Generalized Least Squares and Time Series Regression

1 Generalized Least Squares

A simple and natural extension of weighted least squares (WLS) is *generalized least-squares (GLS) regression.* Formally, GLS looks exactly like WLS:

Suppose that the model is

$$\underline{y} = X\underline{\beta} + \underline{\epsilon}$$

where

$$\underline{\epsilon}_{n\times 1} \sim N_n(\underline{0}, \boldsymbol{\Sigma})$$

i.e. now the random vector $\underline{\epsilon}$ is multivariate normal with mean $\underline{0}$ and variance-covariance matrix $\mathrm{Var}\,(\underline{\epsilon}) = \boldsymbol{\Sigma}$.

If we know the exact value of $\boldsymbol{\Sigma}_{n\times n}$, we can compute a square root $\Gamma_{n\times n}$ of $\boldsymbol{\Sigma}^{-1}$, as we have seen many times before:

$$\boldsymbol{\Sigma}^{-1} = \Gamma^T\Gamma$$

For example we can compute Γ using the eigenvector-eigenvalue decomposition of $\boldsymbol{\Sigma}$. For computational reasons it is usually better to choose Γ to be upper-triangular or lower-triangular; this can be done in Splus using the function `chol()`. Once we have Γ, we can immediately reduce to the standard model by considering instead

$$\underline{y}^* = \Gamma\underline{y} = \Gamma X\underline{\beta} + \Gamma\epsilon = X^*\underline{\beta} + \epsilon^*$$

Note that in our discussion of WLS we simply calculated Γ directly as

$$\Gamma = W^{1/2} = \begin{bmatrix} w_i & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}$$

We can immediately apply our usual least-squares or normal-theory MLE machinery to show that the MLE's are

$$\hat{\underline{\beta}}_{GLS} = (X^{*T}X^*)^{-1}X^*\underline{y}^* = (X^T\Gamma^T\Gamma X)^{-1}X^T\Gamma^T\Gamma\underline{y} = (X^T\boldsymbol{\Sigma}^{-1}X)^{-1}X^T\boldsymbol{\Sigma}^{-1}\underline{y}$$

Once again, we could also obtain these results by directly differentiating the log of the likelihood

$$\begin{aligned} L(\underline{y};\underline{\beta},\boldsymbol{\Sigma}) &= \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\underline{y}-X\underline{\beta})^T\boldsymbol{\Sigma}^{-1}(\underline{y}-X\underline{\beta})\right] \qquad (1) \\ &= \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\Gamma[\underline{y}-X\underline{\beta}])^T(\Gamma[\underline{y}-X\underline{\beta}])\right] \\ &= \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\underline{y}^*-X^*\underline{\beta})^T(\underline{y}^*-X^*\underline{\beta})\right] \end{aligned}$$

As before, the solutions $\hat{\underline{\beta}}_{GLS}$ are obtained by minimizing the "weighted" squared raw residuals

$$Q = \sum_i (y_i^* - \hat{y}_i^*)^2 = (\underline{y}^* - X^*\underline{\beta})^T(\underline{y}^* - X^*\underline{\beta}) = (\underline{y} - X\underline{\beta})^T\boldsymbol{\Sigma}^{-1}(\underline{y} - X\underline{\beta})$$

If we really have the right Σ, it is easy to show that

$$\left.\begin{array}{rcl} E[\hat{\underline{\beta}}_{GLS}|X] & = & \underline{\beta}, \\ \hat{\mathrm{Var}}\,(\hat{\underline{\beta}}_{GLS}) & = & (X^T\Sigma^{-1}X)^{-1}, \end{array}\right\} \tag{2}$$

that the $\hat{\underline{\beta}}_{GLS}$ are BLUE (Gauss-Markov theorem), etc.

In practice we have to either

- Know Σ in advance; or
- Estimate Σ from the data.

Of course, knowing Σ in advance is unlikely for any realistic problem. To estimate Σ from the data, we can precompute an estimate of Σ from the OLS residuals (which has the usual multiple-looks-at-the-data problems) or we can try to maximize the likelihood (1) simultaneously in $\underline{\beta}$ and Σ.

For the latter approach to work, Σ cannot be completely general, because Σ would then have $n(n-1)/2$ parameters to estimate, which grows like the square of the sample size! Instead we need to be able to assert something about the form of Σ. One very common and useful form is obtained by assuming that the errors $\underline{\epsilon}$ form a simple *time series*, and so we now digress to introduce some simple time series ideas.

2 Serially Correlated Errors — Time Series

A *time series* is simply a sequence of observations indexed by time. The errors ϵ_i in a linear regression model may or may not be indexed by time (depends on the application of course!) but the simple kinds of correlation structures that arise in time series models are useful as models of correlation structure in linear regression as well.

The basic idea of time series correlation structure is that ϵ_i and ϵ_j will be highly correlated if i and j are close together, and will have little correlation if i and j are far apart. Clearly this kind of structure works for other interpretations of i and j than points in time. For example, if i and j represent points in space, then we are dealing with an area of statistics called *spatial statistics*; see for example Venables and Ripley (1997, Chapter 16).

In this section we introduce a few simple building blocks of time series models that are useful for building correlation models in linear regression. In all cases we are dealing with the time series $\epsilon_1, \epsilon_2, \ldots$ that arises as errors in linear regression. In particular we will assume

- $E[\epsilon_i] = 0$ for all i
- $\mathrm{Var}\,(\epsilon_i) = \sigma_\epsilon^2$ for all i

and we are mostly interested in learning about the correlations

$$\mathrm{Corr}\,(\epsilon_i, \epsilon_j) = \mathrm{Cov}\,(\epsilon_i, \epsilon_j)/\sigma_\epsilon^2$$

2.1 Stationary Errors and the Autocovariance and Autocorrelation Functions

The error time series $\epsilon_1, \epsilon_2, \ldots$ is *stationary*[1] if the covariance between any two observations

$$\mathrm{Cov}\,(\epsilon_i, \epsilon_j) = E[\epsilon_i \epsilon_j]$$

depends only on the difference $k = |i - j|$ and not on the particular locations i and j.

