ADS 506 - Time Series

Fall 2023 - Week 4 OH

Dave Hurst

Agenda

- Follow-up on ARIMA
  - Last week OH questions
  - Last week assignment
  - model intuition
- Iteration in R
- Assignment 4.1 Hints
- Quiz and Assignment prep –>Erin

ARIMA Questions / Intuitions

- How is PACF calculated?
- How are they interpreted?

AR Models

Generalized AR(p) model:

$$Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \epsilon_t$$

Special case: AR(1) model:

$$Y_t = c + \varphi_1 Y_{t-1} + \epsilon_t$$

AR Models - As code

Special case: AR(1) model:

$$Y_t = c + \varphi_1 Y_{t-1} + \epsilon_t$$

▼ Code

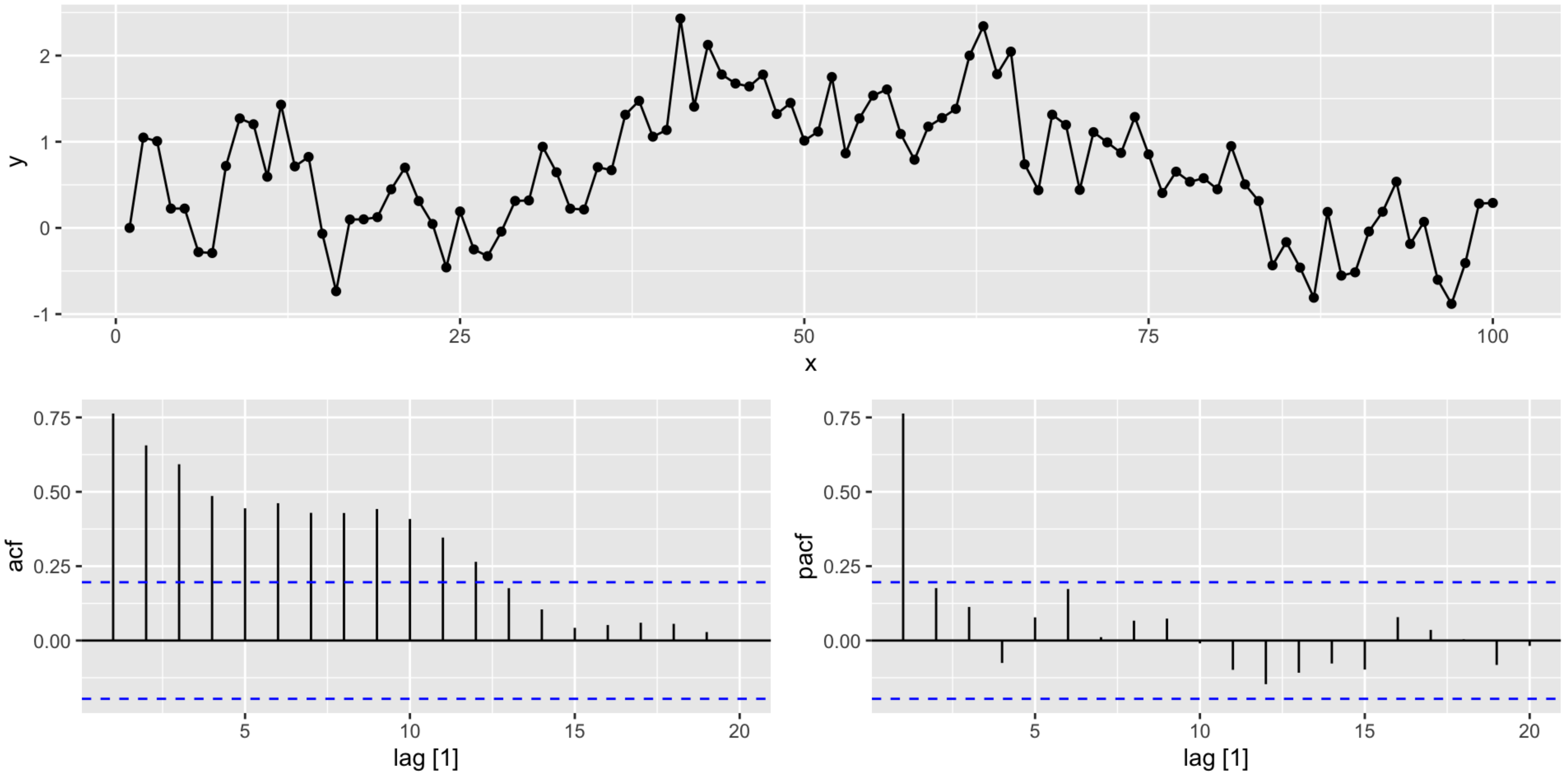
```r
set.seed(506)
ar1_model <- function(phi = 0, c = 0, sd = 0.5,
                      n = 100, seed = NULL) {
    if (!is.null(seed)) set.seed(seed)
    noise <- rnorm(n, sd = sd)
    y <-  c
    for (i in 2:length(noise)) {
        y[i] <- y[i-1] * phi + c + noise[i]
    }
    tsibble(x = 1:n, y = y, index = x)
}
```

```r
ar1_model(phi = 0.8, c = 0)
```

```
# A tsibble: 100 x 2 [1]
       x      y
   <int>  <dbl>
 1     1  0
 2     2 -0.899
 3     3 -0.138
 4     4  0.138
 5     5  0.353
 6     6  1.04
 7     7  0.789
 8     8  1.69
 9     9  1.43
10    10  0.983
# ℹ 90 more rows
```

AR(1) Models

```r
ar1_model(phi = 0.8, c = 0) |>
    gg_tsdisplay(y, plot_type = 'partial')
```

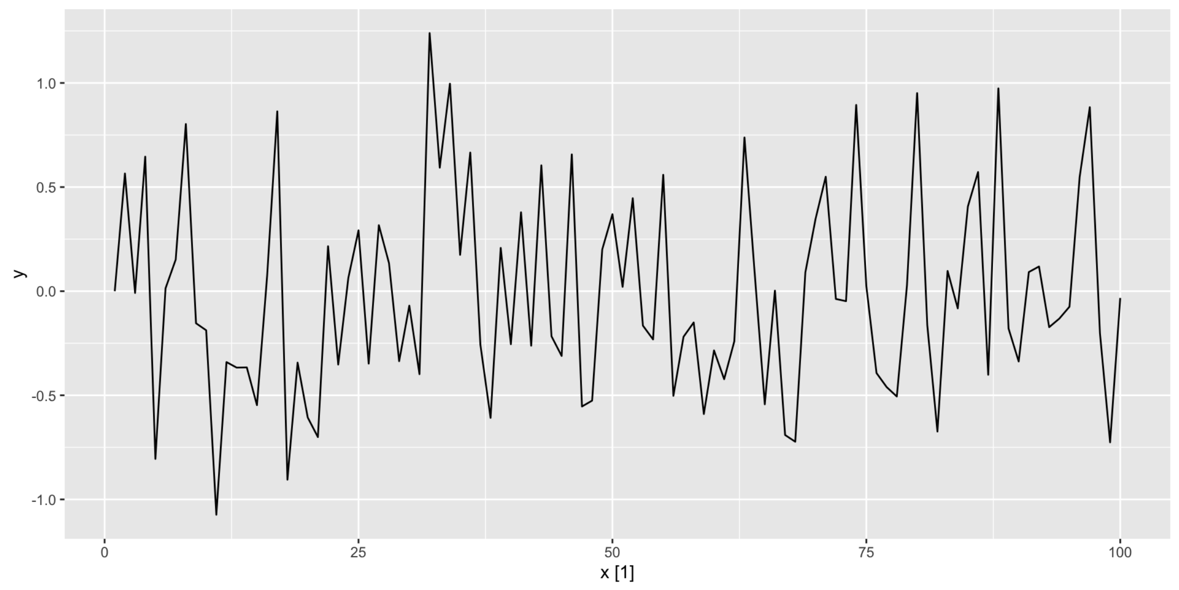


AR(1) Models

White noise

$$AR(1): \varphi = 0, c = 0$$

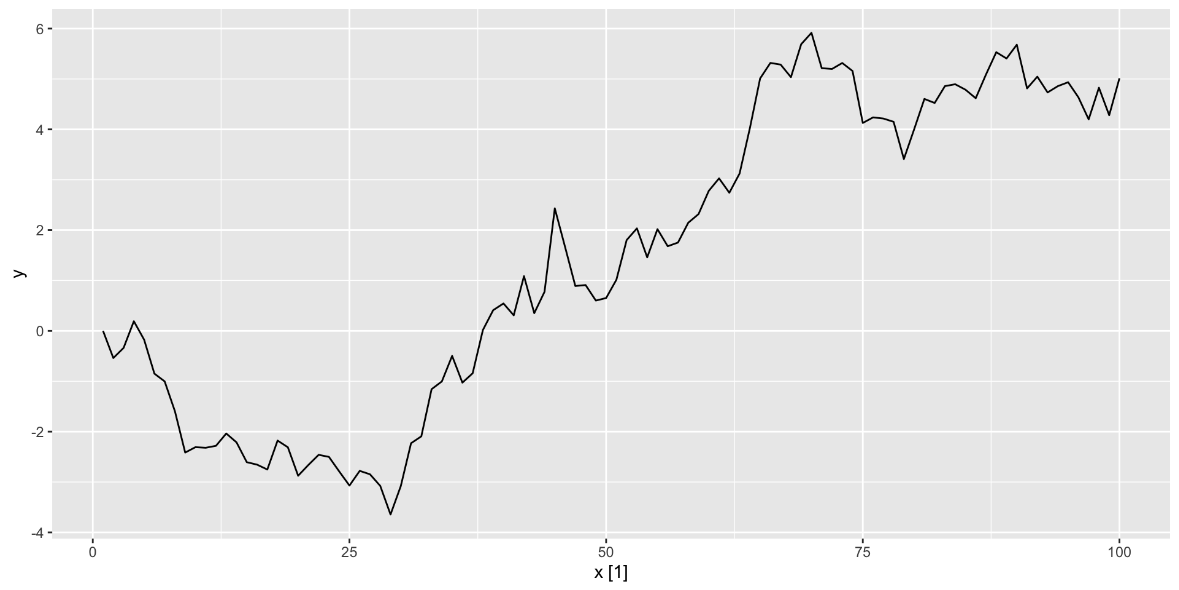
```r
ar1_model(phi = 0, c = 0) |>
    autoplot(y)
```



