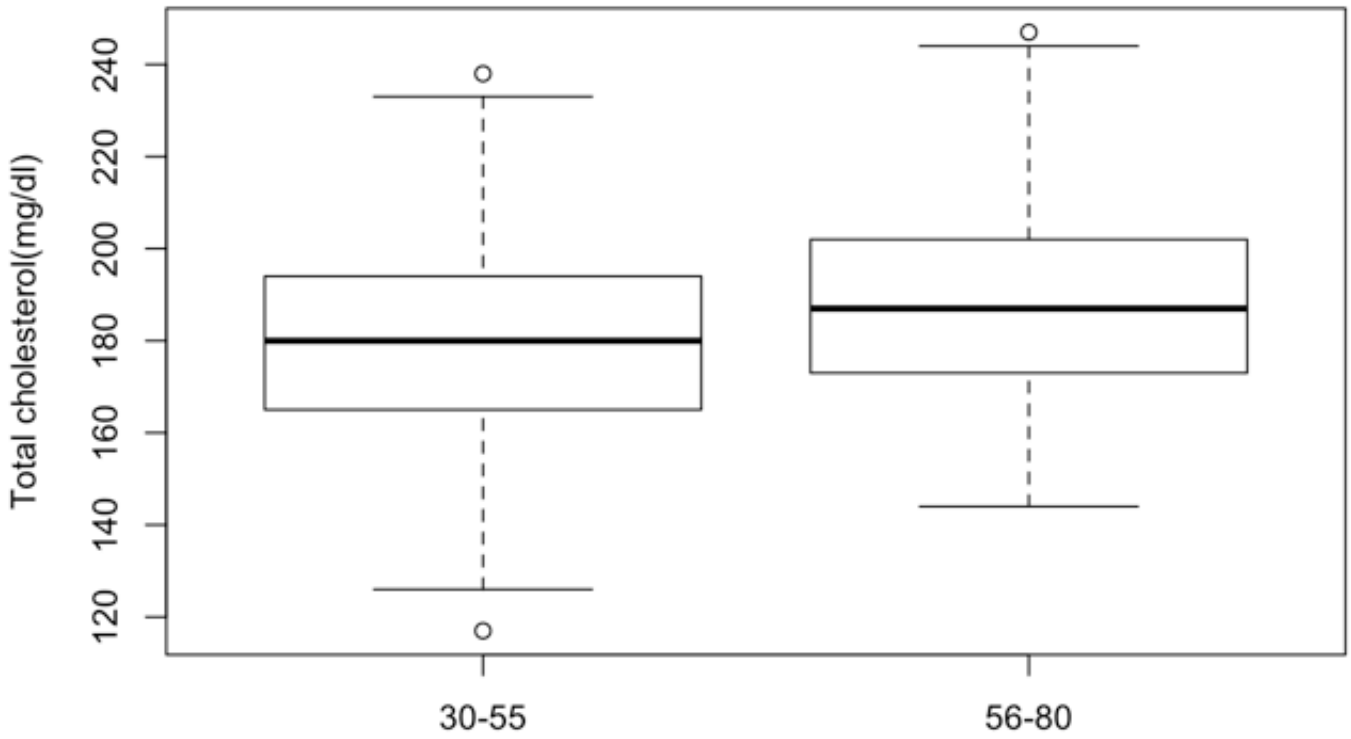
```r
sd(chol[group=="30-55"])
```

```
## [1] 23.02117
```

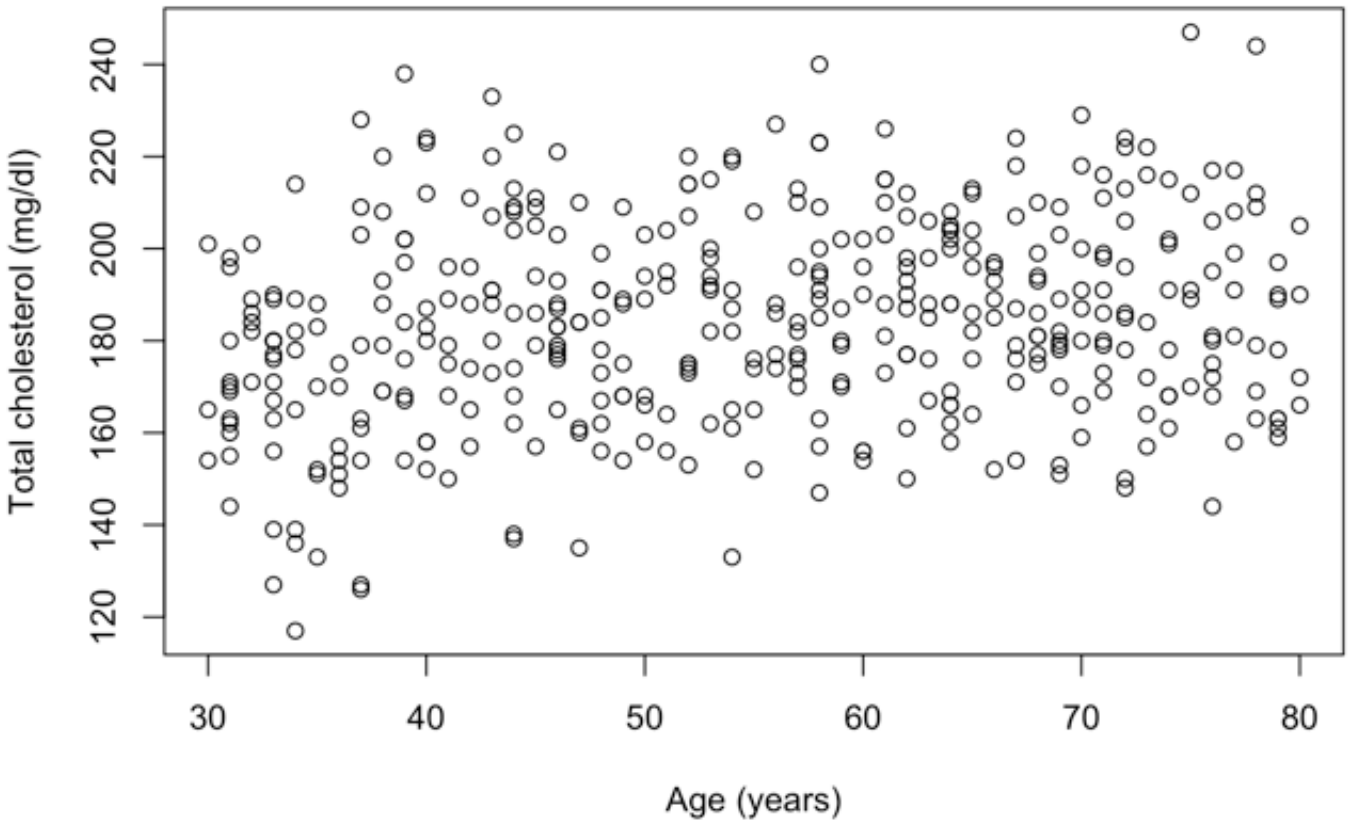
```r
sd(chol[group=="56-80"])
```

```
## [1] 20.46465
```

```r
boxplot(chol~group,ylab="Total cholesterol(mg/dl)")
```




```
plot(chol ~ age, xlab = "Age (years)", ylab = "Total cholesterol (mg/dl)")
```




