Day 15 - Time Series Regression

Introduction

Today we learn how to formally fit a time series regression model by combining a regression model with a serially correlated error process.

Review

The code below fits a time series regression model to the `AirPassengers` series before the year 1960 that models the log count of air passengers by the interaction of month and time index and time index squared.

```
# data
data("AirPassengers")
ap <- AirPassengers
ap_tbl <- tibble(
  ap = c(ap), year = rep(1949:1960, each = 12),
  month = rep(1:12, 12) %>% factor()
) %>% mutate(t = 1:n(), t2 = t^2)  %>%
  mutate(t_scaled = c(scale(t)), t2_scaled = c(scale(t2))) %>%
  mutate(log_ap = log(ap))
ap_sub_tbl <- ap_tbl %>% filter(year < 1960)

# fit model
ols_fit <- lm(log_ap ~ t_scaled*month + t2_scaled , ap_sub_tbl)
confint(ols_fit)
```

```
                     2.5 %       97.5 %
(Intercept)      5.426753311  5.479543166
t_scaled         0.519721230  0.599947224
month2          -0.066680046  0.007406934
month3           0.074633669  0.148484132
month4           0.035882116  0.109519670
month5           0.038696436  0.112144703
month6           0.164088564  0.237371178
month7           0.265642481  0.338783085
month8           0.257952003  0.330974252
month9           0.112863511  0.185791066
month10         -0.028078341  0.044778188
month11         -0.168833980 -0.096024805
month12         -0.053425390  0.019360108
t2_scaled       -0.190308489 -0.119000733
t_scaled:month2 -0.086936582 -0.008717988
t_scaled:month3 -0.071088392  0.007147608
t_scaled:month4 -0.059017789  0.019247211
t_scaled:month5 -0.024461829  0.053843753
t_scaled:month6 -0.005279921  0.073077807
t_scaled:month7  0.007916636  0.086338052
t_scaled:month8  0.010140779  0.088637395
t_scaled:month9 -0.027578677  0.051004618
```

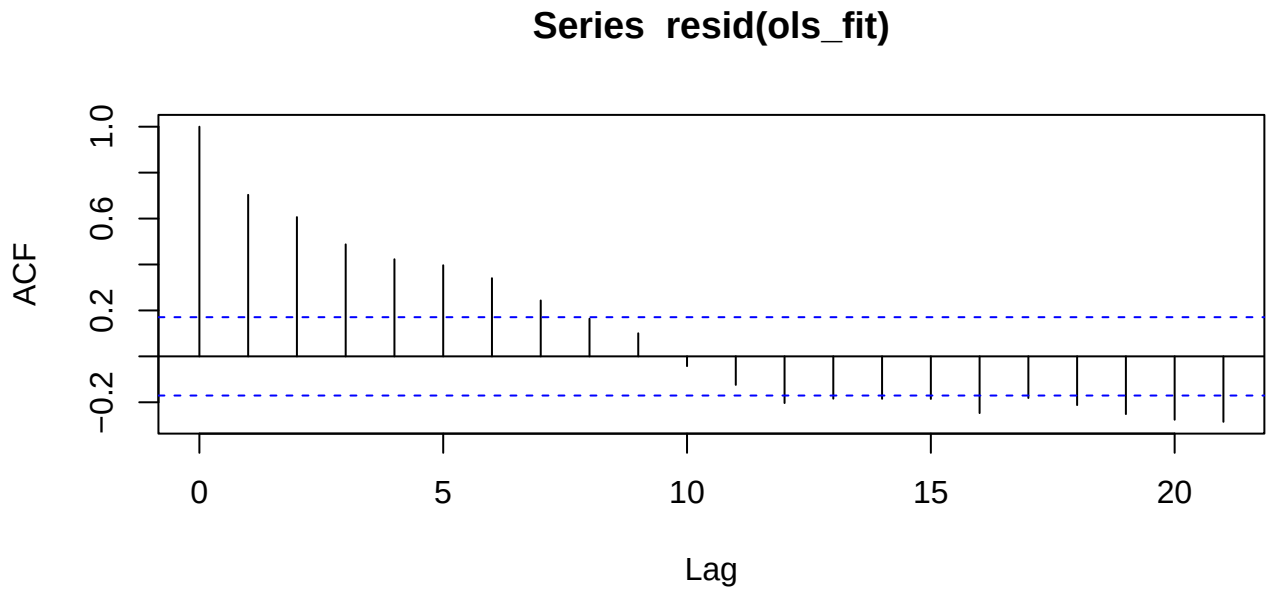
```
t_scaled:month10 -0.024460668  0.054220749
t_scaled:month11 -0.021728303  0.057062634
t_scaled:month12 -0.038724557  0.040187253
```

```
summary(ols_fit)
```