Random walk

$$AR(1): \varphi = 1, c = 0$$

```r
ar1_model(phi = 1, c = 0) |>
    autoplot(y)
```

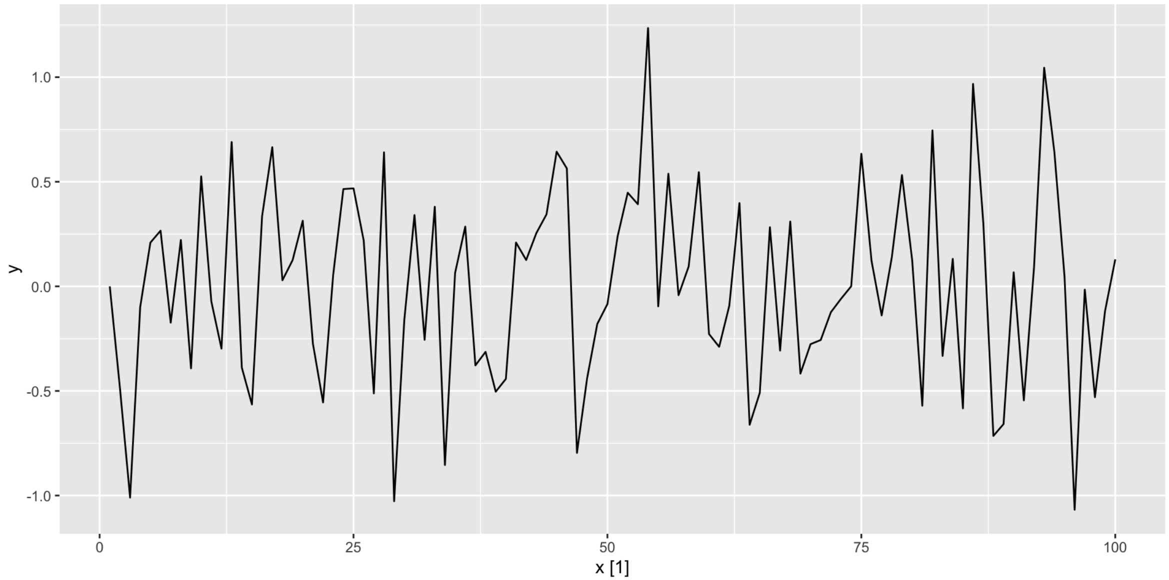


AR(1) Models (repeat w/ different seed)

White noise

$$\mathrm{AR}(1): \varphi = 0, c = 0$$

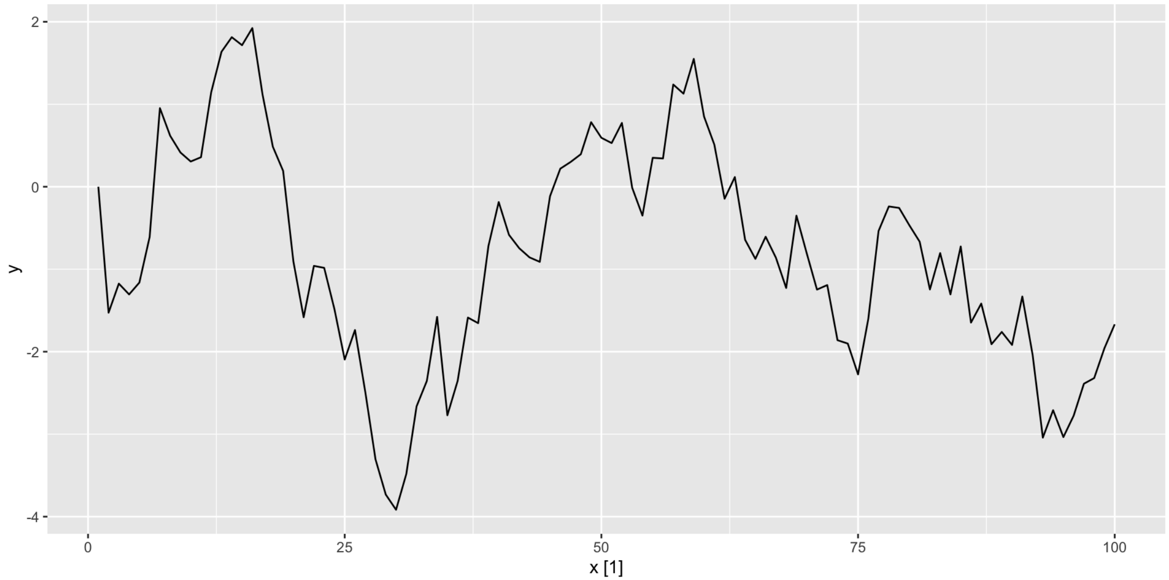
```
ar1_model(phi = 0, c = 0) |>
    autoplot(y)
```



Random walk

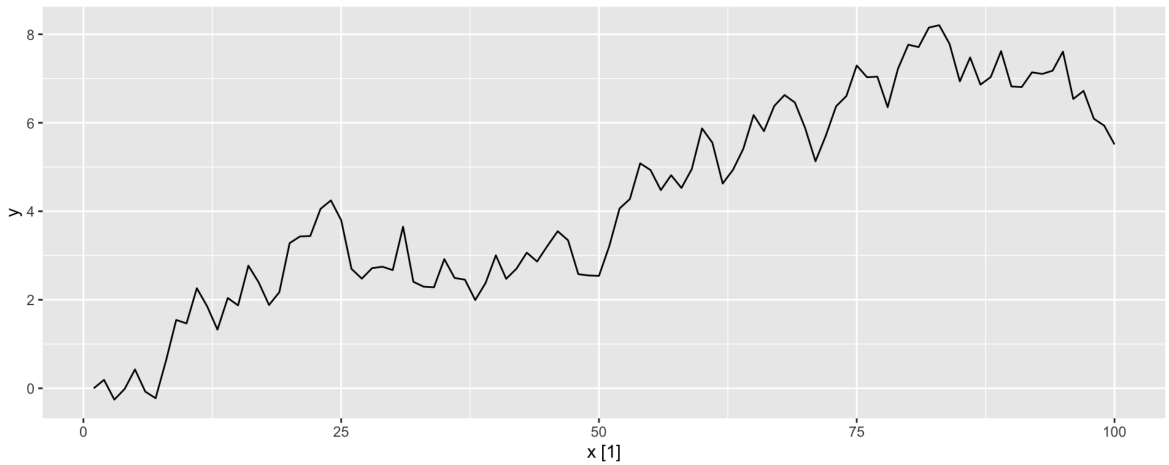
$$\mathrm{AR}(1): \varphi = 1, c = 0$$

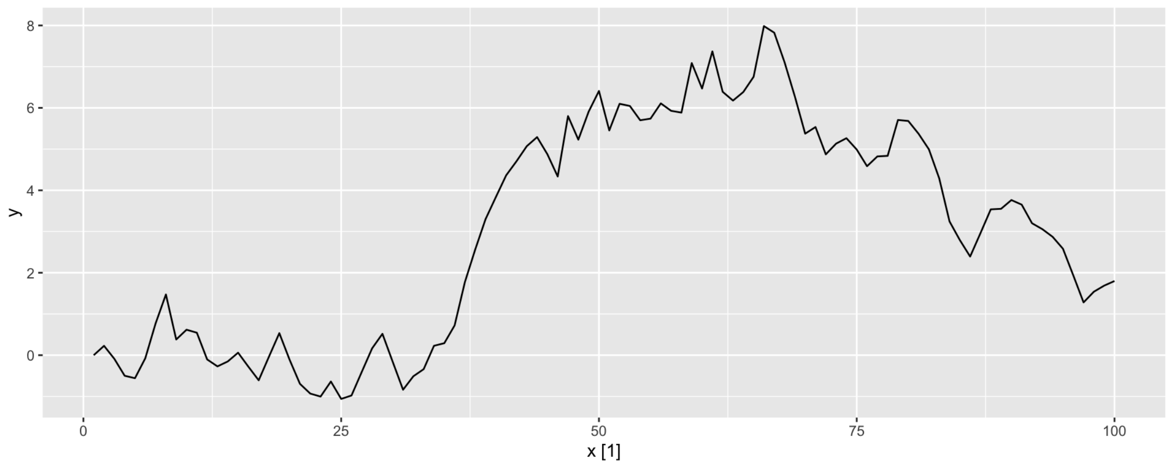
```
ar1_model(phi = 1, c = 0) |>
    autoplot(y)
```