Once we have explored the data a bit we are ready to investigate the association between cholesterol and age using simple linear regression. The following series of commands estimates the linear regression model and assigns it to an R linear model object called *fit*. We can then use the *summary()* function to summarize the regression results and *confint()* to generate confidence intervals for the regression parameters.

```
fit = lm(chol ~ age)
summary(fit)
```

```
##
## Call:
## lm(formula = chol ~ age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.453 -14.643  -0.022  14.659  58.995
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 166.90168    4.26488  39.134  < 2e-16 ***
## age           0.31033    0.07524   4.125 4.52e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.69 on 398 degrees of freedom
## Multiple R-squared:  0.04099,    Adjusted R-squared:  0.03858
## F-statistic: 17.01 on 1 and 398 DF,  p-value: 4.522e-05
```

```
confint(fit)
```

```
##                   2.5 %      97.5 %
## (Intercept) 158.5171656 175.2861949
## age           0.1624211   0.4582481
```

We can construct predictions for mean cholesterol at given values for age as well as for a new individual of given age using the *predict()* function. The *interval="confidence"* option provides confidence intervals for a new prediction of the mean while *interval = "prediction"* provides confidence intervals for a new individual value.

```
# Prediction of the mean
predict(fit, newdata=data.frame(age=c(46,47,48)), interval="confidence")
```

```
##        fit      lwr      upr
## 1 181.1771 178.6776 183.6765
## 2 181.4874 179.0619 183.9129
## 3 181.7977 179.4392 184.1563
```

```
# Prediction of a new individual value
predict(fit, newdata=data.frame(age=c(46,47,48)), interval="prediction")
```

```
##        fit      lwr      upr
## 1 181.1771 138.4687 223.8854
## 2 181.4874 138.7833 224.1915
## 3 181.7977 139.0974 224.4981
```

We can use the *anova()* function to obtain the ANOVA table for our regression model.

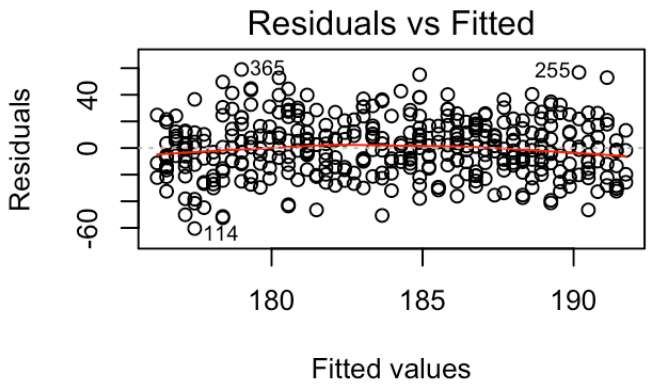
```
# ANOVA table
anova(fit)
```

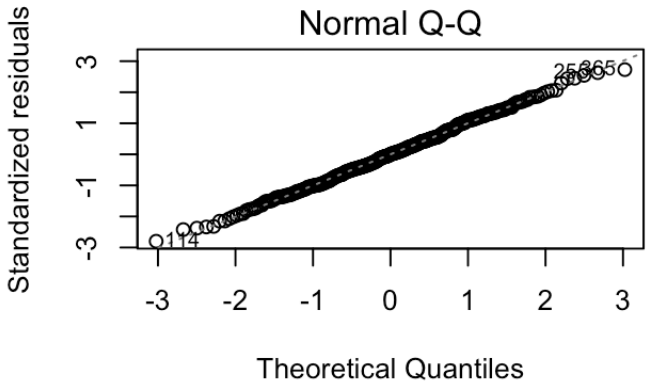
```
## Analysis of Variance Table
##
## Response: chol
##             Df Sum Sq Mean Sq F value    Pr(>F)
## age          1   8002  8001.7  17.013 4.522e-05 ***
## Residuals 398 187187   470.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

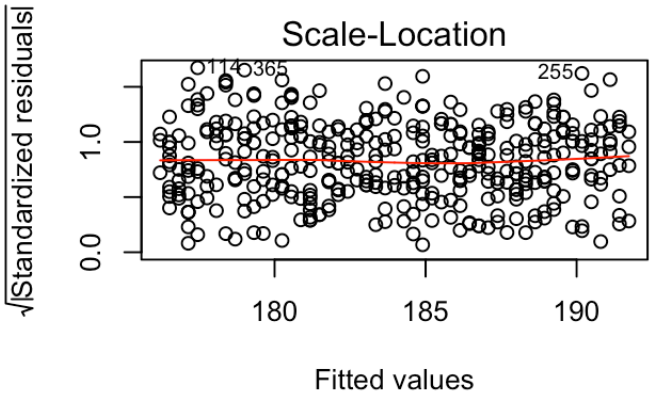
# Linear regression assumptions and diagnostics

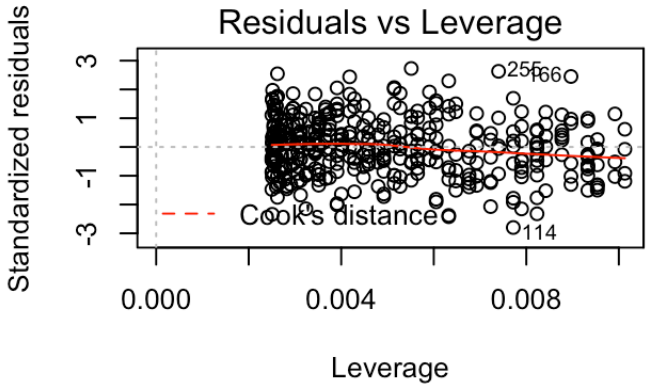
R can be used to generate a variety of diagnostic plots to help us evaluate our regression model assumption. Simply applying the *plot()* function to our regression model object, *fit*, provides a set of standard plots. We can also generate individual and customized plots as shown below.