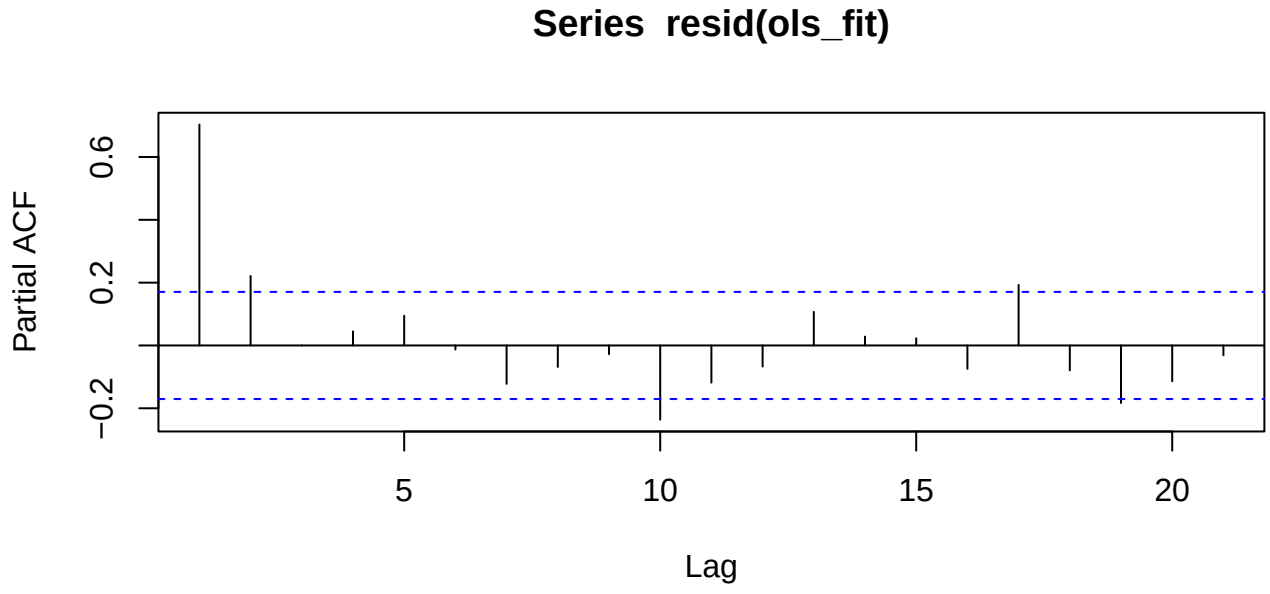
```
Call:
lm(formula = log_ap ~ t_scaled * month + t2_scaled, data = ap_sub_tbl)

Residuals:
      Min        1Q    Median        3Q       Max
-0.135965 -0.026936  0.002268  0.029569  0.093082

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.4531482  0.0133147 409.557  < 2e-16 ***
t_scaled          0.5598342  0.0202347  27.667  < 2e-16 ***
month2           -0.0296366  0.0186863  -1.586  0.11569
month3            0.1115589  0.0186267   5.989 2.87e-08 ***
month4            0.0727009  0.0185730   3.914  0.00016 ***
month5            0.0754206  0.0185252   4.071 8.98e-05 ***
month6            0.2007299  0.0184835  10.860  < 2e-16 ***
month7            0.3022128  0.0184476  16.382  < 2e-16 ***
month8            0.2944631  0.0184178  15.988  < 2e-16 ***
month9            0.1493273  0.0183939   8.118 8.68e-13 ***
month10           0.0083499  0.0183760   0.454  0.65047
month11          -0.1324294  0.0183641  -7.211 8.28e-11 ***
month12          -0.0170326  0.0183581  -0.928  0.35560
t2_scaled        -0.1546546  0.0179854  -8.599 7.39e-14 ***
t_scaled:month2  -0.0478273  0.0197284  -2.424  0.01701 *
t_scaled:month3  -0.0319704  0.0197328  -1.620  0.10814
t_scaled:month4  -0.0198853  0.0197401  -1.007  0.31604
t_scaled:month5   0.0146910  0.0197504   0.744  0.45861
t_scaled:month6   0.0338989  0.0197635   1.715  0.08920 .
t_scaled:month7   0.0471273  0.0197796   2.383  0.01895 *
t_scaled:month8   0.0493891  0.0197985   2.495  0.01414 *
t_scaled:month9   0.0117130  0.0198204   0.591  0.55580
t_scaled:month10  0.0148800  0.0198452   0.750  0.45502
t_scaled:month11  0.0176672  0.0198728   0.889  0.37599
t_scaled:month12  0.0007313  0.0199033   0.037  0.97076
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04209 on 107 degrees of freedom
```

```
Multiple R-squared:  0.9916,    Adjusted R-squared:  0.9898
F-statistic: 529.4 on 24 and 107 DF,  p-value: < 2.2e-16
```

```
acf(resid(ols_fit))
```




```
pacf(resid(ols_fit))
```



Generalized least squares overview

GLS theory

Suppose we extend the typical _____ model to accomodate violations of independence and constant variance. That is, let

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \epsilon_i$$

where ϵ_i is distributed $N(0, \sigma_i^2)$ and may not be indendent. In matrix notation, the above model is equivalent to

$$y = X\beta + \epsilon$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

By properties of normal distributions, the above model is equivalent to

$$y \sim \mathcal{N}(X\beta, \Sigma)$$

where

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{nn} \end{bmatrix}$$

and $\Sigma = \Sigma^\top$. In its general form, we cannot _____ because there are more parameters than observations (estimating Σ requires estimating $n + \frac{n(n-1)}{2}$ parameters). Therefore, to estimate Σ, we must assume that some structure exists. An example of such structure is an _____ process, where:

$$\Sigma = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \dots & \rho^{n-1} \\ \rho & 1 & \rho & \rho^2 & \dots & \rho^{n-2} \\ \rho & \rho^2 & 1 & \rho & \dots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho^2 & \rho^3 & \rho^4 & \dots & 1 \end{bmatrix}$$

The _____ estimate of β is

$$\hat{\beta}_{gls} = \left(X^\top \Omega^{-1} X\right)^{-1} X^\top \Omega^{-1} y$$

Estimating the other model parameters (ex. σ^2 and ρ) can be quite difficult and typically uses a process called restricted maximum likelihood.

