

Lecture 13: Multiple linear regression

Pratheepa Jeganathan

10/21/2019

Recap

- ▶ What is a regression model?
- ▶ Descriptive statistics – graphical
- ▶ Descriptive statistics – numerical
- ▶ Inference about a population mean
- ▶ Difference between two population means
- ▶ Some tips on R

Recap

- ▶ Simple linear regression (covariance, correlation, estimation, geometry of least squares)
 - ▶ Inference on simple linear regression model
 - ▶ Goodness of fit of regression: analysis of variance.
 - ▶ F -statistics.
 - ▶ Residuals.
 - ▶ Diagnostic plots for simple linear regression (graphical methods).

Recap

- ▶ Multiple linear regression
 - ▶ Specifying the model.
 - ▶ Fitting the model: least squares.
 - ▶ Interpretation of the coefficients.

Outline

- ▶ Inference for multiple regression
 - ▶ T -statistics revisited.
 - ▶ More F statistics.
 - ▶ Tests involving more than one β .

Inference for multiple regression

Regression function at one point

- ▶ One thing one might want to *learn* about the regression function in the prostate example is something about the regression function at some fixed values of X_1, \dots, X_7 , i.e. what can be said about the **mean response**

$$\begin{aligned} &\beta_0 + 1.3 \cdot \beta_1 + 3.6 \cdot \beta_2 + 64 \cdot \beta_3 + \\ &0.1 \cdot \beta_4 + 0.2 \cdot \beta_5 - 0.2 \cdot \beta_6 + 25 \cdot \beta_7 \end{aligned}$$

roughly the regression function at “typical” values of the predictors.

- ▶ The expression above is equivalent to

$$\sum_{j=0}^7 a_j \beta_j = \mathbf{a}^T \boldsymbol{\beta}, \quad \mathbf{a} = (1, 1.3, 3.6, 64, 0.1, 0.2, -0.2, 25).$$

Confidence interval for $\sum_{j=0}^p a_j \beta_j$

- ▶ Suppose we want a $(1 - \alpha) \cdot 100\%$ CI for $\sum_{j=0}^p a_j \beta_j$.
- ▶ Just as in simple linear regression:

$$\sum_{j=0}^p a_j \hat{\beta}_j \pm t_{1-\alpha/2, n-p-1} \cdot SE \left(\sum_{j=0}^p a_j \hat{\beta}_j \right).$$

Standard error of $\sum_{j=0}^p a_j \hat{\beta}_j$

- ▶ In order to form these confidence interval, we need the *SE* of our estimate $\sum_{j=0}^p a_j \hat{\beta}_j$.
- ▶ Based on matrix approach to regression

$$\begin{aligned} \text{SE} \left(\sum_{j=0}^p a_j \hat{\beta}_j \right) &= \text{SE} \left(\mathbf{a}^T \hat{\beta} \right) = \sqrt{\text{Cov} \left(\mathbf{a}^T \hat{\beta} \right)} = \sqrt{\mathbf{a}^T \text{Cov} \left(\hat{\beta} \right) \mathbf{a}} \\ &= \sqrt{\hat{\sigma}^2 \mathbf{a}^T (X^T X)^{-1} \mathbf{a}} \end{aligned}$$

. - Don't worry too much about specific implementation – for much of the effects we want R will do this for you in general.

Example

```
library(xtable)
library(ElemStatLearn)
data(prostate)
prostate.lm = lm(lpsa ~ lcavol + lweight +
  age + lbph + svi + lcp + pgg45,
  data = prostate)
n = nrow(prostate)
Y = prostate$lpsa
X = model.matrix(prostate.lm)
beta_hat = as.numeric(solve(t(X) %*% X)
  %*% t(X) %*% Y)
names(beta_hat) = colnames(X)
```

```
Y.hat = X %*% beta_hat  
sigma.hat = sqrt(sum((Y - Y.hat)^2)  
  / (n - ncol(X)))  
cov.beta_hat = sigma.hat^2 * solve(t(X) %*% X)
```

↑ Cov($\hat{\beta}$)

```
print(xtable(data.frame(cov.beta_hat), digits = 4),
      scalebox='0.6')
```

% latex table generated in R 3.6.0 by xtable 1.8-4 package % Mon
Oct 21 01:17:35 2019