For a stationary time series, the *autocovariance function* is

$$\gamma(k) = \mathrm{Cov}\,(\epsilon_i, \epsilon_{i-k})$$

Note that $\sigma_\epsilon^2 = \mathrm{Var}\,(\epsilon_i) = \gamma(0)$. The *autocorrelation function* is defined to be

$$\rho(k) = \gamma(k)/\gamma(0) \equiv \mathrm{Corr}\,(\epsilon_i, \epsilon_{i-k})$$

A plot of $x = k$ vs $y = \rho(k)$ is called an *autocorrelation function plot* (ACF plot) and is a basic tool in displaying the serial dependence in time series data. (We'll see examples below).

2.2 Basic time series models: AR and MA

The two basic building-blocks for time series models are the autoregressive (AR) and moving average (MA) processes. MA processes are easier to describe first.

- A *moving average* process of order q, denoted $MA(q)$, has the form

  $$\epsilon_i = \mu + \beta_0 z_i + \beta_1 z_{i-1} + \ldots + \beta_q z_{i-q}$$

  or, in terms of the *backshift operator* $Bz_i = z_{i-1}$,

  $$\begin{aligned} \epsilon_i &= (\beta_0 + \beta_1 B + \ldots \beta_q B^q) z_i \\ &\equiv \Theta(B) z_i, \end{aligned}$$

  where the z_i are i.i.d. $N(0, \sigma_z^2)$. ("white noise" in time series parlance). One can show that ϵ_i is always stationary, and

  $$\gamma(k) = \begin{cases} \sum_{i=0}^{q-|k|} \beta_i \beta_{i+|k|} & |k| \le q \\ 0 & |k| > q \end{cases}$$

  Figure 1 gives plots of a simulated $MA(1)$ and a simulated $MA(2)$ process.

```
> ma.1 _ arima.sim(model=list(ma=.5))
> ma.2 _ arima.sim(model=list(ma=c(.2,.3)))
> ts.plot(ma.1); title("MA(1) with beta1=0.5")
> ts.plot(ma.2); title("MA(2) with beta1=0.2, beta2=0.3")
> acf(ma.1)
> acf(ma.2)
```

  The autocorrelation plot for a $MA(q)$ process drops to zero (except for noise) after lag q, which make the `acf` plot a good first step in guessing the order q of the process.

[1] This is "second-order stationarity", which is the same as the time series notion of "strong stationarity if the ϵ_i's are jointly normal.

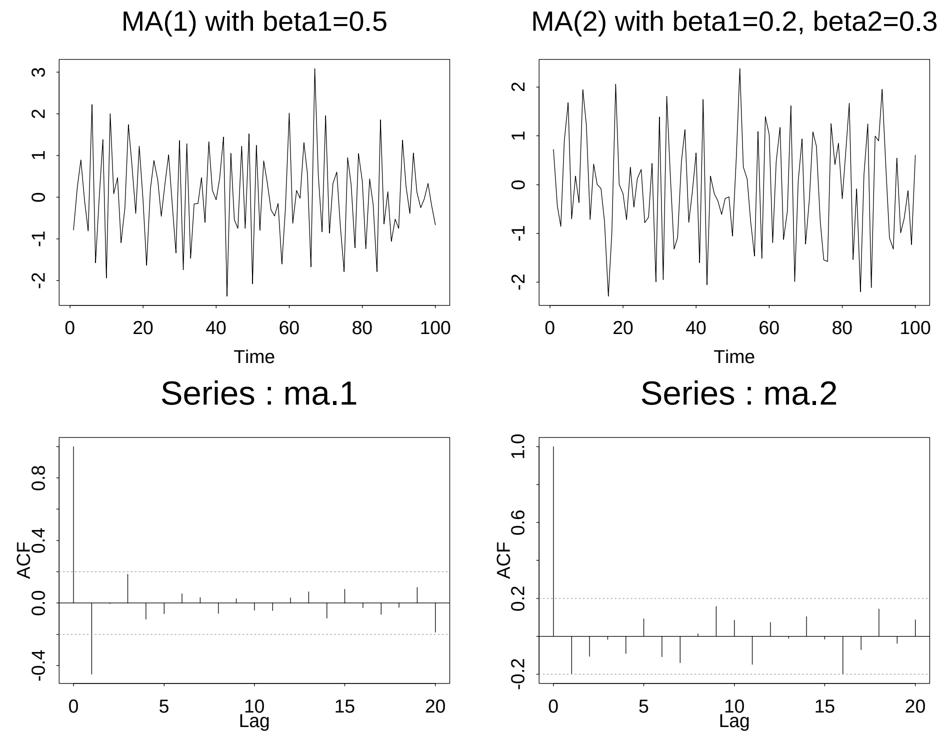

Figure 1: An MA(1) process and an MA(2) process, with corresponding autocorrelation functions.

- An *autoregressive* process of order p, denoted $AR(p)$ has the form

$$\epsilon_i = \alpha_1\epsilon_{i-1} + \alpha_2\epsilon_{i-2} + \cdots + \alpha_p\epsilon_{i-p} + z_i$$

where again z_i are i.i.d. $N(0, \sigma_z^2)$. In terms of the *backshift operator*, $B\epsilon_i = \epsilon_{i-1}$, we may write

$$(1 - \alpha_1 B - \alpha_2 B^2 - \cdots - \alpha_p B^p)\epsilon_i = z_i.$$

Not all values of the coefficients α_k lead to a stationary series. For example the process $\epsilon_i = \epsilon_{i-1} + z_i$ *not* a stationary process, since this would imply $\sigma_\epsilon^2 = \sigma_\epsilon^2 + \sigma_z^2$, which is impossible unless $\sigma_z^2 = 0$. More generally the complex roots of the polynomial

$$\Phi(x) = 1 - \alpha_1 x - \alpha_2 x^2 - \cdots - \alpha_p x^p$$

must have all p of its roots outside the unit circle in the complex plane. In the case of an AR(1) process, this is easy to show!

