STAT 571A — Advanced Statistical Regression Analysis

Chapter 11 NOTES

Model Building – III: Remedial Measures

Heterogeneous Variances

- When diagnostics or other information indicate departure from homogeneity in $\sigma^2\{\varepsilon_i\}$, say, a ‘megaphone’ shape in the resid. plot, we recognize that
  $$\sigma^2\{\varepsilon_i\} = \sigma_i^2,$$
  and remedial action is necessary.
- Previous suggestion: transform Y_i to bring the variation closer to homogeneity.
- This can be effective, but may not always work.

Heterogeneous Variance (cont'd)

- More formally, we update the MLR model:

  $$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1} + \varepsilon_i$$

  where now $\varepsilon_i \sim$ indep. $N(0,\sigma_i^2)$.

- If σ_i^2 is known (not likely) we extend the LS criterion to minimize the Weighted SS:

  $$Q_w = \sum w_i \{Y_i - (\beta_0+\beta_1 X_{i1}+\cdots+\beta_{p-1} X_{i,p-1})\}^2$$

  where we weight each observation inversely to the differential variance: $w_i = 1/\sigma_i^2$.

Weight Inverse to Variance

- This is a standard strategy:
  - If we weight each observation inversely to the differential variance, we give observations with low variance ($\Leftrightarrow$ higher precision) greater weight, and *vice versa*.
- Then minimize Q_w by weighted least squares to find the b_k's.

Matrix Formulation

- In matrix terms:
  - let $\mathbf{W}_{n \times n} = \text{diag}\{w_1,\ldots,w_n\}$
  - the normal equs. become $(\mathbf{X}'\mathbf{W}\mathbf{X})\mathbf{b} = \mathbf{X}'\mathbf{W}\mathbf{Y}$
  - $\Rightarrow$ the WLS sol'n is $\mathbf{b} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{Y}$
  - the covariance matrix is $\boldsymbol{\sigma}^2\{\mathbf{b}\} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$
- Similar to the Gauss-Markov Thm. from Ch. 1, we can show that $E\{\mathbf{b}\} = \boldsymbol{\beta}$, with $\boldsymbol{\sigma}^2\{\mathbf{b}\}$ a min. among all unbiased estimators.

Unknown Heterogeneous Variances: WLS with Replication

If there is replication in the design, or even 'near' replication, we can use it to construct direct estimates of σ_j^2.

As in §3.7, assume the SLR model with $Y_{ij} = \mu_j + \varepsilon_{ij}$, where $i = 1,..,n_j$ and $j = 1,...,c$.

At each j, compute $s_j^2 = \sum(Y_{ij} - \overline{Y}_j)^2/(n_j-1)$ as an (unbiased!) estimator of σ_j^2.

Then, simply use $w_j = 1/s_j^2$ as the weights in the WLS fit.

(Extend this to MLR in an obvious fashion.)

Replication via 'Lots'

- If the study is observational and replication cannot be designed into it, it may still be possible to group the X's into nearly-homogeneous lots.
- If so, find w_j= 1/{sample var. of *j*th lot}.
- Can iterate the process if the WLS estimates of b_k vary greatly at first. (Use the OLS estimates as initial estimates.)

Unknown Variances (cont'd)

- In the more common case where the σ_i^2 terms are unknown, a number of strategies exist for estimating them.
- Recognize: if the X's are correctly modeled in the MLR, then $E\{e_i^2\} = \sigma_i^2$
  - so use e_i^2 as an estimate of σ_i^2,
  - and/or $|e_i|$ as an estimate of σ_i.
    - $\rightarrow$ (The latter is more stable if there are outliers.)

Estimating Variances

- Suppose we find that the e_i's vary in a distinguishable pattern; say, e_i varies more as the fitted values $\uparrow$.
- Depending on the observed pattern, we could perform an intermediate regression of e_i^2 or $|e_i|$ on a component of the model to recover "fitted" values that estimate σ_i^2 or σ_i, resp. Then use these in $w_i = 1/\sigma_i^2$.
- Some possibilities follow $\rightarrow$

Proportional Weighting

- In the simplest case, it may be clear that σ_i^2 changes in some fashion with X_i.
- That is, suppose from a resid. plot we see $|e_i| \propto X_i$. Then, view this as $\sigma_i^2 \propto X_i^2$ and set $w_i = 1/X_i^2$.
- Or, if $e_i^2 \propto X_i$, view this as $\sigma_i^2 \propto X_i$ and set $w_i = 1/X_i$.
- Indeed, if $e_i^2 \propto f(X_i)$ for known $f(\cdot)$, use $w_i = 1/f(X_i)$, etc.

Estimating Variances (cont'd)

Estimating variances (see p. 425)

- If e_i vs. X_{ik} exhibits a 'megaphone' shape, regress $|e_i| = \gamma_0 + \gamma_1 X_{ik}$ and take $s_i = g_0 + g_1 X_{ik}$ in $w_i = 1/s_i^2$.
- If e_i vs. $\hat{Y}_i$ exhibits a 'megaphone' shape, regress $|e_i| = \gamma_0 + \gamma_1 \hat{Y}_i$ and take $s_i = g_0 + g_1 \hat{Y}_i$ in $w_i = 1/s_i^2$.
- If e_i^2 vs. X_{ik} exhibits an increasing trend, regress $e_i^2 = \gamma_0 + \gamma_1 X_{ik}$ and take $s_i^2 = g_0 + g_1 X_{ik}$ in $w_i = 1/s_i^2$.
- (You get the idea...)

Approximate Inferences

Of course, since the w_i's are estimated from the data, the WLS estimates of b_k are only approximate. Bias should be minimal, so

$$E\{b_k\} \approx \beta_k,$$

but $b_k \pm t(1 - \frac{\alpha}{2}; n-p)s_w\{b_k\}$ will only serve as a good approximation for the conf. int. if n is sufficiently large.

<u>Example</u>: Blood Pressure data (CH11TA01)

- Y = (Diastolic) blood pressure
  X = Age
- SLR analysis in R:

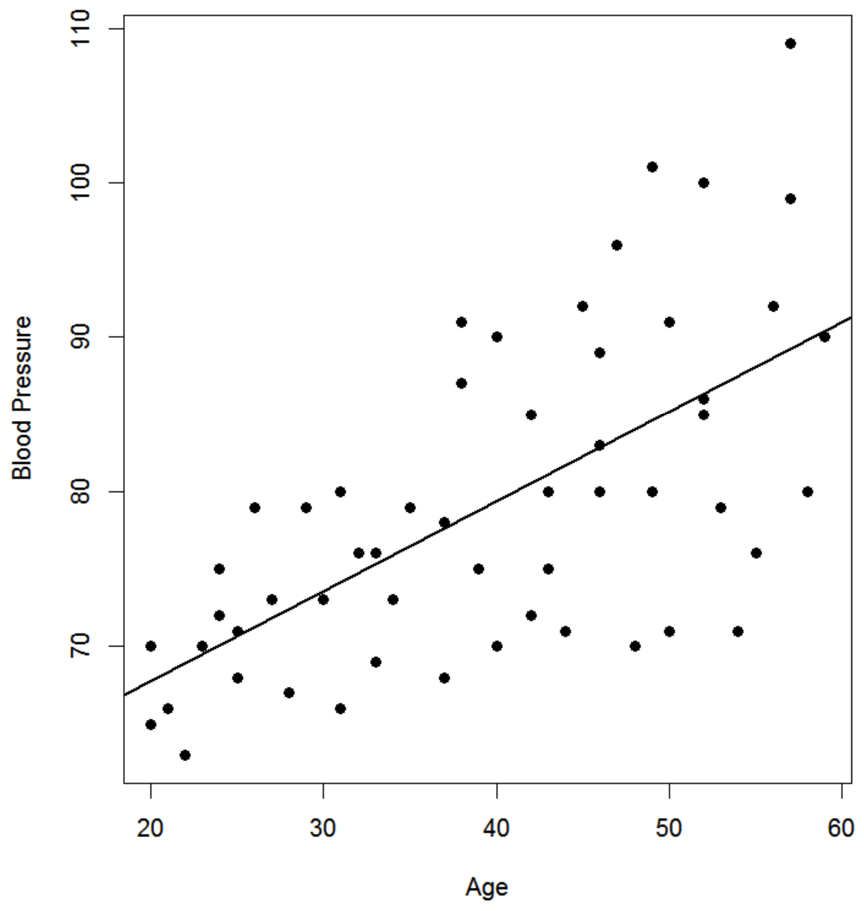
```
> plot( Y ~ X ); abline( lm(Y~X), lwd=2 )
> CH11TA01.lm = lm(Y~X)
> ei = resid( CH11TA01.lm )
> plot(  ei ~ X )
> abline( h=0, lwd=2 )
```

