Day 13 - Intro to Regression

Introduction

Trends in time series can be classified as either *stochastic*, *deterministic*, or both. Recently, we have considered *stochastic* models for time series, culminating in the AR(p) process. Stochastic time series models are adept at explaining serial autocorrelation, but can struggle to explain large structural effects, such as trends or seasonality.

To remedy this problem, we often pair stochastic models with models capable of accounting for deterministic trends, such as regression. In today's lab, we will begin to explore time series as a tool for modeling

---

Regression overview

Linear model theory

A _____ is a model of the form

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

where ϵ_i is _____ distributed $N(0, \sigma^2)$. 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 $\Sigma = \sigma^2 \mathcal{I}_n$ and $\mathcal{I}_n$ is an $n \times n$ identity matrix. To fit the model, we must estimate _____ and _____. The estimated regression equation is written as:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i,1} + \hat{\beta}_2 x_{i,2} + \cdots + \hat{\beta}_p x_{i,p}$$

Parameter estimates are obtained by minimizing the _____. The error in regression is called a _____,

$$e_i = y_i - \hat{y}_i$$

The _____, which minimizes the _____, is

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

Assumptions

1.

2.

3.

4.

Linear models in R

The following functions are useful when working with regression models in R:

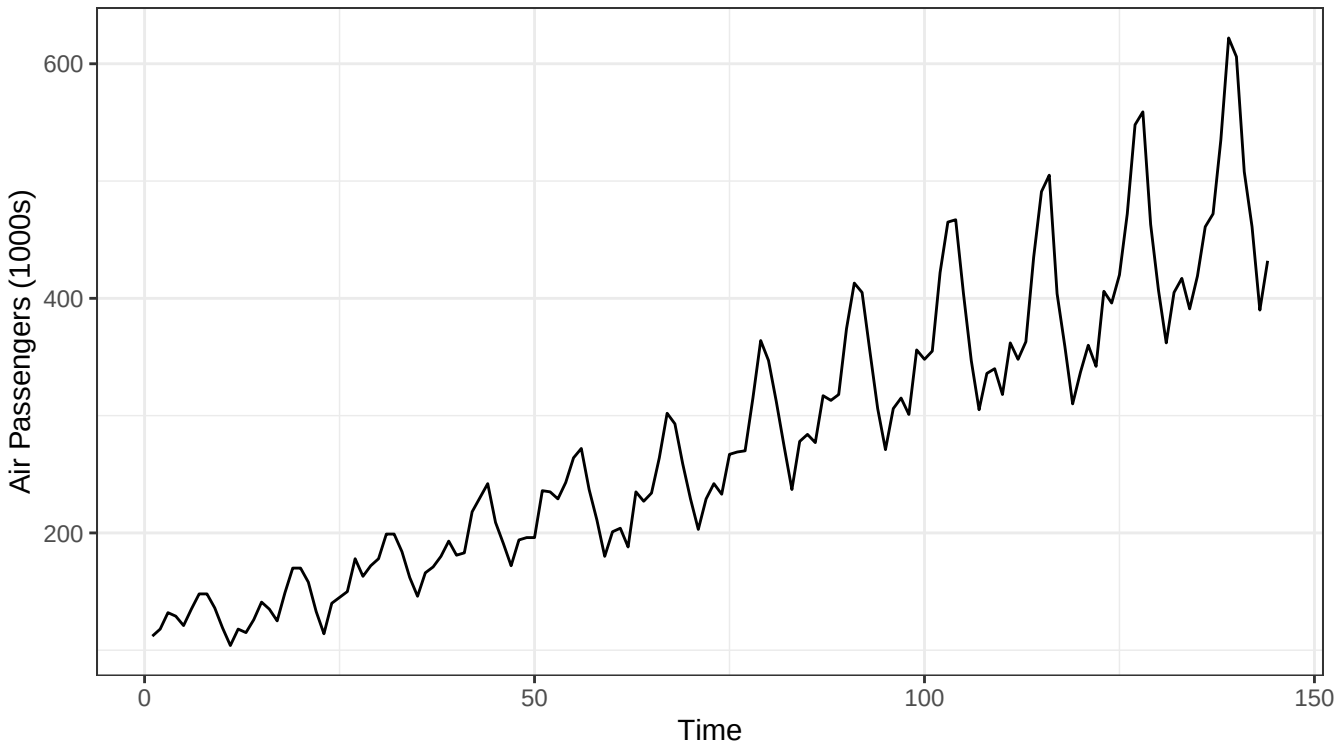
- `lm`
- `plot`
- `fitted`
- `resid`
- `predict`
- `model.matrix`

Air Passengers activity

1. Load the `AirPassengers` data set and create a data frame that includes the count of air passengers, the year, the month, and a time index ($t = 1, 2, \ldots, n$). Plot the count of air passengers by the time index. Then create a second data set, called `ap_sub`, that contains the air passenger measurements between 1949 and 1959.