```
# R's standard linear regression diagnostic plots
par(mfrow = c(2,2)) # create a 2 row x 2 column grid of plots
plot(fit)
```

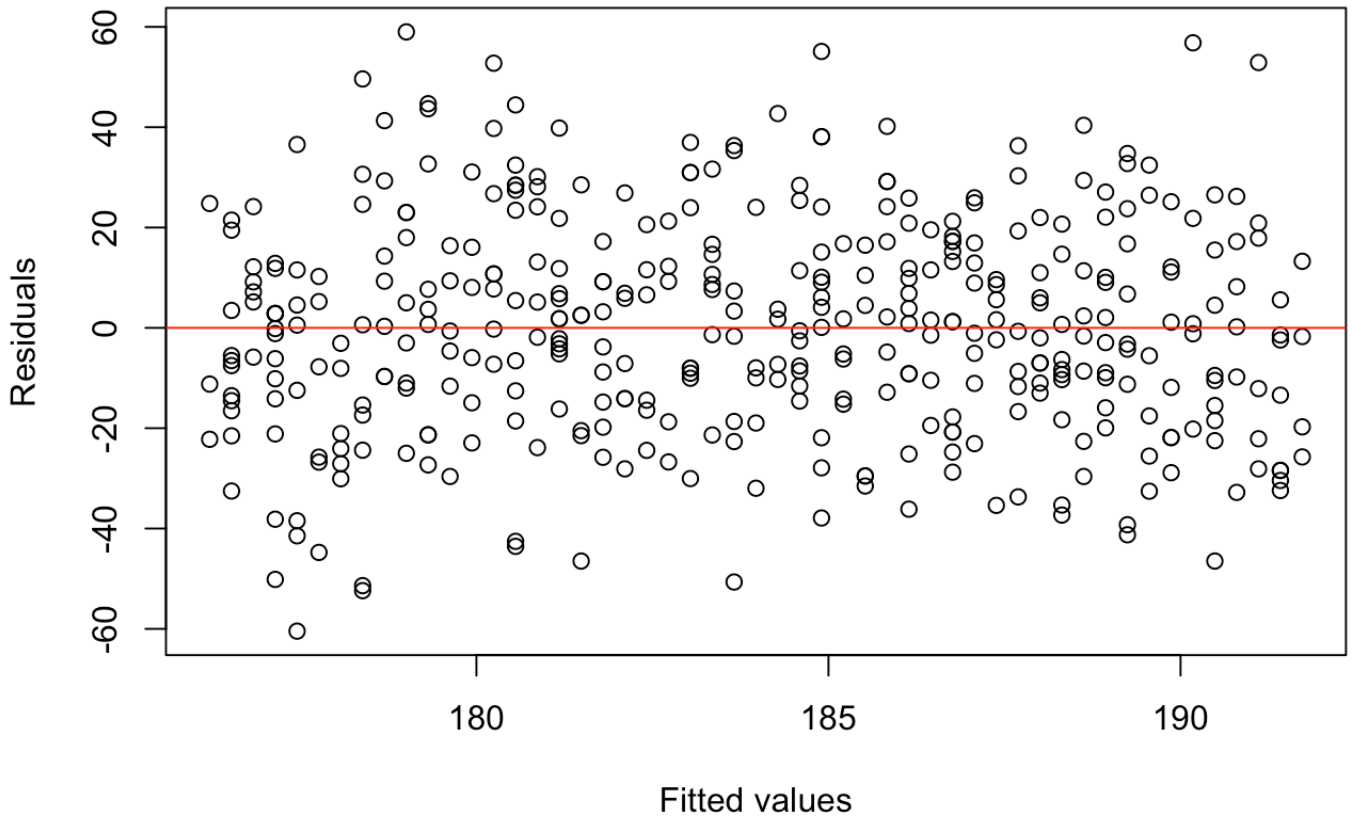


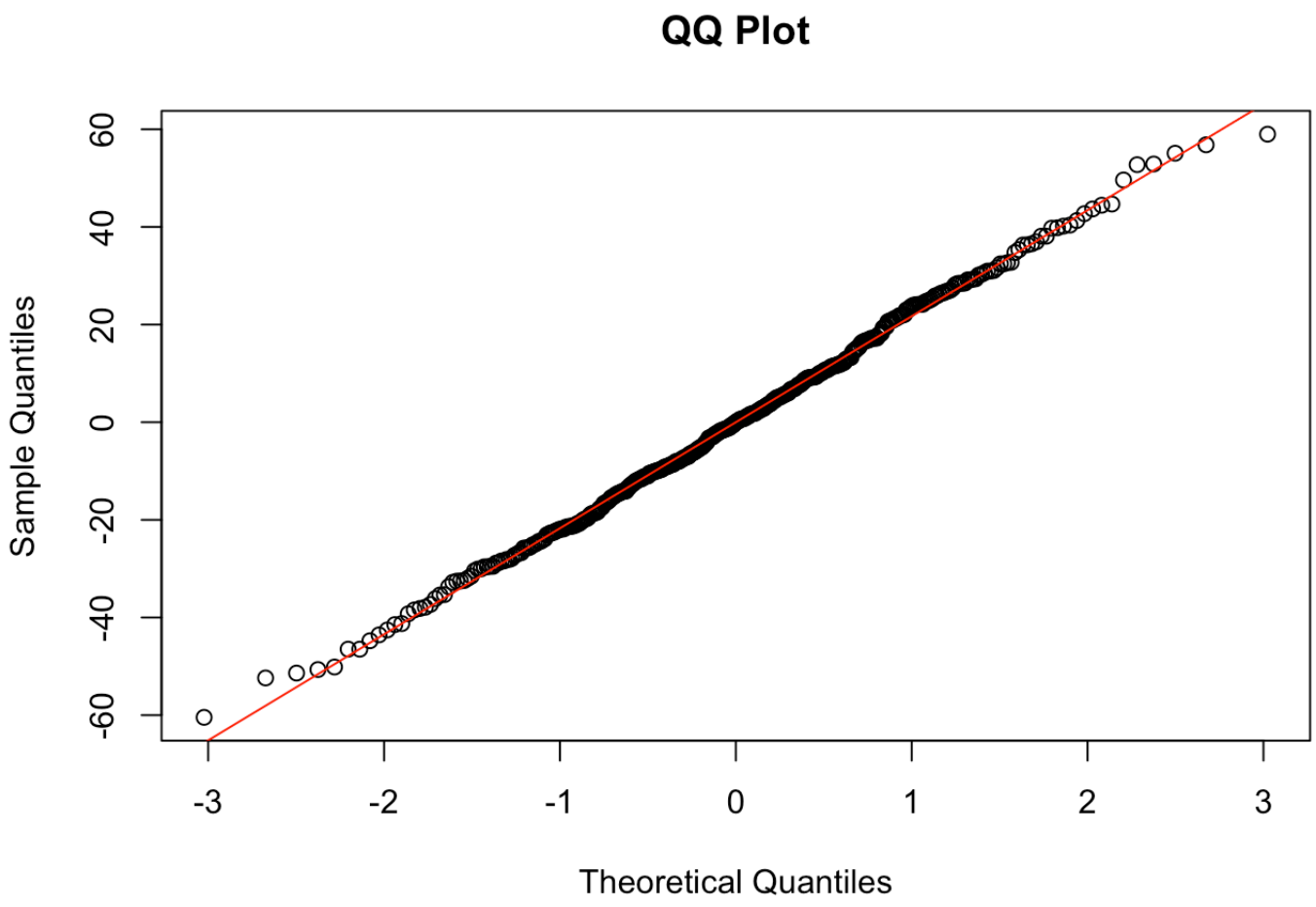




```
# Create individual plots by hand
par(mfrow = c(1,1))
## Fitted values vs. residuals
plot(fit$fitted, fit$residuals, xlab = "Fitted values", ylab = "Residuals")
abline(0,0, col = "red") # plot reference line
```




```
## QQ plot
qqnorm(fit$residuals, main = "QQ Plot")
qqline(fit$residuals, col = "red") # plot reference line
```




If we find that the assumptions of heteroscedasticity or normality are not satisfied, one way to address this is by using robust standard errors, which relax these assumptions. In R we can use the *gee* package to estimate a linear model with robust (also called empirical) standard errors. If you have not already, you will need to load the gee library. This must be repeated each time you open up R.