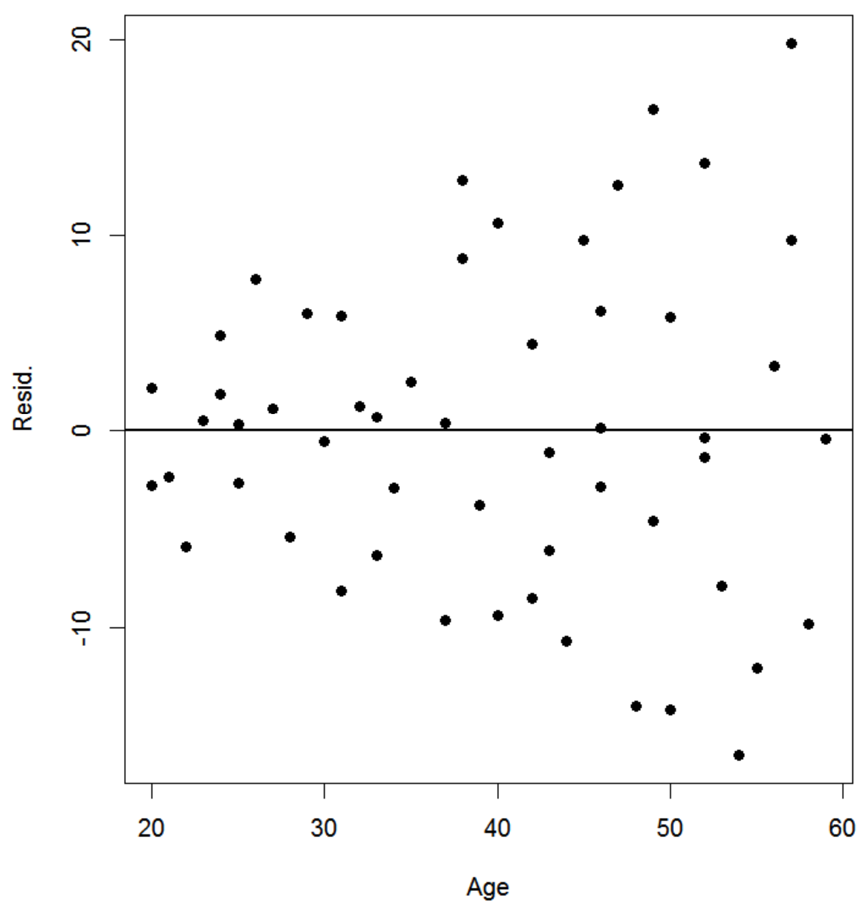
- Plots show increasing trend with X=Age, but also clear ‘megaphone’ spread in residuals
  $\Rightarrow$ variance heterogeneity!

Plots follow $\rightarrow$

Blood Press. data (CH11TA01) (cont'd)

Scatterplot and residual plot (cf. Fig. 11.1):





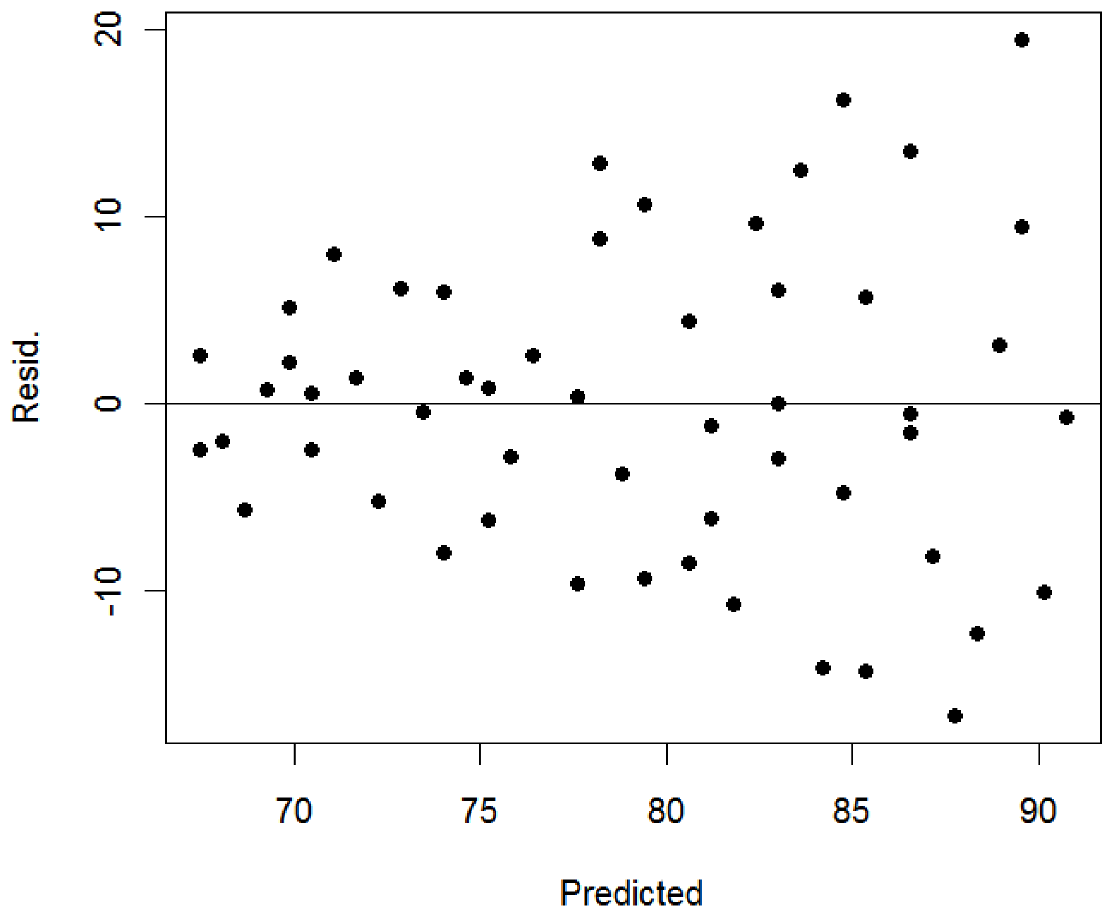
Blood Press. data (CH11TA01) (cont'd)

- Observe 'megaphone' residual spread vs. X
  - $\Rightarrow$ fit SLR of $|e_i| = \gamma_0 + \gamma_1 X_i$ and recover fitted values $s_i = g_0 + g_1 X_i$.
- Apply WLS with weights $w_i = 1/(g_0 + g_1 X_i)^2$.
- The **WLS** analysis in R is simply

```
> si = fitted( lm(abs(ei) ~ X) ); wi = 1/(si^2)
> summary( lm(Y ~ X, weights=wi) )
Call:
lm(formula = Y ~ X, weights = wi)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.56577    2.52092  22.042  < 2e-16
X            0.59634    0.07924   7.526 7.19e-10
```

Blood Press. data (CH11TA01) (cont'd)

Resid. plot from WLS fit doesn't change much, since heterogen-eous variance is still present. But, WLS estimates now adjust for unequal variance.