GLS in R

> **Note**
>
> To implement GLS in R, we use the `gls` function in the `nlme` package.

```
library(nlme)
gls_fit <- gls(
  log_ap ~ t_scaled*month + t2_scaled, correlation = corARMA(p = 1, q = 0), ap_sub_tbl
)
summary(gls_fit)
```

```
Generalized least squares fit by REML
  Model: log_ap ~ t_scaled * month + t2_scaled
  Data: ap_sub_tbl
        AIC       BIC   logLik
  -341.4929 -269.3265 197.7465

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
      Phi
0.7380715

Coefficients:
                     Value  Std.Error  t-value p-value
(Intercept)       5.457462 0.01355812 402.5234  0.0000
t_scaled          0.544777 0.03989347  13.6558  0.0000
month2           -0.030474 0.00982388  -3.1020  0.0025
month3            0.110125 0.01278218   8.6155  0.0000
month4            0.070854 0.01444218   4.9061  0.0000
month5            0.073307 0.01540620   4.7583  0.0000
month6            0.198472 0.01590808  12.4762  0.0000
month7            0.299927 0.01605132  18.6855  0.0000
month8            0.292268 0.01586890  18.4177  0.0000
month9            0.147360 0.01533800   9.6075  0.0000
month10           0.006779 0.01436948   0.4718  0.6381
month11          -0.133382 0.01275730 -10.4554  0.0000
month12          -0.017068 0.00998168  -1.7100  0.0902
t2_scaled        -0.132010 0.04300622  -3.0695  0.0027
t_scaled:month2  -0.049169 0.01034488  -4.7530  0.0000
t_scaled:month3  -0.034434 0.01348176  -2.5541  0.0121
t_scaled:month4  -0.023328 0.01526843  -1.5279  0.1295
t_scaled:month5   0.010349 0.01634002   0.6334  0.5279
```

```
t_scaled:month6   0.028683 0.01694382   1.6928  0.0934
t_scaled:month7   0.041009 0.01718993   2.3857  0.0188
t_scaled:month8   0.042285 0.01711417   2.4707  0.0151
t_scaled:month9   0.003475 0.01669366   0.2082  0.8355
t_scaled:month10  0.005283 0.01583611   0.3336  0.7393
t_scaled:month11  0.006389 0.01432981   0.4459  0.6566
t_scaled:month12 -0.012679 0.01166263  -1.0872  0.2794

 Correlation:
                 (Intr) t_scld month2 month3 month4 month5 month6 month7 month8
t_scaled          0.003
month2           -0.341 -0.068
month3           -0.449 -0.085  0.654
month4           -0.513 -0.092  0.501  0.751
month5           -0.551 -0.095  0.414  0.607  0.791
month6           -0.573 -0.094  0.357  0.514  0.656  0.811
month7           -0.582 -0.090  0.316  0.447  0.561  0.679  0.819
month8           -0.579 -0.085  0.284  0.397  0.489  0.582  0.687  0.820
month9           -0.563 -0.078  0.256  0.354  0.431  0.504  0.584  0.683  0.814
month10          -0.532 -0.067  0.229  0.313  0.377  0.435  0.496  0.568  0.663
month11          -0.481 -0.052  0.197  0.268  0.320  0.365  0.410  0.462  0.529
month12          -0.396 -0.026  0.156  0.210  0.248  0.280  0.311  0.345  0.387
t2_scaled         0.075 -0.938  0.038  0.047  0.050  0.051  0.049  0.046  0.042
t_scaled:month2  -0.071 -0.124  0.268  0.162  0.115  0.090  0.074  0.063  0.056
t_scaled:month3  -0.095 -0.155  0.175  0.248  0.172  0.130  0.104  0.088  0.076
t_scaled:month4  -0.110 -0.166  0.135  0.187  0.229  0.168  0.131  0.108  0.092
t_scaled:month5  -0.121 -0.165  0.111  0.152  0.183  0.211  0.160  0.127  0.106
t_scaled:month6  -0.129 -0.158  0.096  0.130  0.154  0.174  0.194  0.149  0.120
t_scaled:month7  -0.134 -0.146  0.085  0.114  0.134  0.150  0.164  0.177  0.138
t_scaled:month8  -0.139 -0.129  0.077  0.103  0.119  0.132  0.143  0.153  0.162
t_scaled:month9  -0.143 -0.110  0.071  0.094  0.108  0.119  0.128  0.135  0.142
t_scaled:month10 -0.146 -0.086  0.067  0.088  0.101  0.110  0.117  0.123  0.129
t_scaled:month11 -0.152 -0.057  0.064  0.084  0.096  0.104  0.110  0.116  0.120
t_scaled:month12 -0.167 -0.018  0.064  0.084  0.096  0.105  0.111  0.117  0.122
                 month9 mnth10 mnth11 mnth12 t2_scl t_sc:2 t_sc:3 t_sc:4 t_sc:5
t_scaled
month2
month3
month4
month5
month6
month7
month8
month9
month10           0.797
```