```
library(gee)
```

Now rather than using the *lm()* function to estimate our model, we use *gee()*. An additional capability of *gee()* which is beyong the scope of this course, is estimating regression parameters for dependent data. Although we will not be discussing methods for dependent data, since *gee()* has this capability we must identify which ``cluster" or dependent groups the observations in our data set belong to using the *id* argument. Since our data are independent, we will simply label each observation as belonging to its own unique cluster.

```
# Refit our regression model using GEE to obtain empirical standard errors
fit.ese <- gee(chol ~ age, id = seq(1, length(age)))
```

```
## Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
```

```
## running glm to get initial regression estimate
```

```
## (Intercept)         age
## 166.9016802   0.3103346
```

```
summary(fit.ese)
```

```
##
##  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
##  gee S-function, version 4.13 modified 98/01/27 (1998)
##
## Model:
##  Link:                      Identity
##  Variance to Mean Relation: Gaussian
##  Correlation Structure:     Independent
##
## Call:
## gee(formula = chol ~ age, id = seq(1, length(age)))
##
## Summary of Residuals:
##         Min          1Q      Median          3Q         Max
## -60.4530574 -14.6425007  -0.0219055  14.6592464  58.9952695
##
##
## Coefficients:
##                Estimate Naive S.E.   Naive z Robust S.E.  Robust z
## (Intercept) 166.9016802 4.26488334 39.133938  4.46336239 37.393710
## age           0.3103346 0.07523797  4.124707  0.07749153  4.004755
##
## Estimated Scale Parameter:  470.3202
## Number of Iterations:  1
##
## Working Correlation
##      [,1]
## [1,]    1
```

In this case our model assumptions were satisfied so estimates returned when using model-based standard errors and empirical standard errors are almost identical.

One (slightly annoying) feature of R's *gee()* function is that it does not return p-values for regression coefficients. However, they can be calculated within R:

```
beta = fit.ese$coef # Extract coefficients
se = sqrt(diag(fit.ese$robust.variance)) # Extract covariance matrix, take diagonal elements,
  and square root
p = 2*(1-pnorm(abs(beta)/se))
p
```

```
##  (Intercept)          age
## 0.000000e+00 6.208167e-05
```

We can use delta-betas to look for high leverage points:

```
dfb = dfbeta(fit) # create delta-betas for fit
index=order(abs(dfb[,2]),decreasing=T) # sort dfb with largest delta betas at top
cbind(dfb[index[1:15],],age[index[1:15]])
```

```
##     (Intercept)          age
## 114  -0.9893663  0.015268514 34
## 166  -0.6827966  0.014888475 78
## 255  -0.6190643  0.013902713 75
## 186  -0.8544144  0.013279531 33
## 113   0.5376293 -0.011943495 76
## 325  -0.7517511  0.011308451 37
## 365   0.7676508 -0.011297278 39
## 257  -0.7374003  0.011092575 37
## 290  -0.7024787  0.010757541 35
## 144   0.7120264 -0.010710881 37
## 197  -0.6784150  0.010469720 34
## 296  -0.6499386  0.010101515 33
## 231  -0.6293174  0.009712016 34
## 7     0.4403297 -0.009524470 79
## 252  -0.5981020  0.009412761 31
```

In this case the largest delta-beta is only about 5% as large as the age coefficient so we can conclude that there are no high leverage points.

# Multiple linear regression

The same basic set of commands we used for simple linear regression can be extended to the case of multiple linear regression. All we have to do is add additional predictors to the right hand side of our formula inside the *lm()* or *gee()* functions. Additional terms are preceded by "+". Interaction terms are denoted by ":" separating two predictor variables.

Separating two predictors using “*" tells R to add the main effects of both variables as well as their interaction. This is used most commonly since typically we want both main effects and interaction terms in our model.

```
# Model with main effects only
fit2=lm(chol~age+sex)
summary(fit2)
```

```
##
## Call:
## lm(formula = chol ~ age + sex)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.662 -14.482  -1.411  14.682  57.876
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 162.35445    4.24184  38.275  < 2e-16 ***
## age           0.29697    0.07313   4.061 5.89e-05 ***
## sex          10.50728    2.10794   4.985 9.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.06 on 397 degrees of freedom
## Multiple R-squared:  0.09748,    Adjusted R-squared:  0.09293
## F-statistic: 21.44 on 2 and 397 DF,  p-value: 1.44e-09
```

```
# Model with main effects and interaction term
fit3=lm(chol~age*sex)
summary(fit3)
```

```
##
## Call:
## lm(formula = chol ~ age * sex)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.474 -14.377  -1.215  14.764  58.301
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 160.31151    5.86268  27.344  < 2e-16 ***
## age           0.33460    0.10442   3.204  0.00146 **
## sex          14.56271    8.29802   1.755  0.08004 .
## age:sex      -0.07399    0.14642  -0.505  0.61361
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.08 on 396 degrees of freedom
## Multiple R-squared:  0.09806,    Adjusted R-squared:  0.09123
## F-statistic: 14.35 on 3 and 396 DF,  p-value: 6.795e-09
```

We can use *anova* to compare two nested models. For instance, if we want to test the null hypothesis that the coefficient for the interaction term is equal to zero we could compare *fit3* to *fit2*.

```
anova(fit2,fit3)
```

```
## Analysis of Variance Table
##
## Model 1: chol ~ age + sex
## Model 2: chol ~ age * sex
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    397 176162
## 2    396 176049  1    113.52 0.2554 0.6136
```

Note that this gives the same p-value as the hypothesis test for the interaction coefficient in our linear regression model table for *fit3*.

# ANOVA

In R, we can use *lm()* to conduct an analysis of variance. There is no difference in syntax compared to the code for regression models. The only difference is that we are using a categorical predictor. We can either define dummy variables ourselves, or use *factor()* to have R create dummy variables for us.