§11.2: Ridge Regression

- A novel remediation strategy for addressing multicollinearity is known as **Ridge Regression**.
- Recall (Appx. A) that the Mean Squared Error (MSE) of an estimator is

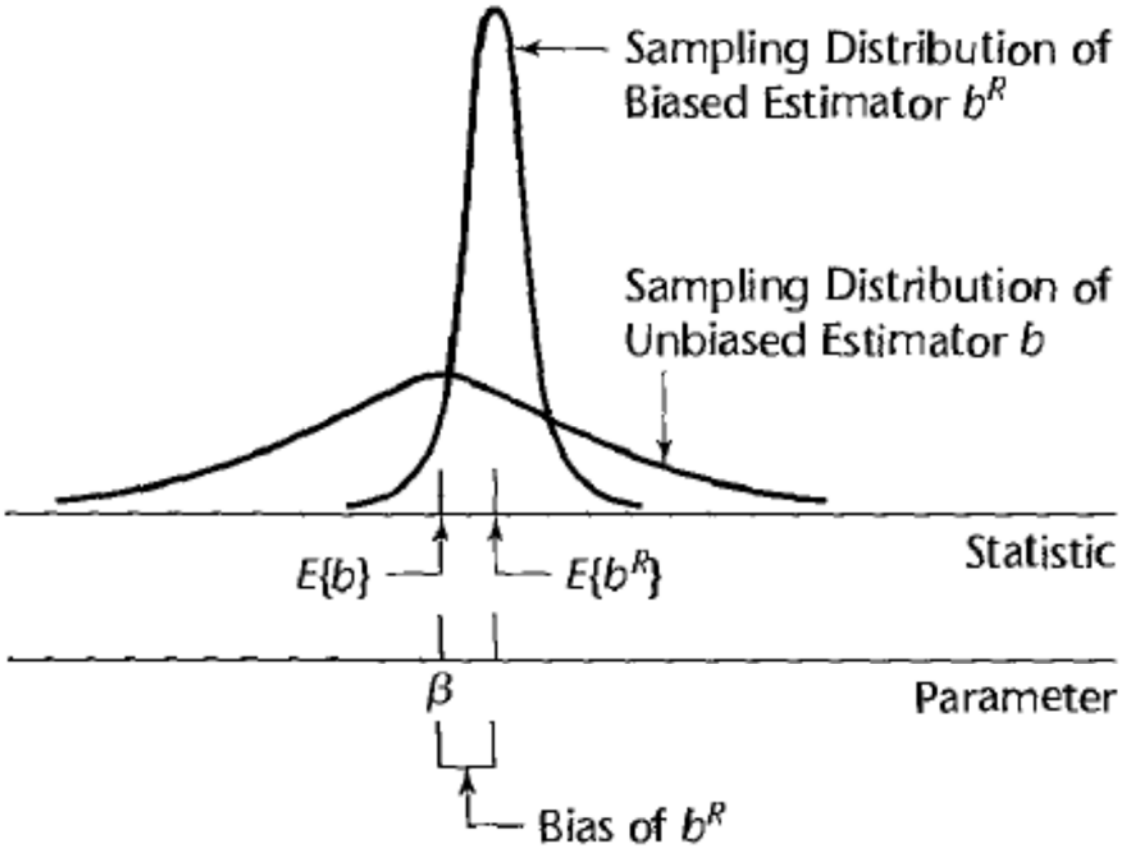
  $$\text{MSE} = \text{Var.} + \text{Bias}^2$$

  So, if we sacrifice a small amt. of bias into the LS estimator we may lessen its variance and overall reduce its MSE.

Bias vs. Variance

- Fig. 11.2 illustrates the effect:

FIGURE 11.2 Biased Estimator with Small Variance May Be Preferable to Unbiased Estimator with Large Variance.


Sampling Distribution of Biased Estimator b^R
Sampling Distribution of Unbiased Estimator b
Statistic
$E\{b\}$
$E\{b^R\}$
Parameter
β
Bias of b^R

Ridge Equations

Hoerl & Kennard (1970) showed that in the presence of multicollinearity, expanding the normal equations into $(\mathrm{X}'\mathrm{X} + c\mathrm{I})\beta = \mathrm{X}'\mathrm{Y}$ can drastically improve the stability of the resulting estimator.

In practice, we first center the Y_i's via $U_i = Y_i - \overline{Y}$, or in vector form $\mathrm{U} = \mathrm{Y} - \overline{Y}\mathrm{1}$, and we standardize the predictors: $Z_{ik} = (X_{ik} - \overline{X}_k)/s_k$. with corresp. standardized design matrix Z.

Ridge Equations (cont'd)

- The Ridge Equations then become

$$(\mathrm{Z}'\mathrm{Z} + \mathrm{cI})\beta_R = \mathrm{Z}'\mathrm{U}$$

  with solution $\mathrm{b}_R = (\mathrm{Z}'\mathrm{Z} + \mathrm{cI})^{-1}\mathrm{Z}'\mathrm{U}$.
  (The inverse matrix can be shown to *always* exist and to be computationally easier to calculate – 'better conditioned'.)

- But, how to choose the constant c?!?

The Ridge Trace

- An existence theorem stipulates that some ridge constant $c > 0$ <u>always</u> exists with a smaller MSE{b_R} than the OLS estimator.
- Unfortunately, it's just an existence thm. It doesn't tell us what c to choose (!).
- One possibility: over increasing $c > 0$ plot the values of all the regression coeff's b_{kR} and look where they all flatten. Choose that c where this ridge trace plot seems to stabilize.

Example: Body Fat Data (CH07TA01)

- Recall that we saw heavy multicollinearity with the Body Fat Data in Ch. 7. Apply a Ridge Regression.
- Use `ridge()` function from the *genridge* package:

