Chapter 6

Linear regression and ANOVA

Regression and analysis of variance form the basis of many investigations. In this chapter we describe how to undertake many common tasks in linear regression (broadly defined), while Chapter 7 discusses many generalizations, including other types of outcome variables, longitudinal and clustered analysis, and survival methods.

Many SAS procedures and R commands can perform linear regression, as it constitutes a special case of which many models are generalizations. We present detailed descriptions for SAS proc reg and proc glm as well as for the R \texttt{lm()} command, as these offer the most flexibility and best output options tailored to linear regression in particular. While ANOVA can be viewed as a special case of linear regression, separate routines are available in SAS (proc anova) and R (\texttt{aov()}) to perform it. In addition, SAS proc mixed is needed for some calculations. We address these additional procedures only with respect to output that is difficult to obtain through the standard linear regression tools.

R supports a flexible modeling language implemented using formulas (see help(formula) and 6.1.1) for regression that is shared with the lattice graphics functions. Many of the routines available within R return or operate on \texttt{lm} class objects, which include objects such as coefficients, residuals, fitted values, weights, contrasts, model matrices, and the like (see help(lm)).

The CRAN statistics for the social sciences task view provides an excellent overview of methods described here and in Chapter 7.

6.1 Model fitting

6.1.1 Linear regression

**SAS**

```
proc glm data=ds;
   model y = x1 ... xk;
run;
```

or

```
proc reg data=ds;
   model y = x1 ... xk;
run;
```

**Note:** Both \texttt{proc glm} and \texttt{proc reg} support linear regression models, while \texttt{proc reg} provides more regression diagnostics. The \texttt{glm} procedure more easily allows categorical covariates.
CHAPTER 6. LINEAR REGRESSION AND ANOVA

R

\[
\begin{align*}
mod1 = \text{lm}(y \sim x1 + \ldots + xk, \text{data}=ds) \\
\text{summary}(mod1) \\
\text{summary.aov}(mod1)
\end{align*}
\]

or

\[
\begin{align*}
form = \text{as.formula}(y \sim x1 + \ldots + xk) \\
mod1 = \text{lm}(form, \text{data}=ds) \\
\text{summary}(mod1) \\
\text{coef}(mod1)
\end{align*}
\]

Note: The first argument of the \text{lm}() function is a formula object, with the outcome specified followed by the \(\sim\) operator then the predictors. It returns a linear model object. More information about the linear model \text{summary()} command can be found using \text{help(summary.lm)}. The \text{coef()} function extracts coefficients from a model (see also the \text{coefplot} package). The \text{biglm()} function in the \text{biglm} package can support model fitting to very large datasets. By default, stars are used to annotate the output of the \text{summary()} functions regarding significance levels: these can be turned off using the command \text{options(show.signif.stars=FALSE)}. The \text{biglm} package can be used to undertake estimation with larger datasets.

6.1.2 Linear regression with categorical covariates

\textit{Example: 6.6.2}

See 6.1.4 (parameterization of categorical covariates).

SAS

\[
\begin{align*}
\text{proc glm data=ds;} \\
\hspace{1cm} \text{class x1;} \\
\hspace{1.5cm} \text{model y = x1 x2 \ldots xk;} \\
\text{run};
\end{align*}
\]

Note: The \text{class} statement specifies covariates that should be treated as categorical. The \text{glm} procedure uses reference cell coding; the reference category can be controlled using the \text{order} option to the \text{proc glm} statement, as in 7.10.11.

R

\[
\begin{align*}
ds = \text{transform}(ds, x1f = \text{as.factor}(x1)) \\
mod1 = \text{lm}(y \sim x1f + x2 + \ldots + xk, \text{data}=ds)
\end{align*}
\]

Note: The \text{as.factor()} function in R creates a categorical variable from a variable. By default, the lowest value (either numerically or lexicographically) is the reference value. The \text{levels} option for the \text{factor()} function can be used to select a particular reference value (see 2.2.19). Ordered factors can be constructed using the \text{ordered()} function.

6.1.3 Changing the reference category

SAS

\[
\begin{align*}
\text{proc sort data=ds;} \\
\hspace{1cm} \text{by classvar;} \\
\text{run};
\end{align*}
\]

\[
\begin{align*}
\text{proc glm data=ds order=data;} \\
\hspace{1cm} \text{class classvar;} \\
\hspace{1.5cm} \text{model y = classvar;} \\
\text{run;}
\end{align*}
\]
6.1. MODEL FITTING

or

```
proc genmod data=ds;
   class classvar (param=ref ref="level");
   model y = classvar;
run;
```

Note: The first code is necessary for procedures that use the more primitive `class` statement, as reviewed in 6.1.4. For these procedures, the default reference category is the last one. The `order` option can take other values, which may be useful. If the desired reference cell cannot be sorted to the end, it may be necessary to recode the category values to, e.g., A. Brown, B. Blue, and C. Green from Blue, Brown, and Green, before sorting. Sorting by descending `classvar` may also be useful. The second set of code will work in the `genmod`, `logistic`, and `surveylogistic` procedures.

```
R
ds = transform(ds, neworder = factor(classvar, levels=c("level", "otherlev1", "otherlev2"))
mod1 = lm(y ~ neworder, data=ds)
```

Note: The first level of a factor (by default that which appears first lexicographically) is the reference group. This can be modified through use of the `factor()` function.

6.1.4 Parameterization of categorical covariates

Example: 6.6.5

SAS and R handle this issue in different ways. In R, `as.factor()` can be applied within any model-fitting function. Parameterization of the covariate can be controlled as below. For SAS, some procedures accept a `class` statement to declare that a covariate is to be treated as categorical. The following procedures will not accept a class statement: `arima`, `catmod`, `factor`, `lifetest`, `loess`, `mcmc`, `nlm`, `nlmixed`, `reg`, and `varclus`. For these procedures, indicator (or “dummy”) variables must be created in a data step, though this should be done with caution. The following procedures accept a class statement which applies reference cell or indicator variable coding (described as `contr.SAS()` in the R note below) to the listed variables: `proc anova`, `candisc`, `discrim`, `fmm`, `gam`, `glimmix`, `glm`, `mixed`, `quantreg`, `robustreg`, `stepdisc`, and `surveyreg`. The value used as the referent can often be controlled, usually as an `order` option to the controlling `proc`, as in 7.10.11. For these procedures, other parameterizations must be coded in a data step. The following procedures accept multiple parameterizations, using the syntax shown below for `proc logistic`: `proc genmod` (defaults to reference cell coding), `proc logistic` (defaults to effect coding), `proc phreg` (defaults to reference cell coding), and `proc surveylogistic` (defaults to effect coding).

**SAS**

```
proc logistic data=ds;
   class x1 (param=paramtype) x2 (param=paramtype);
   ...
run;
```

or

```
proc logistic data=ds;
   class x1 x2 / param=paramtype;
   ...
run;
```

Note: Available `paramtypes` include: 1) `orthpoly`, equivalent to `contr.poly()`; 2) `effect` (the default for `proc logistic` and `proc surveylogistic`), equivalent to `contr.sum()`; and 3) `ref`, equivalent to `contr.SAS()`. In addition, if the same parameterization is desired
for all of the categorical variables in the model, it can be added in a statement such as
the second example. In this case, param=glm can be used to emulate the parameterization
found in the other procedures which accept class statements and in contr.SAS() within
R; this is the default for proc genmod and proc phreg.

R

```r
ds = transform(ds, x1f = as.factor(x1))
mod1 = lm(y ~ x1f, contrasts=list(x1f="contr.SAS"), data=ds)
```

*Note:* The `as.factor()` function creates a factor object, akin to how SAS treats
class variables in proc glm. The `contrasts` option for the `lm()` function specifies how the
levels of that factor object should be used within the function. The `levels` option to the
`factor()` function allows specification of the ordering of levels (the default is lexicographic).
An example can be found in 6.6.