One can show that the autocovariance function for a stationary $AR(p)$ process satisfies the "Yule-Walker" relations

$$\sum_{k=1}^{p} \gamma(k-i)\alpha_k = \gamma(k) \qquad i = 1, 2, \ldots, p,$$

We can use the Yule-Walker equations to solve for $\gamma(k)$ in terms of the α's, if they are known, or solve for the α's in terms of the $\gamma(k)$'s. Also,

$$\sigma_\epsilon^2 = \sum_{k=1}^{p} \gamma(k)\alpha_k + \sigma_z^2.$$

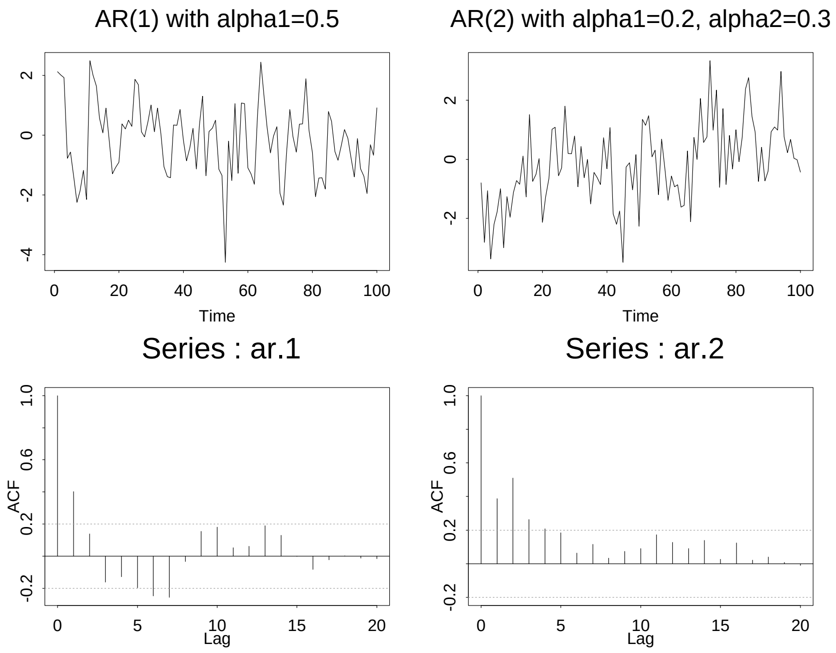

Figure 2: An AR(1) process and an AR(2) process, with corresponding autocorrelation functions.

As an exercise, show that the autocorrelation function of an $AR(1)$ process $\epsilon_i = \alpha\epsilon_{i-1} + z_i$ satisfies $\rho_k = \alpha\rho_{k-1}$, and so, inserting $\rho_0 = 1$, we get $\rho_k = \alpha^k$. What does ρ_k look like for an $AR(2)$ process? Figure 2 contains plots of simulated $AR(1)$ and $AR(2)$ processes, and the corresponding autocorrelation functions.

```
> par(mfrow=c(2,2))
> ar.1 _ arima.sim(model=list(ar=.5))
> ts.plot(ar.1); title("AR(1) with alpha1=0.5")
> ar.2 _ arima.sim(model=list(ar=c(.2,.3)))
> ts.plot(ar.2); title("AR(2) with alpha1=0.2, alpha2=0.3")
> acf(ar.1)
> acf(ar.2)
```

The `acf` plot of an $AR(1)$ decays exponentially to zero, possibly with some oscillation in sign. The $AR(2)$ `acf` plot also decays to zero (this gradual decay in the `acf` plot distinguishes pure-AR from pure-MA processes) but it isn't easy to tell from the plot what the order p of the process should be.

There is another kind of time series plot called a *partial autocorrelation (PACF) plot*, which is basically the graph of the coefficients α_i in an $AR(p_{MAX})$ model where p_{MAX} is much larger than the true order of the process, that can be used to graphically guess the order of an AR process. In a partial autocorrelation plot, the plotted values of α_i fall to zero when $i > p$, the true order of the $AR(p)$ process. You can get a PACF plot in Splus by using `acf(...,type="partial")`.

- For completeness I also mention the <u>*ARMA*</u> process of orders p and q, denoted $ARMA(p, q)$, which simply combines the structure of an $AR(p)$ and an $MA(q)$ model. This is easiest to express in terms of the backshift operator: An $ARMA(p,q)$ process satisifes $\Phi(B)\epsilon_i = \Theta(B)z_i$, where again z_i are i.i.d. $N(0,\sigma^2)$, and $\Phi(B)$ and $\Theta(B)$ are polynomials of the same form as indicated above.

2.3 The Variance-Covariance Matrix of an AR(1) Process

Errors $\epsilon_1,\ldots,\epsilon_n$ that follow an $ARMA(p,q)$ process generally have a fairly simple correlation structure, with only about $p+q$ parameters in Σ, instead of the full $n(n-1)/2$ parameters in an unstructured Σ.

As an example, we note that, from the general descriptions of AR and MA processes above, it is easy to show that, for an $AR(1)$ process

- $\epsilon_i = \alpha\epsilon_{i-1} + z_i$;
- $\sigma_\epsilon^2 = \sigma_z^2/(1-\alpha^2)$;
- $\mathrm{Corr}\,(\epsilon_i,\epsilon_{i-k}) = \alpha^k$.

and thus the form of the covariance matrix is

$$\Sigma = \mathrm{Var}\,(\underline{\epsilon}) = \sigma_z^2/(1-\alpha^2)\begin{bmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ \alpha^1 & 1 & \alpha^1 & \cdots & \alpha^{n-2} \\ \alpha^2 & \alpha^1 & 1 & \cdots & \alpha^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{n-1} & \alpha^{n-2} & \alpha^{n-2} & \cdots & 1 \end{bmatrix} = \sigma_\epsilon^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-2} & \cdots & 1 \end{bmatrix}$$

where we are letting $\rho = \alpha = \mathrm{Corr}\,(\epsilon_i,\epsilon_{i-1})$.