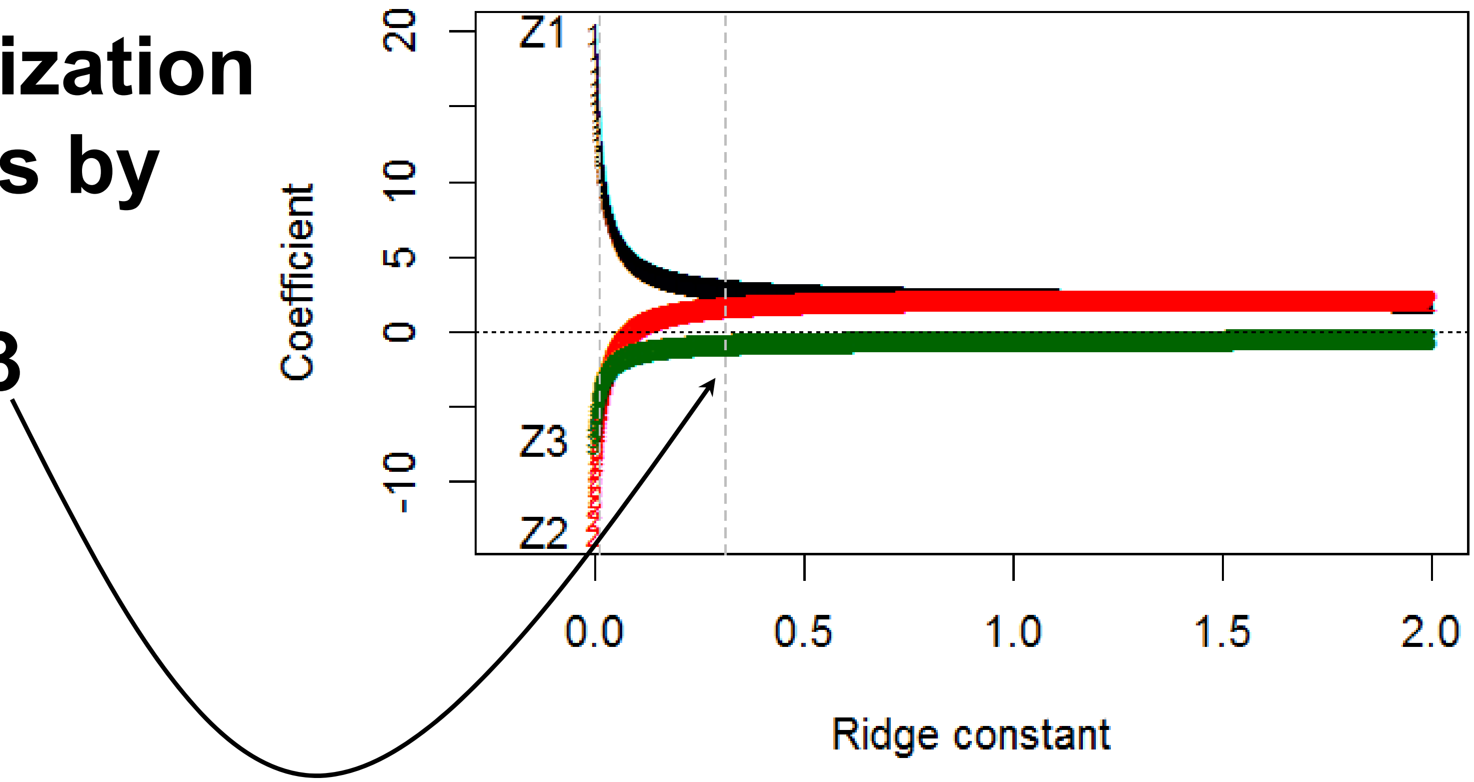
```
> Z1 = scale( X1 ); Z2 = scale( X2 )
> Z3 = scale( X3 ); U = Y - mean(Y)
> require( genridge )
> const = seq(.001,2,.0001) #range for c>0
> fit.ridge = ridge( U ~ Z1 + Z2 + Z3,
                              lambda = const)
> traceplot( fit.ridge, cex=.7 )
```

Plot follows ⟶

Body Fat Data (CH07TA01) (cont'd)

(Stock) ridge trace plot over $0 < c < 2$. Horizontal axis is c; vertical axis is b_{kR}.

Stabilization occurs by about $c \approx 0.3$ or so.



Body Fat Data (CH07TA01) (cont'd)

A warning: the `ridge()` function internally standardizes the predictor variables using a std. deviation with n in the denominator, not $n-1$. But, the `scale()` function uses $n-1$. So the output ridge b_{kR} values will be smaller than we expect by a factor of

$$\sqrt{(n-1)/n}\ .$$

Obviously, this isn't substantial for large n.

Ridge Constant via VIFs

Another approach for selecting c involves study of the VIFs: vary c > $\underline{0}$ until all VIF_k values drop below 10, and $\overline{VIF}$ drops below 6 or so.

⇒ Requires repeated calculation, but can prove valuable. See Table 11.3 ⟶ (c = 0.006 or 0.008 seem to suffice...)

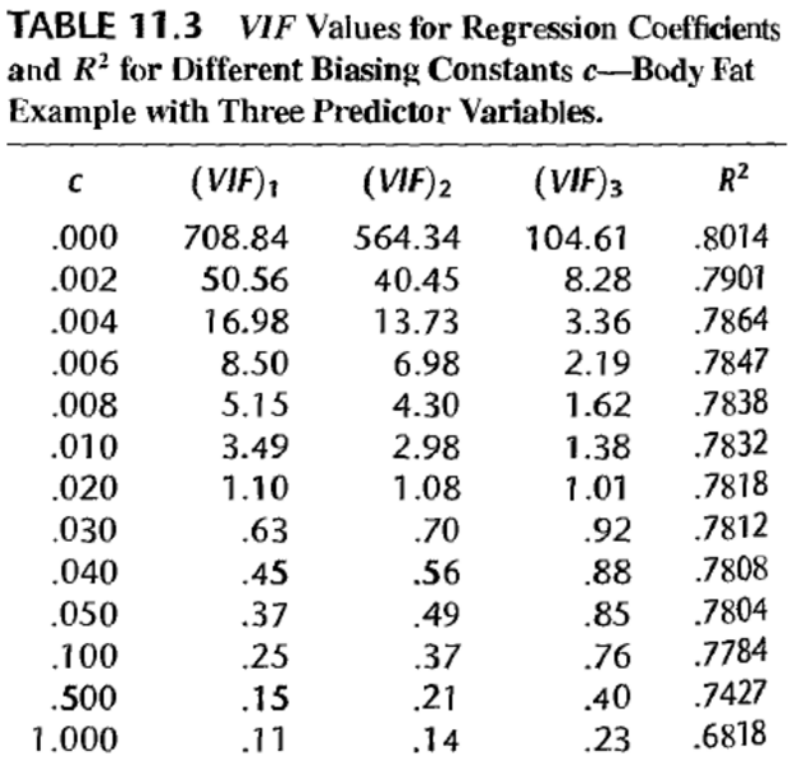


TABLE 11.3 *VIF* Values for Regression Coefficients and R^2 for Different Biasing Constants *c*—Body Fat Example with Three Predictor Variables.

c	$(VIF)_1$	$(VIF)_2$	$(VIF)_3$	R^2
.000	708.84	564.34	104.61	.8014
.002	50.56	40.45	8.28	.7901
.004	16.98	13.73	3.36	.7864
.006	8.50	6.98	2.19	.7847
.008	5.15	4.30	1.62	.7838
.010	3.49	2.98	1.38	.7832
.020	1.10	1.08	1.01	.7818
.030	.63	.70	.92	.7812
.040	.45	.56	.88	.7808
.050	.37	.49	.85	.7804
.100	.25	.37	.76	.7784
.500	.15	.21	.40	.7427
1.000	.11	.14	.23	.6818

Follow-up on Ridge Regr’n

- Ridge regression is a form of *shrinkage* regression, since it literally shrinks the b_{kR} coeff’s towards zero (eventually).
- It is also a form of *regularization*, i.e., <u>penalized</u> <u>regression</u> where large b_{kR} values are penalized. This can help with the instability inherent in multicollinearity.
- A number of estimation strategies are available for finding c, including the Hoerl-Kennard-Baldwin (HKB) and Lawless-Wang (LW) methods.

