CSCI E-106:Assignment 9 Solutions

Ian Kelk

Contents


```
# Ensure required packages are installed, then load them
options(repos = c(CRAN = "https://cloud.r-project.org"))

req_pkgs <- c(
  "knitr", "readr", "dplyr", "ggplot2", "scales",
  "lmtest", "broom", "kableExtra", "tibble",
  "ARDL", "urca"
)

to_install <- setdiff(req_pkgs, rownames(installed.packages()))
if (length(to_install)) {
  install.packages(to_install, dependencies = TRUE)
}

# Load quietly
invisible(lapply(req_pkgs, function(p) {
  suppressPackageStartupMessages(library(p, character.only = TRUE))
}))
```

Due Date: November 14, 2025 at 11:59 pm EST

Instructions

Students should submit their reports on Canvas. The report needs to clearly state what question is being solved, step-by-step walk-through solutions, and final answers clearly indicated. Please solve by hand where appropriate.

Please submit two files: (1) a R Markdown file (.Rmd extension) and (2) a PDF document, word, or html generated using knitr for the .Rmd file submitted in (1) where appropriate. Please, use RStudio Cloud for your solutions.

Problem 1

Refer to the Advertising Agency data. The managing partner of an advertising agency is interested in the possibility of making accurate predictions of monthly billings. Monthly data on amount of billings (Y. in thousands of constant dollars) and on number of hours of staff time (X, in thousand hours) for the 20 most recent months follow. A simple linear regression model is believed to be appropriate. but positively autocorrelated error terms may be present.

a-) Fit a simple linear regression model by ordinary least squares and obtain the residuals. State the regression equation. Also obtain $s(b_0)$ and $s(b_1)$. Plot the residuals against time and explain whether you find any evidence of positive autocorrelation. Conduct a formal test for positive autocorrelation using alpha = .05. State the alternatives (Hypothesis), decision rule and conclusions. Does your visual inspection of the residual plot confirms the formal test results? (20 points)

b-) Using the estimated regression function in part (a) apply the Cochrane-Orcutt procedure to obtain a point estimate of the autocorrelation parameter (RHO). How well does the approximate relationship between the point estimate and the Durbin-Watson test statistic? D ~ 2*(1 - RHO). Use one iteration to obtain the estimates of the regression coefficients $b_0^{'}$ and $b_0^{'}$ and their corresponding $s(b_0^{'})$ and $s(b_1^{'})$ in the transformed model (reparametrized for reasons of autocorrelation). State the new regression function. Test whether any positive autocorrelation remains after the first iteration using alpha = .05 State the alternatives, decision rule, and conclusion.(10 points)

c-) Restate the estimated regression function obtained in part (b) in terms of the original variables. Also obtain corresponding $s(b_0)$ and $s(b_1)$. Compare the estimated regression coefficients obtained with the Cochrane-Orcutt procedure and their estimated standard deviations with those obtained with ordinary least squares in part (a). Based on the results here and in part (b), does the Cochrane-Orcutt procedure appear to have been effective here? (10 points)

d-). Using the model estimated after applying the Cochrane-Orcutt procedure do the following prediction. Staff time in month 21 is expected to be 3.625 thousand hours. Predict the amount of billings in constant dollars for month 21, using a 99 percent prediction interval. Interpret your interval. Estimate β_1, with a 99 percent confidence interval. Interpret your interval estimate. (10 points)

e-)Use the Hildreth-Lu procedure to obtain a point estimate of the autocorrelation parameter. Do a search at the values RHO = .1, .2, . . . , 1.0 and select from these the value of RHO that minimizes SSE. Based on your result of RHO obtain an obtain the estimates of the regression coefficients $b_0^{'}$ and $b_0^{'}$ and their corresponding $s(b_0^{'})$ and $s(b_1^{'})$ in the transformed model (reparametrized for reasons of autocorrelation). Test whether any positive autocorrelation remains in the reparametrized regression model; use alpha = .05 State the alternatives, decision rule, and conclusion (10 points).

f-) Restate the estimated regression function obtained using the Hildreth-Lu procedure in terms of the original variables. Also obtain $s(b_0)$ and $s(b_1)$. Compare the estimated regression coefficients obtained with the Hildreth-Lu procedure and their estimated standard deviations with those obtained with ordinary least squares in (a). Based on the results above has the Hildreth-Lu procedure been effective here? (10 points)

g-) Staff time in month 21 is expected to be 3.625 thousand hours. Predict the amount of billings in constant dollars for month 21, using a 99 percent prediction interval. Interpret your interval (5 points).

h-) Apply the first differences procedure and repeat (f) and (g) above, do not forget to compare with (a) (15 points).

i-) Repeat (g) using an autoregressive distributed lag (ARDL) model and compare the prediction against cochrane orcut, hildreth lu and first differeces procedures. Report your findings (10 points).

Solutions

We first read the data and do a quick sanity check for structure and missing values.

```
dat <- readr::read_csv("Advertising Agency.csv", show_col_types = FALSE)

# Quick checks: structure and missing values
str(dat)
```

```
## spc_tbl_ [20 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Y: num [1:20] 220 204 207 222 211 ...
##  $ X: num [1:20] 2.52 2.17 2.23 2.52 2.31 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Y = col_double(),
##   ..   X = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
colSums(is.na(dat))
```

```
## Y X
## 0 0
```

We then add a simple time index `t` for the 20 months and look at the first few rows.