```r
# Creating dummy variables by hand
dummy1 = 1*(rs174548==1)
dummy2 = 1*(rs174548==2)

fit0 = lm(chol ~ dummy1 + dummy2)
summary(fit0)
```

```
##
## Call:
## lm(formula = chol ~ dummy1 + dummy2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.062 -15.913  -0.062  14.938  59.136
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  181.062      1.455 124.411  < 2e-16 ***
## dummy1         6.802      2.321   2.930  0.00358 **
## dummy2         5.438      4.540   1.198  0.23167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.93 on 397 degrees of freedom
## Multiple R-squared:  0.0221, Adjusted R-squared:  0.01718
## F-statistic: 4.487 on 2 and 397 DF,  p-value: 0.01184
```

```r
anova(fit0)
```

```
## Analysis of Variance Table
##
## Response: chol
##            Df Sum Sq Mean Sq F value   Pr(>F)
## dummy1      1   3624  3624.3  7.5381 0.006315 **
## dummy2      1    690   689.9  1.4350 0.231665
## Residuals 397 190875   480.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Using factor() to create dummy variables
fit1.1 = lm(chol ~ factor(rs174548))
summary(fit1.1)
```

```
##
## Call:
## lm(formula = chol ~ factor(rs174548))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.062 -15.913  -0.062  14.938  59.136
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        181.062      1.455 124.411  < 2e-16 ***
## factor(rs174548)1    6.802      2.321   2.930  0.00358 **
## factor(rs174548)2    5.438      4.540   1.198  0.23167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.93 on 397 degrees of freedom
## Multiple R-squared:  0.0221, Adjusted R-squared:  0.01718
## F-statistic: 4.487 on 2 and 397 DF,  p-value: 0.01184
```

```
anova(fit1.1)
```

```
## Analysis of Variance Table
##
## Response: chol
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## factor(rs174548)   2   4314 2157.10  4.4865 0.01184 *
## Residuals        397 190875  480.79
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If we would like to use the ANOVA parameterization (separate mean for each group) as opposed to the regression parameterization (difference in means relative to a reference group) we can tell R to remove the intercept from the model by using *-1* on the right hand side of the formula.

```
# Model without intercept
fit1.2 = lm(chol ~ -1 + factor(rs174548))
summary(fit1.2)
```

```
##
## Call:
## lm(formula = chol ~ -1 + factor(rs174548))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.062 -15.913  -0.062  14.938  59.136
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## factor(rs174548)0  181.062      1.455  124.41   <2e-16 ***
## factor(rs174548)1  187.864      1.809  103.88   <2e-16 ***
## factor(rs174548)2  186.500      4.300   43.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.93 on 397 degrees of freedom
## Multiple R-squared:  0.9861, Adjusted R-squared:  0.986
## F-statistic:  9383 on 3 and 397 DF,  p-value: < 2.2e-16
```

The *aov()* function can also be used to conduct an analysis of variance. Results are equivalent to those obtained using *lm()* but organized differently.

```
fit1.3 = aov(chol ~ factor(rs174548))
summary(fit1.3)
```

```
##                   Df Sum Sq Mean Sq F value Pr(>F)
## factor(rs174548)   2   4314  2157.1   4.487 0.0118 *
## Residuals        397 190875   480.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit1.3)
```

```
## Analysis of Variance Table
##
## Response: chol
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## factor(rs174548)   2   4314 2157.10  4.4865 0.01184 *
## Residuals        397 190875  480.79
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fit1.3$coeff
```

```
##       (Intercept) factor(rs174548)1 factor(rs174548)2
##         181.061674          6.802272          5.438326
```

Finally, if we do not use *factor()* to convert our SNP to a categorical variable, we can use *lm()* to analyze the association between the predictor and outcome assuming a linear dose response relationship between each copy of the minor allele and the mean of the outcome.

```
fit2 = lm(chol ~ rs174548)
summary(fit2)
```

```
##
## Call:
## lm(formula = chol ~ rs174548)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.575 -16.278  -0.575  15.120  60.722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  181.575      1.411 128.723  < 2e-16 ***
## rs174548       4.703      1.781   2.641  0.00858 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.95 on 398 degrees of freedom
## Multiple R-squared:  0.01723,    Adjusted R-squared:  0.01476
## F-statistic: 6.977 on 1 and 398 DF,  p-value: 0.008583
```

```
anova(fit2)
```

```
## Analysis of Variance Table
##
## Response: chol
##            Df Sum Sq Mean Sq F value   Pr(>F)
## rs174548    1   3363  3362.6  6.9766 0.008583 **
## Residuals 398 191827   482.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can also fit a one-way ANOVA command, relaxing the assumption of equal variance within the groups defined by our factor variable using *oneway.test()*.

```
fit.oneway = oneway.test(chol ~ factor(rs174548))
fit.oneway
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  chol and factor(rs174548)
## F = 4.3258, num df = 2.000, denom df = 73.284, p-value = 0.01676
```

Alternatively, we can use a model with robust standard errors, estimated via *gee()* to relax this assumption.

```
fit.gee = gee(chol ~ factor(rs174548), id=seq(1,length(chol)))
```

```
## Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
```

```
## running glm to get initial regression estimate
```

```
##       (Intercept) factor(rs174548)1 factor(rs174548)2
##        181.061674          6.802272          5.438326
```

```
summary(fit.gee)
```

```
##
##  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
##  gee S-function, version 4.13 modified 98/01/27 (1998)
##
## Model:
##  Link:                      Identity
##  Variance to Mean Relation: Gaussian
##  Correlation Structure:     Independent
##
## Call:
## gee(formula = chol ~ factor(rs174548), id = seq(1, length(chol)))
##
## Summary of Residuals:
##          Min           1Q       Median           3Q          Max
## -64.06167401 -15.91337769  -0.06167401  14.93832599  59.13605442
##
##
## Coefficients:
##                     Estimate Naive S.E.    Naive z Robust S.E.   Robust z
## (Intercept)       181.061674   1.455346 124.411431    1.400016 129.328297
## factor(rs174548)1   6.802272   2.321365   2.930290    2.402005   2.831914
## factor(rs174548)2   5.438326   4.539833   1.197913    3.624271   1.500530
##
## Estimated Scale Parameter:  480.7932
## Number of Iterations:  1
##
## Working Correlation
##      [,1]
## [1,]    1
```

The Kruskal-Wallis test is the non-parametric analogue to one-way ANOVA and can be used to determine whether the ranks of the observations in the groups defined by the exposure variable.

```
fit.kw = kruskal.test(chol ~ factor(rs174548))
fit.kw
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  chol by factor(rs174548)
## Kruskal-Wallis chi-squared = 7.4719, df = 2, p-value = 0.02385
```

# Multiple comparisons

The *multcomp* package can be used to carry out multiple comparisons. If you have not already loaded it, you will need to do so now.

```
library(multcomp)
```

You will first need to use *lm()* to create a regression model object holding the means for each group. To do this you must use the ANOVA parameterization (rather than the regression parameterization) in which you tell R not to include an intercept term.

```
fit1 = lm(chol ~ -1 + factor(rs174548))
```

Next, you must construct the contrast matrix which tells R which group means we wish to compare. For instance, if we want to compare all pairs of means we can use the *contrMat()* function with option *type = "Tukey"*. Alternatively, we can specify the contrasts that we wish to estimate by hand by constructing the appropriate matrix.