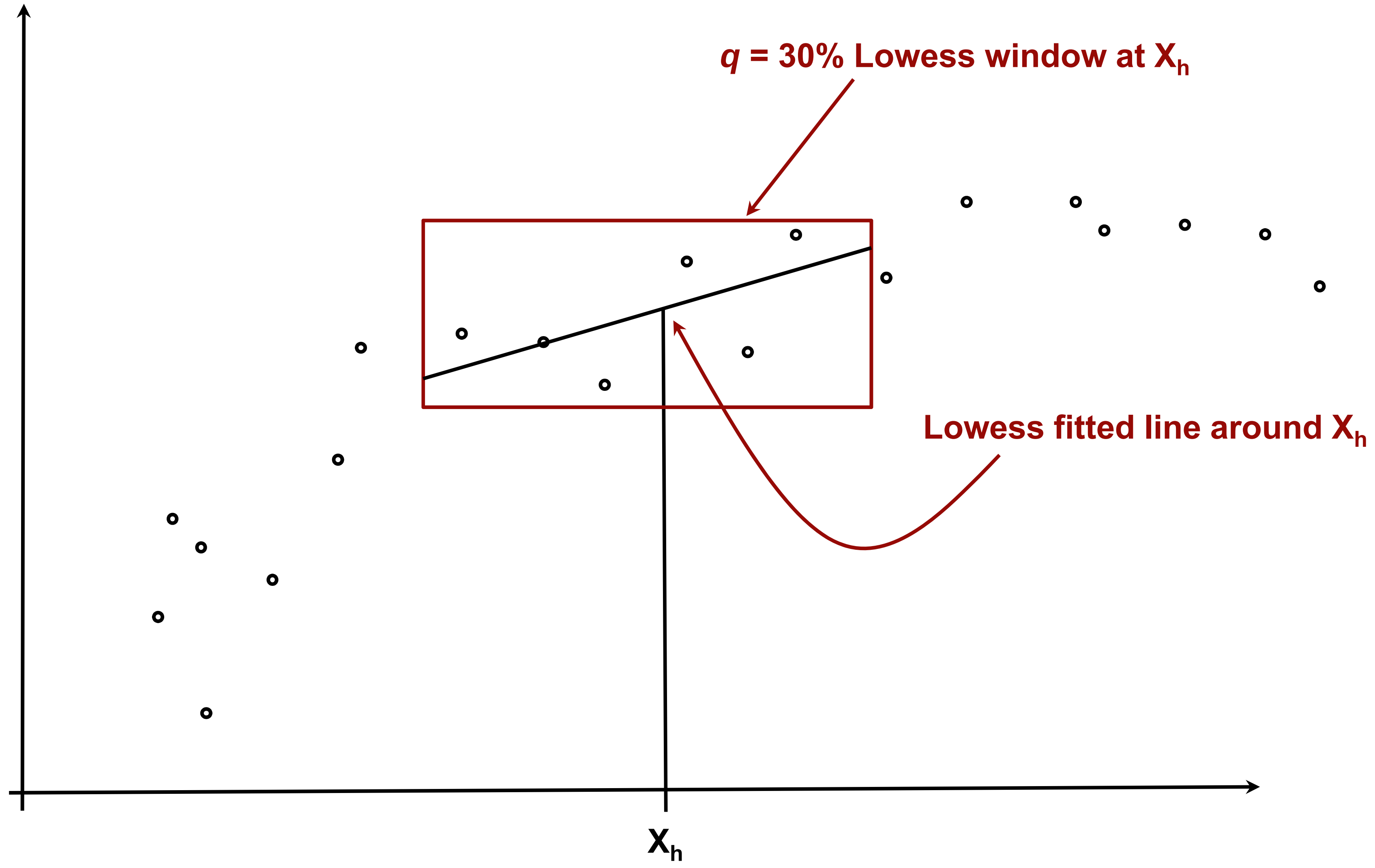
§11.4: Smoothing

- When
  (a) n is large, and
  (b) we are unsure of the form of E{Y_i},
  we can apply non-parametric regression smoothing to fit smooth curves through the data.
- A standard technique is called *lowess* or *loess* (for *lo*cally *we*ighted *s*catterplot *s*moothing).
- Lowess was introduced for the SLR model on pp. 138-139. It sets a 'window' or 'neighborhood' around any X_h and fits a low-order polynomial to the points in the window around X_h.

Lowess Weighted Fit

- In each window, a percentage *q* of points around some X_h is included.
- Lowess weights points in the window closer to X_h more heavily, and performs a WLS fit (only) within the window. The fitted value of $E\{Y_h\}$ is then computed at that X_h.
- Lowess then moves on to the next X_h, creates a new *q*-window, and repeats the process.

Schematic: Lowess Window


q = 30% Lowess window at X_h
Lowess fitted line around X_h
X_h

Lowess Extensions

- Can move to 2nd-order, quadratic fitted curves within each window to add robustness.
- Can iterate the process if outliers are a problem: find residuals from lowess fit and use these to update the original weights in each window. (Usually only two iterations are necessary to clear outlier effects.)
- In R, perform lowess smoothing via the `loess()` function. Can also use the `lowess()` function, or an automated plotter in the `scatter.smooth()` function.

Recall: Toluca data (CH01TA01)

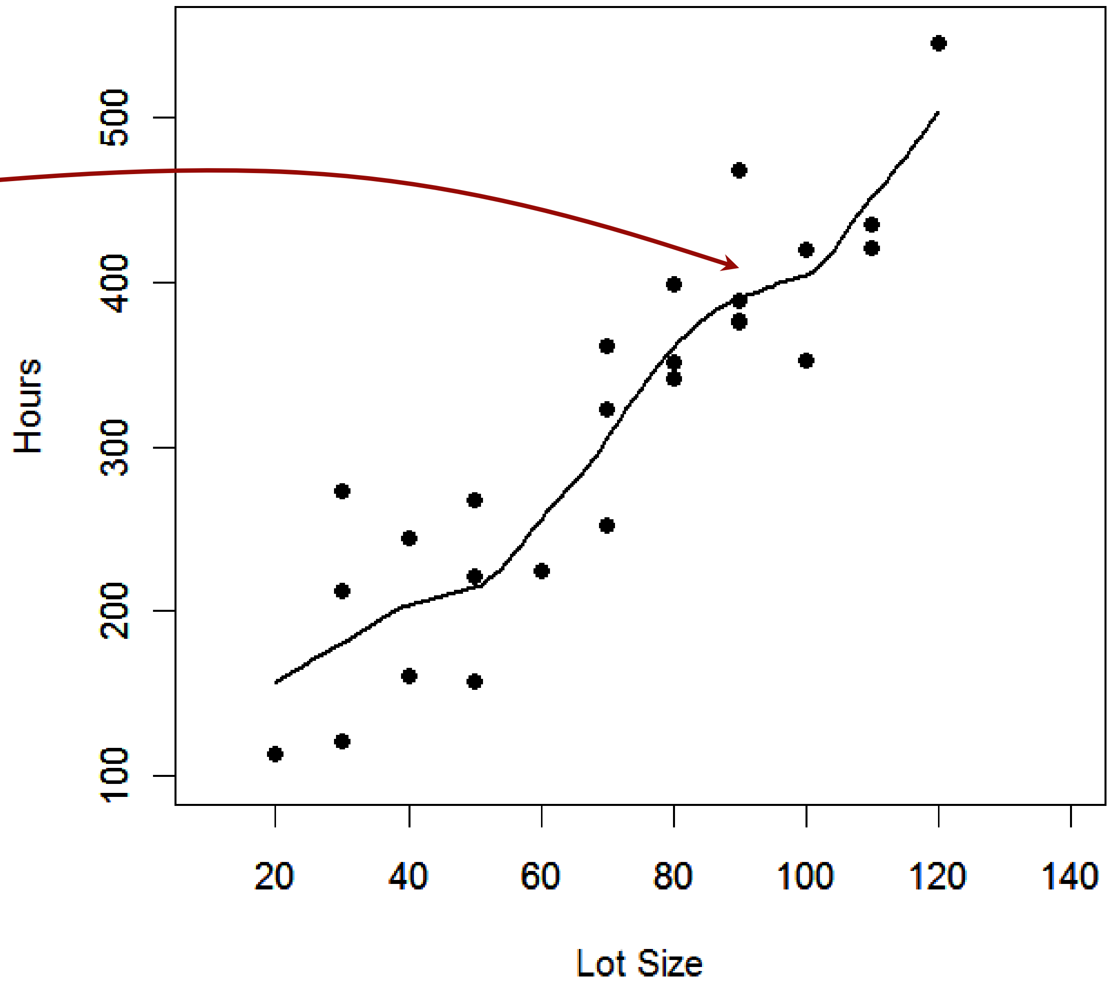
lowess **first-order smoothing for Toluca data from Ch. 1:**

```
> CH01TA01.loess = loess( Y ~ X,
                         span = 1/2, degree = 1 )
> plot( Y ~ X, pch=19 )
> Ysmooth = predict( CH01TA01.loess,
                         data.frame(X = 20:120) )
> par( new=T )
> plot( Ysmooth ~ seq(20,120), type='l',
                                           lwd=2 )
```