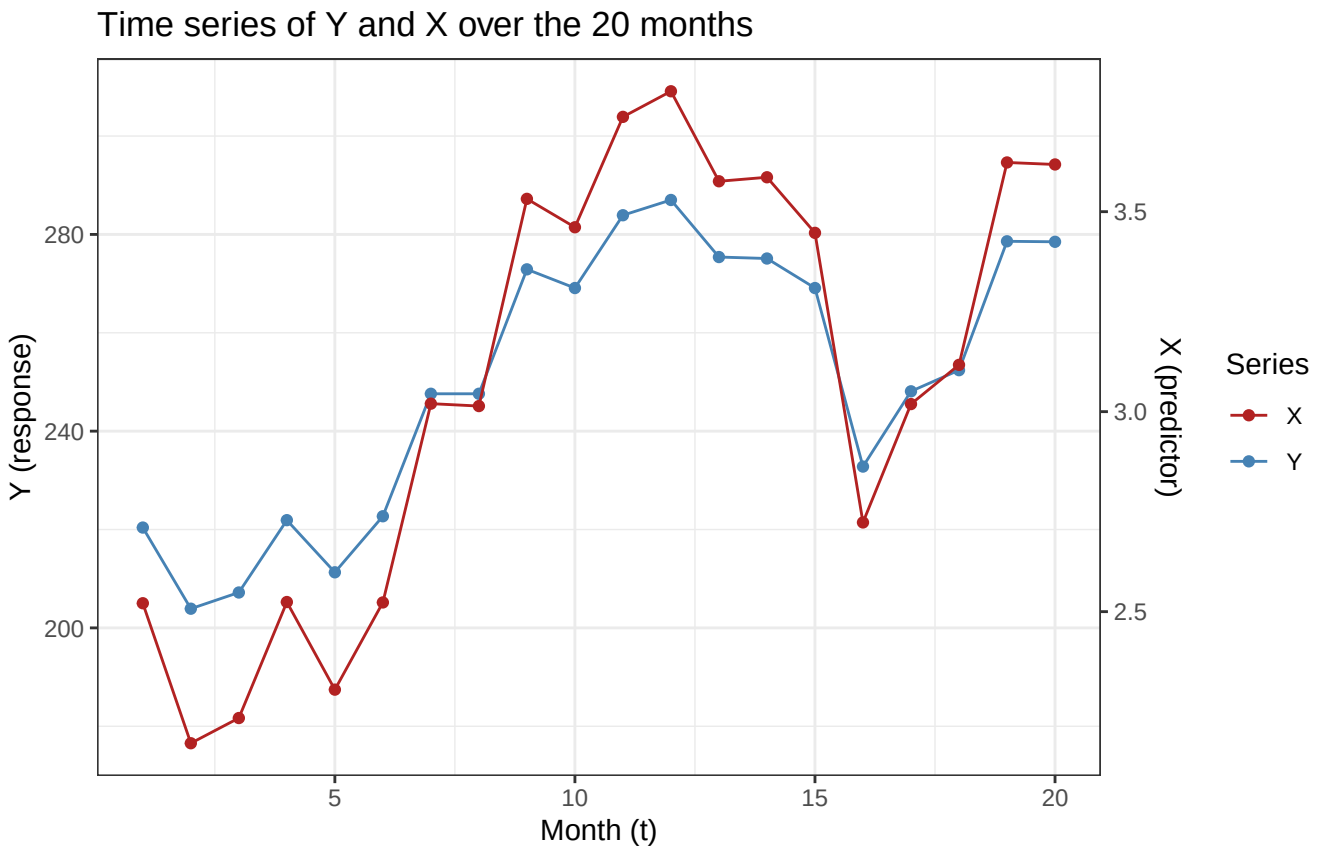
```
dat <- dplyr::mutate(dat, t = dplyr::row_number())

knitr::kable(
  head(dat),
  caption = "First 6 rows of the Advertising Agency data with time index t"
)
```

Table 1: First 6 rows of the Advertising Agency data with time index t

Y	X	t
220.4	2.521	1
203.9	2.171	2
207.2	2.234	3
221.9	2.524	4
211.3	2.305	5
222.7	2.523	6

Before fitting any models, we plot Y and X over time to see how the two series move together over the 20 months. Because Y is around 200–300 and X is around 0–5, a single vertical scale makes X look flat. To show how the series move together while keeping each in its own units, we rescale X by a constant factor and add a secondary axis for X.

```
# Choose a scaling factor so X is roughly on the same vertical range as Y
scale_factor <- mean(dat$Y) / mean(dat$X)

ggplot2::ggplot(dat, ggplot2::aes(x = t)) +
  ggplot2::geom_line(ggplot2::aes(y = Y, color = "Y")) +
  ggplot2::geom_point(ggplot2::aes(y = Y, color = "Y")) +
  ggplot2::geom_line(ggplot2::aes(y = X * scale_factor, color = "X")) +
  ggplot2::geom_point(ggplot2::aes(y = X * scale_factor, color = "X")) +
  ggplot2::scale_y_continuous(
    name = "Y (response)",
    sec.axis = ggplot2::sec_axis(~ . / scale_factor, name = "X (predictor)")
  ) +
  ggplot2::scale_color_manual(
    values = c("Y" = "steelblue", "X" = "firebrick"),
    name = "Series"
  ) +
  ggplot2::labs(
    title = "Time series of Y and X over the 20 months",
    x = "Month (t)"
  ) +
  ggplot2::theme_bw()
```



(a) OLS, residual plot, and Durbin–Watson test

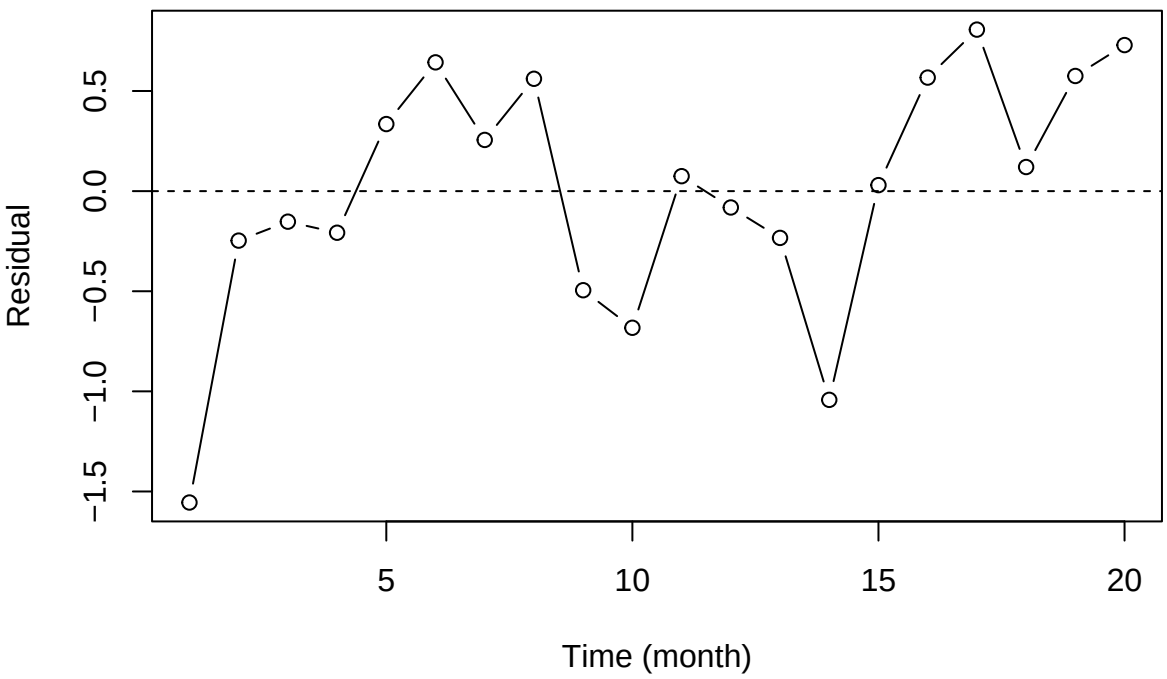
We fit the simple OLS line, report its coefficients and standard errors, write down the regression equation, inspect residuals over time with a plot, and then formally test for positive autocorrelation using the Durbin–Watson test.

```
# Fit OLS
ols <- lm(Y ~ X, data = dat)
sum_ols <- summary(ols)

# Coefficients and their standard errors
coef_table <- broom::tidy(ols)
knitr::kable(coef_table, digits = 4, caption = "OLS coefficients with standard errors")
```