> **!** Important
>
> Be sure to treat the month column as a `factor`!

```
data("AirPassengers")
ap <- AirPassengers

ap_tbl <- tibble(
  val = c(ap), year = rep(1:12, each = 12),
  month = rep(1:12, 12) %>% factor()
) %>% mutate(t = 1:n())

ap_sub <- ap_tbl %>% filter(year < 12)

ap_tbl %>%
  ggplot() +
  geom_line(aes(x = t, y = val)) +
  theme_bw() +
  labs(x = "Time", y = "Air Passengers (1000s)")
```



2. Fit a regression model, called `fit`, that models the passenger count by the time index plus the month (be sure month is treated as a `factor`) for the `ap_sub` data frame. Print out a summary of the model, and use the summary to write out the estimated regression model.

```
fit <- lm(val ~ t + month, ap_sub)
summary(fit)
```

```
Call:
lm(formula = val ~ t + month, data = ap_sub)

Residuals:
   Min     1Q Median     3Q    Max
-36.46 -16.27  -2.68  12.14  77.00

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  69.16294    7.82746   8.836 1.06e-14 ***
t             2.56812    0.05401  47.546  < 2e-16 ***
month2       -7.56812   10.04140  -0.754 0.452523
month3       25.68194   10.04183   2.557 0.011799 *
month4       15.93201   10.04256   1.586 0.115291
month5       17.54571   10.04358   1.747 0.083225 .
month6       52.70486   10.04488   5.247 6.83e-07 ***
```

```
month7       85.50038   10.04648   8.510 6.09e-14 ***
month8       84.11408   10.04837   8.371 1.28e-13 ***
month9       37.36414   10.05054   3.718 0.000308 ***
month10      -0.02216   10.05301  -0.002 0.998245
month11     -32.95391   10.05577  -3.277 0.001375 **
month12      -7.70385   10.05881  -0.766 0.445264
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.55 on 119 degrees of freedom
Multiple R-squared:  0.9557, Adjusted R-squared:  0.9512
F-statistic: 213.9 on 12 and 119 DF,  p-value: < 2.2e-16
```

$$\widehat{ap} = 69.16 + 2.57\text{time} - 7.57\text{feb} + 25.68\text{mar} + 15.93\text{apr} + 17.55\text{may} + 52.70\text{jun} + 85.50\text{jul} + 84.11\text{aug} + 37.36\text{sept} - .02\text{oct} - 32.95\text{nov} - 7.7\text{dec}$$

3. Recreate the slope coefficient estimates by calculating the OLS estimator by hand using `model.matrix`. The following R functions may be useful:

   - `solve` computes a matrix inverse
   - `t` computes the transpose of a matrix
   - `%*%` computes matrix multiplication

```
X <- model.matrix(fit)
solve(t(X) %*% X) %*% t(X) %*% ap_sub$val
```

```
                  [,1]
(Intercept)  69.16294192
t             2.56811869
month2       -7.56811869
month3       25.68194444
month4       15.93200758
month5       17.54570707
month6       52.70486111
month7       85.50037879
month8       84.11407828
month9       37.36414141
month10      -0.02215909
month11     -32.95391414
month12      -7.70385101
```

