Workshop 5:
Model Selection and Residual Analysis

STAT 464/864 | Discrete Time Series Analysis
Skye P. Griffith, Queen's University - Fall 2024

Setup

```
library(itsmr); library(knitr) # packages
load("wine_plotter.RData")     # load plotting functions from Workshop 4
```

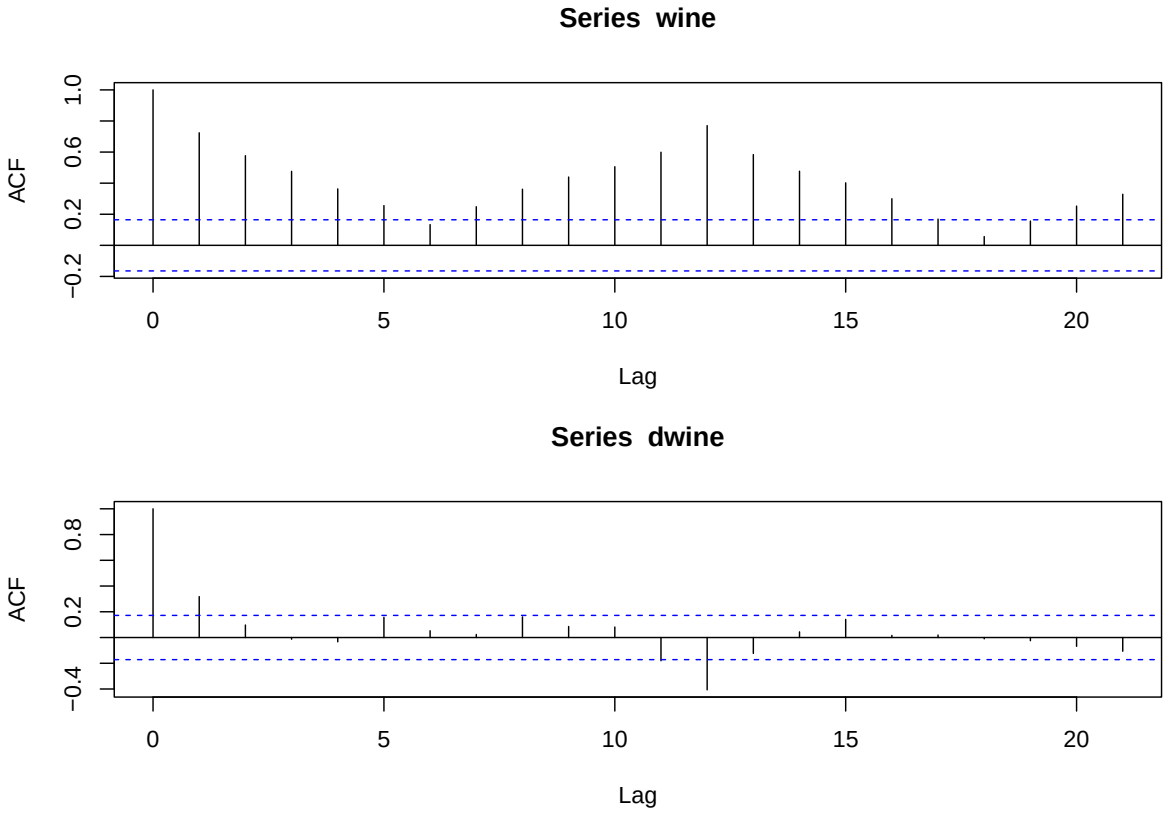
Dataset	Happy Australian Red Wine Sales (unit = kilolitres)
Source	Included in the ITSMR Package (no external files)
Total Times Sampled	(Monthly) Jan, 1980 – Oct, 1991 (142 total obs.)
Truncated Series	Jan, 1980 – Oct, 1989 (118 obs.)
Times to Predict	Nov, 1989 – Oct, 1991 (24 obs.)

```
t.past   <- 1:118    ;  wine.past   <- wine[t.past  ]  # truncated data
t.future <- 119:142  ;  wine.future <- wine[t.future]  # desired prediction
```

A note on Differencing

We didn't talk about differencing (∇_d) as formally as we did the other other methods for seasonal component estimation. But it can be really effective! Look at the "whitening" effect it has on the ACF of the original data.

```r
par(mfrow = c(2,1), mar = c(4,4,4,1))
dwine <- diff(wine, lag=12)

acf(wine)
acf(dwine)
```