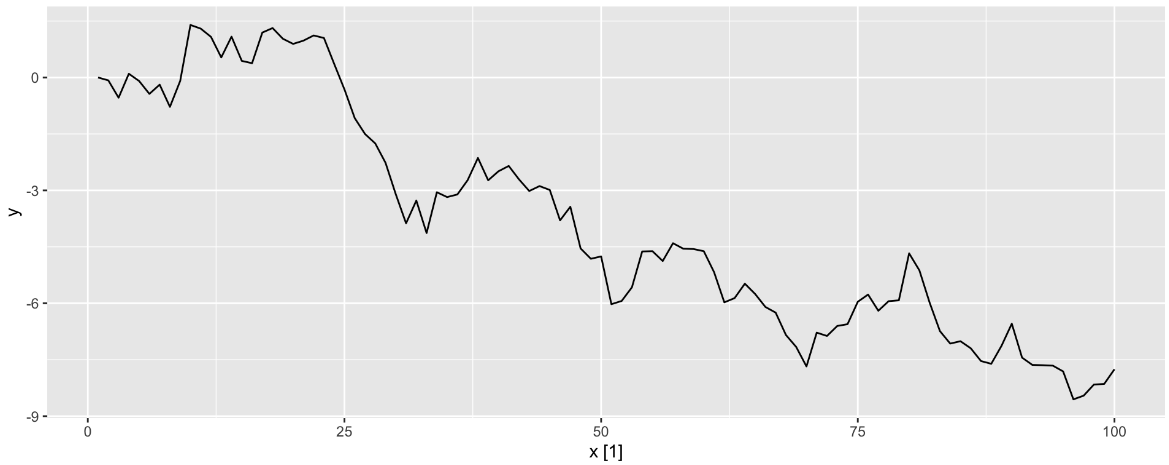
4 Random Walks

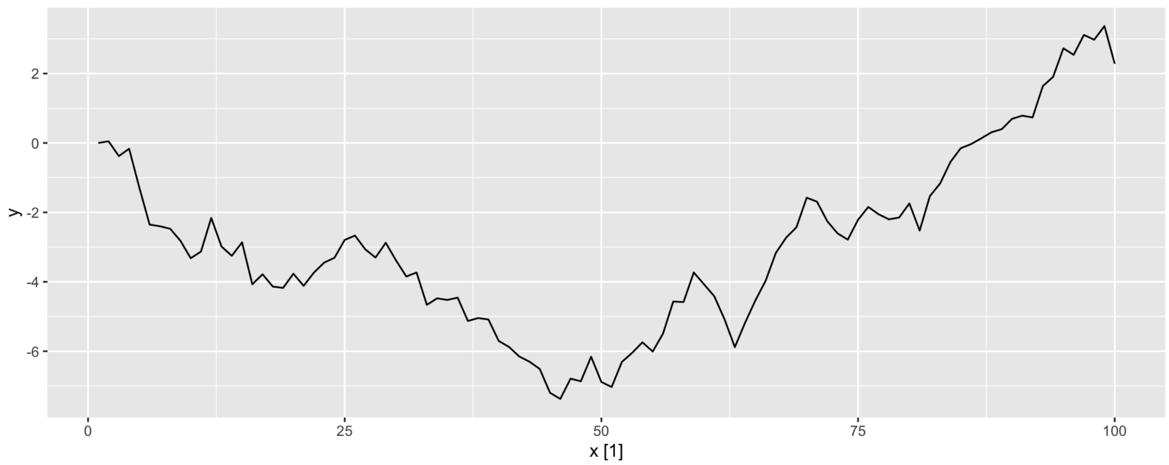
▶ Code


8
6
4
2
0
y
0
25
50
75
100
x [1]


8
6
4
2
0
y
0
25
50
75
100
x [1]

▶ Code

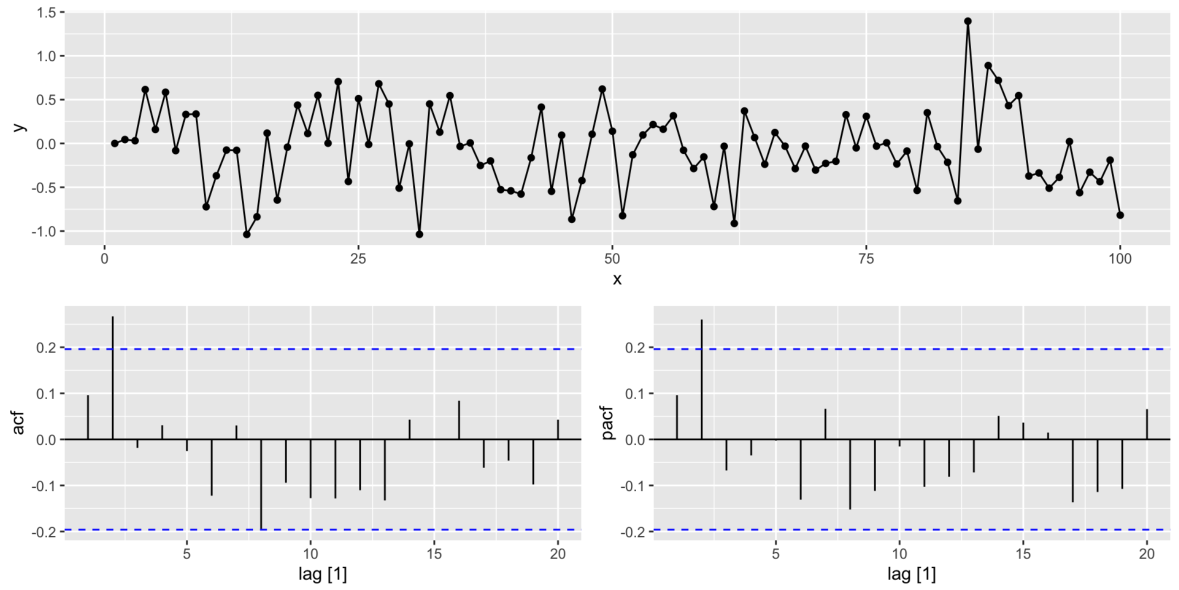

0
-3
-6
-9
y
0
25
50
75
100
x [1]


2
0
-2
-4
-6
y
0
25
50
75
100
x [1]

AR(1) Models - ACF / PACF

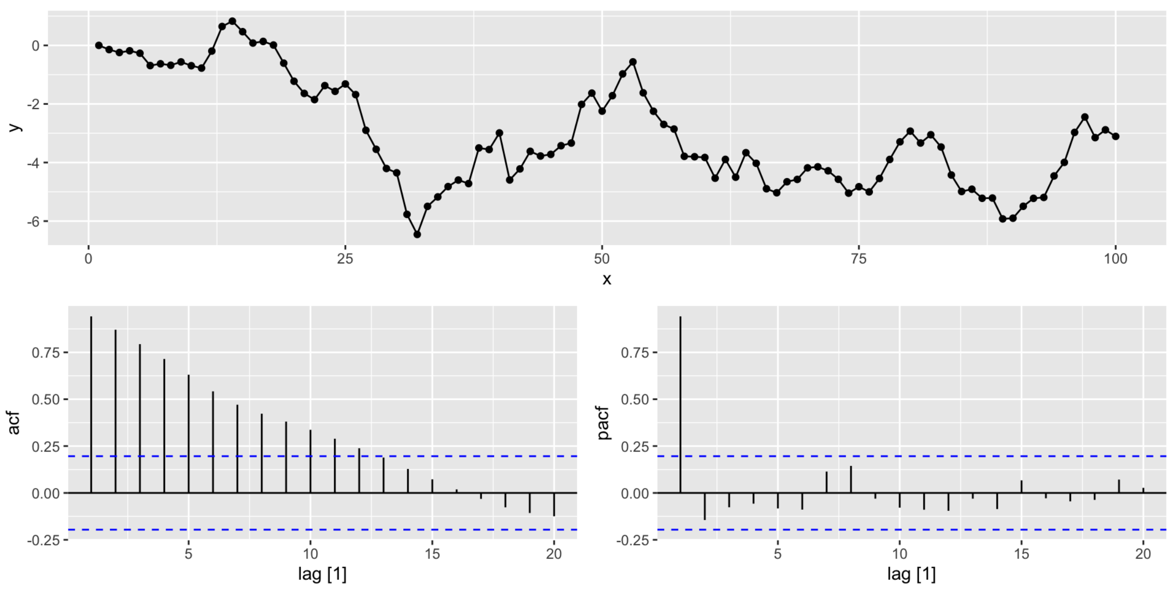
$$\text{AR}(1) : \varphi = 0, c = 0$$

```r
ar1_model(phi = 0, c = 0) |>
    gg_tsdisplay(y, plot_type = 'partial')
```



$$\text{AR}(1) : \varphi = 1, c = 0$$

```r
ar1_model(phi = 1, c = 0) |>
    gg_tsdisplay(y, plot_type = 'partial')
```