4. Interpret the slope coefficient associated with the time index.

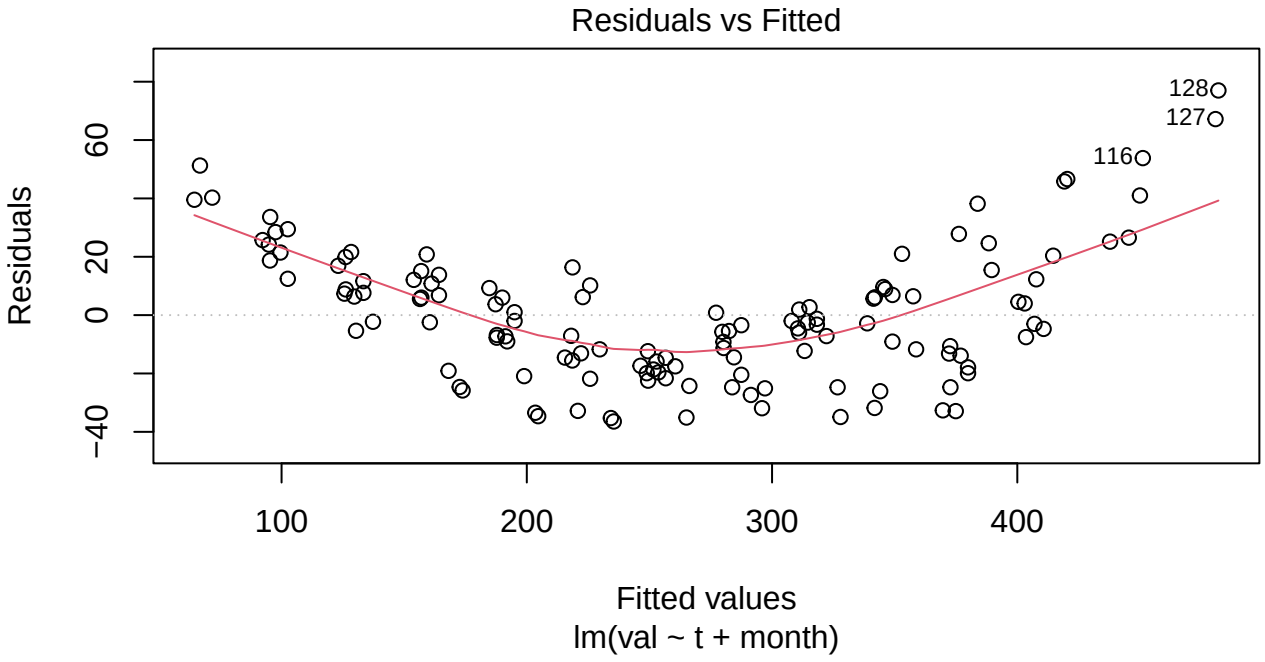
> For a one unit increase in the time index (which is a one month increase in time) and holding the month constant, we expect a 2.57 unit increase in the number of passengers (in thousands), on average.

5. Interpret the slope coefficient associated with the adjustment to the intercept for August.

> Holding the time constant, the number of air passengers (in thousands) in August is 84.11 units greater than in January, on average.

6. Assess the linearity assumption for the regression model by running `plot(fit, which = 1)`

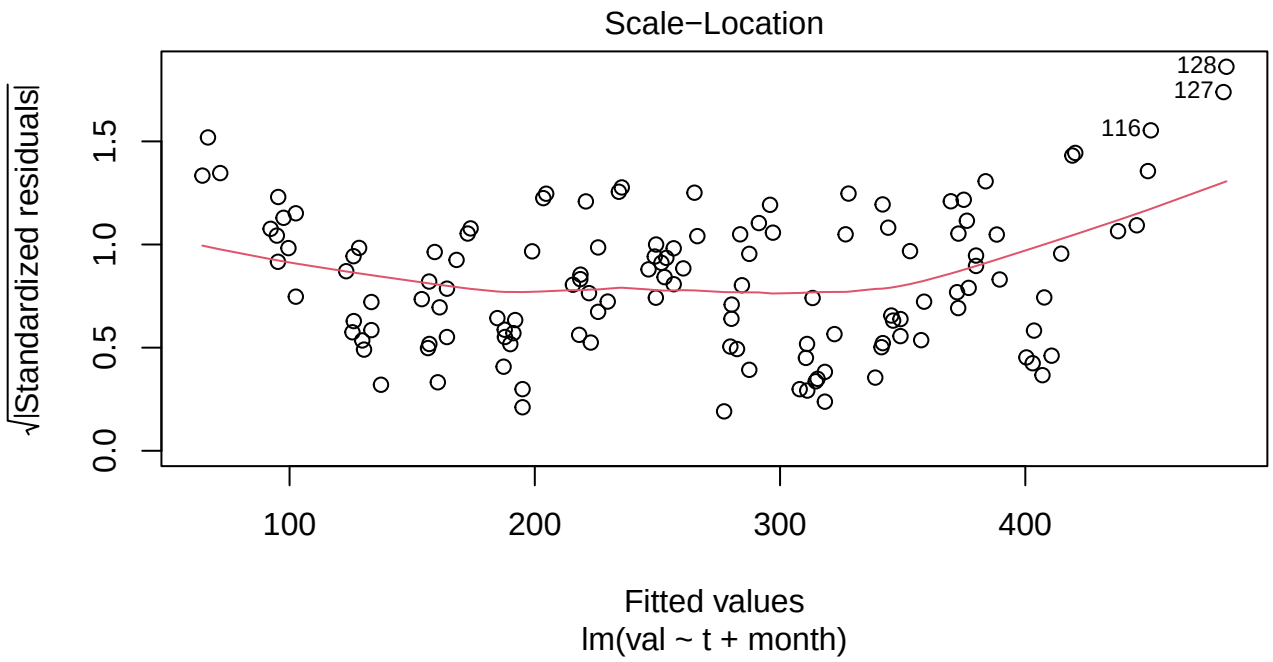
```
plot(fit, which = 1)
```



> Definitely some curvature leftover in the residuals!