The specification of the design matrix for analysis of variance and regression models
can be controlled using the `contrasts` option. Examples of options (for a factor with four
equally spaced levels) are given below.

```r
> contr.treatment(4)
   [,1] [,2] [,3]
1  1  0  0
2  0  1  0
3  0  0  1
4  0  0  0

> contr.poly(4)
   [,1]   .L  .Q  .C
1  1 0.671 0.5 -0.224
2  0 0.224 -0.5 0.671
3  0 -0.224 -0.5 -0.671
4  1  0.671 0.5  0.224

> contr.SAS(4)
1  1  1  0  0
2  1  0  1  0
3  0  1  0  1
4  0  0  1  1

> contr.sum(4)
1   1     1     0     0
2   2     0     1     0
3   3     0     0     1
4   4    -1    -1    -1

> contr.helmert(4)
1  -1    -1    -1    -1
2   1    -1    -1     1
3   0     1    -1    -1
4   0     0     3    -1
```

See `options("contrasts")` for defaults, and `contrasts()` or `lm()` to apply a contrast
function to a factor variable. Support for reordering factors is available within the `factor()`
function.

### 6.1.5 Linear regression with no intercept

**SAS**

```sas
proc glm data=ds;
   model y = x1 ... xk / noint;
run;
```

*Note:* The `noint` option works with many `model` statements.

**R**

```r
mod1 = lm(y ~ 0 + x1 + ... + xk, data=ds)
```
or

```r
mod1 = lm(y ~ x1 + ... + xk - 1, data=ds)
```
6.1. MODEL FITTING

6.1.6 Linear regression with interactions

**SAS**

```sas
proc glm data=ds;
   model y = x1 x2 x1*x2 x3 ... xk;
run;
```

or

```sas
proc glm data=ds;
   model y = x1|x2 x3 ... xk;
run;
```

*Note:* The | operator includes the product and all lower order terms, while the * operator includes only the specified interaction. So, for example, `model y = x1|x2|x3` and `model y = x1 x2 x3 x1*x2 x1*x3 x2*x3 x1*x2*x3` are equivalent statements. The syntax above also works with any covariates designated as categorical using the `class` statement (6.1.2). The `model` statement for many procedures accepts this syntax.

**R**

```r
mod1 = lm(y ~ x1 + x2 + x1:x2 + x3 + ... + xk, data=ds)
```

or

```r
lm(y ~ x1*x2 + x3 + ... + xk, data=ds)
```

*Note:* The * operator includes all lower order terms (in this case main effects), while the : operator includes only the specified interaction. So, for example, the commands `y ~ x1*x2*x3` and `y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:x2:x3` are equivalent. The syntax also works with any covariates designated as categorical using the `as.factor()` command (see 6.1.2).

6.1.7 One-way analysis of variance

**SAS**

```sas
proc glm data=ds;
   class x;
   model y = x / solution;
run;
```

*Note:* The `solution` option to the `model` statement requests that the parameter estimates be displayed. Other procedures which fit ANOVA models include `proc anova` and `proc mixed`.

**R**

```r
ds = transform(ds, xf=as.factor(x))
mod1 = aov(y ~ xf, data=ds)
summary(mod1)
anova(mod1)
```

*Note:* The `summary()` command can be used to provide details of the model fit. More information can be found using `help(summary.aov)`. Note that `summary.lm(mod1)` will display the regression parameters underlying the ANOVA model.

6.1.8 Analysis of variance with two or more factors

**Example:** 6.6.5

Interactions can be specified using the syntax introduced in 6.1.6 (see interaction plots, 8.5.2).
6.2 Tests, contrasts, and linear functions of parameters

6.2.1 Joint null hypotheses: several parameters equal 0

As an example, consider testing the null hypothesis $H_0 : \beta_1 = \beta_2 = 0$.

**SAS**
```
proc reg data=ds;
  model ...;
  nametest: test x1=0, x2=0;
run;
```

*Note:* In the above, `nametest` is an arbitrary label which will appear in the output. Multiple `test` statements are permitted.

**R**
```
mod1 = lm(y ~ x1 + ... + xk, data=ds)
mod2 = lm(y ~ x3 + ... + xk, data=ds)
anova(mod2, mod1)
```

6.2.2 Joint null hypotheses: sum of parameters

As an example, consider testing the null hypothesis $H_0 : \beta_1 + \beta_2 = 1$.

**SAS**
```
proc reg data=ds;
  model ...;
  nametest: test x1 + x2 = 1;
run;
```

*Note:* The `test` statement is prefixed with an arbitrary `nametest` which will appear in the output. Multiple `test` statements are permitted.

**R**
```
mod1 = lm(y ~ x1 + ... + xk, data=ds)
covb = vcov(mod1)
coeff.mod1 = coef(mod1)
    sqrt(covb[2,2] + covb[3,3] + 2*covb[2,3])
pvalue = 2*(1-pt(abs(t), df=mod1$df))
```

*Note:* The `I()` function inhibits the interpretation of operators, to allow them to be used as arithmetic operators.
6.2. TESTS, CONTRASTS, AND LINEAR FUNCTIONS OF PARAMETERS

6.2.3 Tests of equality of parameters

As an example, consider testing the null hypothesis $H_0: \beta_1 = \beta_2$.

**Example: 6.6.7**

**SAS**

```sas
proc reg data=ds;
   model ...;
   nametest: test x1=x2;
run;
```

*Note:* The `test` statement is prefixed with an arbitrary `nametest` which will appear in the output. Multiple `test` statements are permitted.

**R**

```r
mod1 = lm(y ~ x1 + ... + xk, data=ds)
mod2 = lm(y ~ I(x1+x2) + ... + xk, data=ds)
anova(mod2, mod1)
```

*or*

```r
library(gmodels)
estimable(mod1, c(0, 1, -1, 0, ..., 0))
```

*or*

```r
mod1 = lm(y ~ x1 + ... + xk, data=ds)
covb = vcov(mod1)
coeff.mod1 = coef(mod1)
t = (coeff.mod1[2]-coeff.mod1[3])/sqrt(covb[2,2]+covb[3,3]-2*covb[2,3])
pvalue = 2*(1-pt(abs(t), mod1$df))
```

*Note:* The `I()` function inhibits the interpretation of operators, to allow them to be used as arithmetic operators. The `estimable()` function calculates a linear combination of the parameters. The more general R code below utilizes the same approach introduced in 6.2.1 for the specific test of $\beta_1 = \beta_2$ (different coding would be needed for other comparisons).

6.2.4 Multiple comparisons

**Example: 6.6.6**

**SAS**

```sas
proc glm data=ds;
   class x1;
   model y = x1;
   lsmeans x1 / pdiff adjust=tukey;
run;
```

*Note:* The `pdiff` option requests p-values for the hypotheses involving the pairwise comparison of means. The `adjust` option adjusts these p-values for multiple comparisons. Other options available through `adjust` include `bon` (for Bonferroni) and `dunnett`, among others. SAS `proc mixed` also has an `adjust` option for its `lsmeans` statement. A graphical presentation of significant differences among levels can be obtained with the `lines` option to the `lsmeans` statement, as shown in 6.6.6.

**R**

```r
mod1 = aov(y ~ x, data=ds)
TukeyHSD(mod1, "x")
```

*Note:* The `TukeyHSD()` function takes an `aov` object as an argument and calculates the pairwise comparisons of all of the combinations of the factor levels of the variable x (see the `multcomp` package).
CHAPTER 6. LINEAR REGRESSION AND ANOVA

6.2.5 Linear combinations of parameters

It is often useful to calculate predicted values for particular covariate values. Here, we calculate the predicted value \( E[Y|X_1 = 1, X_2 = 3] = \hat{\beta}_0 + \hat{\beta}_1 + 3\hat{\beta}_2 \).

**Example:** 6.6.7

SAS

```sas
proc glm data=ds;
   model y = x1 ... xk;
   estimate 'label' intercept 1 x1 1 x2 3;
run;
```

*Note:* The `estimate` statement is used to calculate a linear combination of parameters (and associated standard errors). The optional quoted text is a label which will be printed with the estimated function.