👆 **MA(q)** **AR(p)** ☝️

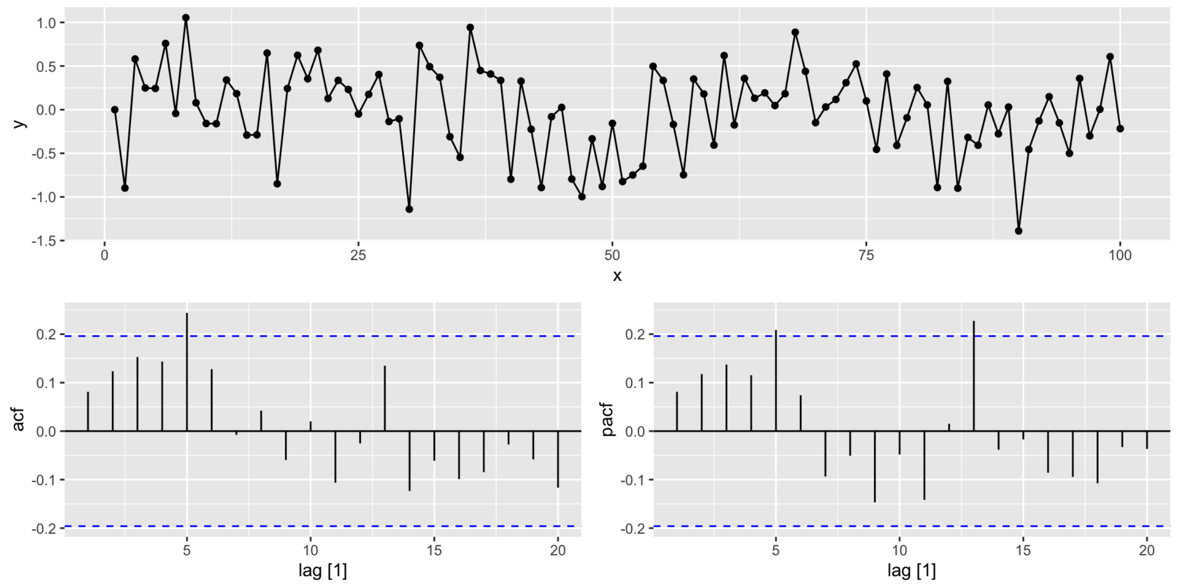
Inferred ARIMA Models

AR(1) Models - ACF / PACF (change seed)

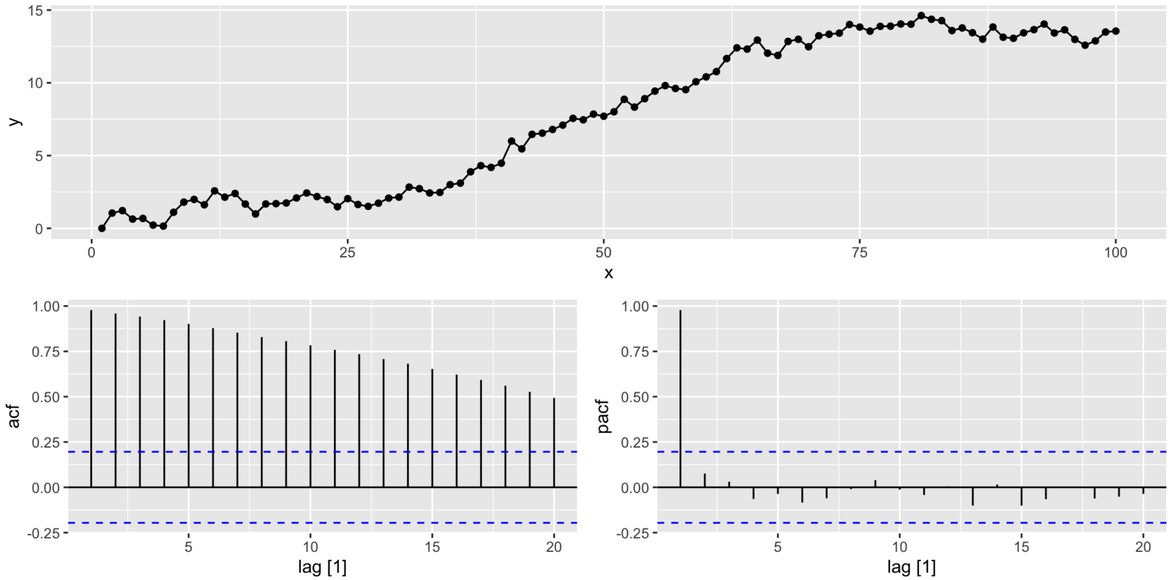
$$\text{AR}(1) : \varphi = 0, c = 0$$

$$\text{AR}(1) : \varphi = 1, c = 0$$

```r
ar1_model(phi = 0, c = 0,
          seed = 506) |>
    gg_tsdisplay(y, plot_type = 'partial')
```




```r
ar1_model(phi = 1, c = 0) |>
    gg_tsdisplay(y, plot_type = 'partial')
```

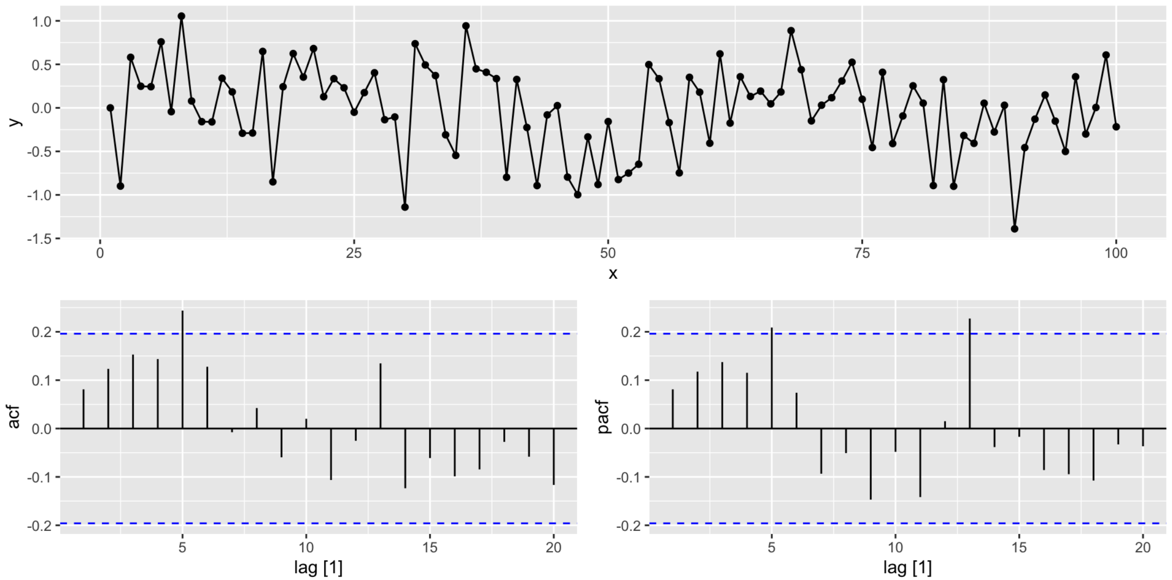


Calculating ACF / PACF

- With a side note on iteration in R (w/ purrr)

{r}

```
ar1_506 <- ar1_model(phi = 0, c = 0, seed =
ar1_506 |>      gg_tsdisplay(y, plot_type =
```




```
ar1_506 |> ACF(y)
```

```
# A tsibble: 20 x 2 [1]
        lag      acf
   <cf_lag>    <dbl>
 1        1   0.0810
 2        2   0.123
 3        3   0.153
 4        4   0.144
 5        5   0.244
 6        6   0.128
 7        7 -0.00765
 8        8   0.0424
 9        9 -0.0595
10       10   0.0201
11       11 -0.106
12       12 -0.0253
```

Calculating ACF

ACF lags(1-5): 0.081, 0.123, 0.153, 0.144, 0.244

@lag1

… as a function …

```r
acf_lag <- function(y, lag = 1) {
    cor(y, lag(y, n = lag),
        use = "complete.obs")
}

acf_lag(ar1_506$y, 1)
```

```
[1] 0.08105321
```

@lags 1-5

With a traditional `for` loop

```r
# traditional for loop
acf_vals <- numeric(length = 5)
for (i in 1:length(acf_vals)) {
    acf_vals[i] <- acf_lag(ar1_506$y, i)
}
acf_vals
```

```
[1] 0.08105321 0.12668603 0.15797526
0.14879380 0.25368453
```

With `purrr::map_...`

```r
map_dbl(1:5, ~acf_lag(ar1_506$y, .x))
```

```
[1] 0.08105321 0.12668603 0.15797526
0.14879380 0.25368453
```

Calculating PACF (using lm)

ACF lags(1-5): 0.081, 0.1176, 0.1373, 0.1152, 0.2088

@lag1

```
with(ar1_506, lm(y ~ lag(y, n = 1)))
```

```
Call:
lm(formula = y ~ lag(y, n = 1))

Coefficients:
  (Intercept)  lag(y, n = 1)
     -0.01846        0.08112
```

```
pacf_lag <- function(y, lag = 1) {
    lm(y ~ lag(y, n = lag))$coef[2] |> as.n
}

pacf_lag(ar1_506$y, 1)
```

```
[1] 0.08111736
```

@lags 1-5

```
purrr::map_dbl(1:5, ~pacf_lag(ar1_506$y, .x
```

```
[1] 0.08111736 0.12578046 0.15568567
0.14664894 0.25040847
```

Calculating PACF (using ARIMA(1,0,0))

ACF lags(1-5): 0.081, 0.118, 0.137, 0.115, 0.209

@lag1

```
ar1_506_ar1 <- ar1_506 |>
    model(ARIMA(y ~ pdq(1, 0, 0)))
ar1_506_ar1 |>
    report()
```

```
Series: y
Model: ARIMA(1,0,0)

Coefficients:
         ar1
      0.0816
s.e.  0.0992

sigma^2 estimated as 0.2485:  log
likelihood=-71.79
AIC=147.57   AICc=147.7   BIC=152.78
```

@lags 1-5

```
pacf_lag <- function(y, lag = 1) {

    ar1 <- tsibble(x = 1:length(y),
                   y = y, index = x) |>
        model(ARIMA(y ~ pdq(lag, 0, 0)))
    ar1 |> coef() |> pull(estimate) |> _[la
}

pacf_lag(ar1_506$y, 1)
```

```
[1] 0.08160791
```

```
purrr::map_dbl(1:5, ~pacf_lag(ar1_506$y, .x
```

```
[1] 0.08160791 0.12173879 0.14033962
0.11544244 0.21337158
```