Plot follows →

Toluca data (CH01TA01) (cont'd)

***loess* smooth** **shows a generally linear pattern; cf. Fig. 3.19a**



Toluca data (CH01TA01) (cont’d)

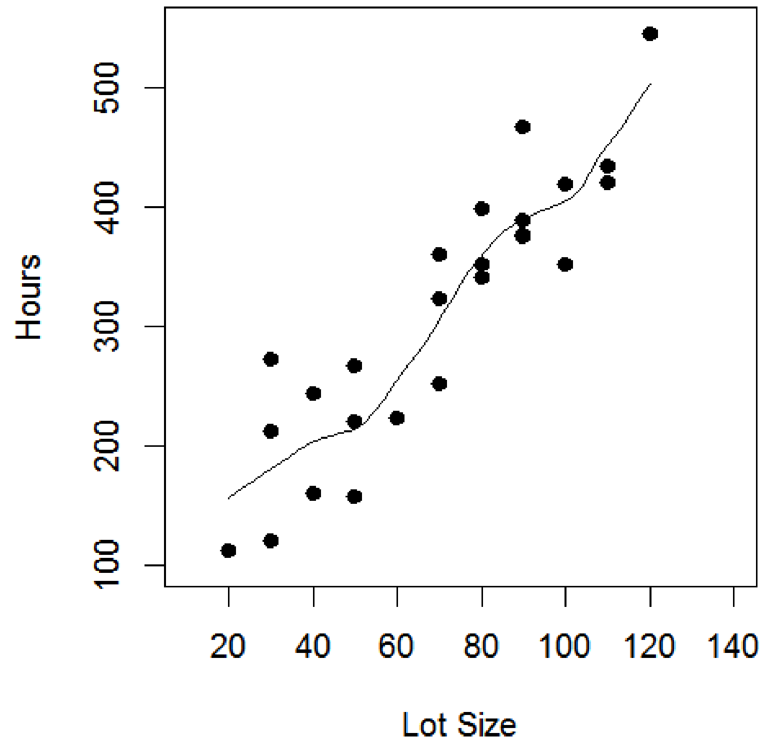
Compare to other functions for *lowess* fit:

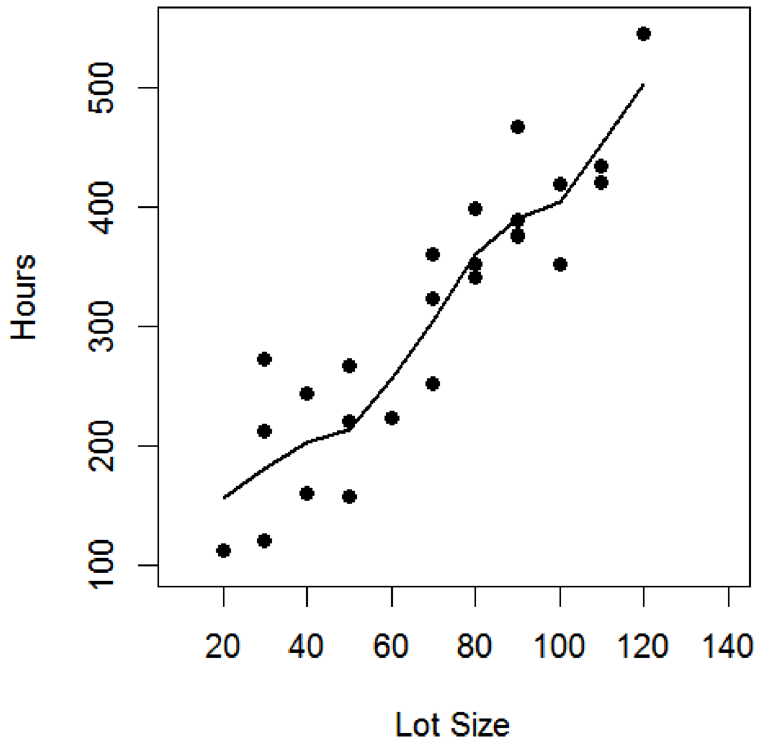
```
> scatter.smooth( Y ~ X, span=.5, pch=19,
   lwd=2, xlab='Lot Size', ylab='Hours‘,
   family=‘gaussian’ )
>
> plot( Y ~ X, pch=19, xlab='Lot Size',
                              ylab='Hours' )
> lines( lowess(Y ~ X, f=.5, iter=0),
                                     lwd=2 )
```

Plots follow →

Toluca data (CH01TA01) (cont'd)

lowess smooth via `scatter.smooth()` (left) and `lowess()` (right). The smoothed curves are essentially identical and also match Fig. 3.19a:





Loess Smoothing

- Lowess was extended into *loess* for multiple X's. The method is more complex, but the concepts are generally unchanged.
- Consider two X's, X_1 and X_2. At any 'new' $\mathbf{X}_h' = [X_{h1}\ X_{h2}]$, loess finds the fitted value for $E\{Y_h\}$ by fitting a **smoothed 1st- or 2nd-order surface** in a *q*-neighborhood around that $\mathbf{X}_h$.
- It then migrates the neighborhood thru the $\mathbf{X}$ space to approximate the $E\{Y\}$ surface.

Loess Smoothing (cont'd)

Needed is:

(a) a distance metric to define the neighborhood (Euclidean distance is common: $d_i = \sqrt{(X_{i1} - X_{h1})^2 + (X_{i2} - X_{h2})^2}$); and

(b) a weight function that is $w_i = 0$ outside the neighborhood and positive otherwise. For the weight, popular is the tricube:

$$w_i = \left(1 - \left|\frac{d_i}{d_{max}}\right|^3\right)^3$$

where d_{max} is the max. distance to any point in the current neighborhood.

Loess Smoothing (cont’d)

- Similar to the single-X case, loess essentially requires three user inputs:
  - (i) a value for *q* (usually $0.2 < q < 0.8$);
  - (ii) choice of 1st-order or 2nd-order smoothing; and
  - (iii) single pass (`family=‘gaussian’`) or robust/multi-pass (`family=‘symmetric’`) iterations. (The same `family=` option exist for `scatter.smooth()`, but not for `lowess()`.)
- In R, use the `loess()` function.