R

```r
mod1 = lm(y ~ x1 + x2, data=ds)
nedf = data.frame(x1=c(1), x2=c(3))
predict(mod1, newdf, se.fit=TRUE, interval="confidence")
```

*Note:* The `predict()` command in R can generate estimates at any combination of parameter values, as specified as a dataframe that is passed as an argument. More information on this function can be found using `help(predict.lm)`.

6.3 Model diagnostics

6.3.1 Predicted values

**Example:** 6.6.2

SAS

```sas
proc reg data=ds;
   model ...
   output out=newds predicted=predicted_varname;
run;
```

or

```sas
proc glm data=ds;
   model ...
   output out=newds predicted=predicted_varname;
run;
```

*Note:* The `output` statement creates a new dataset and specifies variables to be included, of which the predicted values are an example. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

R

```r
mod1 = lm(y ~ x, data=ds)
predicted.varname = predict(mod1)
```

*Note:* The command `predict()` operates on any `lm` object and by default generates a vector of predicted values. Similar commands retrieve other regression output.
6.3. MODEL DIAGNOSTICS

6.3.2 Residuals

**SAS**

Example: 6.6.2

```sas
proc glm data=ds;
model ...;
    output out=newds residual=residual_varname;
run;
or
proc reg data=ds;
model ...;
    output out=newds residual=residual_varname;
run;
```

*Note:* The `output` statement creates a new dataset and specifies variables to be included, of which the residuals are an example. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

**R**

```r
mod1 = lm(y ~ x, data=ds)
residual.varname = residuals(mod1)
```

*Note:* The command `residuals()` operates on any `lm` object and generates a vector of residuals. Other functions for `aov`, `glm`, or `lme` objects exist (see, for example, `help(residuals.glm)`).

6.3.3 Standardized and Studentized residuals

**Example:** 6.6.2

Standardized residuals are calculated by dividing the ordinary residual (observed minus expected, $y_i - \hat{y}_i$) by an estimate of its standard deviation. Studentized residuals are calculated in a similar manner, where the predicted value and the variance of the residual are estimated from the model fit while excluding that observation. In SAS `proc glm` the standardized residual is requested by the `student` option, while the `rstudent` option generates the studentized residual.

**SAS**

```sas
proc glm data=ds;
    model ...;
    output out=newds student=standardized_resid_varname;
run;
or
proc reg data=ds;
    model ...;
    output out=newds rstudent=studentized_resid_varname;
run;
```

*Note:* The `output` statement creates a new dataset and specifies variables to be included, of which the Studentized residuals are an example. Both `proc reg` and `proc glm` include both types of residuals. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

**R**

```r
mod1 = lm(y ~ x, data=ds)
standardized.resid.varname = stdres(mod1)
studentized.resid.varname = studres(mod1)
```

*Note:* The `stdres()` and `studres()` functions operate on any `lm` object and generate a vector of studentized residuals (the former command includes the observation in the
calculation, while the latter does not). Similar commands retrieve other regression output (see help(influence.measures)).

6.3.4 Leverage

Example: 6.6.2

Leverage is defined as the diagonal element of the \((X(X^TX)^{-1}X^T)\) or “hat” matrix.

SAS

```
proc glm data=ds;
  model ...;
  output out=newds h=leverage_varname;
run;
```

or

```
proc reg data=ds;
  model ...;
  output out=newds h=leverage_varname;
run;
```

Note: The output statement creates a new dataset and specifies variables to be included, of which the leverage values are one example. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

R

```
mod1 = lm(y ~ x, data=ds)
leverage.varname = hatvalues(mod1)
```

Note: The command hatvalues() operates on any lm object and generates a vector of leverage values. Similar commands can be utilized to retrieve other regression output (see help(influence.measures)).

6.3.5 Cook’s D

Example: 6.6.2

Cook’s distance (D) is a function of the leverage (see 6.3.4) and the residual. It is used as a measure of the influence of a data point in a regression model.

SAS

```
proc glm data=ds;
  model ...;
  output out=newds cookd=cookd_varname;
run;
```

or

```
proc reg data=ds;
  model ...;
  output out=newds cookd=cookd_varname;
run;
```

Note: The output statement creates a new dataset and specifies variables to be included, of which the Cook’s distance values are an example. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

R

```
mod1 = lm(y ~ x, data=ds)
cookd.varname = cooks.distance(mod1)
```

Note: The command cooks.distance() operates on any lm object and generates a vector of Cook’s distance values. Similar commands retrieve other regression output.
6.3. MODEL DIAGNOSTICS

6.3.6 DFFITS

DFFITS are a standardized function of the difference between the predicted value for the observation when it is included in the dataset and when (only) it is excluded from the dataset. They are used as an indicator of the observation’s influence.

**SAS**

```sas
proc reg data=ds;
    model ...;
    output out=newds dffits=dffits_varname;
run;
```
or

```sas
proc glm data=ds;
    model ...;
    output out=newds dffits=dffits_varname;
run;
```

**Note:** The `output` statement creates a new dataset and specifies variables to be included, of which the DFFITS values are an example. Others can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

**R**

```r
mod1 = lm(y ~ x, data=ds)
dffits.varname = dffits(mod1)
```

**Note:** The command `dffits()` operates on any `lm` object and generates a vector of DFFITS values. Similar commands retrieve other regression output.

6.3.7 Diagnostic plots

**SAS**

```sas
proc reg data=ds;
    model ...;
    output out=newds predicted=pred_varname residual=resid_varname
        h=leverage_varname cookd=cookd_varname;
run;
```

```sas
proc gplot data=ds;
    plot resid_varname * pred_varname;
    plot resid_varname * leverage_varname;
run;
quit;
```

**Note:** To mimic R more closely, use a data step to generate the square root of residuals. QQ plots of residuals can be generated via `proc univariate`. It is not straightforward to plot lines of constant Cook’s D on the residuals vs. leverage plot. The `reg` procedure will produce many diagnostic plots, as will `proc glm` with the `plots=diagnostics` option.

**R**

```r
mod1 = lm(y ~ x, data=ds)
par(mfrow=c(2, 2)) # display 2 x 2 matrix of graphs
plot(mod1)
```

**Note:** The `plot.lm()` function (which is invoked when `plot()` is given a linear regression model as an argument) can generate six plots: 1) a plot of residuals against fitted values, 2) a Scale-Location plot of $\sqrt{(Y_i - \hat{Y}_i)}$ against fitted values, 3) a normal Q-Q plot of the
residuals, 4) a plot of Cook’s distances (6.3.5) versus row labels, 5) a plot of residuals against leverages (6.3.4), and 6) a plot of Cook’s distances against leverage/(1–leverage). The default is to plot the first three and the fifth. The which option can be used to specify a different set (see help(plot.lm)).

6.3.8 Heteroscedasticity tests

SAS

```sas
proc model data=ds;
  parms int slope;
  y = int + slope * x;
  fit y / pagan=(1 x) white;
run; quit;
```

*Note:* In SAS, White’s test [192] and the Breusch–Pagan test [19] can be found in the model procedure in the SAS/ETS product. Note the atypical syntax of `proc model`.

R

```r
library(lmtest)
bptest(y ~ x1 + ... + xk, data=ds)
```

*Note:* The `bptest()` function in the `lmtest` package performs the Breusch–Pagan test for heteroscedasticity [19].

6.4 Model parameters and results

6.4.1 Parameter estimates

SAS

```sas
ods output parameterestimates=newds;
proc glm data=ds;
  model ... / solution;
run;
or
proc reg data=ds outest=newds;
  model ...;
run;
```

*Note:* The `ods output` statement (A.7.1) can be used to save any piece of SAS output as a SAS dataset. The `outest` option is specific to `proc reg`, though many other procedures accept similar syntax.

R

```r
mod1 = lm(y ~ x, data=ds)
coeff.mod1 = coef(mod1)
```

*Note:* The first element of the vector `coeff.mod1` is the intercept (assuming that a model with an intercept was fit).