ARIMA Models

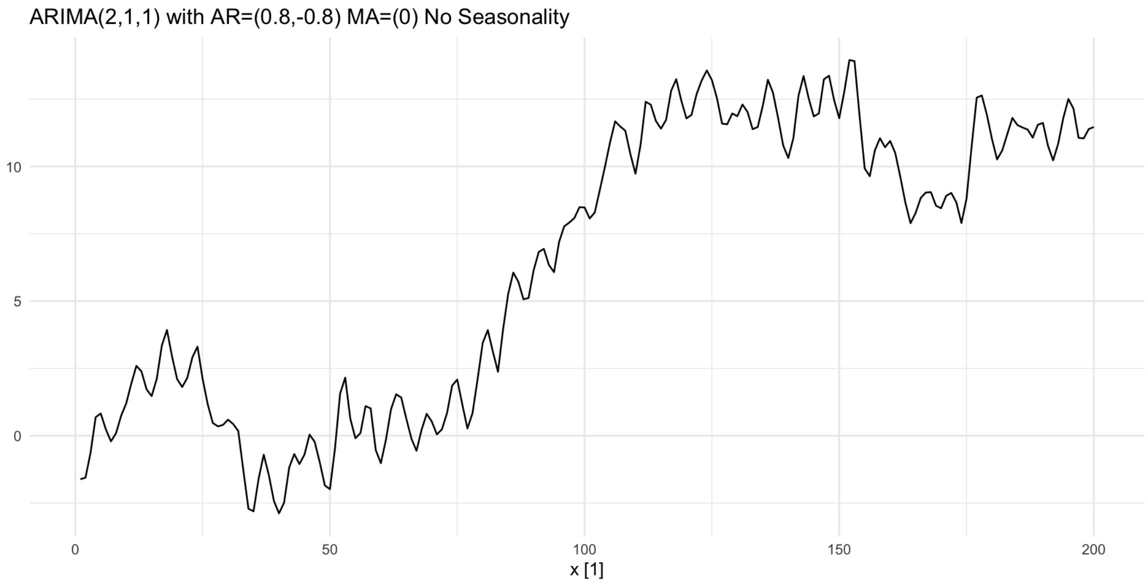
Using `arima.sim()` for more general ARIMA models

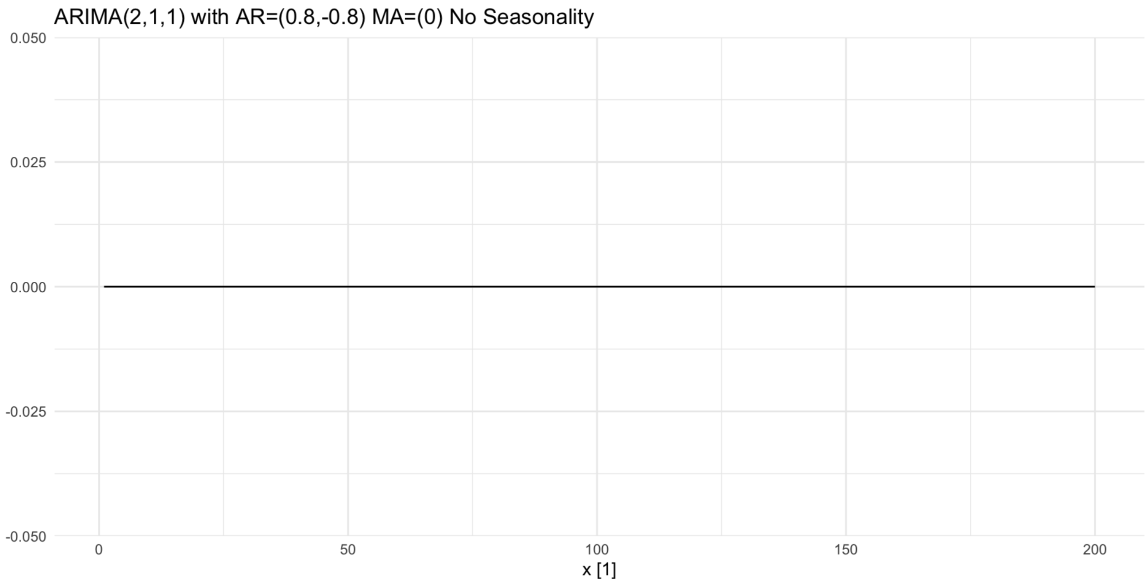
```r
generate_arima_plot <- function(
        diff_order = 1,
        ar_coefficient = c(0.8, -0.4),
        ma_coefficient = c(0.4, -0.8),
        period = 20,  #set period to 1 for no seasonality
        error_sd = 0.5,
        seed = NULL,
        length = 200,
        burnin = 0
) {
    if(! is.null(seed)) set.seed(seed)

    model_order <- c(length(ar_coefficient),
                     diff_order,
                     length(ma_coefficient))


    # Generate the time series
```

ARIMA Models - “error”

```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.8, -0.8),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.5,
    seed = 506
)
```

```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.8, -0.8),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.0,
    seed = 506
)
```

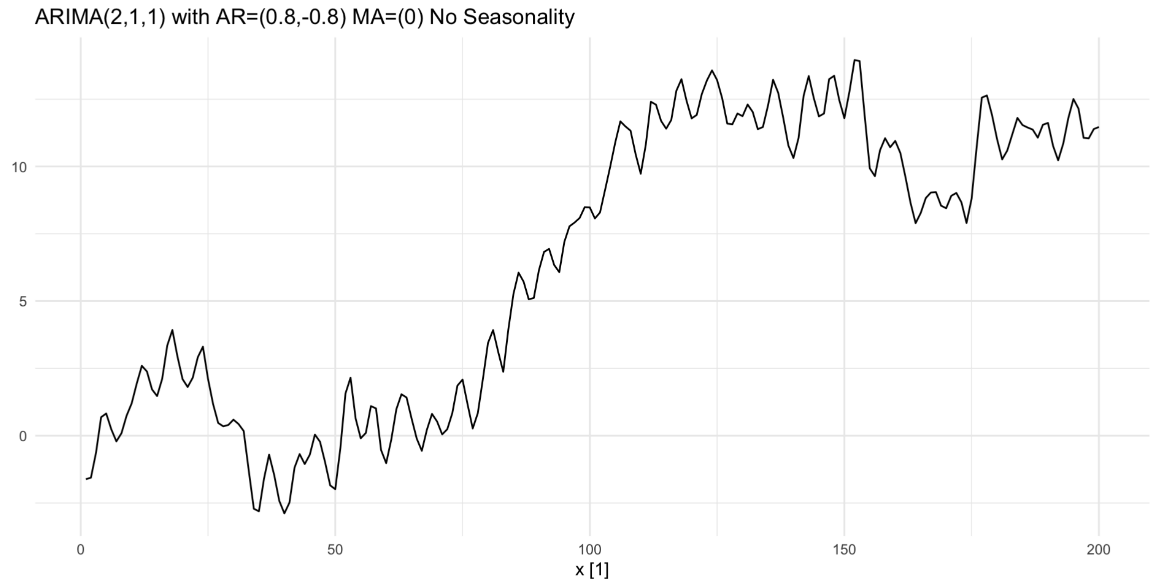


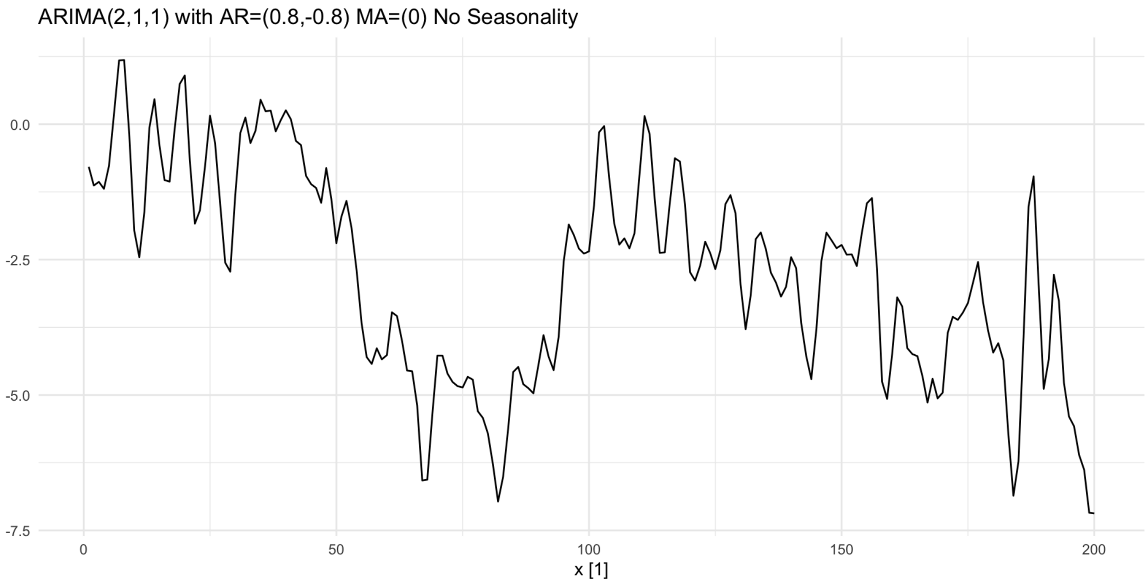


ARIMA Models - “error”

```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.8, -0.8),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.5,
    seed = 506
)
```

```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.8, -0.8),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.5,
    seed = 999
)
```