Life Insur. data (CH10TA01) (cont’d)

- Fit 1st-degree, robust loess smooth with *q* = ½ :

```
> CH10TA01.loess = loess( Y ~ X1+X2, span=1/2,
                  degree=1, family=‘symmetric’ )
```

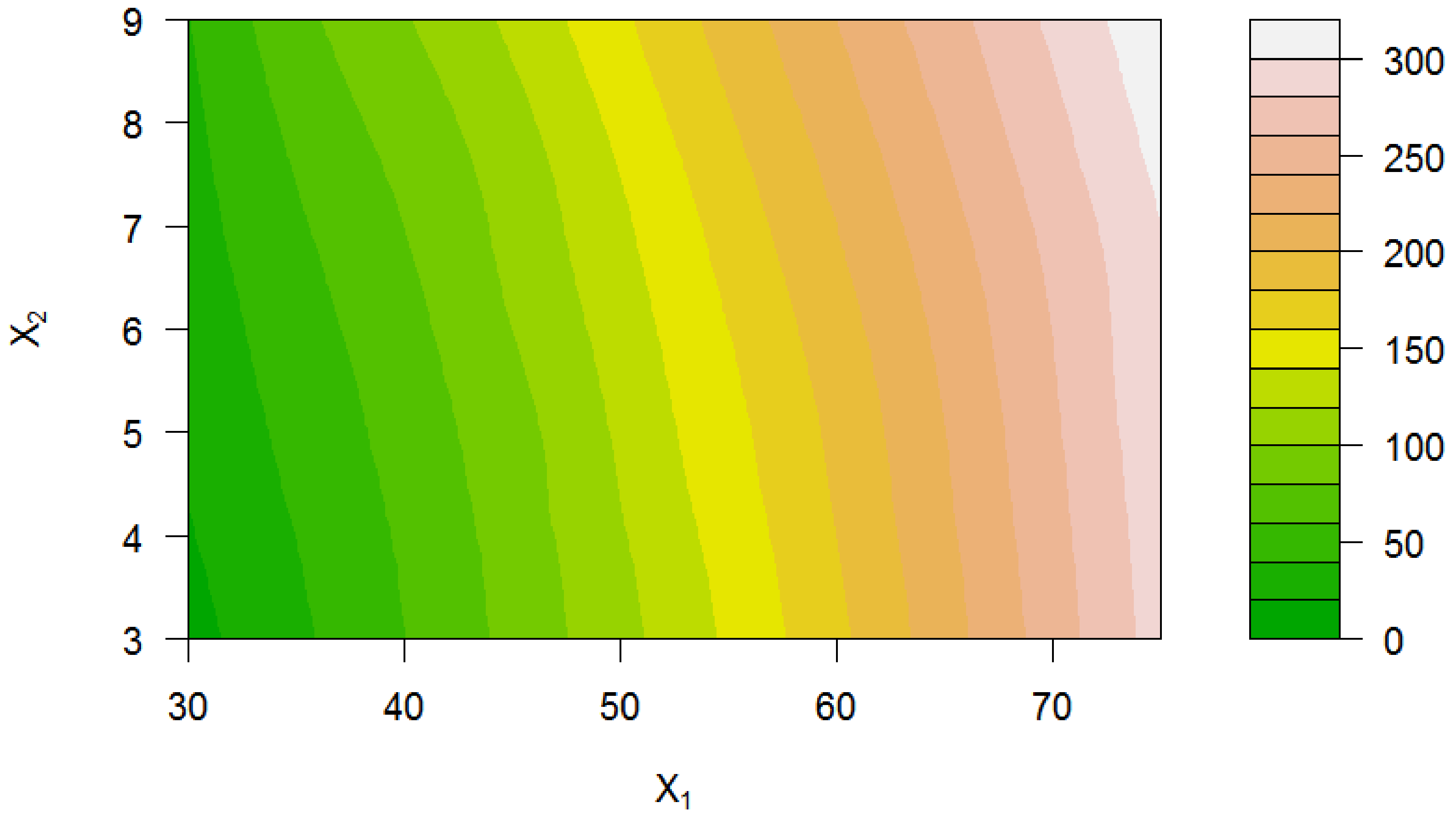
- Contour plot of fitted surface:

```
> X1grid = seq( 30,75,length=50 )
> X2grid = seq(  3, 9,length=50 )
> Ysmooth = matrix(0, nrow=50, ncol=50)
> for(i in 1:50) {
     for(j in 1:50) {
       Ysmooth[i,j] = predict( CH10TA01.loess,
   data.frame(X1=X1grid[i],X2=X2grid[j] )  )  } }
> filled.contour( x=X1grid, y=X2grid, z=Ysmooth,
              color.palette=terrain.colors,
  xlab=expression(X[1]), ylab=expression(X[2])  )
```

Plot follows →

Life Insur. data (CH10TA01) (cont'd)

Contour plot of 1st-degree *loess* smoother



Loess for Residual Analysis

Cleveland (1979) suggests a novel way to use loess to analyze residual patterns.

Given any regression fit, find the absolute residuals $|e_i|$. Then calculate a loess fit of $|e_i|$ against the predicted values $\hat{Y}_i$ and plot the smoothed loess curve.

If the loess curve is approximately horizontal, the loess diagnostic suggests that variation is not heterogeneous!

§11.5: Bootstrapping

- The Bootstrap (a.k.a. bootstrap resampling) is a modern method for performing statistical inferences when the distribution of the data is unknown or uncertain.
- The method is computer-intensive, and is based on the Monte Carlo Method of data simulation. It is elegantly simple: use the computer to sample *with replacement* ("resample") the data as if they were the full population.
- Then, use the simulated *bootstrap distribution* to find confidence intervals for the target parameter.

Bootstrap resampling

Given data Y_1, Y_2, ..., Y_n, the general procedure is as follows:

(1) generate a pseudo-random sample Y_1^*, Y_2^*, ..., Y_n^* by sampling <u>with replacement</u> from the original *n* values $\{Y_1, Y_2, ..., Y_n\}$,

(2) calculate the target estimator/statistic $\hat{\theta}^*$,

(3) repeat steps (1)–(2) a large number of times, say B [often see $B = n(\log n)^2$; book says B = 500 but for conf. intervals we usually take $B \geq 2000$],

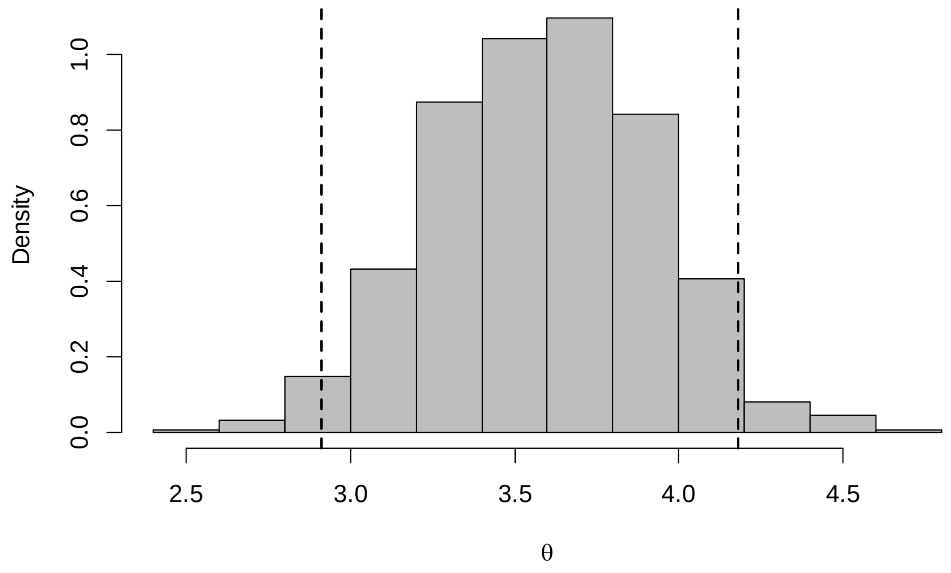
(4) assemble the $\hat{\theta}_b^*$ values (b = 1,...,B) and make inferences based on these B values.

$1 - \alpha$ Confidence Intervals

- From the bootstrap distribution of $\{\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*\}$ find the $\alpha/2$ and $1-(\alpha/2)$ percentiles; e.g., suppose $\alpha = 0.05$ and $B = 2000 \Rightarrow$ find the 51st and 1950th (ordered) $\hat{\theta}_b^*$ values.
- The <u>percentile method</u> uses

$$\hat{\theta}^*_{[51]} < \theta < \hat{\theta}^*_{[1950]}$$

as the 95% boot-strap conf. limits; see dashed lines at right $\longrightarrow$



$1 - \alpha$ Confidence Intervals (cont'd)

- **The reflection method modifies the percentile method slightly: find $\hat{\theta}$ from the original sample and use**

$$2\hat{\theta} - \hat{\theta}^*_{[1950]} < \theta < 2\hat{\theta} - \hat{\theta}^*_{[51]}$$

(see pp. 463-464).