7. Assess the constant variance assumption for the regression model by using the previous plot and running `plot(fit, which = 3)`

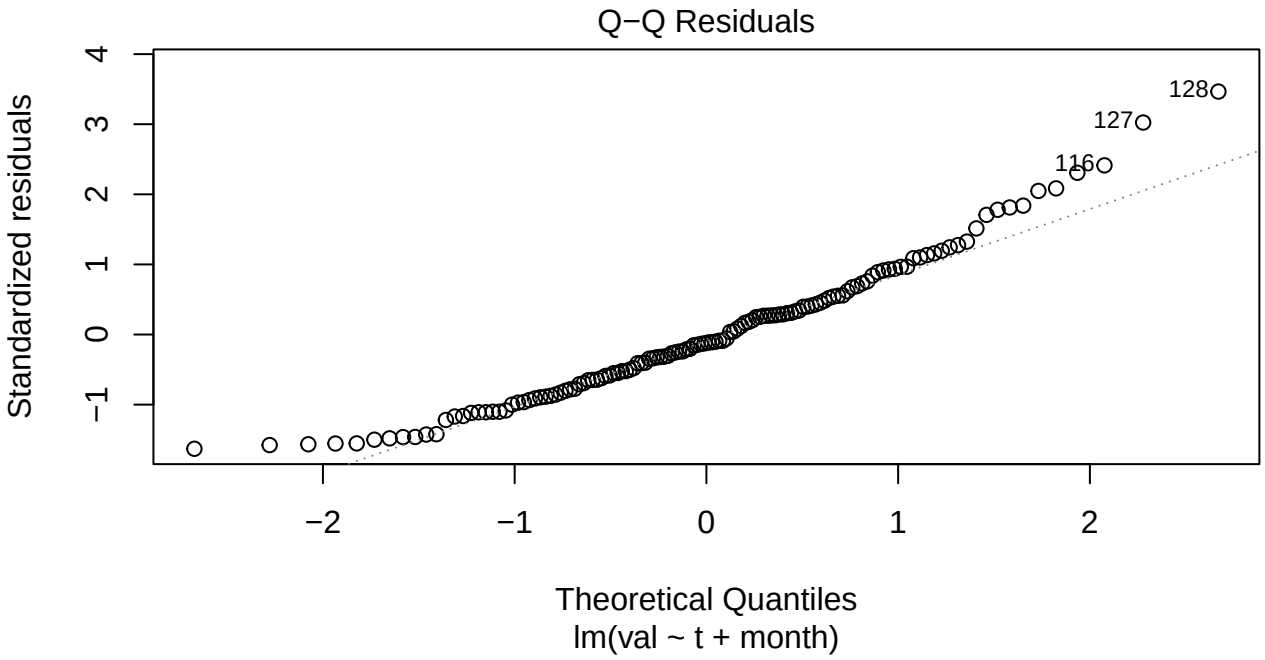
```
plot(fit, which = 3)
```



There is clear evidence of fanning in the residuals vs fitted values plot, which is also indicated by the curvature in the scale-location plot. Both plots suggest that the variance of the residuals is not constant, and appears to increase as the fitted values increase.