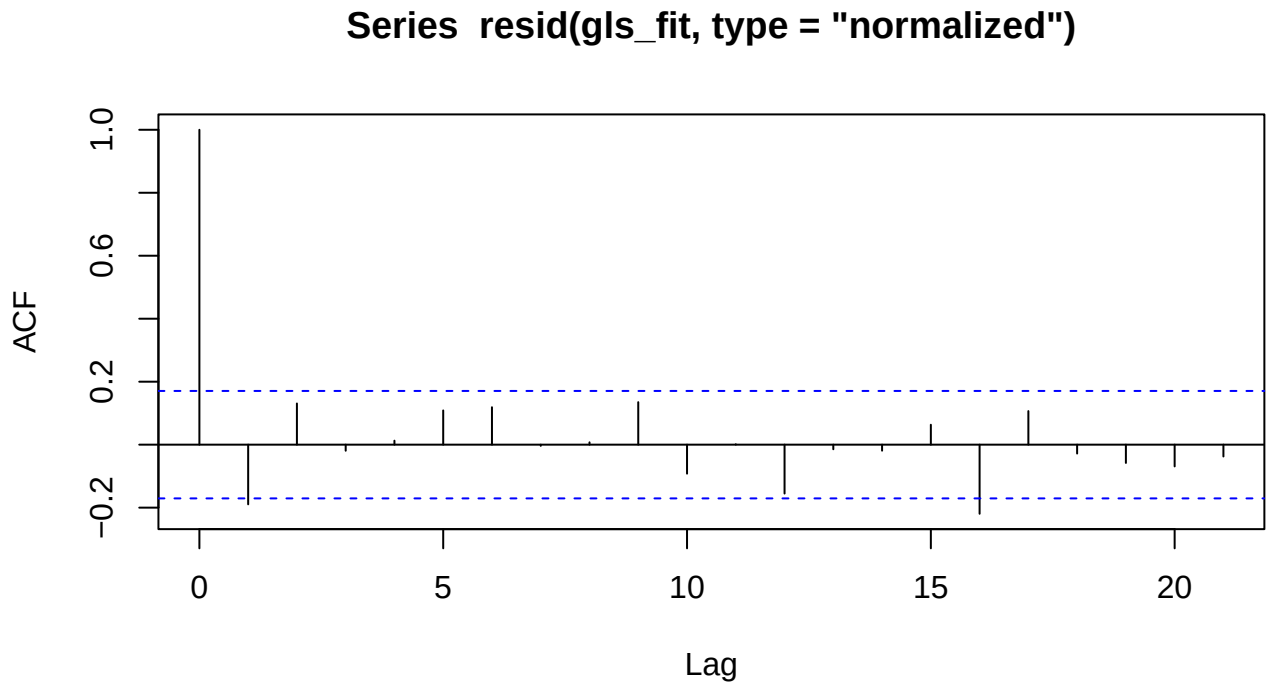
```
month11           0.621  0.761
month12           0.444  0.531  0.678
t2_scaled         0.035  0.025  0.009 -0.023
t_scaled:month2   0.050  0.047  0.045  0.048  0.010
t_scaled:month3   0.068  0.063  0.060  0.064  0.003  0.649
t_scaled:month4   0.081  0.074  0.070  0.075 -0.008  0.493  0.745
t_scaled:month5   0.091  0.082  0.077  0.083 -0.022  0.403  0.597  0.785
t_scaled:month6   0.101  0.089  0.084  0.089 -0.038  0.344  0.500  0.645  0.804
t_scaled:month7   0.113  0.097  0.089  0.095 -0.054  0.303  0.433  0.547  0.668
t_scaled:month8   0.128  0.106  0.096  0.101 -0.071  0.270  0.381  0.474  0.569
t_scaled:month9   0.148  0.118  0.104  0.108 -0.087  0.243  0.339  0.415  0.490
t_scaled:month10  0.133  0.137  0.115  0.117 -0.102  0.217  0.300  0.364  0.423
t_scaled:month11  0.125  0.129  0.134  0.133 -0.116  0.189  0.260  0.312  0.359
t_scaled:month12  0.127  0.134  0.144  0.168 -0.134  0.156  0.213  0.254  0.289
                 t_sc:6 t_sc:7 t_sc:8 t_sc:9 t_s:10 t_s:11
t_scaled
month2
month3
month4
month5
month6
month7
month8
month9
month10
month11
month12
t2_scaled
t_scaled:month2
t_scaled:month3
t_scaled:month4
t_scaled:month5
t_scaled:month6
t_scaled:month7   0.813
t_scaled:month8   0.678  0.815
t_scaled:month9   0.573  0.675  0.810
t_scaled:month10  0.487  0.562  0.660  0.797
t_scaled:month11  0.407  0.462  0.531  0.626  0.767
t_scaled:month12  0.323  0.360  0.406  0.466  0.555  0.701

Standardized residuals:
        Min          Q1         Med          Q3         Max
-3.02816016 -0.59747766 -0.05989898  0.70737318  2.22324088

Residual standard error: 0.04365054
```

```
Degrees of freedom: 132 total; 107 residual
```