3 Computation, for the AR(1) errors model

Now let us consider estimation for

$$\underline{y} = X\underline{\beta} + \underline{\epsilon}, \quad \underline{\epsilon} \sim N(\underline{0},\Sigma)$$

where the errors have AR(1) structure

$$\Sigma \;=\; \sigma_z^2/(1-\rho^2)\begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-2} & \cdots & 1 \end{bmatrix} \;\equiv\; \sigma_\epsilon^2\, R$$

In this case it is easy to verify that a lower-triangular square root of R^{-1} is

$$\Gamma = \begin{bmatrix} \sqrt{1-\rho^2} & 0 & 0 & \cdots & 0 \\ -\rho & 1 & 0 & \cdots & 0 \\ 0 & -\rho & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3}$$

and we can immediately apply OLS to the transformed equation

$$\underline{y}^* = X^* \underline{\beta} + \underline{\epsilon}$$

and use the usual OLS estimates of $\underline{\beta}$, σ^2_ϵ, etc. Thus all we need to know is ρ. There are basically two approaches:

- Fit the uncorrected OLS regression and estimate ρ directly from the equation

$$\hat{\rho} = \frac{\sum_{i=2}^n E_i E_{i-1}}{\sum_{i=1}^n E_i^2} \tag{4}$$

  and then plug this into the estimation framework just outlined.

- Observe that the general likelihood (1) simplifies, in the case of AR(1) structured errors, to the log-likelihood

$$\begin{aligned}\log L(\underline{y}; \underline{\beta}, \boldsymbol{\Sigma}) &= \log\left\{\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\underline{y} - X\underline{\beta})^T \boldsymbol{\Sigma}^{-1}(\underline{y} - X\underline{\beta})\right]\right\} \\ &= -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma_z^2 - \frac{1}{2}\log(1-\rho^2) - \frac{1}{2\sigma_z^2}(\underline{y}^* - X^*\underline{\beta})^T(\underline{y}^* - X^*\underline{\beta})\end{aligned}$$

  (where $\underline{y}^* = \Gamma \underline{y}$ and $X^* = \Gamma X$ and Γ is from equation 3); and this can be maximized directly (we will give an example at the end of the next section).

4 An example

As an example we consider data of Fox and Hartnagel (1979, *Can. Rev. of Sociology and Anthropology, 16*, 96–104) on yearly theft-conviction rates per 100,000 Canadian women aged 15 years and older, in the years 1935–1968. The variables are

```
Year          1935--1968
Fert          Total Fertility Rate per 1000
Partic        Labour-Force Participation Rate per 1000
Coll          Post-Secondary Degree Rate per 10,000
FemAllConv    Female Indictable-Offense Conviction Rate per 100,000
FemTheftConv  Female Theft Conviction Rate per 100,000
MalAllConv    Male Indictable-Offense Conviction Rate per 100,000
MalTheftConv  Male Theft Conviction Rate per 100,000
```

Note that there are some missing values in this data, and Fox's codes "?" have to be changed to "NA" in order to identify (and for the purposes of illustration delete) these rows of the data matrix. Once this is done, we read the data into Splus in the usual way:

```
> hart _ read.table("hartnagl.dat",na.strings="?")
> dim(hart)
[1] 38  8
> names(hart) _ scan(what="")
```

```
1: Year
2: Fert
3: Partic
4: Coll
5: FemAllConv
6: FemTheftConv
7: MalAllConv
8: MalTheftConv
9:
> hart
   Year Fert Partic Coll FemAllConv FemTheftConv MalAllConv MalTheftConv
 1 1931 3200    234 12.4       77.1           NA      778.7           NA
 2 1932 3084    234 12.9       92.9           NA      745.7           NA
 3 1933 2864    235 13.9       98.3           NA      768.3           NA
 4 1934 2803    237 13.6       88.1           NA      733.6           NA
 5 1935 2755    238 13.2       79.4         20.4      765.7        247.1
 6 1936 2696    240 13.2       91.0         22.1      816.5        254.9
 7 1937 2646    241 12.2      100.4         22.4      821.8        272.4
 8 1938 2701    242 12.6      108.9         21.8      956.8        285.8
 9 1939 2654    244 12.3      123.6         21.1     1035.7        292.2
10 1040 2766    245 12.0      157.3         21.4      951.6        256.0
11 1941 2832    246 11.7      154.3         25.3      850.9        205.8
12 1942 2964    268 11.2      143.9         27.1      769.7        188.0
13 1943 3041    333 11.5      147.5         29.0      811.2        205.8
14 1944 3010    335 11.1       97.3         24.2      865.4        207.9
15 1945 3018    331 12.5       76.6         24.7      866.8        197.8
16 1946 3374    253 15.0       72.8         20.5      968.0        195.9
17 1947 3545    243 17.6       68.9         20.7      894.0        198.9
18 1948 3441    241 21.2       66.7         22.8      830.8        198.6
19 1949 3456    242 22.7       55.0         18.5      811.2        216.1
20 1950 3455    237 21.4       55.0         19.3      775.4        212.0
21 1951 3503    242 20.7       55.0         20.0      739.8        208.0
22 1952 3641    240 19.5       54.7         19.4      740.3        199.3
23 1953 3721    233 36.6       47.9         16.1      725.1        176.4
24 1954 3828    232 18.1       49.0         16.5      739.9        166.8
25 1955 3831    236 17.7       44.4         15.2      663.8        179.7
26 1956 3858    245 18.7       40.2         15.3      633.7        198.8
27 1957 3925    256 20.8       43.8         17.1      716.5        262.5
28 1958 3880    261 22.7       49.5         21.2      761.0        263.4
29 1959 3935    265 24.6       50.4         22.3      665.3        253.6
30 1960 3895    278 28.5       59.6         28.5      742.6        280.9
31 1961 3840    291 31.3       72.4         34.4      789.8        290.9
32 1962 3756    290 38.0       70.6         36.0      776.8        274.7
33 1963 3669    290 42.0       82.0         43.8      842.2        296.9
34 1964 3502    307 48.5       89.5         49.7      799.1        281.3
35 1965 3145    313 59.2      101.3         58.6      763.0        264.2
36 1966 2812    325 69.7      116.7         71.4      804.6        286.6
```