	X.Intercept.	lcavol	lweight	age	lbph	svi	lcp	pgg45
(Intercept)	0.7631	0.0100	-0.1127	-0.0058	0.0248	0.0103	0.0021	0.0000
lcavol	0.0100	0.0074	-0.0033	-0.0001	0.0004	-0.0026	-0.0035	0.0000
lweight	-0.1127	-0.0033	0.0394	-0.0004	-0.0045	-0.0041	0.0001	0.0001
age	-0.0058	-0.0001	-0.0004	0.0001	-0.0001	-0.0001	0.0001	-0.0000
lbph	0.0248	0.0004	-0.0045	-0.0001	0.0033	0.0021	-0.0001	-0.0000
svi	0.0103	-0.0026	-0.0041	-0.0001	0.0021	0.0567	-0.0089	-0.0001
lcp	0.0021	-0.0035	0.0001	0.0001	-0.0001	-0.0089	0.0080	-0.0001
pgg45	0.0000	0.0000	0.0001	-0.0000	-0.0000	-0.0001	-0.0001	0.0000

- The standard error of regression function estimate at

$$\mathbf{a} = (1, 1.3, 3.6, 64, 0.1, 0.2, -0.2, 25) \text{ is } \sqrt{\mathbf{a}^T \text{Cov}(\hat{\beta}) \mathbf{a}}$$

```
a = c(1,1.3,3.6,64,0.1,0.2,-0.2,25)
sqrt(t(a)%*%cov.beta_hat**a)
```

```
##           [,1]
## [1,] 0.07101959
```

- The standard errors of each coefficient estimate are the square root of the diagonal entries.

```
round(sqrt(diag(cov.beta_hat)), digits = 4)
```

## (Intercept)	lcavol	lweight	age	l
## 0.8736	0.0858	0.1984	0.0110	0.0
## lcp	pgg45			
## 0.0893	0.0034			

- Generally, we can find our estimate of the covariance function $\text{Cov}(\hat{\beta})$ as follows:

```
print(xtable(vcov(prostate.lm), digits = 4),  
      scalebox='0.6')
```

% latex table generated in R 3.6.0 by xtable 1.8-4 package % Mon
Oct 21 11:03:21 2019

	(Intercept)	lcavol	lweight	age	lbph	svi	lcp	pgg45
(Intercept)	0.7631	0.0100	-0.1127	-0.0058	0.0248	0.0103	0.0021	0.0000
lcavol	0.0100	0.0074	-0.0033	-0.0001	0.0004	-0.0026	-0.0035	0.0000
lweight	-0.1127	-0.0033	0.0394	-0.0004	-0.0045	-0.0041	0.0001	0.0001
age	-0.0058	-0.0001	-0.0004	0.0001	-0.0001	-0.0001	0.0001	-0.0000
lbph	0.0248	0.0004	-0.0045	-0.0001	0.0033	0.0021	-0.0001	-0.0000
svi	0.0103	-0.0026	-0.0041	-0.0001	0.0021	0.0567	-0.0089	-0.0001
lcp	0.0021	-0.0035	0.0001	0.0001	-0.0001	-0.0089	0.0080	-0.0001
pgg45	0.0000	0.0000	0.0001	-0.0000	-0.0000	-0.0001	-0.0001	0.0000

Example (confidence interval for regression at a given point)

```
library(ElemStatLearn)
data(prostate)
prostate.lm = lm(lpsa ~ lcavol + lweight +
  age + lbph + svi + lcp + pgg45,
  data = prostate)
```

Example (confidence interval for regression at a given point)

- ▶ R will form these coefficients (\mathbf{a}) for each regression coefficient separately when using the `confint` function.
- ▶ If we have an observation, $X_1 = 1.3, X_2 = 3.6, X_3 = 64, X_4 = 0.1, X_5 = 0.2, X_6 = -0.2, X_7 = 25$.
- ▶ We can write $\mathbf{a} = (1.3, 3.6, 64, 0.1, 0.2, -0.2, 25)$.
- ▶

```
predict(prostate.lm, list(lcavol = 1.3, lweight = 3.6,
  age = 64, lbph = 0.1,
  svi = 0.2, lcp = -.2, pgg45 = 25),
  interval='confidence',
  level=0.90)
```

```
##          fit          lwr          upr
## 1 2.422332 2.304287 2.540378
```