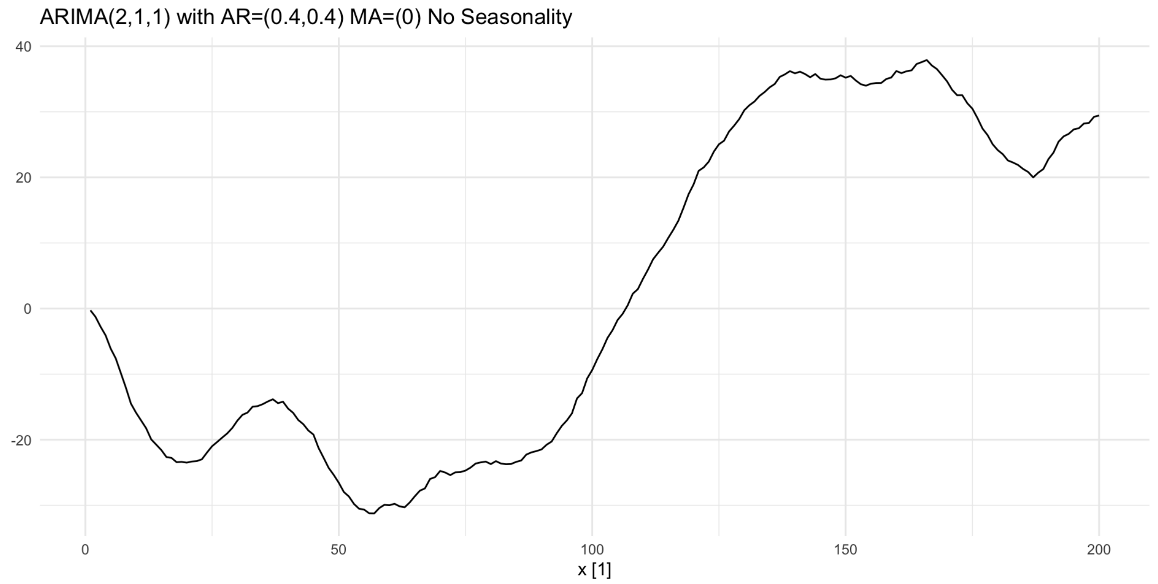


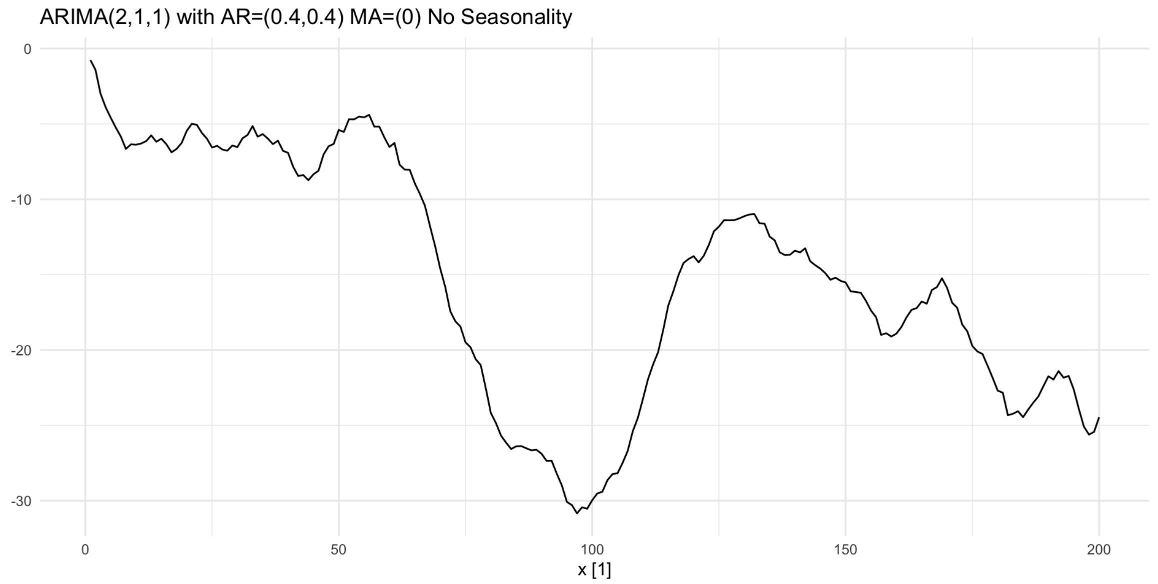


ARIMA Models - parameters

```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.4, 0.4),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.5,
    seed = 506
)
```

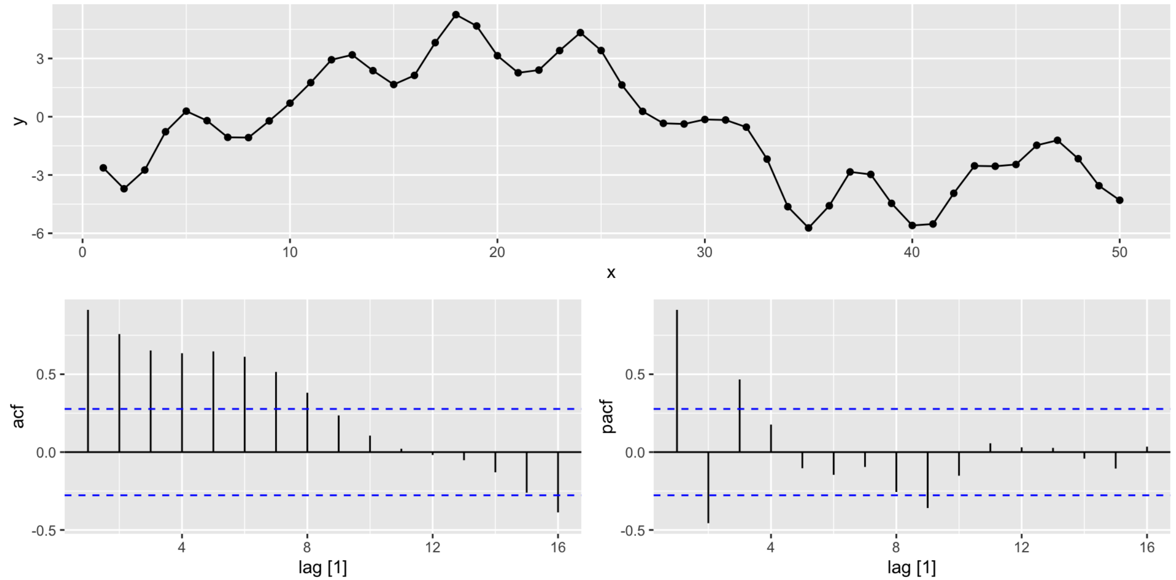
```r
generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.4, 0.4),
    ma_coefficient = c(0.),
    period = 1,
    error_sd = 0.5,
    seed = 999
)
```





ARIMA Models Inference

```r
arima_211a_plt <- generate_arima_plot(
    diff_order = 1,
    ar_coefficient = c(0.8, -0.8),
    ma_coefficient = c(0.7),
    period = 1,
    error_sd = 0.5,
    seed = 506, length = 50
)
arima_211a <- arima_211a_plt$data

arima_211a |>
    gg_tsdisplay(y, plot_type = "partial")
```

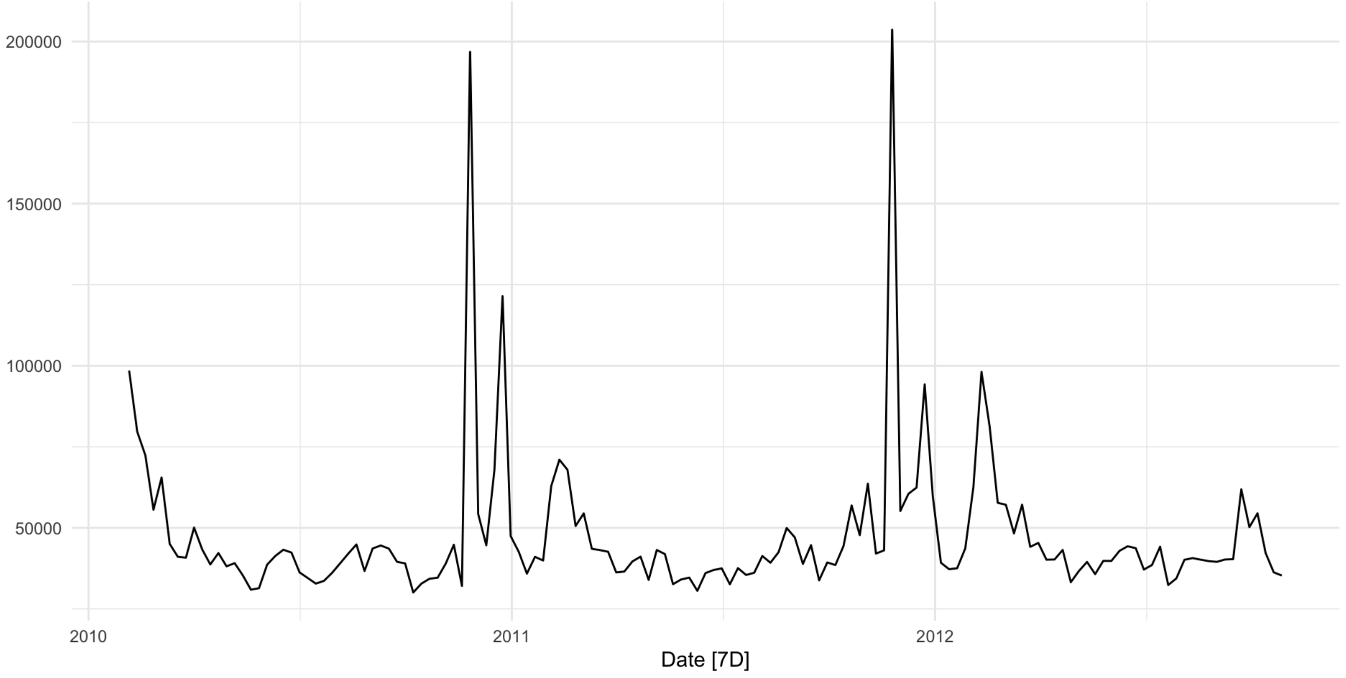



```r
arima_211a |>
    model(ARIMA(y))
```

```
# A mable: 1 x 1
      `ARIMA(y)`
         <model>
1 <ARIMA(2,1,1)>
```