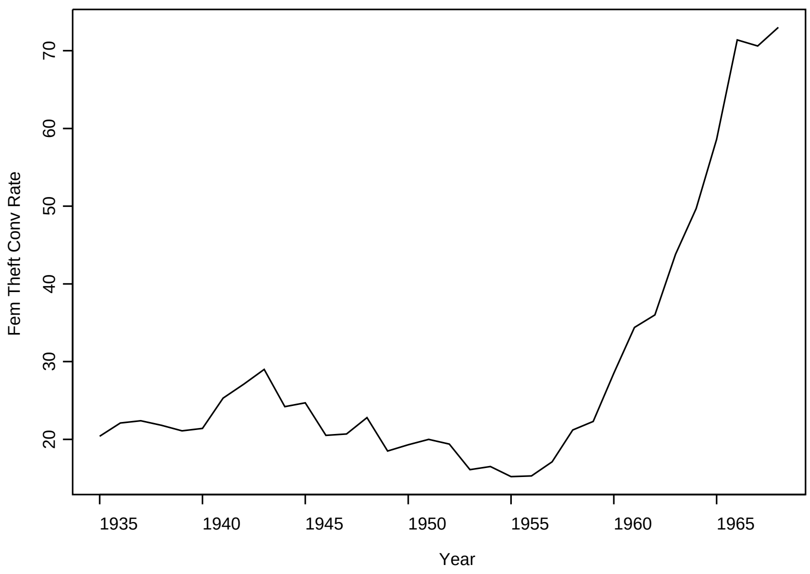

Figure 3: Canadian women's theft-per-100,000 rate from the Fox and Hartnagel data.

```
37 1967 2586    339 80.4      115.2          70.6      781.1         272.0
38 1968 2441    338 90.4      122.9          73.0      849.7         274.7
> hart _ hart[!is.na(apply(hart,1,sum)),]
```

The raw y data (female theft conviction rates) is shown in Figure 3. One can see apparently-strong serial dependence in the data, and also some change in the trend of the data after about 1960.

```
> attach(hart)
> ts.plot(FemTheftConv,axes=F,xlab="Year",ylab="Fem Theft Conv Rate")
Warning messages:
  Non time series data in: ts.plot(FemTheftConv, axes = F, xlab =
        "Year", ylab = "Fem Theft Conv Rate")
> axis(1,at=seq(1,1+40,by=5),label=seq(1935,1935+40,by=5))
> axis(2)
> box()
```

Now we will

- Do an OLS analysis and look for problems in the residuals
- Do the plug-in estimate of ρ for an AR(1) model of the residuals
- Do a joint maximization using an Splus function `nlmin()` that is designed to minimize fairly general nonlinear functions.

4.1 The OLS analysis

We regress female theft conviction rates on fertility rates, participation in the labor force, and post-secondary degree rates, to see if these things account for the rate at which women are convicted for (and hence participate in) thefts. In addition we regress on the male theft conviction rate, as a stand-in for other environmental factors that are not captured by the other three variables.

```
> summary(olsfit _ lm(FemTheftConv ~ Fert + Partic + Coll + MalTheftConv))

Call: lm(formula = FemTheftConv ~ Fert + Partic + Coll + MalTheftConv)
Residuals:
    Min     1Q  Median    3Q   Max
 -8.974 -1.945 -0.4133 2.527 7.165

Coefficients:
                Value Std. Error  t value Pr(>|t|)
 (Intercept)  -7.3341   9.4379    -0.7771   0.4434
        Fert  -0.0061   0.0014    -4.2017   0.0002
      Partic   0.1199   0.0234     5.1236   0.0000
        Coll   0.5516   0.0433    12.7525   0.0000
MalTheftConv   0.0393   0.0186     2.1193   0.0428

[some output omitted]

> par(mfrow=c(2,3))
> plot(olsfit)
> attach("../week10")
> plot(fitted(olsfit),residuals(olsfit))
> abline(h=0)
> qqenv(residuals(olsfit))
> plot(35:68,residuals(olsfit),type="l")
> abline(h=0)
> print(acf(residuals(olsfit)))
$acf:

, , 1
             [,1]
 [1,]  1.0000000000
 [2,]  0.2441946566
 [3,] -0.1919368953
 [4,] -0.2654978931
 [5,] -0.0002950965
 [6,]  0.0247247219

[some output omitted]
```