- **In R, can use the external *boot* package, or just code it directly via the `sample()` function.**

Bootstrapping in Regression

- For the MLR/SLR setting, bootstrapping is a little more complicated.
- We can't just resample the Y_i values, since the LR model is embedded in the data: it's the $E[Y_i]$ in $Y_i = E[Y_i] + \epsilon_i$.
- Instead, we usually <u>resample</u> <u>the</u> <u>residuals</u> to approximate the distribution of ϵ_i.

Resampling SLR Residuals

For simplicity, consider the SLR case:

$\longrightarrow$ Find the fitted values $\hat{Y}_i$ at each X_i (i = 1,...,n).

$\longrightarrow$ Find the raw residuals $e_i = Y_i - \hat{Y}_i$.

$\longrightarrow$ Resample from the residual collection $\{e_1, \ldots, e_n\}$ to find bootstrapped residuals $e_1^*, e_2^*, \ldots, e_n^*$.

$\longrightarrow$ Then, take $Y_i^* = \hat{Y}_i + e_i^*$ as the bootstrapped responses at each X_i.

cont'd $\rightarrow$

Resampling SLR Residuals (cont'd)

→ With the (X_i, Y_i^*) pairs ($i = 1, \ldots, n$), fit the SLR to these bootstrap data and record the LS estimates b_0^* and b_1^*.

→ Repeat this B times to produce the bootstrap distribution of b_0 and b_1.

→ If, say, the goal is inferences on the slope β_1, collect the B values of b_{1b}^* and build a bootstrap confidence interval using these bootstrapped slope estimates.

An alternative method involves resampling with "random X" values; see p. 459.

Example: Toluca Data (CH01TA01)

- Recall the Toluca Data in Ch. 1 and our SLR fit. Apply a bootstrap analysis, with direct R coding:

```
> #set up components from original fit:
> ei = resid(CH01TA01.lm)
> Yhat = fitted( CH01TA01.lm )
> b1orig = coef( CH01TA01.lm )[2]
> n = length(Y)
> B = 2000          #2000 bootstrap resamples
> b1 = numeric(B)  #initialize
>
> set.seed( 571 )  #sets seed for sampler
```

code continues ⟶

Example: Toluca Data (cont'd)

```
> #simple “for” loop:
> for( b in 1:B ) {
>       estar = sample( ei, n, replace=T )
>       Ystar = Yhat + estar
>       b1[b] = coef(lm(Ystar~X))[2]
>                  }  #end “for” loop
>
> summary( b1 )
> b1 = sort( b1 ) #order b1 from small-to-large
```

code continues⟶

<u>Example</u>: Toluca Data (cont'd)

```
> #95% percentile limits if B=2000:
> b1L = b1[51]; b1U = b1[1950]
> c(b1L, b1U)
>
> hist( b1, prob=T )        #visualization
> abline(v=b1L, lty=2, lwd=2)
> abline(v=b1U, lty=2, lwd=2)
>
> #95% reflection limits:
> b1reflectL = 2*b1orig - b1U
> b1reflectU = 2*b1orig - b1L
> c(b1reflectL, b1reflectU)
```

output follows ⟶

Example: Toluca Data (cont'd)

R output (begin with `summary()` results):

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.297   3.331   3.564   3.559   3.792   4.675
```

Next: 95% percentile limits on β_1:

```
        2.913779   4.181367
```

Finally: 95% reflection limits on β_1:

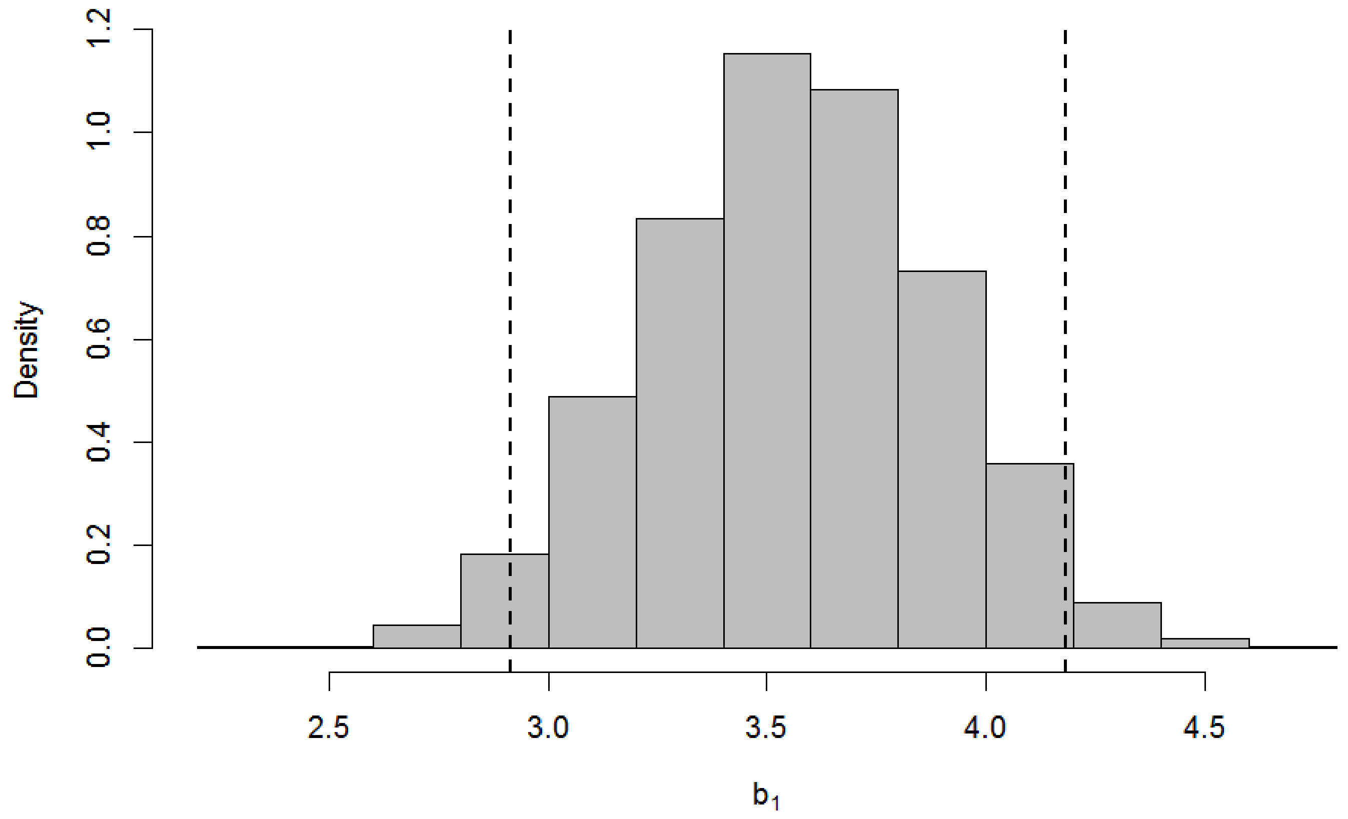
```
        2.959037   4.226625
```

Compare to orig. normal-theory 95% conf. limits:

```
> confint(CH01TA01.lm)[2,]
           2.5 %     97.5 %
        2.852435   4.287969
```

<u>Example</u>: Toluca Data (cont'd)

Visualization: Histogram of bootstrap distribution with 95% percentile limits marked by dashed lines:


Density
b_1