Table 2: OLS coefficients with standard errors

term	estimate	std.error	statistic	p.value
(Intercept)	93.6865	0.8229	113.8458	0
X	50.8801	0.2634	193.1429	0

```
# Regression equation: Y = b0 + b1*X
b0 <- coef(ols)[1]; b1 <- coef(ols)[2]

# Residuals vs. time
plot(dat$t, resid(ols), type = "b", xlab = "Time (month)", ylab = "Residual",
     main = "Residuals vs. Time (OLS)"); abline(h = 0, lty = 2)
```




```r
# Durbin-Watson test for positive autocorrelation (H1: rho > 0)
dw_ols <- lmtest::dwtest(ols, alternative = "greater")
dw_ols
```

```
##
##  Durbin-Watson test
##
## data:  ols
## DW = 0.97374, p-value = 0.002891
## alternative hypothesis: true autocorrelation is greater than 0
```

Commentary (a). The OLS regression is

$$\hat{Y} = 93.6865 + 50.8801\, X.$$

The reported standard errors are $s(b_0) = 0.8229$ and $s(b_1) = 0.2634$ (Table 2). The residuals vs. time plot shows long runs above/below zero, consistent with positive autocorrelation. The Durbin–Watson statistic is 0.9737 (one-sided $p = 0.002891$).

The DW test is:

H_0: $\rho = 0$ (no autocorrelation)

H_1: $\rho > 0$ (positive autocorrelation)

- DW $= 0.9737$ is well below 2, so it strongly suggests positive autocorrelation in the residuals.
- One-sided $p = 0.002891$ means *If, in truth, there were no autocorrelation, there is only about a 0.29% chance of seeing a DW statistic this low or lower, just by random chance.*
- At $\alpha = 0.05$ we reject no-autocorrelation in favor of positive autocorrelation. The visual impression matches the formal test.

With such a small p-value, we reject H_0 and conclude there is strong evidence of positive first-order autocorrelation in our OLS residuals.

Note that we use the `greater` alternative of the `dwtest` which only checks for autocorrelation moving forward in time. This is typical of time series since this is the direction we care about testing.

(b) One-iteration Cochrane–Orcutt (CO), transformed model and post-transform DW

Here we estimate $\hat{\rho}$ from the OLS DW statistic, apply a single Cochrane–Orcutt AR(1) transform to Y and X, fit the transformed regression, and then use a post-transform DW test to confirm that this one-step correction has effectively removed the positive autocorrelation.

```r
# Point estimate from DW: rho_hat is approximately 1 - DW/2
d_stat  <- as.numeric(dw_ols$statistic["DW"])
rho_co1 <- 1 - d_stat/2
rho_co1
```

```
## [1] 0.5131287
```

The Durbin–Watson statistic from part (a) is about $DW \approx 0.9737$.

For an AR(1) error process, there's a common approximation $\hat{\rho} \approx 1 - \frac{DW}{2}$.

We pull out the DW statistic and compute

$$\hat{\rho} \approx 1 - \frac{0.9737}{2} \approx 0.513.$$

This $\hat{\rho}$ is our *one-step estimate* of the first-order autocorrelation in the errors.

This is why it's called "one-iteration" Cochrane–Orcutt. Instead of repeatedly re-estimating ρ and refitting the model, we take the $\hat{\rho}$ from the OLS residuals once.

```r
# One-step CO transform
y <- dat$Y; x <- dat$X
y_t <- y[-1] - rho_co1 * y[-length(y)]
x_t <- x[-1] - rho_co1 * x[-length(x)]
```

Notes about what this means:

`y[-1]` is $Y_2, Y_3, \ldots, Y_T$
(`y[-1]` means to include all of `y` but drop the first element)

`y[-length(y)]` is $Y_1, Y_2, \ldots, Y_{T-1}$. `-length(y)` is `-T`, so "drop the T-th (last) element."

So for $t = 2, \ldots, T$:

$$y_t^* = Y_t - \hat{\rho}, Y_{t-1}, \quad x_t^* = X_t - \hat{\rho}, X_{t-1}.$$

This is the standard Cochrane–Orcutt transformation. If the original error process satisfies

$$e_t = \rho e_{t-1} + u_t,$$

then this transformation approximately turns the correlated errors e_t into uncorrelated u_t in the transformed regression.

```r
co1 <- lm(y_t ~ x_t)  # transformed model: y_t = b0' + b1'*x_t + e_t
sum_co1 <- summary(co1)
b0p <- coef(co1)[1]; b1p <- coef(co1)[2]

knitr::kable(broom::tidy(co1), digits = 4,
             caption = "CO (one-iteration) transformed-model coefficients")
```

Table 3: CO (one-iteration) transformed-model coefficients

term	estimate	std.error	statistic	p.value
(Intercept)	46.4792	0.4834	96.1468	0
x_t	50.3844	0.3061	164.6054	0

We now regress y_t^* on x_t^*:

$$y_t^* = b_0' + b_1' x_t^* + e_t^*.$$

The output we get is

- intercept $b_0' = 46.4792$, with $s(b_0') = 0.4834$,
- slope $b_1' = 50.3844$, with $s(b_1') = 0.3061$.

These are the Cochrane–Orcutt transformed-model coefficients.

```r
# Post-transform DW test (should reduce positive autocorrelation)
dw_co1 <- lmtest::dwtest(co1, alternative = "greater")
dw_co1
```

```
##
##  Durbin-Watson test
##
## data:  co1
## DW = 2.134, p-value = 0.5691
## alternative hypothesis: true autocorrelation is greater than 0
```

We run the Durbin–Watson test again, now on the transformed model's residuals. Result:

$DW = 2.134$ (close to 2), one-sided $p = 0.5691$ for $H_1! : \rho > 0$.

- Interpretation:
  - A DW value near 2 and a large p-value mean **no evidence of remaining positive autocorrelation** after the CO transform.
  - Relative to OLS in (a), the one-step Cochrane–Orcutt correction has done its job: the residuals now look approximately uncorrelated.

Final Commentary (b). Using $\hat{\rho} \approx 1 - \frac{DW}{2}$ from part (a) gives $\hat{\rho} \approx 0.513$. The one-step CO transformed model is

$$y_t = 46.4792 + 50.3844\, x_t + e_t,$$

with standard errors $s(b_0') = 0.4834$ and $s(b_1') = 0.3061$ (Table 3). The post-transform DW is 2.134 with $p = 0.5691$, indicating no remaining positive autocorrelation at $\alpha = 0.05$.

We do not have evidence of positive autocorrelation. The data are consistent with either no autocorrelation or possibly some negative autocorrelation, but we're not formally testing the negative side with this p-value.