6.4.2 Standardized regression coefficients

Standardized coefficients from a linear regression model are the parameter estimates obtained when the predictors and outcomes have been standardized to have a variance of 1 prior to model fitting.
6.4. MODEL PARAMETERS AND RESULTS

SAS
proc reg data=ds;
   model ... / stb;
run;

R
library(QuantPsyc)
mod1 = lm(y ~ x)
lm.beta(lm1)

6.4.3 Standard errors of parameter estimates
See 6.4.9 (covariance matrix).

SAS
proc reg data=ds outest=newds;
   model .../ outseb ...;
run;
or
ods output parameterestimates=newds;
proc glm data=ds;
   model .../ solution;
run;
Note: The ods output statement (A.7.1) can be used to save any piece of SAS output as
a SAS dataset.

R
mod1 = lm(y ~ x, data=ds)
sqrt(diag(vcov(mod1)))
or
coef(summary(mod1))[,2]
Note: The standard errors are the second column of the results from coef().

6.4.4 Confidence interval for parameter estimates

Example: 6.6.2

SAS
proc reg data=ds;
   model ... / clb;
run;

R
mod1 = lm(y ~ x, data=ds)
confint(mod1)

6.4.5 Confidence limits for the mean
These are the lower (and upper) confidence limits for the mean of observations with the
given covariate values, as opposed to the prediction limits for individual observations with
those values (see prediction limits, 6.4.6).

SAS
proc glm data=ds;
   model ...;
       output out=newds lclm=lcl_mean_varname;
run;
or

\begin{verbatim}
proc reg data=ds;
   model ...;
   output out=newds lcl=lcl_mean_varname;
run;
\end{verbatim}

\textit{Note:} The \texttt{output} statement creates a new dataset and specifies output variables to be included, of which the lower confidence limit values are one example. The upper confidence limits can be generated using the \texttt{uclm} option to the \texttt{output} statement. Other possibilities can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

\textbf{R}

\begin{verbatim}
mod1 = lm(y ~ x, data=ds)
pred = predict(mod1, interval="confidence")
lcl.varname = pred[,2]
\end{verbatim}

\textit{Note:} The lower confidence limits are the second column of the results from \texttt{predict()}. To generate the upper confidence limits, the user would replace \texttt{lclm} with \texttt{uclm} for SAS and access the third column of the \texttt{predict()} object in R. The command \texttt{predict()} operates on any \texttt{lm()} object, and with these options generates confidence limit values. By default, the function uses the estimation dataset, but a separate dataset of values to be used to predict can be specified. The \texttt{panel=panel.lmbands} option from the \texttt{mosaic} package can be added to an \texttt{xyplot()} call to augment the scatterplot with confidence interval and prediction bands.

### 6.4.6 Prediction limits

These are the lower (and upper) prediction limits for “new” observations with the covariate values of subjects observed in the dataset, as opposed to confidence limits for the population mean (see confidence limits, 6.4.5).

\textbf{SAS}

\begin{verbatim}
proc glm data=ds;
   model ...;
   output out=newds lcl=lcl_varname;
run;
\end{verbatim}

\begin{verbatim}
or
proc reg data=ds;
   model ...;
   output out=newds lcl=lcl_varname;
run;
\end{verbatim}

\textit{Note:} The \texttt{output} statement creates a new dataset and specifies variables to be included, of which the lower prediction limit values are an example. The upper limits can be requested with the \texttt{ucl} option to the \texttt{output} statement. Other possibilities can be found using the on-line help: Contents; SAS Products; SAS Procedures; REG; OUTPUT.

\textbf{R}

\begin{verbatim}
mod1 = lm(y ~ ..., data=ds)
pred.w.lowlim = predict(mod1, interval="prediction")[2]
\end{verbatim}

\textit{Note:} This code saves the second column of the results from the \texttt{predict()} function into a vector. To generate the upper confidence limits, the user would access the third column of the \texttt{predict()} object in R. The command \texttt{predict()} operates on any \texttt{lm()} object, and with these options generates prediction limit values. By default, the function uses the estimation dataset, but a separate dataset of values to be used to predict can be specified.
6.4. MODEL PARAMETERS AND RESULTS

6.4.7 R-squared

**SAS**

```
proc glm data=ds;
  model ...;
run;
```

*Note:* The coefficient of determination can be found as default output from `proc reg` or `proc glm`.

**R**

```
mod1 = lm(y ~ ..., data=ds)
summary(mod1)$r.squared
```

or

```
library(mosaic)
rsquared(mod1)
```

6.4.8 Design and information matrix

See 3.3 (matrices).

**SAS**

```
proc reg data=ds;
  model .../ xpx ...;
run;
```

or

```
proc glm data=ds;
  model .../ xpx ...;
run;
```

*Note:* A dataset containing the information ($X'X$) matrix can be created using ODS by specifying either `proc` statement or by adding the option `outsscp=newds` to the `proc reg` statement.

**R**

```
mod1 = lm(y ~ x1 + ... + xk, data=ds)
XpX = t(model.matrix(mod1)) %*% model.matrix(mod1)
```

or

```
X = cbind(rep(1, length(x1)), x1, x2, ..., xk)
XpX = t(X) %*% X
rm(X)
```

*Note:* The `model.matrix()` function creates the design matrix from a linear model object. Alternatively, this quantity can be built up using the `cbind()` function to glue together the design matrix X. Finally, matrix multiplication (3.3.6) and the transpose function are used to create the information ($X'X$) matrix.

6.4.9 Covariance matrix of parameter estimates

*Example:* 6.6.2

See 3.3 (matrices) and 6.4.3 (standard errors).

**SAS**

```
proc reg data=ds outest=newds covout;
run;
```
or
ods output covb=newds;
proc reg data=ds;
  model ... / covb ...;
R
mod1 = lm(y ~ x, data=ds)
vcoov(mod1)
or
sumvals = summary(mod1)
covb = sumvals$ cov.unscaled*sumvals$ sigma^2
Note: Running help(summary.lm) provides details on return values.

6.4.10  Correlation matrix of parameter estimates
See 3.3 (matrices) and 6.4.3 (standard errors).
SAS
ods output covb=lmcov corrb=lmcorr ;
proc reg data=ds;
  model ... / covb corrb ...;
or
proc reg data=ds outest=newds covout;
  model ...;
  outest=outds;
run;
Note: the former can be used to generate either the covariance or correlation matrix, or both. The demonstrated ODS command will save the matrix or matrices as datasets. The latter uses the older method of generating SAS output as a dataset, but does not allow the generation of the correlation matrix.
R
mod1 = lm(y ~ x, data=ds)
mod1.cov = vcov(mod1)
mod1.cor = cov2cor(mod1.cov)
Note: The cov2cor() function is a convenient way to convert a covariance matrix into a correlation matrix.

6.5  Further resources
Accessible guides to linear regression in R and SAS can be found in [38] and [108], respectively. Cook [29] reviews regression diagnostics. Frank Harrell’s rms (regression modeling strategies) package [63] features extensive support for regression modeling. The CRAN statistics for the social sciences task view provides an excellent overview of methods described here and in Chapter 7.

6.6  Examples
To help illustrate the tools presented in this chapter, we apply many of the entries to the HELP data. SAS and R code can be downloaded from http://www.amherst.edu/~nhorton/sasr2/examples.
6.6. EXAMPLES

We begin by reading in the dataset and keeping only the female subjects. In R, we prepare for later analyses by creating a version of `substance` as a factor variable (see 6.1.4).

```r
proc import datafile='c:/book/help.dta'
  out=help_a dbms=dta;
run;

data help;
set help_a;
  if female;
run;

> options(digits=3)
> # read in Stata format
> library(foreign)
> ds = read.dta("help.dta", convert.underscore=FALSE)
> ds = transform(ds, sub = factor(substance,
    levels=c("heroin", "alcohol", "cocaine")))
> newds = subset(ds, female==1)
```

6.6.1 Scatterplot with smooth fit

As a first step to help guide fitting a linear regression, we create a scatterplot (8.3.1) displaying the relationship between age and the number of alcoholic drinks consumed in the period before entering detox (variable name: `i1`), as well as primary substance of abuse (alcohol, cocaine, or heroin).