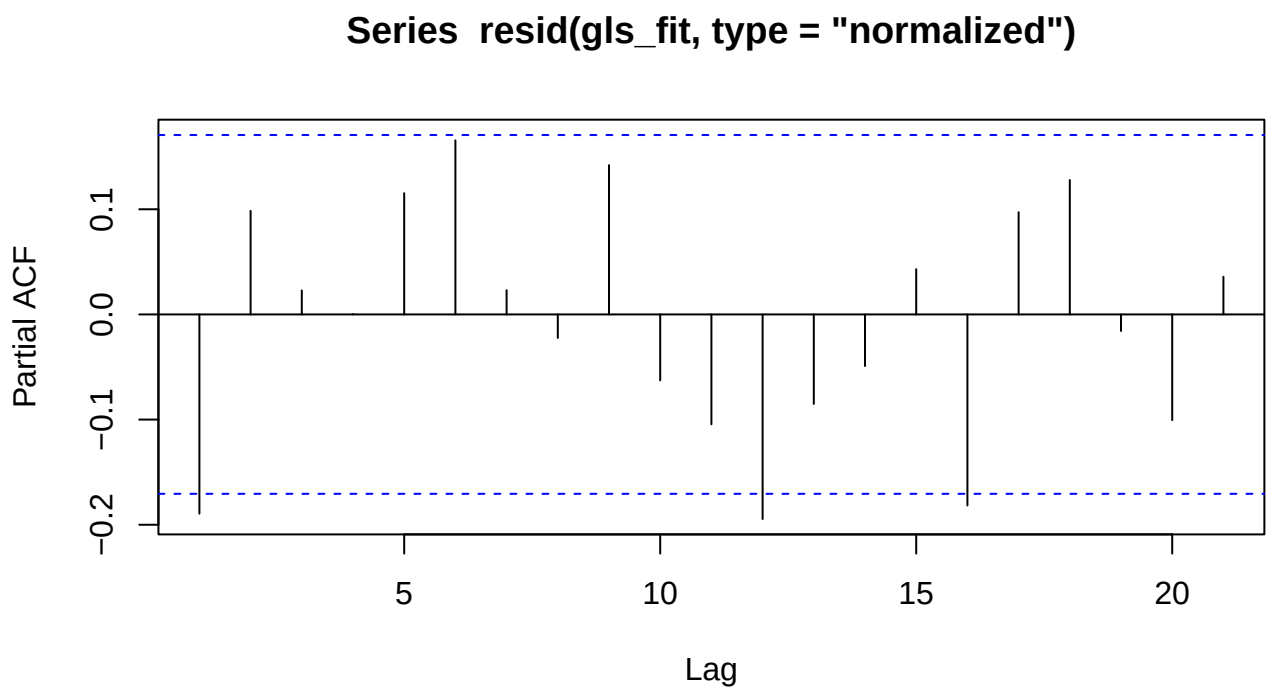
```r
confint(gls_fit)
```

```
                       2.5 %       97.5 %
(Intercept)       5.430888586  5.484035457
t_scaled          0.466587715  0.622967260
month2           -0.049728146 -0.011219260
month3            0.085072746  0.135177981
month4            0.042548180  0.099160495
month5            0.043111089  0.103502283
month6            0.167293109  0.229651636
month7            0.268466828  0.331386840
month8            0.261165518  0.323370448
month9            0.117297637  0.177421496
month10          -0.021384665  0.034942671
month11          -0.158385939 -0.108378248
month12          -0.036632058  0.002495389
t2_scaled        -0.216300278 -0.047719009
t_scaled:month2  -0.069444582 -0.028893387
t_scaled:month3  -0.060858162 -0.008010649
t_scaled:month4  -0.053254007  0.006597144
t_scaled:month5  -0.021676860  0.042374833
t_scaled:month6  -0.004526277  0.061892287
t_scaled:month7   0.007317562  0.074700834
t_scaled:month8   0.008741595  0.075827904
t_scaled:month9  -0.029244067  0.036193889
t_scaled:month10 -0.025754737  0.036321675
t_scaled:month11 -0.021696782  0.034475055
t_scaled:month12 -0.035537592  0.010179075
```

```r
# normalized residuals account for the estimate serial correlation
acf(resid(gls_fit, type = "normalized"))
```




```
pacf(resid(gls_fit, type = "normalized"))
```