As a quick reference:

- Best (for usual regression assumptions): DW ≈ 2 and p > 0.05
- Clearly bad: DW far from 2 and p very small (strong evidence of autocorrelation)
- Ambiguous/suspicious: DW far from 2 but p > 0.05 (might be low power or wrong alternative)

(c) Restate in original variables and compare to OLS

Start from the original model with AR(1) errors:

$$Y_t = \beta_0 + \beta_1 X_t + e_t, \qquad e_t = \rho e_{t-1} + u_t.$$

Multiply the equation at time $t-1$ by ρ:

$$\rho Y_{t-1} = \rho\beta_0 + \rho\beta_1 X_{t-1} + \rho e_{t-1}.$$

Subtract this from the equation at time t:

$$Y_t - \rho Y_{t-1} = (\beta_0 - \rho\beta_0) + \beta_1 X_t - \rho\beta_1 X_{t-1} + (e_t - \rho e_{t-1}) \quad = \beta_0(1-\rho) + \beta_1\big(X_t - \rho X_{t-1}\big) + u_t.$$

The Cochrane–Orcutt regression we actually estimate is

$$Y_t - \rho Y_{t-1} = b_0' + b_1'\big(X_t - \rho X_{t-1}\big) + \hat{u}_t.$$

Matching coefficients gives $b_0' \approx \beta_0(1-\rho)$ and $b_1' \approx \beta_1$, so in the original equation $Y_t = \beta_0 + \beta_1 X_t + e_t$ the intercept and slope are

$$\beta_0 = \frac{b_0'}{1-\rho}, \qquad \beta_1 = b_1'.$$

This is why, in the code below, we rescale the intercept and its standard error by $1/(1-\rho)$ but leave the slope and its standard error unchanged when we convert from the transformed CO fit back to the original Y–X regression.

```r
# Convert the transformed CO coefficients (and their SEs) back to the original regression
se_b0p <- coef(summary(co1))[1, "Std. Error"]
se_b1p <- coef(summary(co1))[2, "Std. Error"]

beta0_hat <- b0p/(1 - rho_co1)
beta1_hat <- b1p
se_beta0  <- se_b0p/(1 - rho_co1)  # delta-method approximation
se_beta1  <- se_b1p
```

Using the converted Cochrane–Orcutt coefficients `beta0_hat`, `beta1_hat` and their standard errors `se_beta0`, `se_beta1` from above, we now assemble a small tibble that places the original OLS intercept and slope (and their SEs) side by side with the corresponding Cochrane–Orcutt values. The table printed below lets us directly compare how the intercept, slope, and standard errors change once we adjust for autocorrelation.

```r
tibble::tibble(
  Model        = c("OLS (original Y, X)", "CO (converted to original Y, X)"),
  Intercept    = c(b0, beta0_hat),
  SE_Intercept = c(coef_table$std.error[coef_table$term == "(Intercept)"], se_beta0),
  Slope        = c(b1, beta1_hat),
  SE_Slope     = c(coef_table$std.error[coef_table$term == "X"], se_beta1)
) |>
  knitr::kable(digits = 4, caption = "Comparison of coefficients and standard errors")
```

Table 4: Comparison of coefficients and standard errors

Model	Intercept	SE_Intercept	Slope	SE_Slope
OLS (original Y, X)	93.6865	0.8229	50.8801	0.2634
CO (converted to original Y, X)	95.4651	0.9929	50.3844	0.3061

Commentary (c). Written back in terms of the original, untransformed variables Y and X, the Cochrane–Orcutt regression is $\hat{Y}_t = 95.4651 + 50.3844 X_t$ (Table 4), compared with the OLS fit $\hat{Y}_t = 93.6865 + 50.8801 X_t$. The Cochrane–Orcutt standard errors are slightly larger ($s(\beta_0) = 0.9929$, $s(\beta_1) = 0.3061$) than the OLS standard errors (0.8229, 0.2634), which is expected once we adjust for autocorrelation. Together with the DW result in (b), this suggests that the Cochrane–Orcutt correction is working as intended.

(d) Forecast month 21 with 99% PI (CO one-iteration)

Use the transformed model's prediction for $\tilde{y}_{T+1} = y_{T+1} - \rho y_T$ at x_{T+1}, then back-transform.

We construct `CO_transformed_new_data` by plugging in the new predictor X_{21} and applying the Cochrane–Orcutt transformation $X_{T+1}^{(CO)} = X_{T+1} - \rho X_T$.

```r
x_T1 <- 3.625           # X_{21}, the new predictor value
x_T  <- tail(x, 1)      # X_T, last observed X
y_T  <- tail(y, 1)      # Y_T, last observed Y

# New data on the Cochrane-Orcutt transformed scale for time T+1
CO_transformed_new_data <- data.frame(
  x_t = x_T1 - rho_co1 * x_T
)
```

We use `predict(co1, ...)` to get `pred_transformed`, which contains the point forecast and 99% prediction interval for the transformed response $Y_{T+1}^{(CO)} = Y_{T+1} - \rho Y_T$.

```r
# Predict on the transformed (CO) scale using the CO model
pred_transformed <- predict(
  co1,
  newdata  = CO_transformed_new_data,
  interval = "prediction",
  level    = 0.99
)
```

We then back-transform to the original Y scale via $Y_{T+1} = Y_{T+1}^{(CO)} + \rho Y_T$ by adding `rho_co1 * y_T` to everything, storing the result in `pred_original_scale`.

`pred_original_scale` is the object we print and refer to in the text as the month-21 forecast and 99% PI on the original scale.

```
# Back-transform the prediction and PI to the original Y scale
pred_original_scale <- pred_transformed + rho_co1 * y_T

knitr::kable(
  as.data.frame(pred_original_scale),
  digits  = 4,
  caption = "Month 21 prediction (original Y scale) from CO one-iteration (99% PI)"
)
```

Table 5: Month 21 prediction (original Y scale) from CO one-iteration (99% PI)

fit	lwr	upr
278.4905	277.1266	279.8544

```
# 99% CI for beta1 after CO (beta1 == b1')
ci_beta1_co <- confint(co1, parm = "x_t", level = 0.99)
ci_tbl <- tibble::tibble(term = "beta1 (CO)", lower = ci_beta1_co[1], upper = ci_beta1_co[2])
knitr::kable(ci_tbl, digits = 4, caption = "99% CI for beta1 after CO (original Y-scale beta1)")
```

Table 6: 99% CI for beta1 after CO (original Y-scale beta1)

term	lower	upper
beta1 (CO)	49.4973	51.2716

Note. Since $\beta_1 = b_1'$ under the CO transform when $\hat{\rho}$ is treated as fixed, this CI is also the 99% CI for β_1 on the original scale.

Commentary (d). For month 21 with $X_{21} = 3.625$, the CO one-iteration point forecast is 278.4905 with a 99% PI [277.1266, 279.8544] (Table 5). The interval is relatively tight (width ≈ 2.73), consistent with a strong linear signal in X and improved noise behavior after removing autocorrelation.