```
# All pairwise comparisons
M1 = contrMat(table(rs174548), type="Tukey")

# Construct contrast matrix by hand for two comparisons
M2 = rbind("mean(C/G) - mean(C/C)" = c(-1, 1, 0),"mean(G/G) - mean(C/C)" = c(-1, 0, 1))
```

We can then use *glht()* to carry out the hypothesis tests for our specified contrasts and finally use *summary* to obtain results with p-values adjusted for multiple comparisons.

```
mc1 = glht(fit1, linfct =M1)
summary(mc1, test = adjusted("none")) # no adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>|t|)
## 1 - 0 == 0    6.802      2.321   2.930  0.00358 **
## 2 - 0 == 0    5.438      4.540   1.198  0.23167
## 2 - 1 == 0   -1.364      4.665  -0.292  0.77015
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

```
summary(mc1, test = adjusted("bonferroni")) # bonferroni adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>|t|)
## 1 - 0 == 0    6.802      2.321   2.930   0.0107 *
## 2 - 0 == 0    5.438      4.540   1.198   0.6950
## 2 - 1 == 0   -1.364      4.665  -0.292   1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- bonferroni method)
```

```
summary(mc1, test = adjusted("fdr")) # FDR adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>|t|)
## 1 - 0 == 0    6.802      2.321   2.930   0.0107 *
## 2 - 0 == 0    5.438      4.540   1.198   0.3475
## 2 - 1 == 0   -1.364      4.665  -0.292   0.7702
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- fdr method)
```

```
mc2 = glht(fit1, linfct =M2)
summary(mc2, test = adjusted("none")) # no adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##                            Estimate Std. Error t value Pr(>|t|)
## mean(C/G) - mean(C/C) == 0    6.802      2.321   2.930  0.00358 **
## mean(G/G) - mean(C/C) == 0    5.438      4.540   1.198  0.23167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

```
summary(mc2, test = adjusted("bonferroni")) # bonferroni adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##                            Estimate Std. Error t value Pr(>|t|)
## mean(C/G) - mean(C/C) == 0    6.802      2.321   2.930  0.00716 **
## mean(G/G) - mean(C/C) == 0    5.438      4.540   1.198  0.46333
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- bonferroni method)
```

```
summary(mc2, test = adjusted("fdr")) # FDR adjustment for multiple comparisons
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = chol ~ -1 + factor(rs174548))
##
## Linear Hypotheses:
##                            Estimate Std. Error t value Pr(>|t|)
## mean(C/G) - mean(C/C) == 0    6.802      2.321   2.930  0.00716 **
## mean(G/G) - mean(C/C) == 0    5.438      4.540   1.198  0.23167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- fdr method)
```

# Two-way ANOVA

In R we extend ANOVA to incorporate multiple predictors by adding additional variables to the right hand side of our model formula, just as we did for linear regression.

```
# One-way ANOVA including main effect of sex
fit0 = lm(chol ~ factor(sex))
summary(fit0)
```

```
##
## Call:
## lm(formula = chol ~ factor(sex))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.299 -14.343  -0.888  14.701  57.701
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   178.477      1.522  117.26  < 2e-16 ***
## factor(sex)1   10.821      2.147    5.04 7.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.47 on 398 degrees of freedom
## Multiple R-squared:  0.05999,    Adjusted R-squared:  0.05763
## F-statistic:  25.4 on 1 and 398 DF,  p-value: 7.087e-07
```

```
# Two-way ANOVA including main effects of sex and SNP
fit1 = lm(chol ~ factor(sex) + factor(rs174548))
summary(fit1)
```

```
##
## Call:
## lm(formula = chol ~ factor(sex) + factor(rs174548))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.653 -14.463  -0.601  15.445  57.635
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        175.365      1.786  98.208  < 2e-16 ***
## factor(sex)1        11.053      2.126   5.199 3.22e-07 ***
## factor(rs174548)1    7.236      2.250   3.215  0.00141 **
## factor(rs174548)2    5.184      4.398   1.179  0.23928
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.24 on 396 degrees of freedom
## Multiple R-squared:  0.08458,    Adjusted R-squared:  0.07764
## F-statistic:  12.2 on 3 and 396 DF,  p-value: 1.196e-07
```

We can compare nested models using *anova()*.

```
# Compare model with sex only to model including sex and SNP
anova(fit0,fit1)
```

```
## Analysis of Variance Table
##
## Model 1: chol ~ factor(sex)
## Model 2: chol ~ factor(sex) + factor(rs174548)
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)
## 1    398 183480
## 2    396 178681  2    4799.1 5.318 0.005259 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To include interaction terms, predictor variables should be separated by ``*".

```
# Two-way ANOVA with interaction between sex and SNP
fit2 = lm(chol ~ factor(sex) * factor(rs174548))
summary(fit2)
```

```
##
## Call:
## lm(formula = chol ~ factor(sex) * factor(rs174548))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.529 -13.604  -0.974  14.171  54.882
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    178.1182     2.0089  88.666  < 2e-16 ***
## factor(sex)1                     5.7109     2.7982   2.041  0.04192 *
## factor(rs174548)1                0.9597     3.1306   0.307  0.75933
## factor(rs174548)2               -0.2015     6.4053  -0.031  0.97492
## factor(sex)1:factor(rs174548)1  12.7398     4.4650   2.853  0.00456 **
## factor(sex)1:factor(rs174548)2  10.2296     8.7482   1.169  0.24297
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.07 on 394 degrees of freedom
## Multiple R-squared:  0.1039, Adjusted R-squared:  0.09257
## F-statistic:  9.14 on 5 and 394 DF,  p-value: 3.062e-08
```

```
# Compare models with and without interaction
anova(fit1,fit2)
```

```
## Analysis of Variance Table
##
## Model 1: chol ~ factor(sex) + factor(rs174548)
## Model 2: chol ~ factor(sex) * factor(rs174548)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    396 178681
## 2    394 174902  2    3778.9 4.2564 0.01483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# ANCOVA

ANCOVA models include both categorical and continuous predictors on the right hand side of the model formula.

```r
# Linear model including age only
fit0 = lm(chol ~ age)
summary(fit0)
```

```
##
## Call:
## lm(formula = chol ~ age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.453 -14.643  -0.022  14.659  58.995
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 166.90168    4.26488  39.134  < 2e-16 ***
## age           0.31033    0.07524   4.125 4.52e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.69 on 398 degrees of freedom
## Multiple R-squared:  0.04099,    Adjusted R-squared:  0.03858
## F-statistic: 17.01 on 1 and 398 DF,  p-value: 4.522e-05
```

```r
# ANCOVA including age and SNP
fit1 = lm(chol ~ factor(rs174548) + age)
summary(fit1)
```

```
##
## Call:
## lm(formula = chol ~ factor(rs174548) + age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -57.209 -14.429   0.444  14.265  55.898
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       163.28125    4.36422  37.414  < 2e-16 ***
## factor(rs174548)1   7.30137    2.27457   3.210  0.00144 **
## factor(rs174548)2   5.08431    4.44331   1.144  0.25321
## age                 0.32140    0.07457   4.310 2.06e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.46 on 396 degrees of freedom
## Multiple R-squared:  0.06592,    Adjusted R-squared:  0.05884
## F-statistic: 9.316 on 3 and 396 DF,  p-value: 5.778e-06
```