Figure 6.1 displays a scatterplot of observed values for `i1` (along with separate smooth fits by primary substance). To improve legibility, the plotting region is restricted to those with number of drinks between 0 and 40 (see plotting limits, 9.2.9).

```r
axis1 order = (0 to 40 by 10) minor=none;
axis2 minor=none;
legend1 label=none value=(h=1.5) shape=symbol(10,1.2)
  down=3 position=(top right inside) frame mode=protect;
symbol1 v=circle i=sm70s c=black l=1 h=1.1 w=5;
symbol2 v=diamond i=sm70s c=black l=33 h=1.1 w=5;
symbol3 v=square i=sm70s c=black l=8 h=1.1 w=5;
proc gplot data=help;
  plot i1*age = substance / vaxis=axis1 haxis=axis2 legend=legend1;
run; quit;
```
> with(newds, plot(age, i1, ylim=c(0,40), type="n", cex.lab=1.4, cex.axis=1.4))
> with(newds, points(age[substance=="alcohol"], i1[substance=="alcohol"], pch="a"))
> with(newds, lines(lowess(age[substance=="alcohol"], i1[substance=="alcohol"]), lty=1, lwd=2))
> with(newds, points(age[substance=="cocaine"], i1[substance=="cocaine"], pch="c"))
> with(newds, lines(lowess(age[substance=="cocaine"], i1[substance=="cocaine"], lty=2, lwd=2))
> with(newds, points(age[substance=="heroin"], i1[substance=="heroin"], pch="h"))
> with(newds, lines(lowess(age[substance=="heroin"], i1[substance=="heroin"], lty=3, lwd=2))
> legend(44, 38, legend=c("alcohol", "cocaine", "heroin"), lty=1:3, cex=1.4, lwd=2, pch=c("a", "c", "h"))

The pch option to the legend() command can be used to insert plot symbols in R legends (Figure 6.1 displays the different line styles).

Figure 6.1: Scatterplot of observed values for age and I1 (plus smoothers by substance)

Not surprisingly, Figure 6.1 suggests that there is a dramatic effect of primary substance, with alcohol users drinking more than others. There is some indication of an interaction with age. It is important to note that SAS uses only the points displayed (i.e., within the specified axes) when smoothing, while R uses all points, regardless of whether they appear in the plot.

6.6.2 Linear regression with interaction

Next we fit a linear regression model (6.1.1) for the number of drinks as a function of age, substance, and their interaction (6.1.6). To assess the need for the interaction, we use the F test from the Type III sums of squares in SAS. In R, we additionally fit the model with
no interaction and use the `anova()` function to compare the models (the `drop1()` function could also be used). To save space, some results of `proc glm` have been suppressed using the `ods select` statement (see A.7).

```plaintext
options ls=74; /* reduces width of output to make it fit in gray area */
ods select overallanova modelanova parameterestimates;
proc glm data=help;
class substance;
   model i1 = age substance age * substance / solution;
   output out=helpout cookd=cookd_ch4 dffits=dffits_ch4
       student=sresids_ch4 residual=resid_ch4
       predicted=pred_ch4 h=lev_ch4;
run; quit;
ods select all;

The GLM Procedure

Dependent Variable: i1 i1

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>5</td>
<td>12275.17570</td>
<td>2455.03514</td>
<td>9.99</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>101</td>
<td>24815.36635</td>
<td>245.69670</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>106</td>
<td>37090.54206</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The GLM Procedure

Dependent Variable: i1 i1

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type I SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>1</td>
<td>384.75504</td>
<td>384.75504</td>
<td>1.57</td>
<td>0.2137</td>
</tr>
<tr>
<td>substance</td>
<td>2</td>
<td>10509.56444</td>
<td>5254.78222</td>
<td>21.39</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>age*substance</td>
<td>2</td>
<td>1380.85622</td>
<td>690.42811</td>
<td>2.81</td>
<td>0.0649</td>
</tr>
</tbody>
</table>

Source  | DF | Type III SS | Mean Square | F Value | Pr > F |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>1</td>
<td>27.157727</td>
<td>27.157727</td>
<td>0.11</td>
<td>0.7402</td>
</tr>
<tr>
<td>substance</td>
<td>2</td>
<td>3318.992822</td>
<td>1659.496411</td>
<td>6.75</td>
<td>0.0018</td>
</tr>
<tr>
<td>age*substance</td>
<td>2</td>
<td>1380.85622</td>
<td>690.428111</td>
<td>2.81</td>
<td>0.0649</td>
</tr>
</tbody>
</table>
```
The GLM Procedure

Dependent Variable: i1  i1

| Parameter          | Estimate  | Standard Error | t Value | Pr > |t| |
|--------------------|-----------|----------------|---------|------|-----|
| Intercept          | -7.77045212 | 12.8785672    | -0.60   | 0.5476 |
| age                | 0.39337843 | 0.36221749    | 1.09    | 0.2801 |
| substance alcohol  | 64.88044165 | 18.48733701   | 3.51    | 0.0007 |
| substance cocaine  | 13.02733169 | 19.13852222   | 0.68    | 0.4976 |
| substance heroin   | 0.00000000 | .              | .       | .     |
| age*substance alcohol| -1.11320795 | 0.49135408    | -2.27   | 0.0256 |
| age*substance cocaine| -0.27758561 | 0.53967749    | -0.51   | 0.6081 |
| age*substance heroin| 0.00000000 | .              | .       | .     |

> options(show.signif.stars=FALSE)
> lm1 = lm(i1 ~ sub * age, data=newds)
> lm2 = lm(i1 ~ sub + age, data=newds)
> anova(lm2, lm1)

Analysis of Variance Table

<table>
<thead>
<tr>
<th>Model 1: i1 ~ sub + age</th>
<th>Model 2: i1 ~ sub * age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Df</td>
<td>RSS</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>103</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
</tr>
</tbody>
</table>

> summary.aov(lm1)

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub</td>
<td>2</td>
<td>10810</td>
<td>5405</td>
<td>22.00</td>
</tr>
<tr>
<td>age</td>
<td>1</td>
<td>84</td>
<td>84</td>
<td>0.34</td>
</tr>
<tr>
<td>sub:age</td>
<td>2</td>
<td>1381</td>
<td>690</td>
<td>2.81</td>
</tr>
<tr>
<td>Residuals</td>
<td>101</td>
<td>24815</td>
<td>246</td>
<td></td>
</tr>
</tbody>
</table>

There is some indication of a borderline significant interaction between age and substance group (p=0.065).

In SAS, the ods output statement can be used to save any printed result as a SAS dataset. In the following code, all printed output from proc glm is suppressed, but the parameter estimates are saved as a SAS dataset, then printed using proc print. In addition, various diagnostics are saved via the output statement.

ods select none;
ods output parameterestimates=helpmodelanova;
proc glm data=help;
class substance;
model i1 = age|substance / solution;
output out=helpout cookd=cookd_ch4 dffits=dffits_ch4
   student=sresids_ch4 residual=resid_ch4
   predicted=pred_ch4 h=lev_ch4;
run; quit;
ods select all;
### 6.6. EXAMPLES

```sas
proc print data=helpmodelanova;
   var parameter estimate stderr tvalue probt;
   format _numeric_ 6.3;
run;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>Parameter</th>
<th>Estimate</th>
<th>StdErr</th>
<th>tValue</th>
<th>Probt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intercept</td>
<td>-7.770</td>
<td>12.879</td>
<td>-0.603</td>
<td>0.548</td>
</tr>
<tr>
<td>2</td>
<td>age</td>
<td>0.393</td>
<td>0.362</td>
<td>1.086</td>
<td>0.280</td>
</tr>
<tr>
<td>3</td>
<td>substance alcohol</td>
<td>64.880</td>
<td>18.487</td>
<td>3.509</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>substance cocaine</td>
<td>13.027</td>
<td>19.139</td>
<td>0.681</td>
<td>0.498</td>
</tr>
<tr>
<td>5</td>
<td>substance heroin</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>age*substance alcohol</td>
<td>-1.113</td>
<td>0.491</td>
<td>-2.266</td>
<td>0.026</td>
</tr>
<tr>
<td>7</td>
<td>age*substance cocaine</td>
<td>-0.278</td>
<td>0.540</td>
<td>-0.514</td>
<td>0.608</td>
</tr>
<tr>
<td>8</td>
<td>age*substance heroin</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In R, the `summary()` function provides similar information.