(e) Hildreth–Lu (grid over rho = 0.1, 0.2, . . . , 1.0)

First, we run a Hildreth–Lu grid search over a set of candidate ρ values. For each candidate ρ in $0.1, 0.2, \ldots, 1.0$ we apply the same AR(1) transform as in Cochrane–Orcutt, fit a transformed regression, and record the sum of squared residuals (SSE) and coefficient estimates for that ρ.

```
rhos <- seq(0.1, 1.0, by = 0.1)

fit_list <- lapply(rhos, function(rho) {
  # AR(1) transform for this candidate rho
  yt <- y[-1] - rho * y[-length(y)]
  xt <- x[-1] - rho * x[-length(x)]

  # Fit transformed regression at this rho
  m   <- lm(yt ~ xt)
  sse <- sum(resid(m)^2)

  tibble::tibble(
    rho    = rho,
    sse    = sse,
    b0p    = coef(m)[1],
    b1p    = coef(m)[2],
```

```
    se_b0p = coef(summary(m))[1, 2],
    se_b1p = coef(summary(m))[2, 2]
  )
})

hl_tbl <- dplyr::bind_rows(fit_list)

# Show all candidate rho values and their SSE / coefficients
hl_tbl |>
  knitr::kable(
    digits  = 4,
    caption = "Hildreth-Lu grid search over rho (candidate AR(1) values)"
  )
```

Table 7: Hildreth–Lu grid search over rho (candidate AR(1) values)

rho	sse	b0p	b1p	se_b0p	se_b1p
0.1	4.0450	84.9741	50.6764	0.6498	0.2284
0.2	3.7414	75.6629	50.6301	0.6091	0.2401
0.3	3.5511	66.3543	50.5697	0.5711	0.2561
0.4	3.4685	57.0406	50.4925	0.5329	0.2770
0.5	3.4889	47.7071	50.3980	0.4897	0.3025
0.6	3.6126	38.3321	50.2909	0.4363	0.3313
0.7	3.8511	28.8910	50.1846	0.3678	0.3612
0.8	4.2292	19.3703	50.1005	0.2836	0.3887
0.9	4.7772	9.7830	50.0585	0.1923	0.4109
1.0	5.5140	0.1673	50.0652	0.1329	0.4256

```
# Choose the rho that minimizes SSE
rho_hl <- hl_tbl$rho[which.min(hl_tbl$sse)]
rho_hl
```

```
## [1] 0.4
```

Using the Hildreth–Lu choice of ρ from the grid search above, we now apply the corresponding AR(1) transformation to the original Y and X series and fit the transformed regression $Y_t - \rho Y_{t-1}$ on $X_t - \rho X_{t-1}$. The code below refits this Hildreth–Lu model at the selected ρ and prints a tidy coefficient table so we can see the estimated intercept, slope, and their standard errors on the transformed scale.

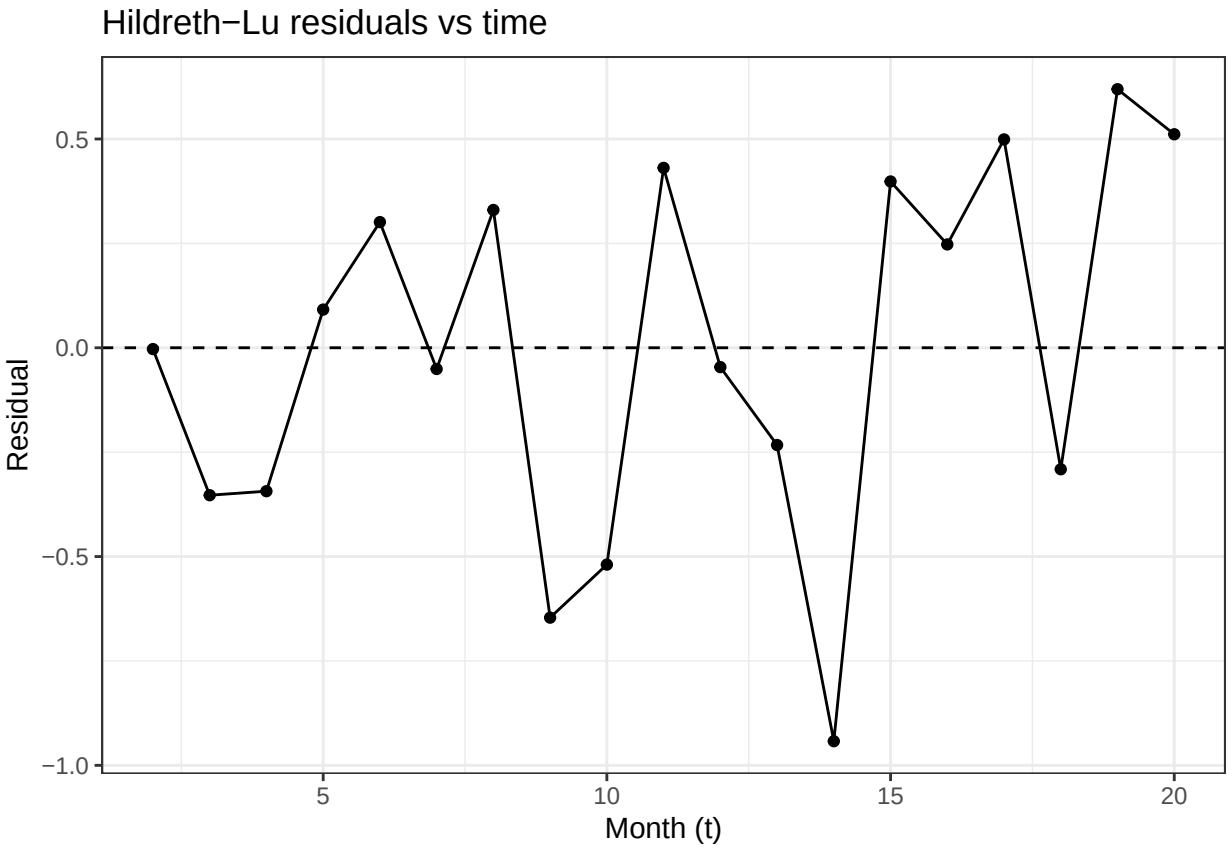
```
# Fit transformed model at the best Hildreth-Lu rho
yt_hl <- y[-1] - rho_hl * y[-length(y)]
xt_hl <- x[-1] - rho_hl * x[-length(x)]
co_hl <- lm(yt_hl ~ xt_hl)

# Coefficient table for the HL transformed regression
hl_coef_tbl <- broom::tidy(co_hl)

knitr::kable(
  hl_coef_tbl,
  digits  = 4,
  caption = "Hildreth-Lu transformed-model coefficients"
)
```