8. Assess the normality assumption for the regression model by running `plot(fit, which = 2)`

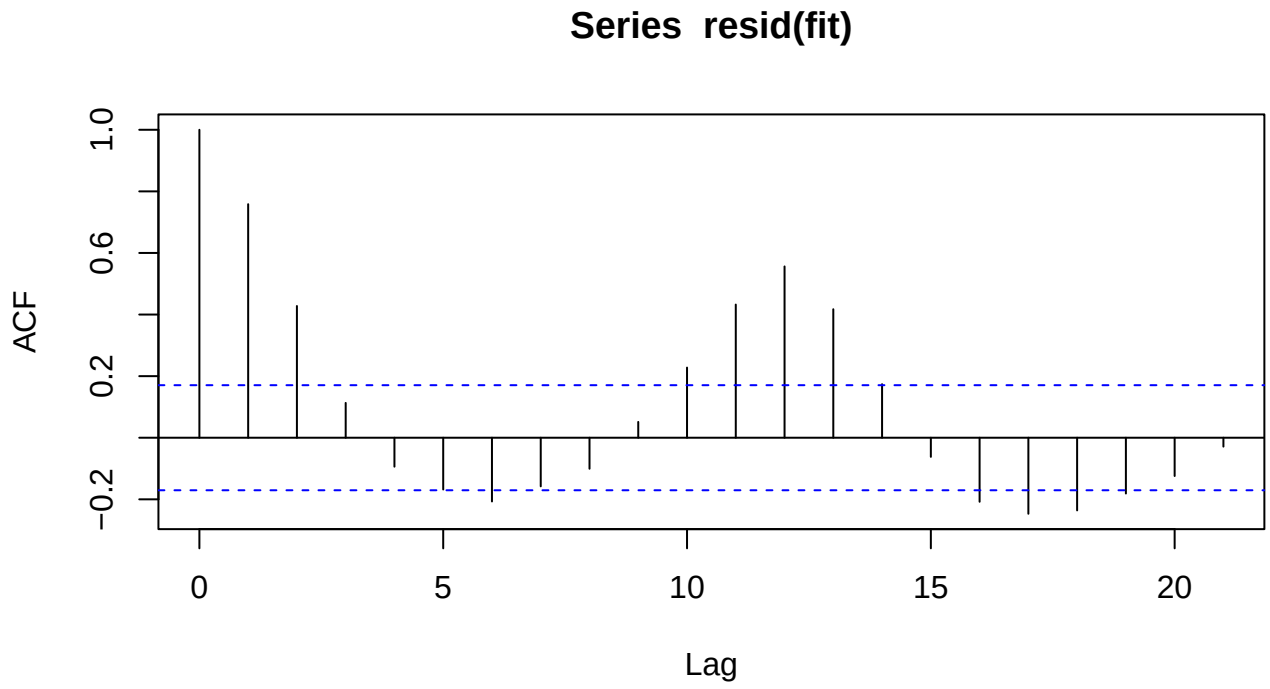
```
plot(fit, which = 2)
```



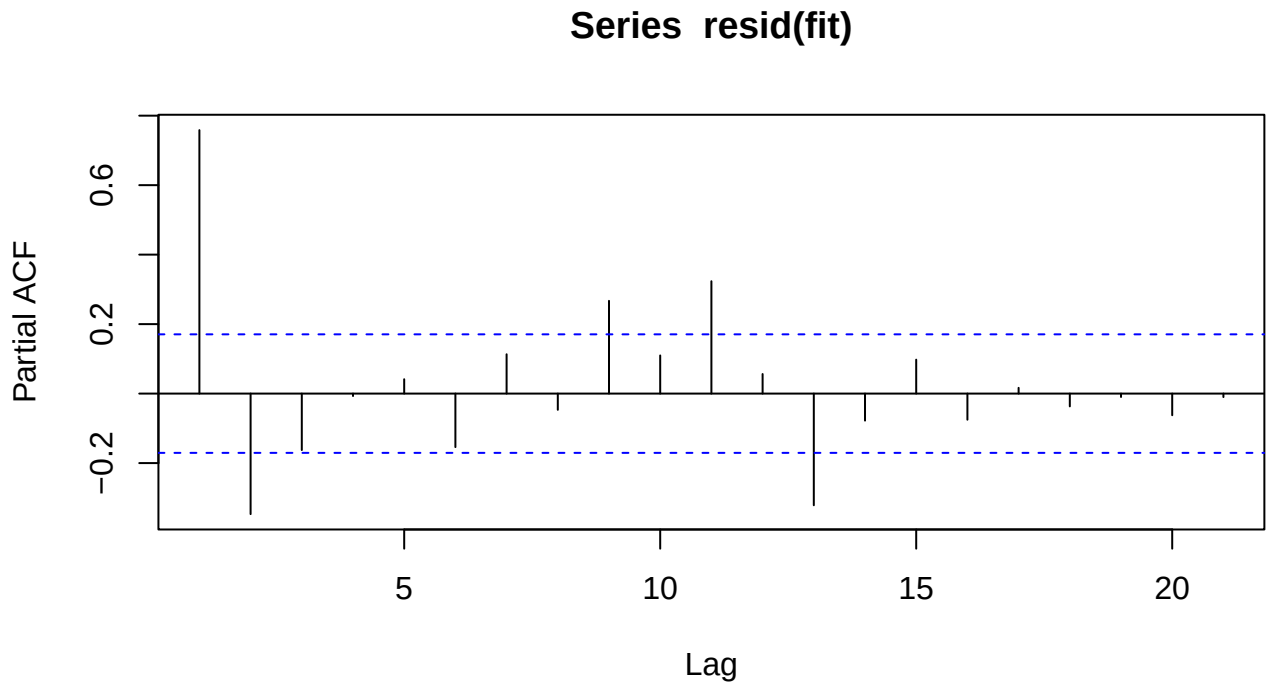
The points fall of the hypothesized quantile lines in the tails, suggesting a violation of normality.