```
# ANCOVA including age, SNP, and interaction
fit2 = lm(chol ~ factor(rs174548) * age)
summary(fit2)
```

```
##
## Call:
## lm(formula = chol ~ factor(rs174548) * age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -57.542 -14.300   0.713  14.214  55.709
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           164.14677    5.79545  28.323  < 2e-16 ***
## factor(rs174548)1       3.42799    8.79946   0.390  0.69707
## factor(rs174548)2      16.53004   18.28067   0.904  0.36642
## age                     0.30576    0.10154   3.011  0.00277 **
## factor(rs174548)1:age   0.07159    0.15617   0.458  0.64692
## factor(rs174548)2:age  -0.20255    0.31488  -0.643  0.52043
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.49 on 394 degrees of freedom
## Multiple R-squared:  0.06777,    Adjusted R-squared:  0.05594
## F-statistic: 5.729 on 5 and 394 DF,  p-value: 4.065e-05
```

```
# Test of coincident lines
anova(fit0,fit2)
```

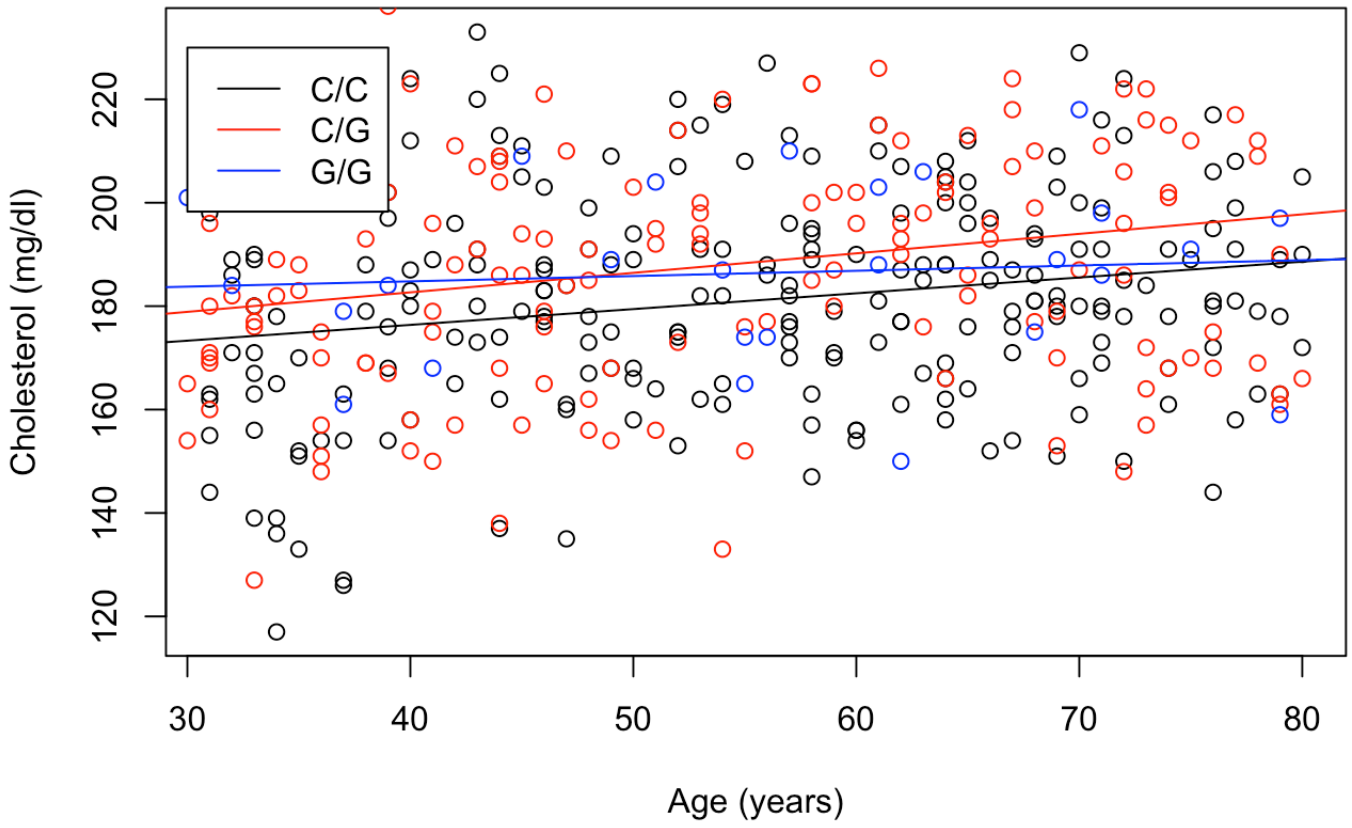
```
## Analysis of Variance Table
##
## Model 1: chol ~ age
## Model 2: chol ~ factor(rs174548) * age
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    398 187187
## 2    394 181961  4    5226.6 2.8293 0.02455 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Test of parallel lines
anova(fit1,fit2)
```

```
## Analysis of Variance Table
##
## Model 1: chol ~ factor(rs174548) + age
## Model 2: chol ~ factor(rs174548) * age
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1    396 182322
## 2    394 181961  2    361.11 0.391 0.6767
```

We can use *plot()* to visualize the data and fitted lines.

```
plot(age[rs174548==0], chol[rs174548==0], xlab = "Age (years)", ylab = "Cholesterol (mg/dl)")
points(age[rs174548==1], chol[rs174548==1], col = "red")
points(age[rs174548==2], chol[rs174548==2], col = "blue")
abline(fit2$coef[1],fit2$coef[4])
abline((fit2$coef[1]+fit2$coef[2]),(fit2$coef[4]+fit2$coef[5]), col = "red")
abline((fit2$coef[1]+fit2$coef[3]),(fit2$coef[4]+fit2$coef[6]), col = "blue")
legend(30,230,lty = 1, col = c("black","red","blue"),legend = c("C/C","C/G","G/G"))
```



Finally, if we conclude that the slopes are the same then we can obtain predictions and confidence intervals for a new observation for each genotype adjusted for age by predicting cholesterol at the mean value of age using our fitted ANCOVA.

```
## mean cholesterol for different genotypes adjusted by age
predict(fit1, new=data.frame(age=mean(age),rs174548=0))
```

```
##        1
## 180.9013
```

```
predict(fit1, new=data.frame(age=mean(age),rs174548=1))
```

```
##        1
## 188.2026
```

```
predict(fit1, new=data.frame(age=mean(age),rs174548=2))
```

```
##        1
## 185.9856
```

# Logistic Regression

In R, the syntax for logistic regression is very similar to that for linear regression. Instead of using the *lm()* function, we will use *glm()*. We will also need to specify the specific type of regression model we want to fit using *family = "binomial"*.