Table 8: Hildreth–Lu transformed-model coefficients

term	estimate	std.error	statistic	p.value
(Intercept)	57.0406	0.5329	107.0436	0
xt_hl	50.4925	0.2770	182.3057	0

We next check whether the Hildreth–Lu transformation has successfully removed the positive autocorrelation. We first look at a residuals-versus-time plot for the transformed regression, then confirm with a Durbin–Watson test.

```
# Residuals vs time for the Hildreth-Lu transformed regression
hl_resid_df <- tibble::tibble(
  t        = dat$t[-1],        # time index for the transformed sample
  residual = resid(co_hl)
)

ggplot2::ggplot(hl_resid_df, ggplot2::aes(x = t, y = residual)) +
  ggplot2::geom_hline(yintercept = 0, linetype = "dashed") +
  ggplot2::geom_line() +
  ggplot2::geom_point() +
  ggplot2::labs(
    title = "Hildreth-Lu residuals vs time",
    x = "Month (t)",
    y = "Residual"
  ) +
  ggplot2::theme_bw()
```



From the residuals-versus-time plot above, we see that the Hildreth–Lu residuals now bounce around zero with no long smooth trend and no long runs of the same sign, in contrast to the OLS residuals. This visual pattern suggests that the AR(1) correction has largely removed the serial correlation. To confirm this formally, we use the fitted HL model co_hl to run a Durbin–Watson test for positive autocorrelation and then summarize the selected ρ_{HL}, the DW statistic, and its one-sided p-value in a small table so we can see at a glance whether any substantial autocorrelation remains.

```
# Durbin-Watson test on HL residuals
dw_hl <- lmtest::dwtest(co_hl, alternative = "greater")

dw_summary <- data.frame(
  rho_HL   = rho_hl,
  DW_stat  = as.numeric(dw_hl$statistic["DW"]),
  p_value  = dw_hl$p.value
)
```

```
knitr::kable(
  dw_summary,
  digits  = 4,
  caption = "Hildreth-Lu choice of rho and post-transform Durbin-Watson test"
)
```

Table 9: Hildreth–Lu choice of rho and post-transform Durbin–Watson test

rho_HL	DW_stat	p_value
0.4	1.9054	0.3509

Commentary (e). The HL grid search selects $\rho = 0.4$ as minimizing SSE. The post-transform DW is 1.9054 with $p = 0.3509$, so there is no evidence of remaining positive autocorrelation at $\alpha = 0.05$.

(f) Restate HL on the original Y scale and compare to OLS.

In part (e) we estimated the Hildreth–Lu transformed regression using the selected value of ρ_{HL}. To put Hildreth–Lu and OLS on the same footing, we now rewrite the HL model in terms of the original regression

$$Y_t = \beta_0 + \beta_1 X_t + e_t.$$

For the AR(1) transformation used in Hildreth–Lu, the transformed-model coefficients b_0' and b_1' from `co_hl` correspond to an original-scale intercept and slope given by $\beta_0 = b_0'/(1-\rho_{HL})$ and $\beta_1 = b_1'$. In the code below we extract b_0', b_1' and their standard errors from the HL fit, convert them to the original Y, X scale, and approximate the standard errors of these converted coefficients while treating ρ_{HL} as fixed.

```
# Convert the transformed Hildreth-Lu coefficients (and their SEs)
# back to the original Y, X regression
b0p_hl  <- coef(co_hl)[1]
b1p_hl  <- coef(co_hl)[2]
se0p_hl <- coef(summary(co_hl))[1, "Std. Error"]
se1p_hl <- coef(summary(co_hl))[2, "Std. Error"]

# Map transformed HL coefficients to the original Y_t = beta0 + beta1 * X_t + e_t equation
beta0_hl <- b0p_hl / (1 - rho_hl)
beta1_hl <- b1p_hl

# Approximate SEs on the original scale (treat rho_hl as fixed)
se0_hl <- se0p_hl / (1 - rho_hl)  # delta-method style rescaling
se1_hl <- se1p_hl
```

Using the converted Hildreth–Lu coefficients `beta0_hl`, `beta1_hl` and their standard errors `se0_hl`, `se1_hl` from above, we now assemble a small tibble that places the original OLS intercept and slope (and their SEs) side by side with the corresponding Hildreth–Lu values. The table printed below lets us directly compare how the intercept, slope, and standard errors change once we adjust for autocorrelation using the Hildreth–Lu choice of `rho_hl`.

```
tibble::tibble(
  Model        = c("OLS (original Y, X)", "Hildreth-Lu (original Y, X)"),
  Intercept    = c(b0, beta0_hl),
  SE_Intercept = c(
    coef_table$std.error[coef_table$term == "(Intercept)"],
    se0_hl
  ),
  Slope        = c(b1, beta1_hl),
  SE_Slope     = c(
    coef_table$std.error[coef_table$term == "X"],
    se1_hl
  )
) |>
  knitr::kable(
```

```
    digits  = 4,
    caption = "Hildreth-Lu vs. OLS on the original Y scale"
  )
```

Table 10: Hildreth–Lu vs. OLS on the original Y scale

Model	Intercept	SE_Intercept	Slope	SE_Slope
OLS (original Y, X)	93.6865	0.8229	50.8801	0.2634
Hildreth–Lu (original Y, X)	95.0676	0.8881	50.4925	0.2770

Commentary (f). In the original Y–X regression (Table 10), the Hildreth–Lu model gives $\beta_0 = 95.0676$ and $\beta_1 = 50.4925$ with $s(\beta_0) = 0.8881$ and $s(\beta_1) = 0.2770$, compared with the OLS fit $b_0 = 93.6865$, $b_1 = 50.8801$ and standard errors $(0.8229, 0.2634)$. The HL coefficient estimates remain close to the OLS values, while the standard errors are modestly larger, as expected after adjusting for autocorrelation. Together with the HL post-transform DW of 1.9054 (no evidence of positive autocorrelation), this suggests that the Hildreth–Lu correction is effective and produces predictions similar to the one-iteration Cochrane–Orcutt procedure.

(g) Forecast month 21 with 99% PI (HL)

This part is nearly identical to part **d)**. Once we've picked a value of ρ, the calculation for HL is exactly the same as for CO; the only differences are which ρ we plug in (`rho_co1` vs `rho_hl`), and which fitted transformed model we use (`co1` vs `co_hl`).

The **transform** is the same:

- $Y_t^* = Y_t - \rho Y_{t-1}$
- $X_t^* = X_t - \rho X_{t-1}$

The **regression form** is the same:

- $Y_t^* = b_0' + b_1' X_t^* + e_t$

The **forecast/back-transform** is the same:

- predict Y_{T+1}^* at $X_{T+1}^* = X_{T+1} - \rho X_T$
- then $Y_{T+1} = Y_{T+1}^* + \rho Y_T$

The only conceptual difference between CO and HL in this assignment is

- CO: we get ρ from the DW approximation (one-step CO here).
- HL: we get ρ from a grid search over $0.1, 0.2, \ldots, 1.0$ that minimizes SSE.