```r
> summary(lm1)
```

Call:
`lm(formula = i1 ~ sub * age, data = newds)`

Residuals:

<table>
<thead>
<tr>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>-31.92</td>
<td>-8.25</td>
<td>-4.18</td>
<td>3.58</td>
<td>49.88</td>
</tr>
</tbody>
</table>

Coefficients:

|                | Estimate | Std. Error | t value | Pr(>|t|) |
|----------------|----------|------------|---------|---------|
| (Intercept)    | -7.770   | 12.879     | -0.60   | 0.54763 |
| subalcohol     | 64.880   | 18.487     | 3.51    | 0.00067 |
| subcocaine     | 13.027   | 19.139     | 0.68    | 0.49763 |
| age            | 0.393    | 0.362      | 1.09    | 0.28005 |
| subalcohol:age | -1.113   | 0.491      | -2.27   | 0.02561 |
| subcocaine:age | -0.278   | 0.540      | -0.51   | 0.60813 |

Residual standard error: 15.7 on 101 degrees of freedom
Multiple R-squared: 0.331, Adjusted R-squared: 0.298
F-statistic: 9.99 on 5 and 101 DF, p-value: 8.67e-08

```r
> confint(lm1)
```

<table>
<thead>
<tr>
<th></th>
<th>2.5 %</th>
<th>97.5 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-33.319</td>
<td>17.778</td>
</tr>
<tr>
<td>subalcohol</td>
<td>28.207</td>
<td>101.554</td>
</tr>
<tr>
<td>subcocaine</td>
<td>-24.938</td>
<td>50.993</td>
</tr>
<tr>
<td>age</td>
<td>-0.325</td>
<td>1.112</td>
</tr>
<tr>
<td>subalcohol:age</td>
<td>-2.088</td>
<td>-0.138</td>
</tr>
<tr>
<td>subcocaine:age</td>
<td>-1.348</td>
<td>0.793</td>
</tr>
</tbody>
</table>

It may also be useful to produce the table in \textit{LaTeX} format. In SAS, we can do this using the `latex` destination for the ODS system. When compiled, the resulting table is displayed in Figure 6.2.
In R, we can use the `xtable` package to display the regression results in \LaTeX, as shown in Table 6.1.

```r
> library(xtable)
> lmtab = xtable(lm1, digits=c(0,3,3,2,4), label="better",
>                  caption="Formatted results using the \{xtable\} package")
> print(lmtab) # output the LaTeX
```

Table 6.1: Formatted results using the `xtable` package

|                                | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------------------------|----------|------------|---------|----------|
| (Intercept)                    | -7.770   | 12.879     | -0.60   | 0.5476   |
| subalcohol                     | 64.880   | 18.487     | 3.51    | 0.0007   |
| subcocaine                     | 13.027   | 19.139     | 0.68    | 0.4976   |
| age                            | 0.393    | 0.362      | 1.09    | 0.2801   |
| subalcohol:age                 | -1.113   | 0.491      | -2.27   | 0.0256   |
| subcocaine:age                 | -0.278   | 0.540      | -0.51   | 0.6081   |

There are many quantities of interest stored in the linear model object `lm1`, and these can be viewed or extracted for further use.

```r
> names(summary(lm1))
```

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

```r
> summary(lm1)$sigma
```

```
15.7
```

```r
> names(lm1)
```

```r
[1] "coefficients"  "residuals"  "effects"  "rank"
[5] "fitted.values" "assign"  "qr"        "df.residual"
[9] "contrasts"  "xlevels"  "call"       "terms"
[13] "model"
```
6.6. EXAMPLES

> coef(lm1)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-7.770</td>
<td>64.880</td>
<td>13.027</td>
<td>0.393</td>
<td>-1.113</td>
</tr>
<tr>
<td>subalcohol:age</td>
<td>-0.278</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> vcov(lm1)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>165.86</td>
<td>-165.86</td>
<td>-165.86</td>
<td>4.548</td>
<td>4.548</td>
</tr>
<tr>
<td>subalcohol</td>
<td>-165.86</td>
<td>341.78</td>
<td>165.86</td>
<td>4.548</td>
<td>-8.866</td>
</tr>
<tr>
<td>subcocaine</td>
<td>-165.86</td>
<td>165.86</td>
<td>366.28</td>
<td>4.548</td>
<td>-4.548</td>
</tr>
<tr>
<td>age</td>
<td>-4.55</td>
<td>4.55</td>
<td>4.55</td>
<td>0.131</td>
<td>-0.131</td>
</tr>
<tr>
<td>subalcohol:age</td>
<td>4.55</td>
<td>-8.87</td>
<td>-4.55</td>
<td>-0.131</td>
<td>0.241</td>
</tr>
<tr>
<td>subcocaine:age</td>
<td>4.55</td>
<td>-4.55</td>
<td>-10.13</td>
<td>-0.131</td>
<td>0.131</td>
</tr>
</tbody>
</table>

6.6.3 Regression diagnostics

Assessing the model is an important part of any analysis. We begin by examining the residuals (6.3.2). First, we calculate the quantiles of their distribution (5.1.4), then display the smallest residual.

```plaintext
options ls=74;
proc means data=helpout min q1 median q3 max maxdec=2;
  var resid_ch4;
run;
```

The MEANS Procedure

<table>
<thead>
<tr>
<th>Analysis Variable : resid_ch4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>-31.92</td>
</tr>
</tbody>
</table>

```
> newds = transform(newds, pred = fitted(lm1))
> newds = transform(newds, resid = residuals(lm1))
> with(newds, quantile(resid))

0%    25%    50%    75%    100%
-31.92 -8.25 -4.18  3.58  49.88
```

We could examine the output, then condition to find the value of the residual that is less than -31. Instead the dataset can be sorted so the smallest observation is first and then print one observation.
CHAPTER 6. LINEAR REGRESSION AND ANOVA

```r
> tmpds = with(newds,
>   data.frame(id, age, i1, sub, pred, resid, rstandard(lm1)))
> subset(tmpds, resid==max(resid))

   id age i1  sub  pred resid  rstandard.lm1.
9  9  50 71 alcohol  21.1 49.9     3.32
```

Graphical tools are one of the best ways to examine residuals. Figure 6.3 displays the default diagnostic plots (6.3) from the model (for R) and the Q-Q plot generated from the saved diagnostics (for SAS).

Sometimes in SAS it is necessary to clear out old graphics settings. This is easiest to do with the `goptions reset=all` statement (9.2.8).

```r
> oldpar = par(mfrow=c(2, 2), mar=c(4, 4, 2, 2)+.1)
> plot(lm1)
> par(oldpar)
```

In SAS, we get assorted diagnostic plots by default, but here we demonstrate a manual approach using the previously saved diagnostics. Figure 6.4 displays the empirical density of the standardized residuals, along with an overlaid normal density. The assumption that the residuals are approximately Gaussian does not appear to be tenable.
6.6. EXAMPLES

Figure 6.3: Q-Q plot from SAS, default diagnostics from R

Figure 6.4: Empirical density of residuals, with superimposed normal density

```r
axis1 label="Standardized residuals";
ods select "Histogram 1";
proc univariate data=helpout;
    var sresids_ch4;
    histogram sresids_ch4 / normal(mu=est sigma=est color=black)
        kernel(color=black) haxis=axis1;
run;
ods select all;
```
> library(MASS)
> std.res = rstandard(lm1)
> hist(std.res, breaks=seq(-2.5, 3.5, by=.5), main="",
> xlab="standardized residuals", col="gray80", freq=FALSE)
> lines(density(std.res), lwd=2)
> xvals = seq(from=min(std.res), to=max(std.res), length=100)
> lines(xvals, dnorm(xvals, mean(std.res), sd(std.res)), lty=2)

The residual plots indicate some potentially important departures from model assumptions, and further exploration should be undertaken.

### 6.6.4 Fitting the regression model separately for each value of another variable

One common task is to perform identical analyses in several groups. Here, as an example, we consider separate linear regressions for each substance abuse group. In SAS, we show only the parameter estimates, using ODS.