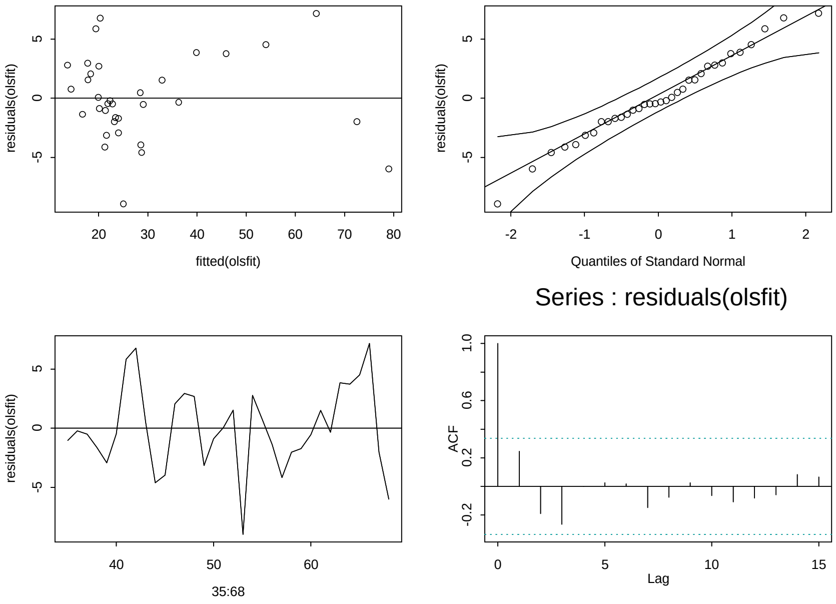

Figure 4: Some residual diagnostics for the OLS regression `lm(formula = FemTheftConv ~ Fert + Partic + Coll + MalTheftConv)`.

The "usual" residual diagnostics shown in the first two panels of Figure 4 don't show much. However, a time-series plot of the residual does appear to reveal some serial dependence (though with only 35 data points it is a bit hard to say). The dependence is not terribly strong, as the autocorrelation plot in Figure 4 doesn't show any autocorrelations outside approximate 95% testing bands for H_0: correlation equals zero[2].

From the Splus code above, you can see that the `acf()` function also prints out the autocorrelations at lags 1, 2, ... In particular we can deduce from the output that

$$\hat{\rho} = \widehat{\mathrm{Cov}}(\epsilon_i, \epsilon_{i-1}) = 0.2442$$

It can be shown that the OLS $\underline{\hat{\beta}}$'s are still unbiased and consistent estimates of $\underline{\beta}$, but they will no longer be BLUE (the proper GLS estimates are BLUE), and their standard error estimates will be biased in general; the correct standard errors of the OLS estimates when $\underline{\epsilon} \sim N(\underline{0}, \boldsymbol{\Sigma})$ can be easily shown to be

$$\mathrm{Var}\,(\underline{\hat{\beta}}) \;=\; (X^T X)^{-1} X^T \boldsymbol{\Sigma} X (X^T X)^{-1}$$

Using the estimated $\hat{\rho} = 0.2442$, we can construct the matrix Γ in (3) directly, form an estimate of the variance-covariance matrix $\boldsymbol{\Sigma}$, and apply this formula to get corrected SE's for the OLS $\hat{\beta}$'s.

```
> n _ length(FemTheftConv)
> rho _ 0.2442
> Gam _ diag(c(sqrt(1-rho^2),rep(1,length=n-1))) - rho*rbind(0,cbind(diag(n-1),0))
```

[2] These testing bands are somewhat crude however and a better approach might be to use the Durbin-Watson style test for nonzero autocorrelation mentioned in Rawlings, Pantula and Dickey (1998, pp. 354–355).

```
> SE2 _ sum(residuals(olsfit)^2)/olsfit$df.residual
> VC _ SE2*solve(t(Gam)%*%Gam)
> X _ model.matrix(olsfit)
> XTXI _ solve(t(X)%*%X)
> VarBeta _ XTXI %*% t(X) %*% VC %*% X %*% XTXI
> sqrt(diag(VarBeta))

[1] 11.745943500  0.001859433  0.028712245  0.053070901  0.023239818

> # now compare with the OLS standard errors...

> rbind(summary(olsfit)$coefficients[,2],sqrt(diag(VarBeta)))

     (Intercept)        Fert     Partic       Coll MalTheftConv
[1,]     9.43792 0.001449503 0.02340973 0.04325091   0.01855587
[2,]    11.74594 0.001859433 0.02871224 0.05307090   0.02323982
```

Why don't we stop here, since we already seem to have a solution?

- The OLS estimates $\underline{\hat{\beta}}$ are not particularly efficient; and less-so as ρ grows;
- The estimate σ_ϵ^2 is also not particularly efficient; and hence the corrected OLS standard error estimates are not terribly efficient;
- We have not incorporated uncertainty in estimating ρ into our SE estimates, and so the "real" SE estimates are likely to be somewhat higher yet.

4.2 The Plug-In GLS analysis

Using the matrix Γ in (3) that we constructed above to adjust the OLS SE's for $\hat{\beta}$'s, we can form the adjusted regression $\underline{y}^* = X^*\underline{\beta} + \underline{\epsilon}$ and apply OLS to this adjusted problem.

```
> ystar _ Gam %*% FemTheftConv
> Xstar _ Gam %*% X
> summary(glsfit1 _ lm(ystar ~ Xstar - 1))

Call: lm(formula = ystar ~ Xstar - 1)
Residuals:
    Min     1Q  Median    3Q   Max
 -9.294 -1.738 -0.2283 2.157 6.915

Coefficients:
                    Value Std. Error  t value Pr(>|t|)
 Xstar(Intercept)  -6.6417  11.1218   -0.5972   0.5550
        XstarFert  -0.0059   0.0018   -3.3042   0.0025
      XstarPartic   0.1156   0.0271    4.2672   0.0002
        XstarColl   0.5359   0.0501   10.7058   0.0000
```

```
XstarMalTheftConv   0.0399   0.0220     1.8168   0.0796

[some output omitted]

> # compare corrected-SE OLS estimates with plug-in GLS estimates

> cbind(cbind(summary(olsfit)$coefficients[,1],sqrt(diag(VarBeta))),
+ summary(glsfit1)$coefficients[,1:2])
                                                      Value   Std. Error
 Xstar(Intercept) -7.33414345 11.745943500 -6.641719523 11.121808677
        XstarFert -0.00609037  0.001859433 -0.005879305  0.001779358
      XstarPartic  0.11994156  0.028712245  0.115624068  0.027095855
        XstarColl  0.55155755  0.053070901  0.535866449  0.050053675
XstarMalTheftConv  0.03932477  0.023239818  0.039933442  0.021980373
```

The results are quite close to the OLS estimates with corrected SE's. The corrected OLS standard errors are quite similar to the plug-in GLS standard errors; and because ρ is not large in this problem, OLS $\hat{\beta}$'s are quite close to the GLS $\hat{\beta}$'s as well.