Confidence interval for individual regression coefficients

- If we want a confidence interval for β_1 . We can write \mathbf{a} as follows

$$\mathbf{a}_{\text{lcvol}} = (0, 1, 0, 0, 0, 0, 0, 0)^T$$

so that

$$\mathbf{a}_{\text{lcvol}}^T \boldsymbol{\beta} = \beta_1$$

and

$$\mathbf{a}_{\text{lcvol}}^T \hat{\boldsymbol{\beta}} = \hat{\beta}_1 = \text{coef}(\text{prostate.lm})[2]$$

Confidence interval for regression coefficient

- ▶ Suppose we want a $(1 - \alpha) \cdot 100\%$ CI for β_1 .
- ▶ Just as in simple linear regression:

$$\mathbf{a}_{1\text{cavol}}^T \hat{\boldsymbol{\beta}} \pm t_{1-\alpha/2, n-p-1} \cdot SE \left(\mathbf{a}_{1\text{cavol}}^T \hat{\boldsymbol{\beta}} \right)$$
$$\hat{\beta}_1 \pm t_{1-\alpha/2, n-p-1} \cdot SE \left(\hat{\beta}_1 \right).$$

Example (confidence interval for regression coefficient)

- Confidence interval for each $\beta_j, j = 0, 1, \dots, p$.

```
confint(prostate.lm, level=0.90)
```

##	5 %	95 %
## (Intercept)	-0.9578488958	1.946158404
## lcavol	0.4268548240	0.712237239
## lweight	0.2845659251	0.944273708
## age	-0.0391601782	-0.002666755
## lbph	0.0016386253	0.193066445
## svi	0.3565053323	1.148289353
## lcp	-0.2534678904	0.043549074
## pgg45	-0.0003011464	0.010950077



T-statistics revisited

- ▶ Of course, these confidence intervals are based on the standard ingredients of a T -statistic.
- ▶ Suppose we want to test

$$H_0 : \sum_{j=0}^p a_j \beta_j = h.$$

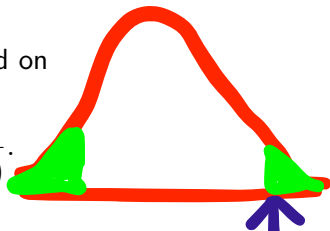
- As in simple linear regression, it is based on

$$T = \frac{\sum_{j=0}^p a_j \hat{\beta}_j - h}{SE(\sum_{j=0}^p a_j \hat{\beta}_j)}.$$

- ▶ If H_0 is true, then $T \sim t_{n-p-1}$, so we reject H_0 at level α if

$$|T| \geq t_{1-\alpha/2, n-p-1}, \quad \text{OR}$$

$$\text{p-value} = 2 * (1 - \text{pt}(|T|, n - p - 1)) \leq \alpha.$$



Example (T-statistic)

- R produces these in the coef table summary of the linear regression model. Again, each of these linear combinations is a vector \mathbf{a} with only one non-zero entry like $\mathbf{a}_{1\text{cavol}}$ above.

```
print(xtable(summary(prostate.lm)$coef,  
  digits = 3), scalebox='0.6')
```

% latex table generated in R 3.6.0 by xtable 1.8-4 package % Mon
Oct 21 11:03:21 2019

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.494	0.874	0.566	0.573
lcavol	0.570	0.086	6.634	0.000
lweight	0.614	0.198	3.096	0.003
age	-0.021	0.011	-1.905	0.060
lbph	0.097	0.058	1.691	0.094
svi	0.752	0.238	3.159	0.002
lcp	-0.105	0.089	-1.175	0.243
pgg45	0.005	0.003	1.573	0.119

Example (T-statistic)

- ▶ Let's do a quick calculation to remind ourselves the relationships of the columns in the table above.

```
T1 = 0.570 / 0.086  
P1 = 2 * (1 - pt(abs(T1), 89))  
print(round(c(T1, P1), digits = 3))
```

```
## [1] 6.628 0.000
```

- ▶ These were indeed the values for `lcavol` in the `summary` table.