9. Assess the independence assumption by thinking about the problem and running `acf(resid(fit))` and `pacf(resid(fit))`

```
acf(resid(fit))
```




```
pacf(resid(fit))
```



> Generally, we would not expect observations to be independent of one another, since we would expect observations collected sequentially in time to be similar. The ACF plot confirms this suspicion, and indicated that there is autocorrelation left in the residuals at multiple lags (1, 2, 11, 12, 13, …). The PACF plot suggests that the bulk of this residual autocorrelation is driven by lags 1 and 2.

10. One way to address the violations of the normality and constant variance assumptions is to log transform the response. To address the violation of the linearity assumption, it may be helpful to create a squared time index variable. Create new columns in the `ap_sub` data frame that log the number of passengers and computes the square of the time index. Then fit a new model, called `fit_log`, that models the logged passenger count by the time index, time index squared, and month.

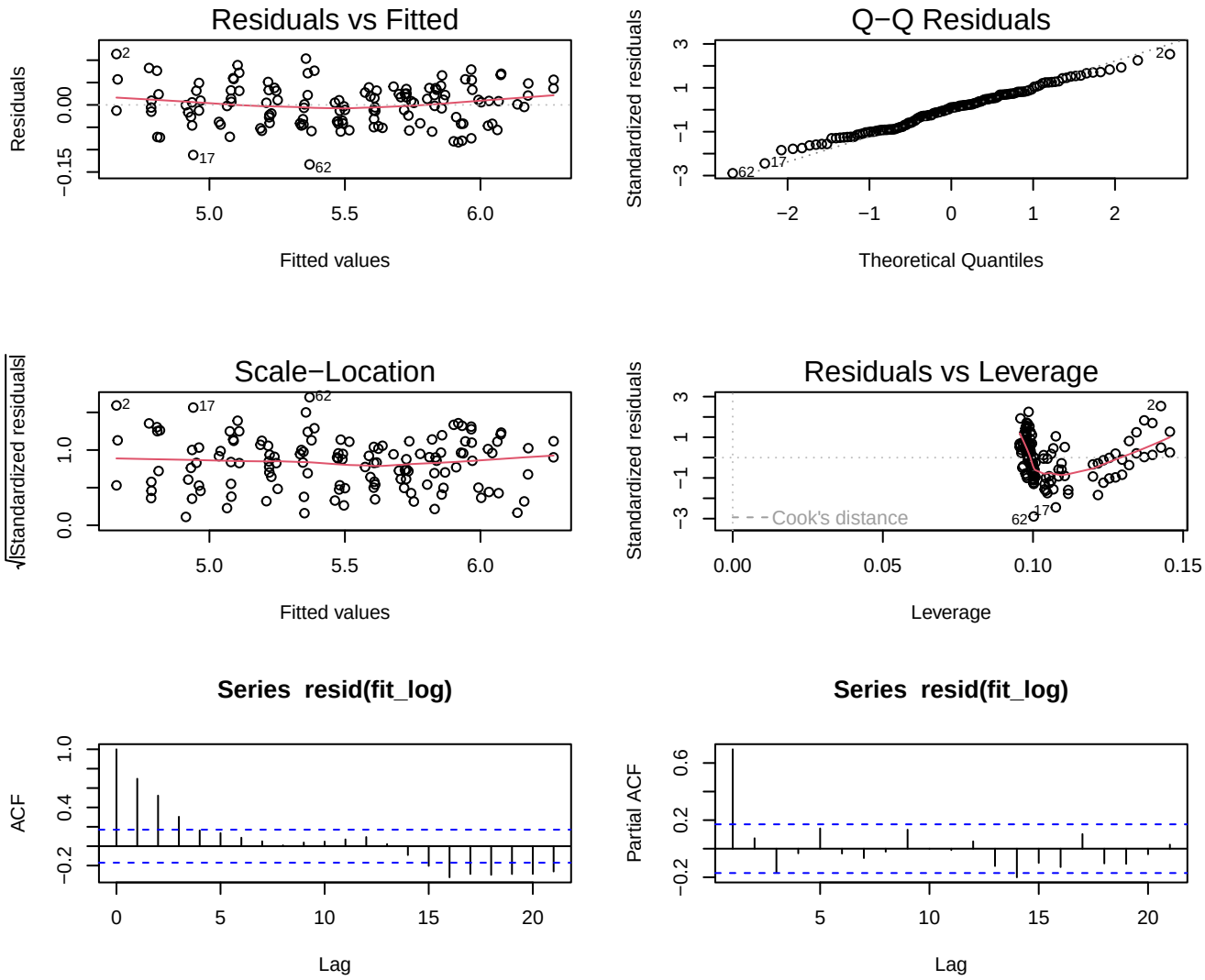
```
ap_sub <- ap_sub %>%
  mutate(
    log_ap = log(val),
    t2 = t^2
  )
fit_log <- lm(log_ap ~ t + t2 + month, ap_sub)
summary(fit_log)
```

```
Call:
lm(formula = log_ap ~ t + t2 + month, data = ap_sub)

Residuals:
     Min       1Q   Median       3Q      Max
-0.13301 -0.03867  0.00465  0.03212  0.11373

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.648e+00  1.875e-02 247.965  < 2e-16 ***
t            1.333e-02  4.461e-04  29.881  < 2e-16 ***
t2          -2.272e-05  3.248e-06  -6.994 1.72e-10 ***
month2      -1.776e-02  2.066e-02  -0.859 0.391862
month3       1.185e-01  2.066e-02   5.735 7.64e-08 ***
month4       7.626e-02  2.067e-02   3.690 0.000340 ***
month5       7.211e-02  2.067e-02   3.489 0.000682 ***
month6       1.946e-01  2.067e-02   9.415 4.87e-16 ***
month7       2.950e-01  2.067e-02  14.268  < 2e-16 ***
month8       2.879e-01  2.068e-02  13.925  < 2e-16 ***
month9       1.469e-01  2.068e-02   7.105 9.83e-11 ***
month10      5.869e-03  2.069e-02   0.284 0.777119
month11     -1.349e-01  2.069e-02  -6.517 1.85e-09 ***
month12     -1.903e-02  2.070e-02  -0.919 0.359814
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04846 on 118 degrees of freedom
Multiple R-squared:  0.9878,    Adjusted R-squared:  0.9864
F-statistic: 734.4 on 13 and 118 DF,  p-value: < 2.2e-16
```