Once ρ is chosen, everything else (transform, fit, forecast, back-transform) is the same!

So… we again want a forecast for month 21 at $X_{21} = 3.625$, but now using the Hildreth–Lu fit `co_hl` and its AR(1) parameter ρ_{HL} stored in `rho_hl`. As with Cochrane–Orcutt, we work on the transformed scale first and then back-transform to the original Y scale. We start by constructing `HL_transformed_new_data` by plugging in the new predictor X_{21} and applying the Hildreth–Lu AR(1) transformation

$$X_{T+1}^{(HL)} = X_{T+1} - \rho_{HL} X_T.$$

```
# We reuse the same X_{21}, X_T, Y_T as in the CO forecast
x_T1 <- 3.625           # X_{21}, the new predictor value
x_T  <- tail(x, 1)      # X_T, last observed X
y_T  <- tail(y, 1)      # Y_T, last observed Y

# New data on the Hildreth-Lu transformed scale for time T+1
HL_transformed_new_data <- data.frame(
  xt_hl = x_T1 - rho_hl * x_T
)
```

We then use `predict(co_hl, ...)` to get `pred_transformed_hl`, which contains the point forecast and 99% prediction interval for the transformed response

$$Y_{T+1}^{(HL)} = Y_{T+1} - \rho_{HL} Y_T.$$

```r
# Predict on the transformed (HL) scale using the HL model
pred_transformed_hl <- predict(
  co_hl,
  newdata  = HL_transformed_new_data,
  interval = "prediction",
  level    = 0.99
)
```

Finally, we back-transform to the original Y scale using

$$Y_{T+1} = Y_{T+1}^{(HL)} + \rho_{HL} Y_T,$$

which we implement by adding `rho_hl * y_T` to the entire prediction object. The resulting `pred_original_scale_hl` object contains the month-21 forecast and 99% prediction interval on the original scale.

```r
# Back-transform the prediction and PI to the original Y scale
pred_original_scale_hl <- pred_transformed_hl + rho_hl * y_T

knitr::kable(
  as.data.frame(pred_original_scale_hl),
  digits  = 4,
  caption = "Month 21 prediction (original Y scale) from Hildreth-Lu (99% PI)"
)
```

Table 11: Month 21 prediction (original Y scale) from Hildreth–Lu (99% PI)

fit	lwr	upr
278.4031	277.0399	279.7663

Commentary (g). For $X_{21} = 3.625$, the Hildreth–Lu forecast is 278.4031 with a 99% prediction interval [277.0399, 279.7663] (Table 11). The interval width is similar to Cochrane–Orcutt.

(h) First differences; restate and predict as in (f)–(g)

This part is nearly identical to what we did for Cochrane–Orcutt and Hildreth–Lu. For CO and HL, once we have picked a value of ρ, the calculation is exactly the same; the only differences are which ρ we plug in (`rho_co1` vs `rho_hl`), and which fitted transformed model we use (`co1` vs `co_hl`).

For CO and HL the **transform** is the same:

- $Y_t^* = Y_t - \rho Y_{t-1}$
- $X_t^* = X_t - \rho X_{t-1}$

The **regression form** is the same:

- $Y_t^* = b_0' + b_1' X_t^* + e_t$

The **forecast/back-transform** is the same:

- predict Y_{T+1}^* at $X_{T+1}^* = X_{T+1} - \rho X_T$
- then $Y_{T+1} = Y_{T+1}^* + \rho Y_T$

First differences follows the same overall template, but with a different transform. Instead of quasi-differencing with an estimated ρ, we work with pure differences:

- $\Delta Y_t = Y_t - Y_{t-1}$
- $\Delta X_t = X_t - X_{t-1}$

We fit

$$\Delta Y_t = b0_{\text{fd}} + b1_{\text{fd}}\, \Delta X_t + \text{error}.$$

then for forecasting we

- predict ΔY_{T+1} at $\Delta X_{T+1} = X_{T+1} - X_T$,
- and back-transform with $Y_{T+1} = Y_T + \Delta Y_{T+1}$.

So across CO, HL, and first differences we always:

1. transform Y and X,
2. regress transformed Y on transformed X,
3. transform the new X_{T+1} the same way,
4. predict on the transformed scale,
5. undo the transform to get Y_{T+1} on the original scale.

The differences are just in *how* we transform (quasi-differences with an estimated ρ for CO/HL vs pure first differences for FD) and how ρ is chosen (DW-based, grid search, or implicitly $\rho \approx 1$ when we difference).

First we apply the first-differences procedure to remove autocorrelation by differencing both Y and X, then fit a regression of ΔY_t on ΔX_t and inspect the estimated intercept and slope.

```
# First-difference the series
dy <- diff(y)
dx <- diff(x)

# First-differences regression: dY_t = b0_fd + b1_fd * dX_t + error
fd     <- lm(dy ~ dx)
sum_fd <- summary(fd)

knitr::kable(
  broom::tidy(fd),
  digits  = 4,
  caption = "First-differences regression: dY_t on dX_t"
)
```

Table 12: First-differences regression: dY_t on dX_t

term	estimate	std.error	statistic	p.value
(Intercept)	0.1673	0.1329	1.2583	0.2253
dx	50.0652	0.4256	117.6423	0.0000

To forecast month 21, we use the first-differences model to predict the change in Y between T and $T+1$,

$$\Delta Y_{T+1} = b_{0,\text{fd}} + b_{1,\text{fd}}(X_{T+1} - X_T),$$

then convert back to the original Y scale via

$$Y_{T+1} = Y_T + \Delta Y_{T+1}.$$

In the code below we predict ΔY_{T+1} and its 99% prediction interval, then add Y_T to obtain the month-21 forecast and 99% PI on the original Y scale. We store this in `pred_original_scale_fd` to keep the naming consistent with the CO and HL sections.

```
# Coefficients from the first-differences model
b0_fd <- coef(fd)[1]
b1_fd <- coef(fd)[2]

# Reuse the same X_21, X_T, Y_T as in the CO/HL forecasts
x_T1 <- 3.625           # X_21, the new predictor value
x_T  <- tail(x, 1)      # X_T, last observed X
y_T  <- tail(y, 1)      # Y_T, last observed Y

# Predict change in Y at T+1 given dX = X_21 - X_T
pred_diff <- predict(
  fd,
  newdata  = data.frame(dx = x_T1 - x_T),
  interval = "prediction",
  level    = 0.99
```

```
)

# Back-transform to the original Y scale: Y_T1 = Y_T + predicted change in Y
pred_original_scale_fd <- pred_diff + y_T

knitr::kable(
  as.data.frame(pred_original_scale_fd),
  digits  = 4,
  caption = "Month 21 prediction (original Y scale) from first differences (99% PI)"
)
```

Table 13: Month 21 prediction (original Y scale) from first differences (99% PI)

fit	lwr	upr
279.0177	277.3231	280.7124

Commentary (h). In the first-differences model, the slope is 50.0652 (SE 0.4256), very close to the OLS slope; the intercept 0.1673 is not significant ($p = 0.2253$). The month-21 forecast on the original Y scale is 279.0177 with a 99% prediction interval [277.3231, 280.7124] (Table 13), noticeably wider than the CO and HL intervals, reflecting higher variability after differencing. Compared with (a), first differences remove autocorrelation by differencing rather than modeling it explicitly, trading a bit of efficiency for simplicity.