One-sided tests

- ▶ Suppose, instead, we wanted to test the one-sided hypothesis

$$H_0 : \sum_{j=0}^p a_j \beta_j \leq h, \text{ vs. } H_a : \sum_{j=0}^p a_j \beta_j > h$$

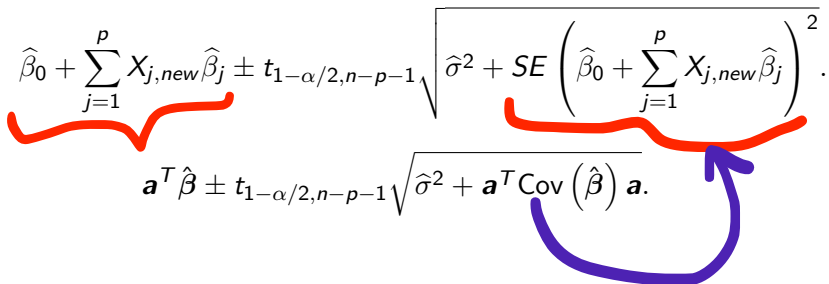
- ▶ We reject H_0 at level α if

$$T \geq t_{1-\alpha, n-p-1}, \quad \text{OR} \\ p - \text{value} = (1 - \text{pt}(T, n - p - 1)) \leq \alpha.$$

- ▶ **Note:** the decision to do a one-sided T test should be made *before* looking at the T statistic. Otherwise, the probability of a type I error is doubled!

Prediction interval

- ▶ Basically identical to simple linear regression.
- ▶ Prediction interval at $X_{1,new}, \dots, X_{p,new}$:

$$\underbrace{\hat{\beta}_0 + \sum_{j=1}^p X_{j,new} \hat{\beta}_j}_{\mathbf{a}^T \hat{\beta}} \pm t_{1-\alpha/2, n-p-1} \sqrt{\hat{\sigma}^2 + \underbrace{SE \left(\hat{\beta}_0 + \sum_{j=1}^p X_{j,new} \hat{\beta}_j \right)^2}_{\mathbf{a}^T \text{Cov}(\hat{\beta}) \mathbf{a}}}$$


Forming intervals by hand

- ▶ While R computes most of the intervals we need, we could write a function that explicitly computes a confidence interval (and can be used for prediction intervals with the “extra” argument).
- ▶ This exercise shows the calculations that R is doing under the hood: the function *predict* is generally going to be fine for our purposes.

```

interaval.lm = function(cur.lm, a, level=0.95, extra=0) {
  # the center of the confidence interval
  center = sum(a*cur.lm$coef)
  # the estimate of sigma^2
  sigma.hat.sq = sum(resid(cur.lm)^2) /
    cur.lm$df.resid
  # the standard error of sum(a*cur.lm$coef)
  se = sqrt(extra * sigma.hat.sq +
    sum((a %>% vcov(cur.lm)) * a))
  # the degrees of freedom for the t-statistic
  df = cur.lm$df
  # the quantile used in the confidence interval
  q = qt((1 - level)/2, df,
    lower.tail=FALSE)
  # upper, lower limits
  upper = center + se * q
  lower = center - se * q
  return(data.frame(center,
    lower, upper))
}

```

Example (prediction intervals)

- By using the `extra = 1` argument, we can make prediction intervals.

```
print(intervall.lm(prostate.lm, c(1, 1.3, 3.6,  
  64, 0.1, 0.2, -0.2, 25),  
  extra=1))
```

```
##      center    lower    upper  
## 1 2.422332 1.032301 3.812363
```

```
predict(prostate.lm,list(lcavol=1.3,  
  lweight=3.6,age=64,lbph=0.1,  
  svi=0.2,lcp=-0.2,pgg45=25),  
  interval='prediction')
```

```
##      fit      lwr      upr  
## 1 2.422332 1.032301 3.812363
```

Example (confidence interval for mean response)

```
print(interaval.lm(prostate.lm,  
  c(1, 1.3, 3.6, 64, 0.1,  
    0.2, -0.2, 25), extra = 0))
```

```
##      center    lower    upper  
## 1 2.422332 2.281218 2.563447
```

```
predict(prostate.lm, list(lcavol=1.3, lweight=3.6,  
  age=64, lbph=0.1, svi=0.2,  
  lcp=-0.2, pgg45=25),  
  interval='confidence')
```

```
##      fit      lwr      upr  
## 1 2.422332 2.281218 2.563447
```