```sas
ods select none;
proc sort data=help;
   by substance;
run;

ods output parameterestimates=helpsubstparams;
proc glm data=help;
   by substance;
   model i1 = age / solution;
run;
ods select all;
options ls=74;
proc print data=helpsubstparams;
run;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>substance</th>
<th>Dependent</th>
<th>Parameter</th>
<th>Estimate</th>
<th>StdErr</th>
<th>tValue</th>
<th>Probt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>alcohol</td>
<td>i1</td>
<td>Intercept</td>
<td>57.10998953</td>
<td>18.00474934</td>
<td>3.17</td>
<td>0.0032</td>
</tr>
<tr>
<td>2</td>
<td>alcohol</td>
<td>i1</td>
<td>age</td>
<td>-0.71982952</td>
<td>0.45069028</td>
<td>-1.60</td>
<td>0.1195</td>
</tr>
<tr>
<td>3</td>
<td>cocaine</td>
<td>i1</td>
<td>Intercept</td>
<td>5.25687957</td>
<td>11.52989056</td>
<td>0.46</td>
<td>0.6510</td>
</tr>
<tr>
<td>4</td>
<td>cocaine</td>
<td>i1</td>
<td>age</td>
<td>0.11579282</td>
<td>0.32582541</td>
<td>0.36</td>
<td>0.7242</td>
</tr>
<tr>
<td>5</td>
<td>heroin</td>
<td>i1</td>
<td>Intercept</td>
<td>-7.77045212</td>
<td>8.59729637</td>
<td>-0.90</td>
<td>0.3738</td>
</tr>
<tr>
<td>6</td>
<td>heroin</td>
<td>i1</td>
<td>age</td>
<td>0.39337843</td>
<td>0.24179872</td>
<td>1.63</td>
<td>0.1150</td>
</tr>
</tbody>
</table>

For R, a matrix of the correct size is created, then a for loop is run for each unique value of the grouping variable.
6.6.5 Two-way ANOVA

Is there a statistically significant association between gender and substance abuse group with depressive symptoms? In SAS, we can make an interaction plot (8.5.2) by hand, as below, or proc glm will make a similar one automatically.

```sas
libname k 'c:/book';

proc sort data=k.help;
   by substance female;
run;

ods select none;
proc means data=k.help;
   by substance female;
   var cesd;
   output out=helpmean mean=;
run;
ods select all;
axis1 minor=none;
symbol1 i=j v=none l=1 c=black w=5;
symbol2 i=j v=none l=2 c=black w=5;
proc gplot data=helpmean;
   plot cesd*substance = female / haxis=axis1 vaxis=axis1;
run; quit;
```

R has a function `interaction.plot()` to carry out this task. Figure 6.5 displays an interaction plot for CESD as a function of substance group and gender.

```r
> ds = transform(ds, genf = as.factor(ifelse(female, "F", "M")))
> with(ds, interaction.plot(substance, genf, cesd,
                       xlab="substance", las=1, lwd=2))
```

There are indications of large effects of gender and substance group, but little suggestion of interaction between the two. The same conclusion is reached in Figure 6.6, which displays boxplots by substance group and gender.
CHAPTER 6. LINEAR REGRESSION AND ANOVA

Figure 6.5: Interaction plot of CESD as a function of substance group and gender

Figure 6.6: Boxplot of CESD as a function of substance group and gender

data h2; set k.help;
   if female eq 1 then sex='F';
   else sex='M';
run;
proc sort data=h2; by sex; run;

symbol1 v='x' c=black;
proc boxplot data=h2;
   plot cesd * substance(sex) / notches boxwidthscale=1;
run;
6.6. EXAMPLES

```r
> library(memisc)
> ds = transform(ds, subs = cases(
    "Alc" = substance=="alcohol",
    "Coc" = substance=="cocaine",
    "Her" = substance=="heroin"))
> boxout = with(ds,
    boxplot(cesd ~ subs + genf, notch=TRUE, varwidth=TRUE,
    col="gray80"))
> boxmeans = with(ds, tapply(cesd, list(subs, genf), mean))
> points(seq(boxout$n), boxmeans, pch=4, cex=2)
```

The width of each box is proportional to the size of the sample, with the notches denoting confidence intervals for the medians and X's marking the observed means. Next, we proceed to formally test whether there is a significant interaction through a two-way analysis of variance (6.1.8). In SAS, the Type III sums of squares table can be used to assess the interaction; we restrict output to this table to save space. In R we fit models with and without an interaction, and then compare the results. We also construct the likelihood ratio test manually.

```r
options ls=74;
ods select modelanova;
proc glm data=k.help;
    class female substance;
    model cesd = female substance female*substance / ss3;
run;
```

The GLM Procedure

Dependent Variable: CESD

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMALE</td>
<td>1</td>
<td>2463.232928</td>
<td>2463.232928</td>
<td>16.84</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>SUBSTANCE</td>
<td>2</td>
<td>2540.208432</td>
<td>1270.104216</td>
<td>8.69</td>
<td>0.0002</td>
</tr>
<tr>
<td>FEMALE*SUBSTANCE</td>
<td>2</td>
<td>145.924987</td>
<td>72.962494</td>
<td>0.50</td>
<td>0.6075</td>
</tr>
</tbody>
</table>

```r
> aov1 = aov(cesd ~ sub * genf, data=ds)
> aov2 = aov(cesd ~ sub + genf, data=ds)
> resid = residuals(aov2)
> anova(aov2, aov1)
```