11. Reassess the assumptions using the new model.

```
par(mfrow = c(3,2))
plot(fit_log)
acf(resid(fit_log))
pacf(resid(fit_log))
```



- The independence assumption remains mostly unchanged. We still expect serial correlation since observations are collected over time, and the ACF and PACF plots still suggest that there is serial autocorrelation present in the residuals. Based on the PACF plot, it appears to be largely driven by an AR(1) process.
- The residuals vs fitted values plot looks quite a bit better, showing very slight curvature and no evidence of fanning, suggesting that the changes we made to the model resulted in greater congruence with the linearity and constant variance assumptions.
- The points in the QQ plot still show a slight violation of the normality assumption, but we should keep in mind that we have over 100 observations in this data set. Making normal inference is likely still reasonable here, since the Central Limit Theorem tells us that the sampling distribution of our sample means (the coefficients) are likely approximately normally distributed with over 100 observations.

12. If the residuals are positively serially correlated, we will tend to underestimate the standard errors of the regression coefficients. What impact does this have when determining statistical significance of regression coefficients? It may be helpful to know that the t-test provided in the R output is calculated as

$$t = \frac{\hat{\beta}}{SE(\hat{\beta})}$$

If we under estimate the standard errors, we overestimate the t statistic, leading to erroneously small p-values. Therefore, we are more likely to conclude that regression coefficients are "statistically significant" in the presence of positive serial autocorrelation, even if they shouldn't be.

13. With the exception of the independence assumption, hopefully you are convinced that the model created in 10 is a reasonable model for the `AirPassengers` data set. Next, calculate the fitted values by hand and compare them to the values from `fitted`.