Arbitrary contrasts

- ▶ If we want, we can set the intercept term to 0. This allows us to construct confidence interval for, say, how much the `lpsa` score will change will increase if we change `age` by 2 years and `svi` by 0.5 units, leaving everything else unchanged.
- ▶ Therefore, what we want is a confidence interval for 2 times the coefficient of `age` + 0.5 times the coefficient of `lbph`:

$$2 \cdot \beta_{\text{age}} + 0.5 \cdot \beta_{\text{svi}}$$

- ▶ Most of the time, *predict* will do what you want so this won't be used too often.

```
interval.lm(prostate.lm, c(0,0,0,2,0,0.5,0,0))
```

```
##           center        lower        upper
## 1 0.3343717 0.09496226 0.5737812
```

Questions about many (combinations) of β_j 's

- ▶ In multiple regression we can ask more complicated questions than in simple regression.
- ▶ For instance, we could ask whether lcp and pgg45 explains little of the variability in the data, and might be dropped from the regression model.
- ▶ These questions can be answered by F -statistics.
- ▶ **Note: This hypothesis should really be formed *before* looking at the output of summary.**
- ▶ Later we'll see some examples of the messiness when forming a hypothesis after seeing the summary.

Dropping one or more variables

- ▶ Suppose we wanted to test the above hypothesis.
- ▶ Formally, the null hypothesis is:

$$H_0 : \beta_{1cp}(= \beta_6) = \beta_{pgg45}(= \beta_7) = 0$$

and the alternative is

$$H_a = \text{one of } \beta_{1cp}, \beta_{pgg45} \text{ is not 0.}$$

- ▶ This test is an F -test based on two models

$$\text{Full : } Y_i = \beta_0 + \sum_{j=1}^7 X_{ij}\beta_j + \varepsilon_i$$

$$\text{Reduced : } Y_i = \beta_0 + \sum_{j=1}^5 \beta_j X_{ij} + \varepsilon_i$$

- ▶ **Note: The reduced model R must be a special case of the full model F to use the F -test.**

SSE of a model

- ▶ A “model”, \mathcal{M} is a subspace of \mathbb{R}^n (e.g. column space of X).
- ▶ Least squares fit = projection onto the subspace of \mathcal{M} , yielding predicted values $\hat{Y}_{\mathcal{M}}$
- ▶ Error sum of squares:

$$SSE(\mathcal{M}) = \|Y - \hat{Y}_{\mathcal{M}}\|^2.$$

F -statistic for $H_0 : \beta_{1cp} = \beta_{pgg45} = 0$

- ▶ We compute the F statistic the same to compare any models

$$F = \frac{\frac{SSE(R) - SSE(F)}{2}}{\frac{SSE(F)}{n-1-p}} \sim F_{2, n-p-1} \quad (\text{if } H_0 \text{ is true})$$

Note: In the original image, a blue bracket connects the denominator of the F-statistic formula to the handwritten red text "n-6-(n-8)".

- ▶ Reject H_0 at level α if $F \geq F_{1-\alpha, 2, n-1-p}$.
- ▶ When comparing two models, one a special case of the other (i.e. one nested in the other), we can test if the smaller model (the special case) is roughly as good as the larger model in describing our outcome. This is typically tested using an F test based on comparing the two models. The following function does this.

```
f.test.lm = function(R.lm, F.lm) {  
  SSE.R = sum(resid(R.lm)^2)  
  SSE.F = sum(resid(F.lm)^2)  
  df.num = R.lm$df - F.lm$df  
  df.den = F.lm$df  
  F = ((SSE.R - SSE.F) / df.num) / (SSE.F / df.den)  
  p.value = 1 - pf(F,  
    df.num, df.den)  
  return(data.frame(F,  
    df.num, df.den, p.value))  
}
```

- R has a function that does essentially the same thing as `f.test.lm`: the function is `anova`. It can be used several ways, but it can be used to compare two models.

```
prostate.lm.reduced = lm(lpsa ~ lcavol +  
  lbph + lweight + age + svi,  
  data=prostate)  
print(f.test.lm(prostate.lm.reduced, prostate.lm))
```

```
##           F df.num df.den   p.value  
## 1 1.372057      2      89 0.2588958
```

```
anova(prostate.lm.reduced, prostate.lm)
```