Analysis of Variance Table

<table>
<thead>
<tr>
<th>Model 1: cesd ~ sub + genf</th>
<th>Model 2: cesd ~ sub * genf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.Df</td>
<td>RSS</td>
</tr>
<tr>
<td>1</td>
<td>449</td>
</tr>
</tbody>
</table>
```
> options(digits=6)
> logLik(aov1)
'log Lik.' -1768.92 (df=7)
> logLik(aov2)
'log Lik.' -1769.42 (df=5)
> lldiff = logLik(aov1)[1] - logLik(aov2)[1]
> lldiff
[1] 0.505055
> 1 - pchisq(2*lldiff, df=2)
[1] 0.603472
> options(digits=3)
> summary(aov2)

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub</td>
<td>2</td>
<td>2704</td>
<td>1352</td>
<td>9.27</td>
<td>0.00011</td>
</tr>
<tr>
<td>genf</td>
<td>1</td>
<td>2569</td>
<td>2569</td>
<td>17.61</td>
<td>3.3e-05</td>
</tr>
<tr>
<td>Residuals</td>
<td>449</td>
<td>65515</td>
<td>146</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is little evidence (p=0.61) of an interaction, so this term can be dropped. For SAS, this means estimating the reduced model.

options ls=74; /* stay in gray box */
ods select overallanova parameterestimates;
proc glm data=k.help;
   class female substance;
   model cesd = female substance / ss3 solution;
run;

The GLM Procedure

Dependent Variable: CESD

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>3</td>
<td>5273.13263</td>
<td>1757.71088</td>
<td>12.05</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>449</td>
<td>65515.35744</td>
<td>145.91394</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>452</td>
<td>70788.49007</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.6. EXAMPLES

The GLM Procedure

Dependent Variable: CESD

| Parameter          | Estimate     | Standard Error | t Value | Pr > |t| |
|--------------------|--------------|----------------|---------|-------|
| Intercept          | 39.13070331 B| 1.48571047     | 26.34   | <.0001|
| FEMALE 0           | -5.61922564 B| 1.33918653     | -4.20   | <.0001|
| FEMALE 1           | 0.00000000 B | .              | .       | .     |
| SUBSTANCE alcohol  | -0.28148966 B| 1.41554315     | -0.20   | 0.8425|
| SUBSTANCE cocaine  | -5.60613722 B| 1.46221461     | -3.83   | 0.0001|
| SUBSTANCE heroin   | 0.00000000 B | .              | .       | .     |

The model was already fit in R to allow assessment of the interaction.

> aov2

Call:
  aov(formula = cesd ~ sub + genf, data = ds)

Terms:
  sub genf Residuals
Sum of Squares  2704 2569  65515
Deg. of Freedom  2   1  449

Residual standard error: 12.1
Estimated effects may be unbalanced

If results with the same referent categories used by SAS are desired, the default R design matrix (see 6.1.4) can be changed and the model re-fit.

> contrasts(ds$sub) = contr.SAS(3)
> aov3 = lm(cesd ~ sub + genf, data=ds)
> summary(aov3)

Call:
  lm(formula = cesd ~ sub + genf, data = ds)

Residuals:
  Min  1Q Median  3Q Max
  -32.13 -8.85  1.09  8.48  27.09

Coefficients:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|---------|
| (Intercept) | 33.52 | 1.38 | 24.22 | < 2e-16 |
| sub1 | 5.61 | 1.46 | 3.83 | 0.00014 |
| sub2 | 5.32 | 1.34 | 3.98 | 8.1e-05 |
| genfM | -5.62 | 1.34 | -4.20 | 3.3e-05 |

Residual standard error: 12.1 on 449 degrees of freedom
Multiple R-squared:  0.0745,  Adjusted R-squared:  0.0683
F-statistic: 12 on 3 and 449 DF,  p-value: 1.35e-07

The AIC criteria (7.8.3) can also be used to compare models. In SAS it is available in proc reg and proc mixed. Here we use proc mixed, omitting other output.
ods select fitstatistics;
proc mixed data=k.help method=ml;
   class female substance;
   model cesd = female|substance;
run; quit;

The Mixed Procedure

Fit Statistics

-2 Log Likelihood 3537.8
AIC (smaller is better) 3551.8
AICC (smaller is better) 3552.1
BIC (smaller is better) 3580.6

ods select fitstatistics;
proc mixed data=k.help method=ml;
   class female substance;
   model cesd = female substance;
run; quit;
ods select all;

The Mixed Procedure

Fit Statistics

-2 Log Likelihood 3538.8
AIC (smaller is better) 3548.8
AICC (smaller is better) 3549.0
BIC (smaller is better) 3569.4

> AIC(aov1)
[1] 3552

> AIC(aov2)

[1] 3549

The AIC criterion also suggests that the model without the interaction is most appropriate.

6.6.6 Multiple comparisons

We can also carry out multiple comparison (6.2.4) procedures to test each of the pairwise differences between substance abuse groups. In SAS this utilizes the \texttt{lsmeans} statement within proc glm.

ods select diff lsmeandiffcl lsmlines diffplot;
proc glm data=k.help;
   class substance;
   model cesd = substance;
   lsmeans substance / pdiff adjust=tukey cl lines;
run; quit;
ods select all;
6.6. EXAMPLES

The GLM Procedure
Least Squares Means
Adjustment for Multiple Comparisons: Tukey-Kramer

Least Squares Means for effect SUBSTANCE
Pr > |t| for H0: LSMean(i)=LSMean(j)

Dependent Variable: CESD

<table>
<thead>
<tr>
<th>i/j</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0009</td>
<td>0.9362</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0009</td>
<td>0.0008</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.9362</td>
<td>0.0008</td>
<td></td>
</tr>
</tbody>
</table>

The GLM Procedure
Least Squares Means
Adjustment for Multiple Comparisons: Tukey-Kramer

Least Squares Means for Effect SUBSTANCE

| Difference Means Simultaneous 95% Between Confidence Limits for LSMean(i)-LSMean(j) |
|---------------------------------|---------------------------------|
| i j                             |                                 |
| 1 2                            | 4.951829 1.753296 8.150362      |
| 1 3                            | -0.498086 -3.885335 2.889162    |
| 2 3                            | -5.449915 -8.950037 -1.949793   |

The GLM Procedure
Least Squares Means
Adjustment for Multiple Comparisons: Tukey-Kramer

Tukey-Kramer Comparison Lines for Least Squares Means of SUBSTANCE

LS-means with the same letter are not significantly different.

<table>
<thead>
<tr>
<th>CESD</th>
<th>SUBSTANCE</th>
<th>LSMEAN</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>heroin</td>
<td>34.87097</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>alcohol</td>
<td>34.37288</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>cocaine</td>
<td>29.42105</td>
<td>2</td>
</tr>
</tbody>
</table>

The above output demonstrates the results of the lines option using the lsmeans statement. The letter A shown on the left connecting the heroin and alcohol substances implies that there is not a statistically significant difference between these two groups. Since the cocaine substance has the letter B and no other group has one, the cocaine group is significantly different from each of the other groups. If instead the cocaine and alcohol substances both had a letter B attached, while the heroin and alcohol substances retained the letter A they have in the actual output, only the heroin and cocaine groups would differ significantly, while the alcohol group would differ from neither. This presentation becomes
CHAPTER 6. LINEAR REGRESSION AND ANOVA

CESD Comparisons for SUBSTANCE

Significant
Not significant
Differences for alpha=0.05 (Tukey-Kramer Adjustment)

alcohol
cocaine
eroin

28 30 32 34 36

28 30 32 34 36

(a) SAS

Figure 6.7: Pairwise comparisons

particularly useful as the number of groups increases. A graphical version called a diffplot is also produced; it is shown in Figure 6.7.

In R, we use the TukeyHSD() function.

> mult = TukeyHSD(aov(cesd ~ sub, data=ds), "sub")
> mult

Tukey multiple comparisons of means
95% family-wise confidence level

Fit: aov(formula = cesd ~ sub, data = ds)

$sub
  diff  lwr upr p adj
alcohol-heroin -0.498 -3.89 2.89 0.936
coke-cocaine -5.450 -8.95 -1.95 0.001
cocaine-alcohol -4.952 -8.15 -1.75 0.001

The alcohol group and heroin group both have significantly higher CESD scores than the cocaine group, but the alcohol and heroin groups do not significantly differ from each other (95% CI for the difference ranges from −3.9 to 2.9). Figure 6.7 provides a graphical display of the pairwise comparisons.

> plot(mult)

6.6.7 Contrasts

We can also fit contrasts (6.2.3) to test hypotheses involving multiple parameters. In this case, we can compare the CESD scores for the alcohol and heroin groups to the cocaine group. In SAS, to allow checking the contrast, we use the e option to the estimate statement.
6.6. EXAMPLES

ods select contrastcoef estimates;
proc glm data=k.help;
   class female substance;
   model cesd = female substance;
   output out=outanova residual=resid_ch4anova;
   estimate 'A+H = C?' substance 1 -2 1 / e;
run; quit;
ods select all;

The GLM Procedure

Coefficients for Estimate A+H = C?

Row 1

Intercept 0
FEMALE 0 0
FEMALE 1 0
SUBSTANCE alcohol 1
SUBSTANCE cocaine -2
SUBSTANCE heroin 1

The GLM Procedure

Dependent Variable: CESD

| Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|-----------|----------|----------------|---------|------|---|
| A+H = C?  | 10.9307848 | 2.42008987 | 4.52 | <.0001 |

> library(gmodels)
> levels(ds$sub)

[1] "heroin" "alcohol" "cocaine"

> fit.contrast(aov2, "sub", c(1,1,-2), conf.int=0.95 )

| Estimate | Std. Error | t value | Pr(>|t|) | lower CI | upper CI |
|----------|------------|---------|---------|----------|----------|
| sub c=( 1 1 -2 ) | 10.9 | 2.42 | 4.52 | 8.04e-06 | 6.17 | 15.7 |

As expected from the interaction plot (Figure 6.5), there is a statistically significant difference in this 1 degree of freedom comparison ($p < 0.0001$).