> *Note that this is* not *expected to be a great analysis, since we are not incorporating the uncertainty in estimating ρ into the estimates of the other parameters; in particular I would expect the standard errors of the regression coefficients to be underestimated in this approach.*

4.3 The GLS MLE analysis

A maximum likelihood analysis depends on maximizing the likelihood

$$\begin{aligned}\log L(\underline{y};\underline{\beta},\boldsymbol{\Sigma}) &= \log\left\{\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\underline{y}-X\underline{\beta})^T\boldsymbol{\Sigma}^{-1}(\underline{y}-X\underline{\beta})\right]\right\} \\ &= -\frac{n}{2}\log 2\pi-\frac{n}{2}\log\sigma_z^2-\frac{1}{2}\log(1-\rho^2)-\frac{1}{2\sigma_z^2}(\underline{y}^*-X^*\underline{\beta})^T(\underline{y}^*-X^*\underline{\beta})\end{aligned}$$

(where $\underline{y}^*=\Gamma\underline{y}$ and $X^*=\Gamma X$ and Γ is from equation 3) simultaneously in

$$\beta_0,\quad \beta_1,\quad \ldots,\quad \beta_k,\quad \sigma_z^2\quad \text{and}\quad \rho$$

For a problem with relatively few parameters like our analysis of female theft convictions in Canada, there is an Splus function `nlmin()` that can be used to maximize the likelihood above directly in all these parameters. The function `nlmin()` minimizes nonlinear functions in whatever parameters are appropriate. In this case, if we minimize something proportional to $-\log L(\underline{y};\underline{\beta},\boldsymbol{\Sigma})$ we will have maximized the likelihood.

First we need to set up some helper functions, and pre-compute some quantities that are repeatedly used in an `nlmin()` analysis. The following computations and function definition are in the file `gls-code.splus`. Note that the code is fairly specific to our theft conviction data.

```
# Set up for "ar1fun"

# attach(hart)  # (don't forget!)

olsfit _ lm(FemTheftConv ~ Fert + Partic + Coll + MalTheftConv)
X _ model.matrix(olsfit)
y _ FemTheftConv
n _ length(y)

# initial values for "nlmin"

rho _ acf(residuals(olsfit),plot=F)$acf[2,1,1]
sig2z _ sum(residuals(olsfit)^2)/n * (1-rho^2)
b.start _ c(coefficients(olsfit),rho,sig2z)

# "ar1fun" is proportional to -(log-likelihood) so by
# minimizing "ar1fun" we'll maximize the log-likelihood for
# regression with ar1 errors.

ar1fun _ function(b) {
#
# Here are the params:
# b[1] - beta0
# b[2] - beta1
# b[3] - beta2
# b[4] - beta3
# b[5] - beta4
# b[6] - rho
# b[7] - sig^2_z
#
# this assumes that "hart" is attached, and that
#
# olsfit _ lm(FemTheftConv ~ Fert + Partic + Coll + MalTheftConv))
# X _ model.matrix(olsfit)
# y _ FemTheftConv
# n _ length(y)
#
# have already been computed.
#
# WARNING: The following code is likely very inefficient.  But I bet it
# works!
#
        rho _ b[6]
        s2z _ b[7] #
#
        Gam _ diag(c(sqrt(1-rho^2),rep(1,length=n-1))) -
                rho*rbind(0,cbind(diag(n-1),0)) #
#
#       note that minimizing Q below maximizes the log-likelihood
#       for linear regression with AR(1) errors
#
        Q _ n/2 * log(s2z) + 1/2 * log(1-rho^2) +
                        1/(2*s2z) * sum((Gam%*%(y - X%*%b[1:5]))^2)
```

```
        return(Q)
}

#
# now try, e.g.
#
# nlmin(ar1fun,b.start,print.level=1)
# and help(nlmin)
#
#
```

After a little fooling around, the following seemed to work well

```
> result _ nlmin(ar1fun,b.start,print.level=1,max.fcal=100,max.iter=100)
    it    nf      f       reldf      preldf     reldx
     0     1  5.863e1
     1     7  5.863e1    1.970e-5   3.944e-5   4.587e-7
     2    10  5.863e1    3.560e-7   2.549e-6   7.485e-8
     3    15  5.861e1    2.649e-4   4.922e-4   2.014e-5

[many iterations omitted]

    30    60  5.822e1    2.414e-8   2.225e-8   6.693e-4
    31    61  5.822e1    4.016e-10  3.786e-10  1.048e-4
    32    62  5.822e1    2.945e-12  2.864e-12  5.495e-6
 ***** relative function convergence *****
 function     5.821717e1    reldx        5.4952e-6
 function evaluations:  62   gradient evaluations: 303
 preldf       2.8637e-12  npreldf      2.8637e-12

> result
$x:
[1]  -5.079305725  -0.005753718   0.110158212   0.520073564   0.039469484
[6]   0.398738667  11.354708284

$converged:
[1] T

$conv.type:
[1] "relative function convergence"
```