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

```
##
```

```
## Model 1: lpsa ~ lcavol + lbph + lweight + age + svi
```

```
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi + l
```

```
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
```

```
## 1      91 44.437
```

```
## 2      89 43.108  2    1.3291 1.3721 0.2589
```

Dropping an arbitrary subset

- ▶ For an arbitrary model, suppose we want to test

$$H_0 : \beta_{i_1} = \dots = \beta_{i_j} = 0$$

$$H_a : \text{one or more of } \beta_{i_1}, \dots, \beta_{i_j} \neq 0$$

*J ≠ β's
zero*

for some subset $\{i_1, \dots, i_j\} \subset \{0, \dots, p\}$.

- ▶ You guessed it: it is based on the two models:

$$R : Y_i = \sum_{l=0, l \notin \{i_1, \dots, i_j\}}^p \beta_l X_{il} + \varepsilon_i$$

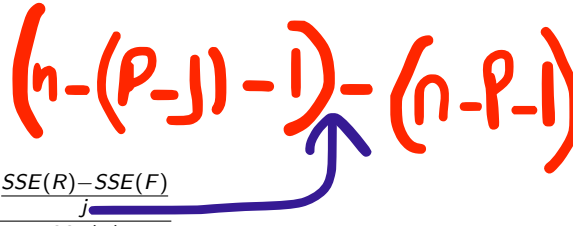
$$F : Y_i = \sum_{l=0}^p \beta_l X_{il} + \varepsilon_i$$

where $X_{i0} = 1$ for all i .

- ▶ **Note:** This hypothesis should really be formed *before* looking at the output of summary. Looking at summary before deciding which to drop is problematic!

Dropping an arbitrary subset

- ▶ Statistic:

$$F = \frac{\frac{SSE(R) - SSE(F)}{j}}{\frac{SSE(F)}{n-p-1}} \sim F_{j, n-p-1} \quad (\text{if } H_0 \text{ is true})$$


The image contains handwritten red text $(n - (p - j) - 1) - (n - p - 1)$ at the top. A blue arrow originates from the denominator of the F-statistic formula, $\frac{SSE(F)}{n-p-1}$, and points upwards towards the red text.

- ▶ Reject H_0 at level α if $F \geq F_{1-\alpha, j, n-1-p}$.

General F -tests

- ▶ Given two models $R \subset F$ (i.e. R is a subspace of F), we can consider testing

$$H_0 : R \text{ is adequate (i.e. } \mathbb{E}(Y) \in R)$$

vs.

$$H_a : F \text{ is adequate (i.e. } \mathbb{E}(Y) \in F)$$

- ▶ The test statistic is

$$F = \frac{(SSE(R) - SSE(F))/(df_R - df_F)}{SSE(F)/df_F}.$$

- ▶ If H_0 is true, $F \sim F_{df_R - df_F, df_F}$ so we reject H_0 at level α if $F \geq F_{1-\alpha, df_R - df_F, df_F}$.

Constraining $\beta_{\text{l cavol}} = \beta_{\text{s vi}}$

- ▶ In this example, we might suppose that the coefficients for `lcavol` and `s vi` are the same and want to test this. We do this, again, by comparing a “full model” and a “reduced model”.
- ▶ Full model:

$$Y_i = \beta_0 + \beta_1 X_{i,\text{lcavol}} + \beta_2 X_{i,\text{lweight}} + \beta_3 X_{i,\text{age}} \\ + \beta_4 X_{i,\text{lbph}} + \beta_5 X_{i,\text{s vi}} + \beta_6 X_{i,\text{lcp}} + \beta_7 X_{i,\text{pgg45}} + \varepsilon_i$$

- ▶ Reduced model:

$$Y_i = \beta_0 + \tilde{\beta}_1 X_{i,\text{lcavol}} + \beta_2 X_{i,\text{lweight}} + \beta_3 X_{i,\text{age}} \\ + \beta_4 X_{i,\text{lbph}} + \tilde{\beta}_1 X_{i,\text{s vi}} + \beta_6 X_{i,\text{lcp}} + \beta_7 X_{i,\text{pgg45}} + \varepsilon_i$$

Example

```
prostate$Z = prostate$lcavol + prostate$svi  
equal.lm = lm(Y ~ Z + lweight + age +  
             lbph + lcp + pgg45, data = prostate)  
f.test.lm(equal.lm, prostate.lm)
```