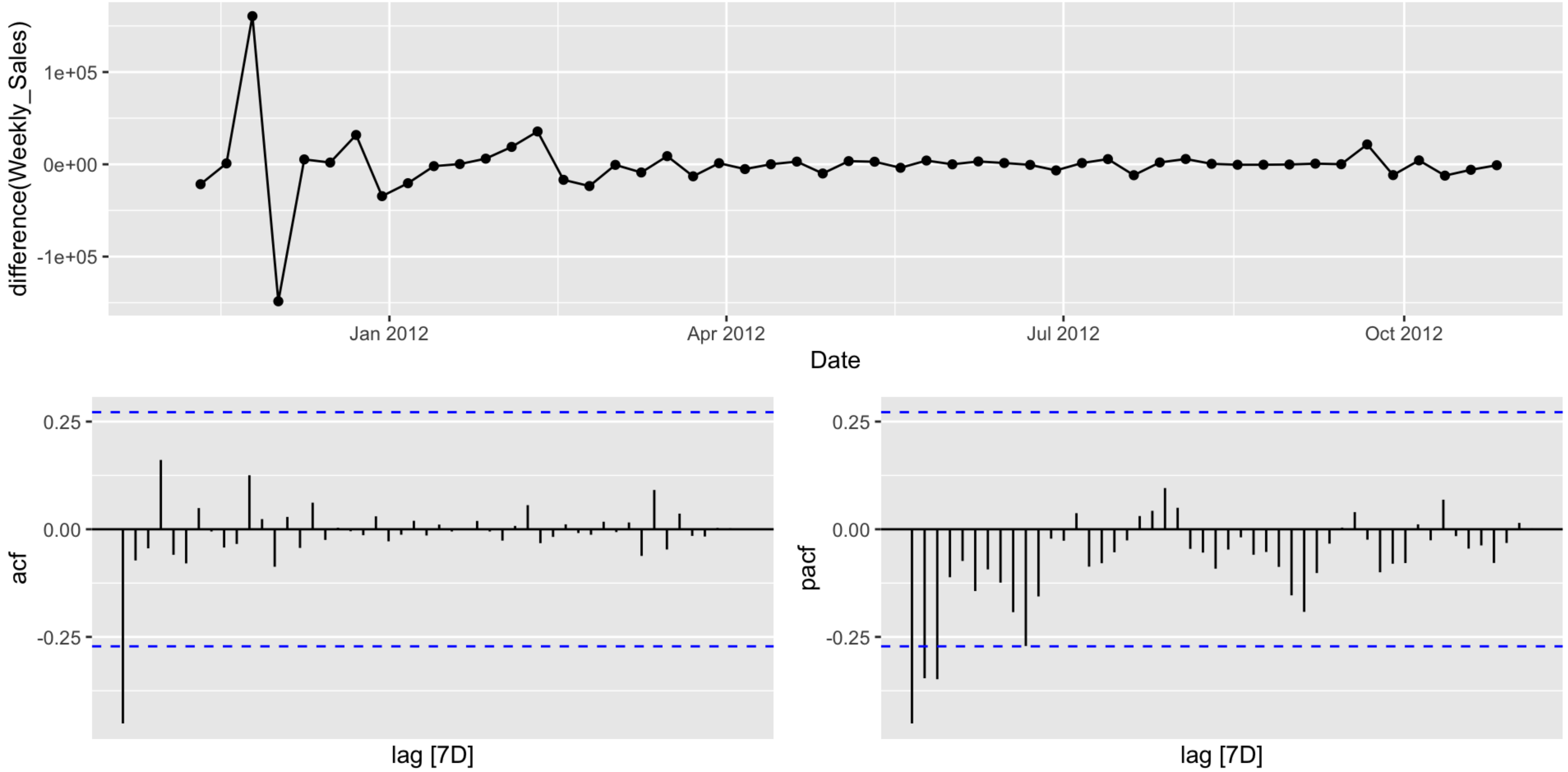
Assignment 3.1 - Retrospective

```r
ws1d72 <- read_csv("WalmartStore1Dept72.csv", show_col_types = FALSE) |>
    mutate(Date = mdy(Date)) |>
    as_tsibble(index = Date)
ws1d72 |>
    autoplot(Weekly_Sales) +
    labs("Weekly Sales in Department #27 of Walmart Store 1",
         y = "") +
    theme_minimal()
```



Assignment 3.1 – restricted training set

```
ws1d72_trn <- ws1d72 |> filter_index("2011-11-04" ~ "2012-10-26")
ws1d72_trn |>
    gg_tsdisplay(
        Weekly_Sales |> difference() ,
        plot_type = "partial", lag_max = 55)
```



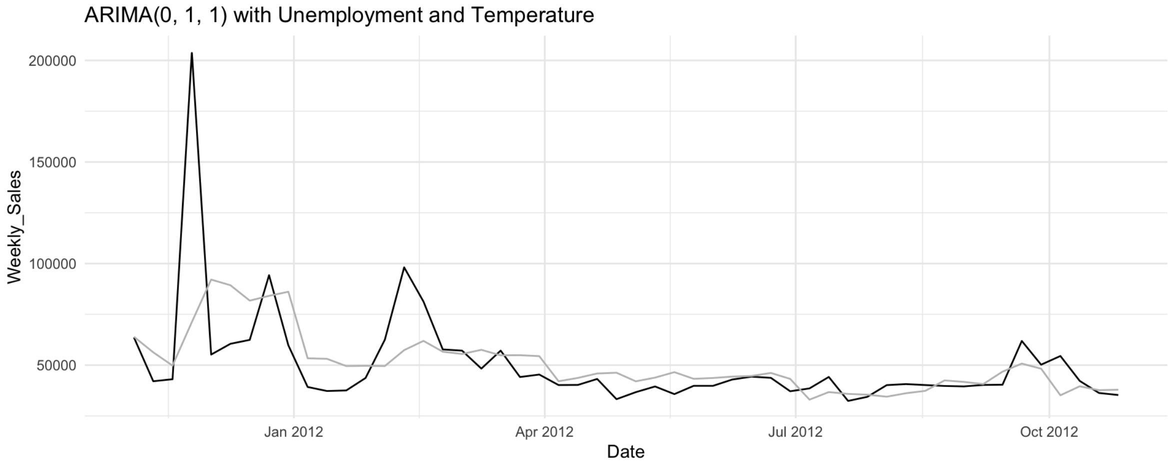
Assignment 3.1 Hints – ARIMA model selection

```
ws1d72_trn_fit <- ws1d72_trn |>
    model(
        x2_ar3 = ARIMA(Weekly_Sales ~ pdq(3, 1, 0) + 1 +
                             Unemployment + Temperature),
        x2_ma1 = ARIMA(Weekly_Sales ~ pdq(0, 1, 1) + 1 +
                             Unemployment + Temperature),
        auto_arima = ARIMA(Weekly_Sales),
        auto_arima_x2 = ARIMA(Weekly_Sales ~ 1 +
                                    Unemployment + Temperature)
    )
```

▶ Code

.model	.type	RMSE
x2_ma1	Training	22122.77
x2_ar3	Training	22305.41
auto_arima_x2	Training	22511.44
auto_arima	Training	24622.78

▶ Code



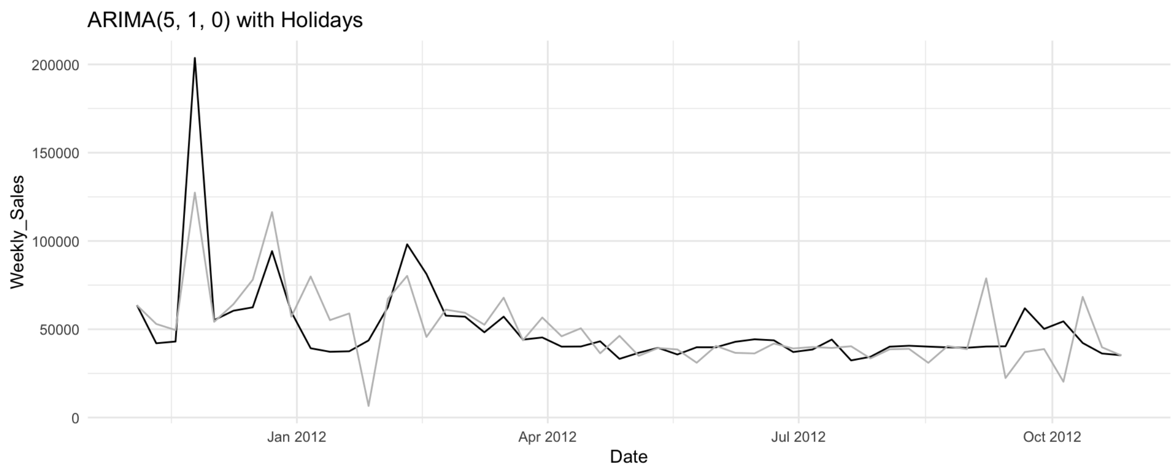
Assignment 3.1 Better Hints – faux seasonality

```
ws1d72_trn_fit2 <- ws1d72_trn |>
    model(
        auto_arima = ARIMA(Weekly_Sales),
        auto_holiday = ARIMA(Weekly_Sales ~ 1 + IsHoliday),
        ma1_holiday = ARIMA(Weekly_Sales ~ pdq(0, 1, 1) + 1 + IsHoliday),
        ar3_holiday = ARIMA(Weekly_Sales ~ pdq(3, 1, 0) + 1 + IsHoliday)
    )
```