(i) ARDL model and prediction; compare forecasts

In this last approach we use an autoregressive distributed lag (ARDL) model. An ARDL model is just a regression that includes:

- **lags of the dependent variable** (autoregressive part), and
- **current and lagged values of one or more predictors** (distributed lag part).

For a single predictor X_t, an ARDL(p, q) model has p lags of Y and q lags of X as regressors. In our case we use an ARDL$(1, 1)$, which means one lag of Y and one lag of X:

$$Y_t = \alpha + \phi, Y_{t-1} + \beta_0 X_t + \beta_1 X_{t-1} + e_t.$$

Here:

- Y_t is the current response,
- Y_{t-1} captures *persistence* or "memory" in Y,
- X_t is the contemporaneous effect of the predictor,
- X_{t-1} captures a *lagged effect* of X, and
- e_t is the error term.

In the code, we implement this by creating lagged columns `Y_lag1` and `X_lag1`, then fitting

$$Y_t \sim Y_{t-1} + X_t + X_{t-1},$$

which is exactly an ARDL$(1, 1)$ model for (Y_t, X_t).

Since we only have 20 time points, adding lots of lags would eat degrees of freedom very quickly and risk overfitting. That is, every extra parameter we estimate uses up one piece of information from the data, leaving us with fewer data points to estimate the remaining noise. ARDL(1,1) is the simplest model that still allows persistence in Y via Y_{t-1}, and lets X have both an immediate effect (X_t) and a one-period lagged effect (X_{t-1}). ARDL(1,1) is chosen as a deliberately simple prototype that is consistent with the AR(1) correction theme of the earlier parts.

We now fit an ARDL(1,1) model on the original Y and X series by adding explicit lag columns. This gives a regression of Y_t on Y_{t-1}, X_t, and X_{t-1}, which lets us capture persistence in Y and both contemporaneous and lagged effects of X.

```
dat2 <- dplyr::mutate(
  dat,
  Y_lag1 = dplyr::lag(Y, 1),
```

```
  X_lag1 = dplyr::lag(X, 1)
)

# ARDL(1,1): Y_t ~ Y_{t-1} + X_t + X_{t-1}
ardl11 <- lm(Y ~ Y_lag1 + X + X_lag1, data = dat2[-1, ])  # drop first row with NA lag

knitr::kable(
  broom::tidy(ardl11),
  digits  = 4,
  caption = "ARDL(1,1) coefficients"
)
```

Table 14: ARDL(1,1) coefficients

term	estimate	std.error	statistic	p.value
(Intercept)	54.1735	19.6105	2.7625	0.0145
Y_lag1	0.4304	0.2099	2.0503	0.0582
X	50.4448	0.4203	120.0335	0.0000
X_lag1	-21.6860	10.5128	-2.0628	0.0569

To forecast month 21, we plug in the last observed values Y_T and X_T and the new predictor value $X_{21} = 3.625$ into the ARDL(1,1) regression. This gives a direct forecast of Y_{21} on the original scale, along with a 99% prediction interval.

```
# Reuse the same X_21, X_T, Y_T as in the other forecasts
x_T1 <- 3.625           # X_21, the new predictor value
x_T  <- tail(x, 1)      # X_T, last observed X
y_T  <- tail(y, 1)      # Y_T, last observed Y

# New data row for ARDL(1,1) forecast at T+1
new_ardl <- data.frame(
  Y_lag1 = y_T,
  X      = x_T1,
  X_lag1 = x_T
)

pred_original_scale_ardl <- predict(
  ardl11,
  newdata  = new_ardl,
  interval = "prediction",
  level    = 0.99
)

knitr::kable(
  as.data.frame(pred_original_scale_ardl),
  digits  = 4,
  caption = "Month 21 prediction (original Y scale) from ARDL(1,1) (99% PI)"
)
```

Table 15: Month 21 prediction (original Y scale) from ARDL(1,1) (99% PI)

fit	lwr	upr
278.4337	276.8655	280.0018

Commentary (i). The ARDL(1,1) fit indicates persistence (lagged Y coefficient 0.4304, $p \approx 0.058$) and a substantial contemporaneous X effect (50.4448), partly offset by a negative lagged X effect (-21.6860, $p \approx 0.057$). The month-21 forecast is 278.4337 with a 99% prediction interval [276.8655, 280.0018], slightly wider than the CO and HL intervals and comparable to first differences.

Summary table of 99% predictions

Finally, we collect the month-21 point forecasts and 99% prediction intervals from all four methods—CO, Hildreth–Lu, first differences, and ARDL(1,1)—into a single table to compare their predictions on the original Y scale.

```
summ <- tibble::tibble(
  Method = c("CO (1-iter)", "Hildreth-Lu", "First Diff", "ARDL(1,1)"),
  Point  = c(
    pred_original_scale[,"fit"],
    pred_original_scale_hl[,"fit"],
    pred_original_scale_fd[,"fit"],
    pred_original_scale_ardl[,"fit"]
  ),
  PI_L = c(
    pred_original_scale[,"lwr"],
    pred_original_scale_hl[,"lwr"],
    pred_original_scale_fd[,"lwr"],
    pred_original_scale_ardl[,"lwr"]
  ),
  PI_U = c(
    pred_original_scale[,"upr"],
    pred_original_scale_hl[,"upr"],
    pred_original_scale_fd[,"upr"],
    pred_original_scale_ardl[,"upr"]
  )
)

knitr::kable(
  summ,
  digits  = 4,
  caption = "99% prediction for month 21 by method (original Y scale)"
)
```

Table 16: 99% prediction for month 21 by method (original Y scale)

Method	Point	PI_L	PI_U
CO (1-iter)	278.4905	277.1266	279.8544
Hildreth–Lu	278.4031	277.0399	279.7663
First Diff	279.0177	277.3231	280.7124
ARDL(1,1)	278.4337	276.8655	280.0018

Commentary (summary). All methods produce very similar point forecasts near 278.4–279.0. The narrowest 99% prediction intervals are from CO (one-iteration) and Hildreth–Lu (width around 2.7), while first differences and ARDL(1,1) give wider intervals. Given the strong evidence of positive autocorrelation in the original OLS fit (DW ≈ 0.97) and its removal under CO/HL (DW near 2), the CO and HL corrections are the most defensible for inference and forecasting here; between them, their point and interval estimates are virtually identical.