We can directly calculate standard errors using the formula $\mathrm{Var}\,(\underline{\hat{\beta}}_{GLS}) = (X^T \boldsymbol{\Sigma}^{-1} X)^{-1}$ from Equation (2) and compare with our earlier plug-in estimates.

```
> betaGLS _ result$x[1:5]
> rhoGLS _ result$x[6]
> s2zGLS _ result$x[7]
```

```
> s2eGLS _ s2zGLS/(1-rhoGLS^2)
> s2eGLS
[1] 13.50132
> rho _ rhoGLS
> Gam _ diag(c(sqrt(1-rho^2),rep(1,length=n-1))) -
+       rho*rbind(0,cbind(diag(n-1),0))
> VCI _ t(Gam)%*%Gam/s2eGLS
> VarBetaGLS _ solve(t(X)%*%VCI%*%X)
> ests _ cbind(cbind(summary(olsfit)$coefficients[,1],sqrt(diag(VarBeta))),
+ summary(glsfit1)$coefficients[,1:2],betaGLS,sqrt(diag(VarBetaGLS)))
> dimnames(ests) _ list(c("Intercept","Fert","Partic","Coll","MalTheftConv"),
+ c("OLS","SE(corrected)","P-I GLS","P-I SE","MLE GLS","MLE SE"))
> round(ests,4)
                 OLS SE(corrected) P-I GLS  P-I SE MLE GLS  MLE SE
   Intercept -7.3341       11.7459 -6.6417 11.1218 -5.0793 12.8435
        Fert -0.0061        0.0019 -0.0059  0.0018 -0.0058  0.0022
      Partic  0.1199        0.0287  0.1156  0.0271  0.1102  0.0304
        Coll  0.5516        0.0531  0.5359  0.0501  0.5201  0.0569
MalTheftConv  0.0393        0.0232  0.0399  0.0220  0.0395  0.0253
```

In every case, as expected, the SE's from the maximum-likelihood version of GLS are larger than either the "corrected" OLS SE's or the plug-in GLS SE's.

How do we assess the success of a GLS fit? The easiest residuals to look at are

$$\underline{E} = \underline{y}^* - \underline{X}^* \underline{\hat{\beta}}_{GLS} = \Gamma(\underline{y} - X\underline{\hat{\beta}}_{GLS})$$

since they are supposed to be estimating uncorrelated, equal-variance $N(0, \sigma_\epsilon^2)$. Note that these are the "usual" residuals that we would "expect", but tilted by Γ so as to be estimating uncorrelated equal-variance quantities.

```
> yhat _ X%*%betaGLS[1:5]
> resids _ Gam %*% (y - yhat)
> par(mfrow=c(2,2))
> plot(yhat,resids)
> abline(h=0)
> plot(resids)
> abline(h=0)
> plot(yhat,y)
> abline(0,1)
> qqenv(resids)
```

Some potentially useful plots are in Figure 5. Note that the y vs $\hat{y}$ plot in the lower left displays essentially $\underline{y} - X\underline{\hat{\beta}}_{GLS}$ instead of $\Gamma(\underline{y} - X\underline{\hat{\beta}}_{GLS})$. Quantities like standardized residuals and Cook's D's could also be calculated, and might well be useful, but these plots illustrate the general ideas.

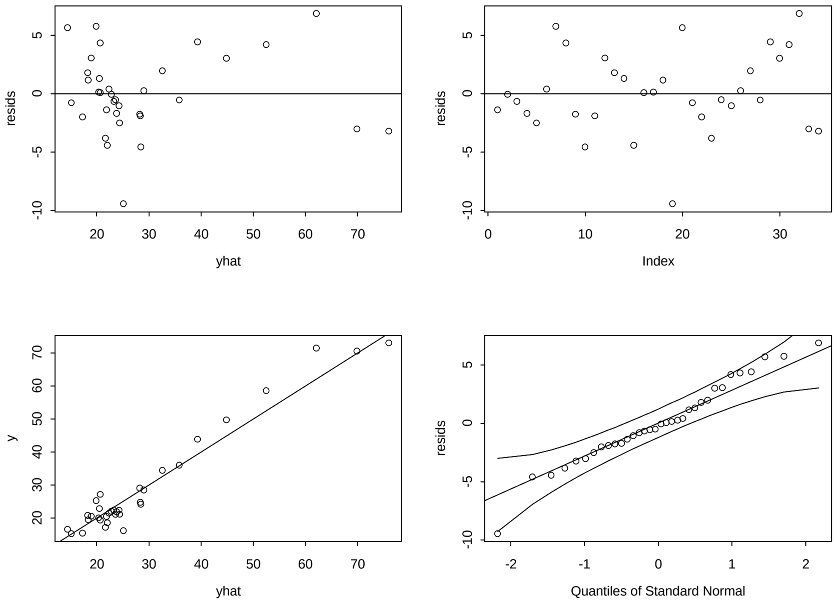

Figure 5: Some uncorrelated raw-residual diagnostic plots for the MLE GLS fit `FemTheftConv ~ Fert + Partic + Coll + MalTheftConv`.

5 Some Closing Remarks

GLS is a useful general technique for solving linear regression problems when the correlations structure of the regression errors is not too complicated. Maximum likelihood GLS is a reasonable approach to fitting linear regression models with time series structured errors.

For problems with many more parameters, `nlmin()` may need to be replaced with a method you code yourself. Two common methods are

- Set up a grid of proposed correlations, say, $\rho_1, \rho_2, \ldots, \rho_M$, calculate $\underline{\hat{\beta}}_m$ for each ρ_m and choosing the $\underline{\hat{\beta}}_m$; or
- Iterate back and forth between estimating $\underline{\beta}$ from the current guess for ρ, and re-estimating ρ by applying formula (4) to the new set of residuals obtained.

(Neither of the above are the best possible approach but they do illustrate valid approaches to the joint-MLE method). The second method above is closely related to a general technique for maximizing likelihoods (and posterior distributions) over different "types" of parameters (like regression coefficients, vs. error correlations), called the E-M ("expectation-maximization") algorithm.

GLS and related methods also turn out to be useful in nonlinear optimization problems, since many nonlinear problems can be converted into linear ones by application of Newton's method and/or Taylor's Theorem. One version of this, which we may see later, is called *iteratively reweighted least squares* (IRLS).