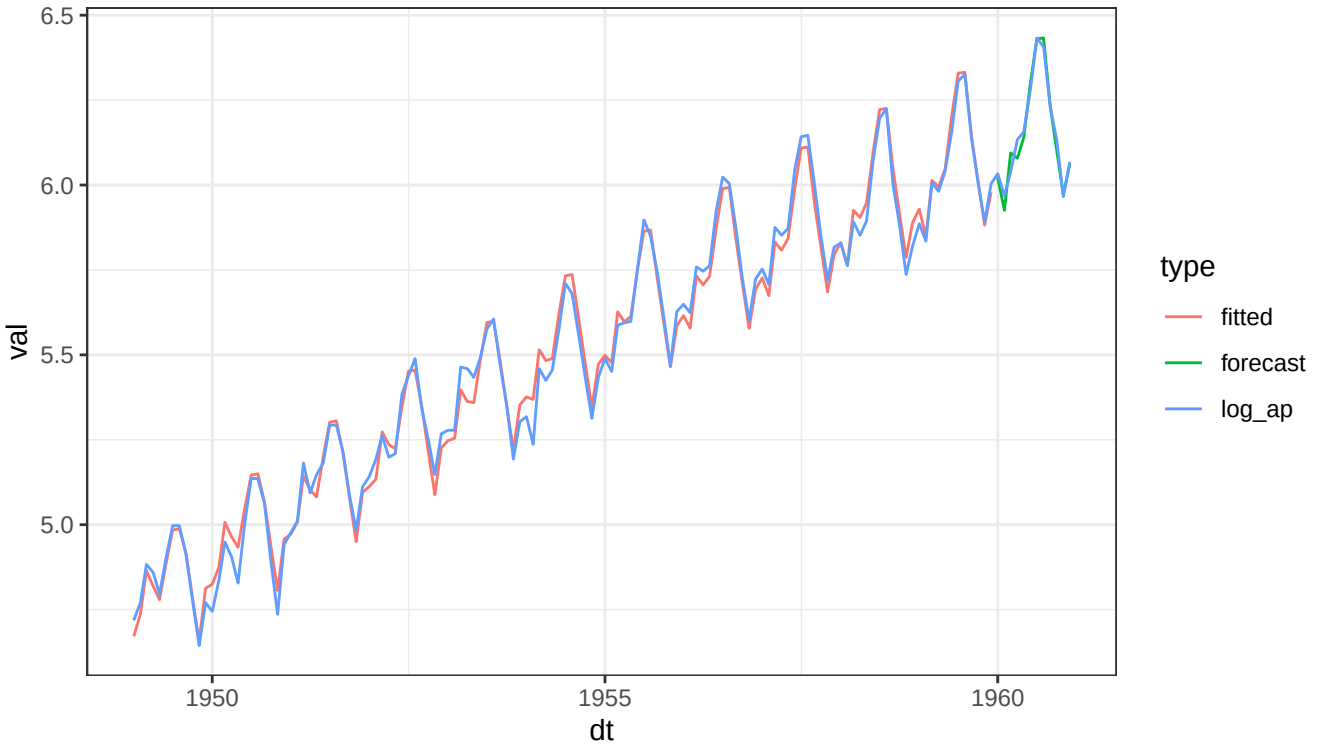
Forecasts with GLS

> **Note**
>
> To obtain forecasts, we use the `predict` function.

```
# create table for prediction
pred_tbl <- ap_tbl %>%
  filter(year >= 1960)

# obtain prediction
## note: you cannot obtain SEs!!!
(pred <- predict(gls_fit, pred_tbl))
```

```
 [1] 6.021277 5.925726 6.094597 6.078738 6.139993 6.301442 6.430439 6.432989
 [9] 6.232481 6.102074 5.970943 6.061718
attr(,"label")
[1] "Predicted values"
```

```
# plot them
ap_tbl %>%
  mutate(
    fitted = c(fitted(gls_fit), rep(NA, 12)),
    forecast = c(rep(NA, 132), pred)
  ) %>%
  pivot_longer(log_ap:forecast, names_to = "type", values_to = "val") %>%
  mutate(dt = ymd(paste0(year, "-", month, "-1"))) %>%
  ggplot() +
  geom_line(aes(x = dt, y = val, col = type)) +
  theme_bw()
```



Important

Unfortunately, there is no closed form solution for the _____ in a GLS model. Therefore, to obtain estimates of uncertainty in our forecasts from a GLS model, you must:

- Figure out how to bootstrap the SE of the prediction (disgusting)
- Use the Delta method (somehow more disgusting)
- Fit a Bayesian model

Why use GLS?

GLS is most useful for making inference about regression coefficients with complicated correlation structures (GLS can accommodate hierarchical models, time series models, longitudinal models, and any combination of the three). GLS allows us to adjust the SEs of the coefficient estimates and obtain confidence intervals and p-values that account for the serial correlation that is present.

State-space model using `arima`

State-space models

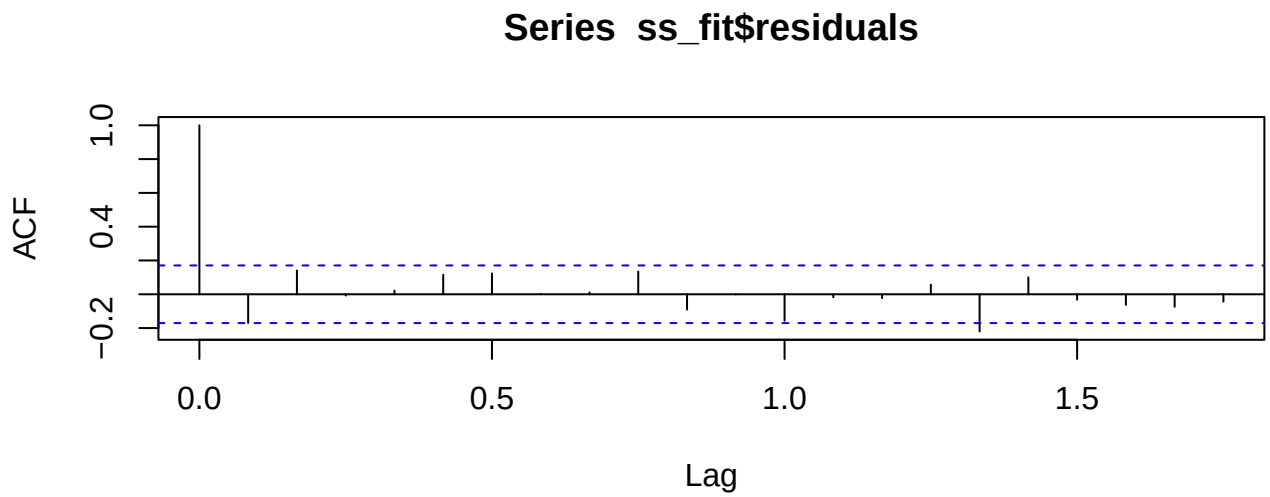
The `arima` function offers a way to fit time series regression models and obtain estimates of the _____ using a _____ representation of the model. The details are beyond the scope of this class, but representing the model in this way allows the model to be estimated using a Kalman-filter, which enables estimates of the _____.

```
# prepare a few things
ts <- ts(
  ap_sub_tbl$log_ap,
  start = c(1949, 1),
  freq = 12
)

ss_fit <- arima(
  x = ts,
  order = c(1, 0, 0),
  xreg = model.matrix(
    ~ t_scaled*month + t2_scaled,
    ap_sub_tbl
  ),
  include.mean = F
)
ss_fit
```