```
# Logistic regression for association between CHD and cholesterol
glm.mod1  <- glm(chd ~ chol, family = "binomial")
summary(glm.mod1)
```

```
##
## Call:
## glm(formula = chd ~ chol, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7437  -0.8219  -0.4852   0.9096   2.4536
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.09600    1.29881  -8.543  < 2e-16 ***
## chol          0.05498    0.00678   8.109 5.12e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 409.71  on 398  degrees of freedom
## AIC: 413.71
##
## Number of Fisher Scoring iterations: 4
```

We can extract odds ratios and confidence intervals by exponentiating the model coefficients and their confidence intervals.

```
exp(glm.mod1$coef)
```

```
##  (Intercept)         chol
## 1.517293e-05 1.056515e+00
```

```
exp(confint(glm.mod1))
```

```
## Waiting for profiling to be done...
```

```
##                    2.5 %       97.5 %
## (Intercept) 1.061838e-06 0.0001744859
## chol        1.043101e+00 1.0712556915
```

We can also take a look at the odds ratio for larger differences in exposure level by multiplying the coefficient by the corresponding factor. For instance, to get the odds ratio associated with a 10 mg/dl difference in cholesterol we multiply the coefficients by 10.

```
exp(10*glm.mod1$coef)
```

```
##  (Intercept)         chol
## 6.466861e-49 1.732831e+00
```

Just as in *lm()* we can extend our model by adding additional predictor variables to the right hand side of the model formula.

```
glm.mod2 <- glm(chd ~ chol+factor(rs4775401), family = "binomial", data = cholesterol)
summary(glm.mod2)
```

```
##
## Call:
## glm(formula = chd ~ chol + factor(rs4775401), family = "binomial",
##     data = cholesterol)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5528  -0.7810  -0.4585   0.8037   2.6275
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -11.625209   1.335335  -8.706  < 2e-16 ***
## chol                 0.055443   0.006872   8.069 7.11e-16 ***
## factor(rs4775401)1   0.794212   0.259257   3.063  0.00219 **
## factor(rs4775401)2   1.138308   0.464317   2.452  0.01422 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 397.27  on 396  degrees of freedom
## AIC: 405.27
##
## Number of Fisher Scoring iterations: 4
```

```
exp(glm.mod2$coef)
```

```
##        (Intercept)               chol factor(rs4775401)1 factor(rs4775401)2
##       8.937908e-06       1.057009e+00       2.212697e+00       3.121483e+00
```

```
exp(confint(glm.mod2))
```

```
## Waiting for profiling to be done...
```

```
##                             2.5 %       97.5 %
## (Intercept)         5.776075e-07 0.0001096301
## chol                1.043422e+00 1.0719733312
## factor(rs4775401)1 1.336145e+00 3.6998174205
## factor(rs4775401)2 1.250542e+00 7.8187264825
```

For logistic regression models, we can compare nested models using the likelihood ratio test. We first need to load the *lmtest* package if you have not already done so

```
library(lmtest)
```

```
# Compare models with and without SNP
lrtest(glm.mod1,glm.mod2)
```

```
## Likelihood ratio test
##
## Model 1: chd ~ chol
## Model 2: chd ~ chol + factor(rs4775401)
##   #Df  LogLik Df Chisq Pr(>Chisq)
## 1   2 -204.85
## 2   4 -198.63  2 12.44   0.001989 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Relative risk regression

```
glm.rr <- gee(chd ~ chol+factor(rs4775401), family = "poisson", id = seq(1,nrow(cholesterol))
  , data = cholesterol)
```

```
## Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
```

```
## running glm to get initial regression estimate
```

```
##        (Intercept)               chol factor(rs4775401)1 factor(rs4775401)2
##        -7.06494205         0.02963414         0.41510943         0.63841622
```

```
summary(glm.rr)
```

```
##
##  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
##  gee S-function, version 4.13 modified 98/01/27 (1998)
##
## Model:
##  Link:                      Logarithm
##  Variance to Mean Relation: Poisson
##  Correlation Structure:     Independent
##
## Call:
## gee(formula = chd ~ chol + factor(rs4775401), id = seq(1, nrow(cholesterol)),
##     data = cholesterol, family = "poisson")
##
## Summary of Residuals:
##        Min         1Q     Median         3Q        Max
## -0.7876407 -0.2528067 -0.1356891  0.2918421  0.9313744
##
##
## Coefficients:
##                        Estimate  Naive S.E.    Naive z Robust S.E.   Robust z
## (Intercept)        -7.06494205 0.673034850 -10.497141 0.586040831 -12.055375
## chol                0.02963414 0.003372133   8.787949 0.002752366  10.766786
## factor(rs4775401)1  0.41510943 0.155971511   2.661444 0.144444876   2.873826
## factor(rs4775401)2  0.63841622 0.256170049   2.492158 0.200023351   3.191708
##
## Estimated Scale Parameter:  0.671285
## Number of Iterations:  1
##
## Working Correlation
##      [,1]
## [1,]    1
```

```
exp(glm.rr$coef)
```

```
##        (Intercept)               chol factor(rs4775401)1 factor(rs4775401)2
##       0.0008545444       1.0300775972       1.5145364657       1.8934796543
```

# Risk difference regression

We can estimate risk differences by applying a linear regession model (Normal GLM with identity link) to a binary outcome. As with reltive risk regression using a Poisson GLM, we need to use robust standard errors to account for the fact that we are assuming we have a Normally distributed outcome but in fact our outcome is binary.

```
glm.rd <- gee(chd ~ chol+factor(rs4775401), id = seq(1,nrow(cholesterol)), data =
  cholesterol)
```

```
## Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
```

```
## running glm to get initial regression estimate
```

```
##        (Intercept)               chol factor(rs4775401)1 factor(rs4775401)2
##       -1.485416572        0.009392399        0.142743141        0.212108382
```

```
summary(glm.rd)
```

```
##
##  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
##  gee S-function, version 4.13 modified 98/01/27 (1998)
##
## Model:
##  Link:                      Identity
##  Variance to Mean Relation: Gaussian
##  Correlation Structure:     Independent
##
## Call:
## gee(formula = chd ~ chol + factor(rs4775401), id = seq(1, nrow(cholesterol)),
##     data = cholesterol)
##
## Summary of Residuals:
##        Min          1Q      Median          3Q         Max
## -0.6297305 -0.3183884 -0.1122199  0.3514847  1.0953414
##
##
## Coefficients:
##                         Estimate   Naive S.E.   Naive z  Robust S.E.   Robust z
## (Intercept)        -1.485416572 0.1729186937 -8.590260 0.1372414095 -10.823385
## chol                0.009392399 0.0009293498 10.106420 0.0007615605  12.333097
## factor(rs4775401)1  0.142743141 0.0427305918  3.340537 0.0423172293   3.373168
## factor(rs4775401)2  0.212108382 0.0827885757  2.562049 0.0822370553   2.579231
##
## Estimated Scale Parameter:  0.1684965
## Number of Iterations:  1
##
## Working Correlation
##      [,1]
## [1,]    1
```