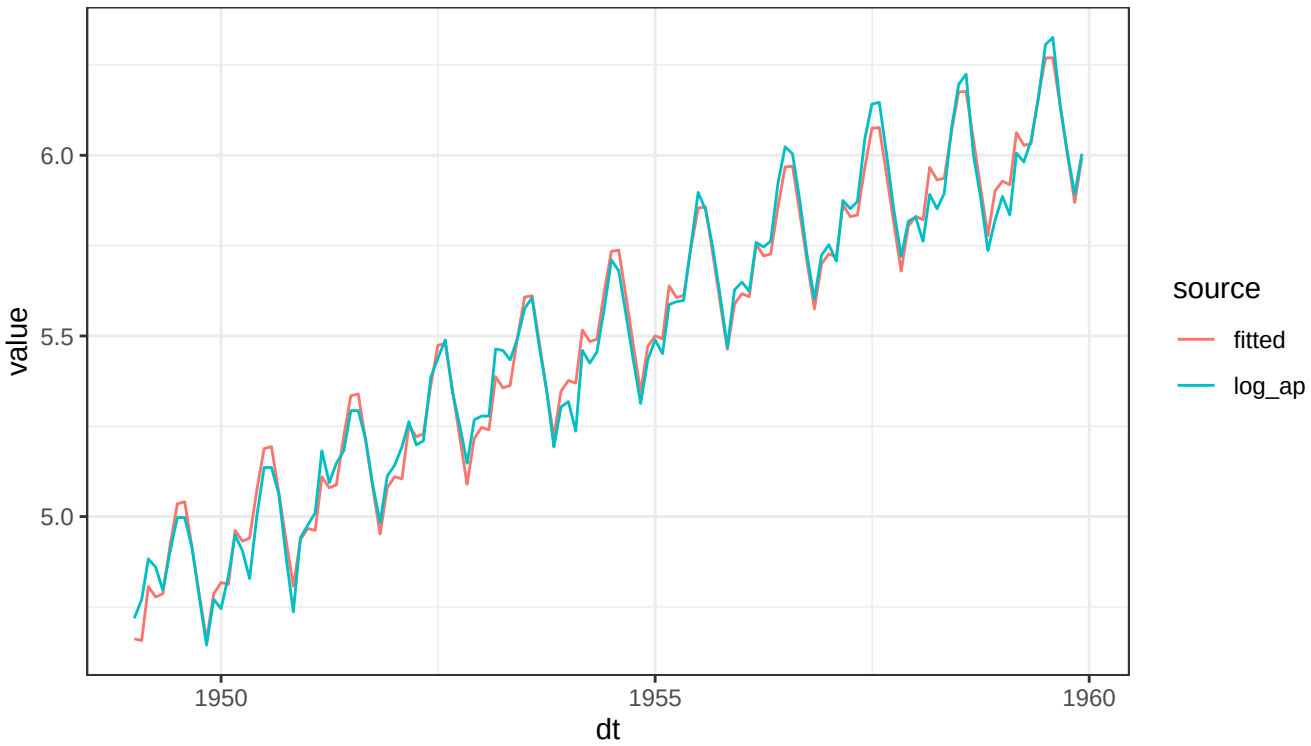
```
# use matrix algebra to spend it up
X <- model.matrix(fit_log)
fitted_byhand <- c(X %*% matrix(coef(fit_log), ncol = 1))
all((fitted_byhand - fitted(fit_log)) < 1e6)
```

```
[1] TRUE
```

14. Plot the observed and fitted time series on a single plot and comment on how well the model estimates the observed relationship.

```
ap_sub %>%
  mutate(fitted = fitted(fit_log)) %>%
  pivot_longer(
    c("log_ap", "fitted"),
    names_to = "source",
```

```
    values_to = "value"
  ) %>%
  mutate(
    year_true = 1948 + year,
    dt = mdy(paste0(month, "-1-", year_true))
  ) %>%
  ggplot() +
  geom_line(aes(x = dt, y = value, col = source)) +
  theme_bw()
```



> Looks pretty good to me! The fitted matches the observed logged passengers quite well.

15. Calculate the residuals by hand and compare them to the values obtained from `resid`.

```
resid_byhand <- ap_sub$log_ap - fitted_byhand
all((resid_byhand - resid(fit_log)) < 1e6)
```

```
[1] TRUE
```

16. Use the `fit_log` model to forecast the time series for the year of 1960 (which we had previously excluded) using the `predict` function. The predict function requires a `newdata` argument to obtain forecasts - this data frame must include all the predictors used in the model. Plot the

observed, fitted, and forecasted series on a single plot and include a 95% prediction interval for the forecasted series. For more about predicting from `lm` models, run `?predict.lm`.

```
# new data
ap_pred_tbl <- ap_tbl %>%
  filter(year == 12) %>%
  mutate(
    log_ap = log(val),
    t2 = t^2
  )

# obtain predictions
ap_pred <- predict(
  fit_log, newdata = ap_pred_tbl, interval = "prediction"
) %>%
  as_tibble %>%
  mutate(
    source = "fitted",
    year = 1960,
    month = 1:12
  ) %>%
  mutate(dt = ymd(paste0(year, "-", month, "-1")))

# plot nicely
ap_tbl %>%
  mutate(log_ap = log(val)) %>%
  mutate(fitted = c(fitted(fit_log), rep(NA, 12))) %>%
  pivot_longer(
    c("log_ap", "fitted"),
    names_to = "source",
    values_to = "value"
  ) %>%
  mutate(
    year_true = 1948 + year,
    dt = mdy(paste0(month, "-1-", year_true))
  ) %>%
  ggplot() +
  geom_line(aes(x = dt, y = value, col = source)) +
  geom_vline(aes(xintercept = ymd("1960-1-1")), linetype = "dotdash") +
  geom_ribbon(
    data = ap_pred,
    aes(x = dt, ymin = lwr, ymax = upr), alpha = .20
  ) +
  geom_line(
```

```
    data = ap_pred,
    aes(x = dt, y = fit, col = source), linetype = "dotted"
  ) +
  theme_bw() +
  labs(
    x = "Date",
    y = "log(Air Passengers (1000s))"
  )
```