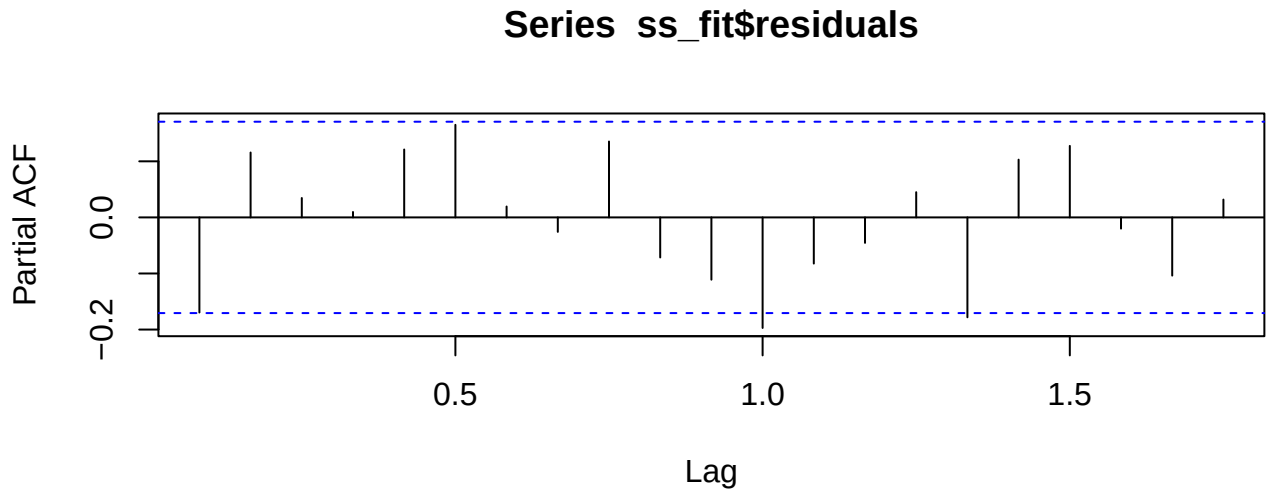
```
Call:
arima(x = ts, order = c(1, 0, 0), xreg = model.matrix(~t_scaled * month + t2_scaled,
    ap_sub_tbl), include.mean = F)

Coefficients:
         ar1  (Intercept)  t_scaled   month2  month3  month4  month5  month6
      0.7210       5.4573    0.5461  -0.0305  0.1101  0.0708  0.0733  0.1984
s.e.  0.0611       0.0119    0.0344   0.0089  0.0115  0.0130  0.0138  0.0143
      month7  month8  month9  month10  month11  month12  t2_scaled
      0.2999  0.2922  0.1473   0.0067  -0.1334  -0.0171    -0.1335
s.e.  0.0144  0.0142  0.0138   0.0129   0.0115   0.0090     0.0371
      t_scaled:month2  t_scaled:month3  t_scaled:month4  t_scaled:month5
              -0.0492          -0.0344          -0.0233           0.0104
s.e.           0.0094           0.0122           0.0137           0.0147
      t_scaled:month6  t_scaled:month7  t_scaled:month8  t_scaled:month9
               0.0288           0.0412           0.0426           0.0038
```

```
s.e.           0.0152           0.0154           0.0154           0.0150
      t_scaled:month10  t_scaled:month11  t_scaled:month12
                0.0056            0.0068           -0.0124
s.e.            0.0143            0.0130            0.0106

sigma^2 estimated as 0.000703:  log likelihood = 291.5,  aic = -529.01
```

```
acf(ss_fit$residuals)
```




```
pacf(ss_fit$residuals)
```



Forecasts with state-space models

Forecasts with state-space models

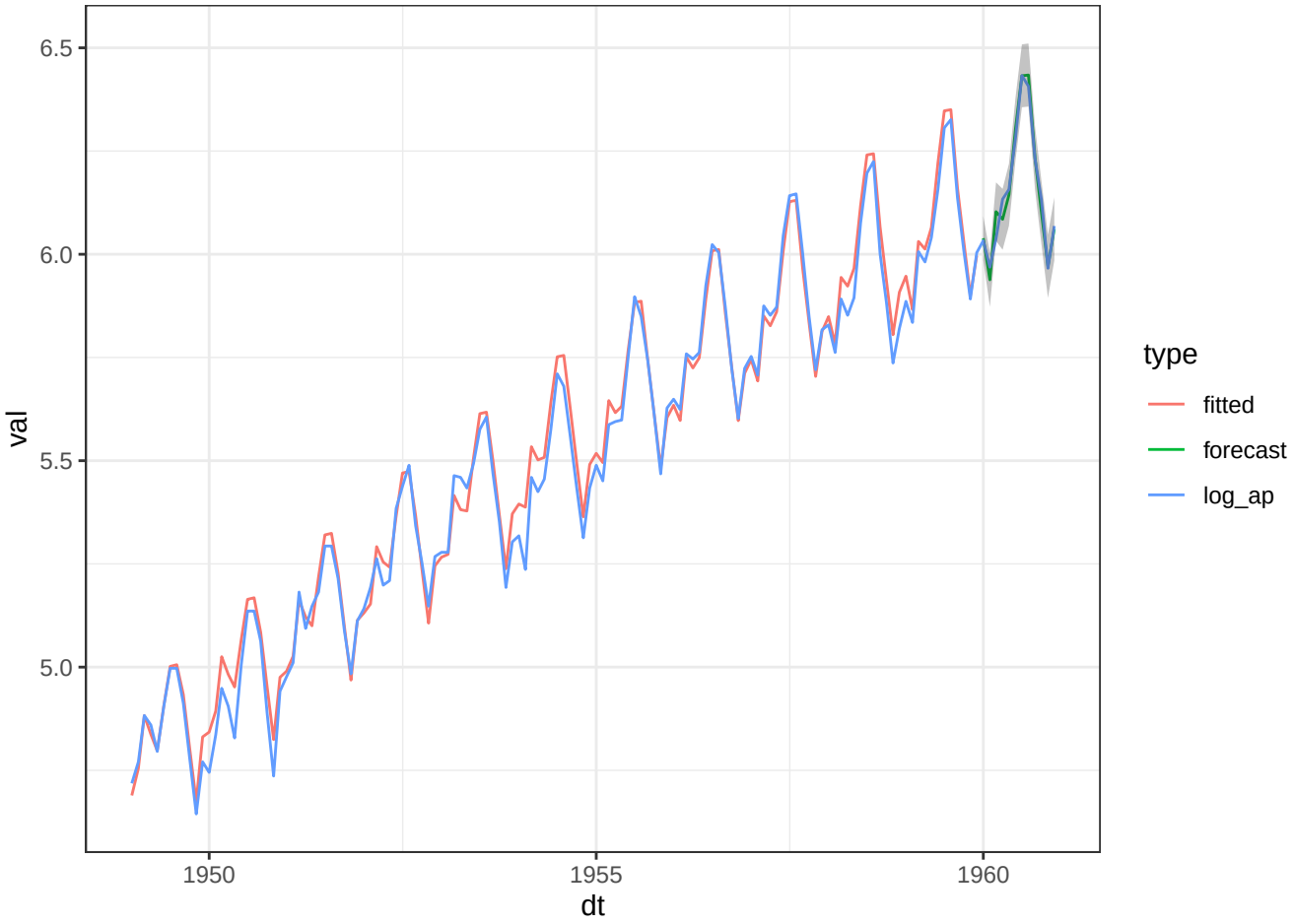
To obtain forecasts, we again use the `predict` function, specifying the number of time points ahead (`n.ahead`) and the matrix of new regression coefficients (`newxreg`).

```
# forecasts
ss_fitted <- predict(
  ss_fit,
  newxreg = model.matrix(
    ~ t_scaled*month + t2_scaled,
    ap_sub_tbl
  )
)

(ss_pred <- predict(
  ss_fit, n.ahead = 12,
  newxreg = model.matrix(
    ~ t_scaled*month + t2_scaled,
    ap_tbl %>% filter(year >= 1960)
  )
))
```

```
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1960 6.038872 5.938064 6.103185 6.084665 6.144054 6.304165 6.432193 6.434102
          Sep      Oct      Nov      Dec
1960 6.233129 6.102359 5.970941 6.061428

$se
             Jan         Feb         Mar         Apr         May         Jun
1960 0.02651422 0.03268772 0.03547530 0.03684129 0.03753181 0.03788583
             Jul         Aug         Sep         Oct         Nov         Dec
1960 0.03806858 0.03816325 0.03821237 0.03823788 0.03825114 0.03825803
```

```
# plot
ap_tbl %>%
  mutate(
    fitted = c(ss_fitted$pred, rep(NA, 12)),
    forecast = c(rep(NA, 132), ss_pred$pred)
  ) %>%
  pivot_longer(log_ap:forecast, names_to = "type", values_to = "val") %>%
```

```
  mutate(dt = ymd(paste0(year, "-", month, "-1"))) %>%
  ggplot() +
  geom_line(aes(x = dt, y = val, col = type)) +
  geom_ribbon(
    data = tibble(
      dt = ymd(paste0(rep(1960, 12), "-", 1:12, "-1")),
      lwr = c(ss_pred$pred - 2*ss_pred$se),
      upr = c(ss_pred$pred + 2*ss_pred$se),
      type = "forecast"
    ),
    aes(x = dt, ymin = lwr, ymax = upr),
    alpha = .30
  ) +
  theme_bw()
```