##		F	df.num	df.den	p.value
##	1	0.4818657	1	89	0.4893864

Constraining $\beta_{\text{l cavol}} + \beta_{\text{s vi}} = 1$

- Full model:

$$Y_i = \beta_0 + \beta_1 X_{i,\text{l cavol}} + \beta_2 X_{i,\text{l weight}} + \beta_3 X_{i,\text{age}} \\ + \beta_4 X_{i,\text{lbph}} + \beta_5 X_{i,\text{s vi}} + \beta_6 X_{i,\text{l cp}} + \beta_7 X_{i,\text{pgg45}} + \varepsilon_i$$

- Reduced model:

$$Y_i = \beta_0 + \tilde{\beta}_1 X_{i,\text{l cavol}} + \beta_2 X_{i,\text{l weight}} + \beta_3 X_{i,\text{age}} \\ + \beta_4 X_{i,\text{lbph}} + (1 - \tilde{\beta}_1) X_{i,\text{s vi}} + \beta_6 X_{i,\text{l cp}} + \beta_7 X_{i,\text{pgg45}} + \varepsilon_i$$

Example

```
prostate$Z2 = prostate$lcavol - prostate$svi
constrained.lm = lm(lpsa ~ Z2 + lweight + age +
  lbph + lcp + pgg45,
  data=prostate, offset=svi)
anova(constrained.lm, prostate.lm)
```

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

```
##
```

```
## Model 1: lpsa ~ Z2 + lweight + age + lbph + lcp + pgg45
```

```
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi + lcp + pgg45
```

```
##   Res.Df    RSS Df Sum of Sq    F Pr(>F)
```

```
## 1      90 43.961
```

```
## 2      89 43.108   1   0.85359 1.7623 0.1877
```

```
f.test.lm(constrained.lm, prostate.lm)
```

```
##           F df.num df.den  p.value
```

```
## 1 1.762322      1      89 0.1877303
```

- ▶ What we had to do above was subtract X_3 from Y on the right hand side of the formula. R has a way to do this called using an *offset*. What this does is it subtracts this vector from Y before fitting.

General linear hypothesis

- ▶ An alternative version of the F test can be derived that does not require refitting a model.
- ▶ Suppose we want to test

$$H_0 : C_{q \times (p+1)} \beta_{(p+1) \times 1} = h$$

versus

$$H_a : C_{q \times (p+1)} \beta_{(p+1) \times 1} \neq h.$$

- ▶ This can be tested via an F test:

$$F = \frac{(C\hat{\beta} - h)^T \left(C(X^T X)^{-1} C^T \right)^{-1} (C\hat{\beta} - h)/q}{SSE(F)/df_F} \stackrel{H_0}{\sim} F_{q, n-p-1}.$$

Note: we are assuming that $\text{rank}(C(X^T X)^{-1} C^T) = q$.

- Here's a function that implements this computation.

```
general.linear = function(model.lm,
  linear_part, null_value=0) {
  # shorthand
  C = linear_part
  h = null_value

  beta.hat = coef(model.lm)
  V = as.numeric(C %*% beta.hat - null_value)
  # the MSE is included in vcov
  invcovV = solve(C %*% vcov(model.lm) %*% t(C))

  df.num = nrow(C)
  df.den = model.lm$df.resid
  F = t(V) %*% invcovV %*% V / df.num
  p.value = 1 - pf(F, df.num, df.den)
  return(data.frame(F, df.num,
    df.den, p.value))
}
```

- Let's test verify this work with our test for testing

$$\beta_{1cp} = \beta_{pgg45} = 0.$$

```
A = matrix(0, nrow = 2, ncol = 8)
A[1,7] = 1
A[2,8] = 1
print(A)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    0    0    0    0    0    1    0
## [2,]    0    0    0    0    0    0    0    1
```

- Rank of **A**

```
qr(A)$rank
```

```
## [1] 2
```

```
general.linear(prostate.lm, A)
```

```
##           F df.num df.den  p.value  
## 1 1.372057      2      89 0.2588958
```

```
f.test.lm(prostate.lm.reduced, prostate.lm)
```

```
##           F df.num df.den  p.value  
## 1 1.372057      2      89 0.2588958
```