▶ Code

▶ Code

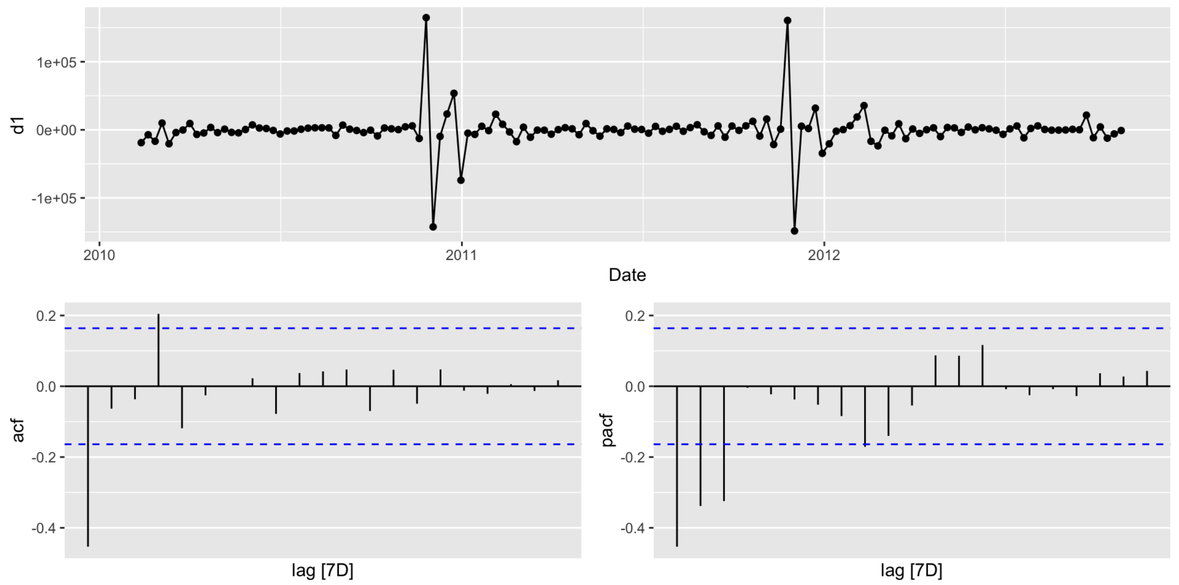
.model	.type	RMSE
auto_arima	Training	24622.78
auto_holiday	Training	18393.43
ma1_holiday	Training	19429.49
ar3_holiday	Training	20131.32

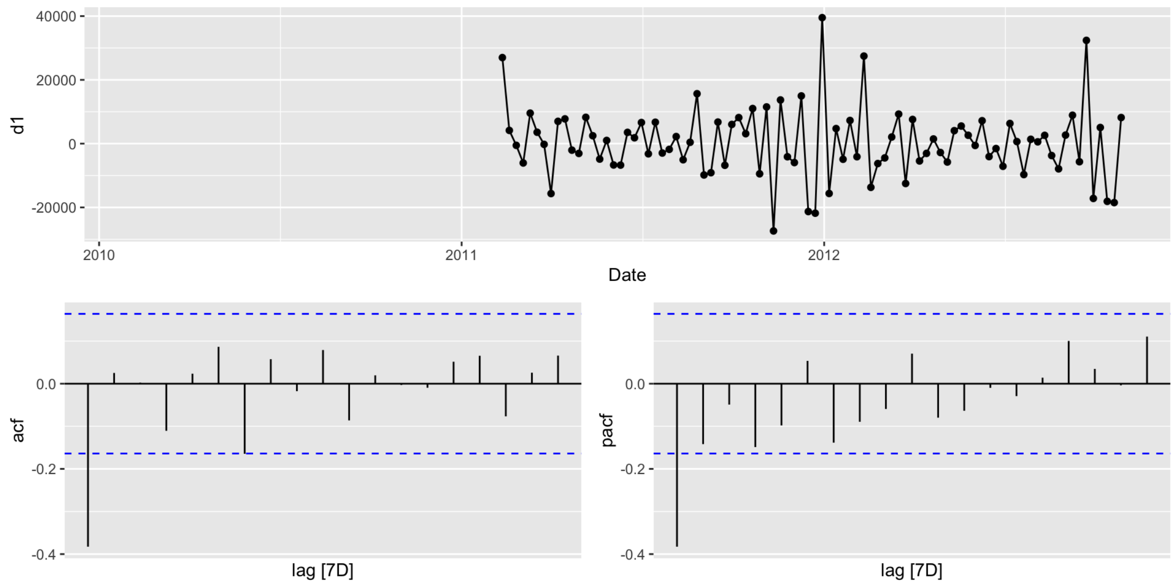


Assignment 3.1 – True seasonality (model selection)

```r
ws1d72 |>
    mutate(d1 = Weekly_Sales |>
                difference(lag = 1)) |>

    gg_tsdisplay(d1, plot_type = "partial"
```

```r
ws1d72 |>
    mutate(d1 = Weekly_Sales |>
                difference(lag = 1) |>
                difference(lag = 52)) |>
    gg_tsdisplay(d1, plot_type = "partial"
```





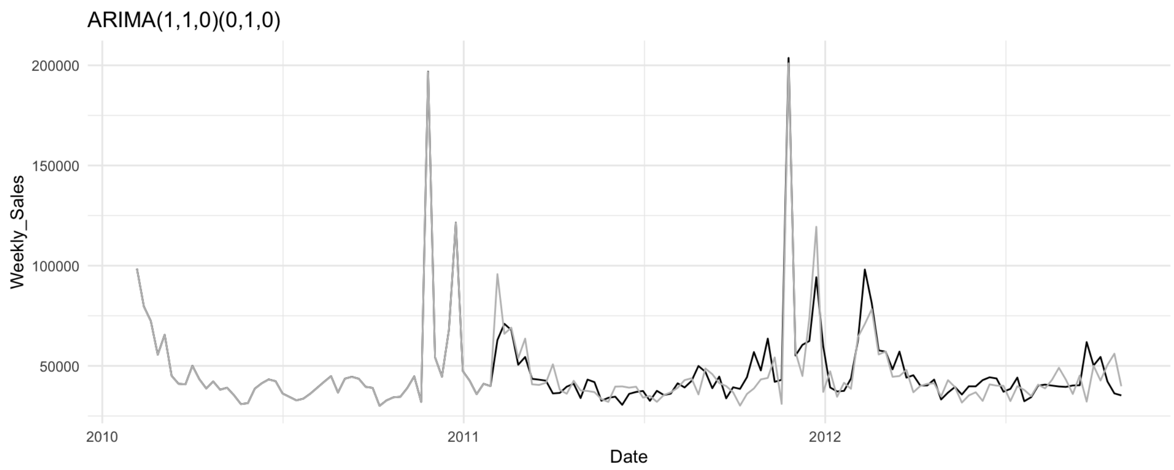
Assignment 3.1 – True Seasonality

```
ws1d72_fit <-  ws1d72 |>
    model(
        auto_arima = ARIMA(Weekly_Sales),
        ma1_12 = ARIMA(Weekly_Sales ~ pdq(0,1,1) + PDQ(0,1,0, period = 52)  ) ,
        ar1_12 = ARIMA(Weekly_Sales ~ pdq(1,0,0) + PDQ(0,1,0, period = 52)  ),
        auto_12 = ARIMA(Weekly_Sales ~ pdq() + PDQ(period = 52)  )
    )
```

▶ Code

▶ Code

.model	.type	RMSE
auto_arima	Training	22042.148
ma1_12	Training	7648.828
ar1_12	Training	7396.473
auto_12	Training	7396.473



Assignment 4.1 - Hints

```r
pme_history <-
    read_csv("PowderyMildewEpidemic.csv",
             col_types = cols(Outbreak = col_factor(levels = c("No", "Yes"))))
```

Year	Outbreak	MaxTemp	RelHumidity
1987	Yes	30.14	82.86
1988	No	30.66	79.57
1989	No	26.31	89.14
1990	Yes	28.43	91.00
1991	No	29.57	80.57
1992	Yes	31.25	67.82
1993	No	30.35	61.76
1994	Yes	30.71	81.14
1995	No	30.71	61.57
1996	Yes	33.07	59.76
1997	No	31.50	68.29
2000	No	29.50	79.14

- treat 2000 as if it were 1998
- sliding window for training periods

```r
forecast_year <- function(yr) {
    trn <- pme_history |> filter(Year < yr)
    tst <- pme_history |> filter(Year == yr) |> select(Actual
= Outbreak)
    ... fit model using trn
    ... forecast tst$Predicted using tst
    return(tst)
}

pyears <- c(1996:1998, 2000)
outbreaks <- purrr::map_dfr(pyears, forecast_year)
outbreaks
```