Some comparisons

```
tibble(
  coef = names(coef(ols_fit)),
  ols = coef(ols_fit),
  ols_se = summary(ols_fit)$coefficients[,2],
  gls = coef(gls_fit),
  gls_se = sqrt(diag(summary(gls_fit)$varBeta)),
  ss = ss_fit$coef[-1],
  ss_se = sqrt(diag(ss_fit$var.coef))[-1]
) %>%
  print(n = "all")
```

```
# A tibble: 25 x 7
   coef                   ols ols_se      gls  gls_se       ss   ss_se
   <chr>                <dbl>  <dbl>    <dbl>   <dbl>    <dbl>   <dbl>
 1 (Intercept)       5.45     0.0133  5.46    0.0136   5.46    0.0119
 2 t_scaled          0.560    0.0202  0.545   0.0399   0.546   0.0344
 3 month2           -0.0296   0.0187 -0.0305  0.00982 -0.0305  0.00890
 4 month3            0.112    0.0186  0.110   0.0128   0.110   0.0115
 5 month4            0.0727   0.0186  0.0709  0.0144   0.0708  0.0130
 6 month5            0.0754   0.0185  0.0733  0.0154   0.0733  0.0138
 7 month6            0.201    0.0185  0.198   0.0159   0.198   0.0143
 8 month7            0.302    0.0184  0.300   0.0161   0.300   0.0144
 9 month8            0.294    0.0184  0.292   0.0159   0.292   0.0142
10 month9            0.149    0.0184  0.147   0.0153   0.147   0.0138
11 month10           0.00835  0.0184  0.00678 0.0144   0.00673 0.0129
12 month11          -0.132    0.0184 -0.133   0.0128  -0.133   0.0115
13 month12          -0.0170   0.0184 -0.0171  0.00998 -0.0171  0.00903
14 t2_scaled        -0.155    0.0180 -0.132   0.0430  -0.134   0.0371
15 t_scaled:month2  -0.0478   0.0197 -0.0492  0.0103  -0.0492  0.00937
16 t_scaled:month3  -0.0320   0.0197 -0.0344  0.0135  -0.0344  0.0122
17 t_scaled:month4  -0.0199   0.0197 -0.0233  0.0153  -0.0233  0.0137
18 t_scaled:month5   0.0147   0.0198  0.0103  0.0163   0.0104  0.0147
19 t_scaled:month6   0.0339   0.0198  0.0287  0.0169   0.0288  0.0152
20 t_scaled:month7   0.0471   0.0198  0.0410  0.0172   0.0412  0.0154
21 t_scaled:month8   0.0494   0.0198  0.0423  0.0171   0.0426  0.0154
22 t_scaled:month9   0.0117   0.0198  0.00347 0.0167   0.00380 0.0150
23 t_scaled:month10  0.0149   0.0198  0.00528 0.0158   0.00564 0.0143
24 t_scaled:month11  0.0177   0.0199  0.00639 0.0143   0.00675 0.0130
25 t_scaled:month12  0.000731 0.0199 -0.0127  0.0117  -0.0124  0.0106
```