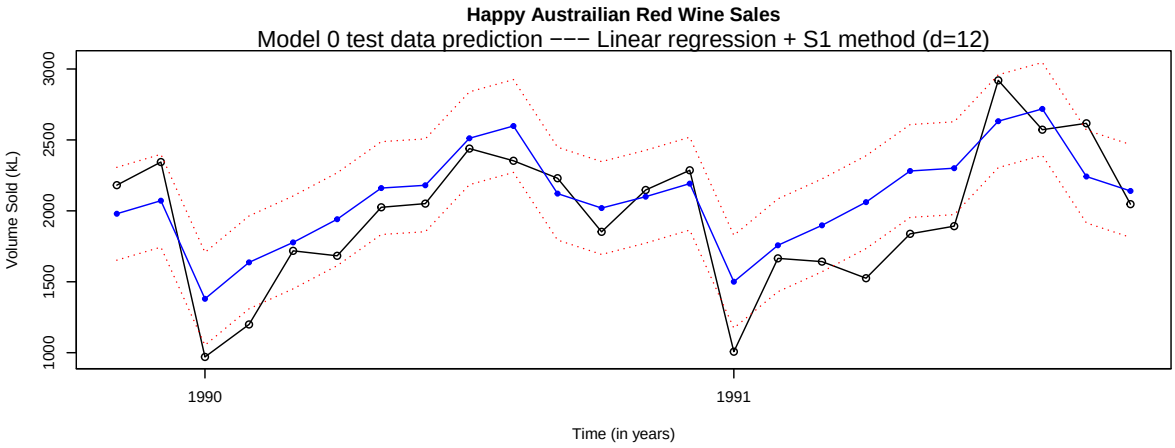
Returning to our Happy Wine Forecasting Models

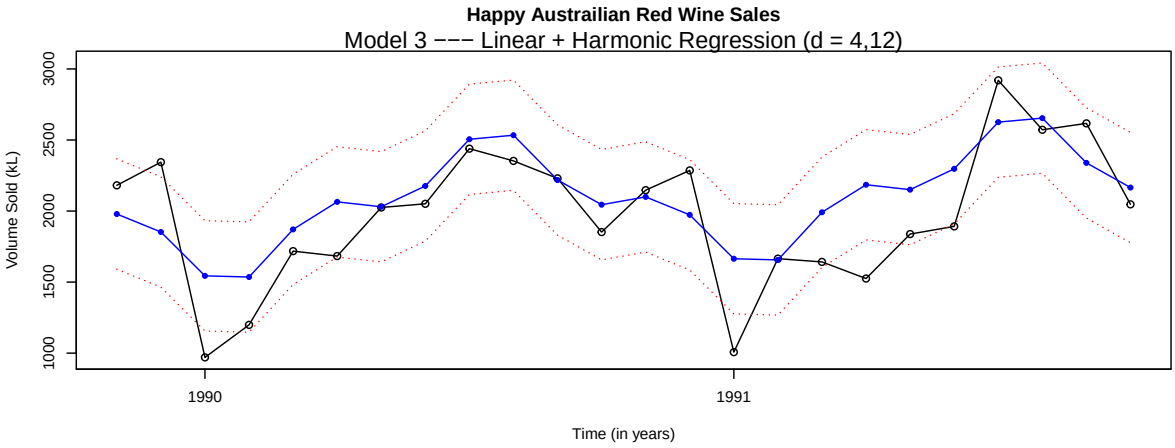
Let's look at the best models from workshop 4, plus one other model: in *Model 4*, we remove the trend m_t by taking the *logarithm* of the original data.

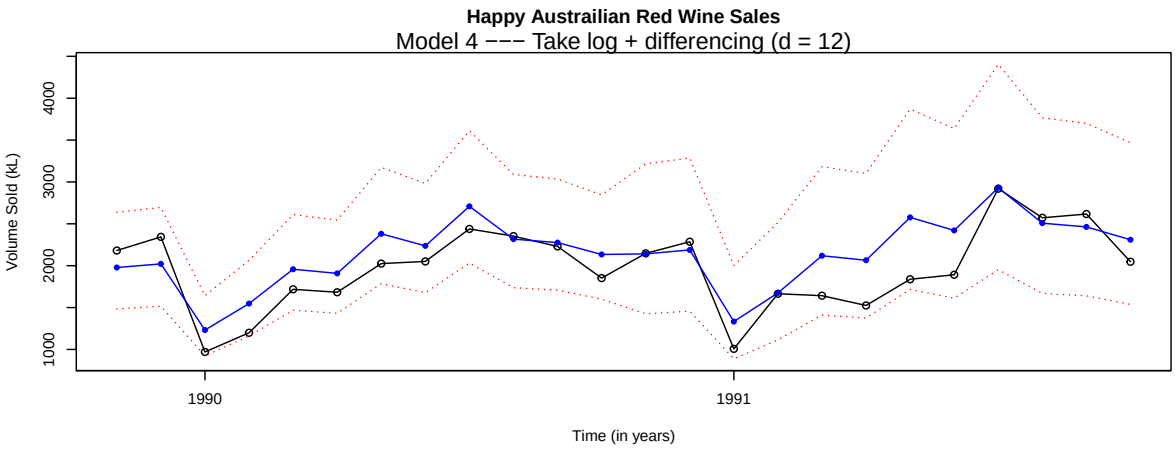
Model	m_t estimation	s_t Estimation
0	linear regression	S1 method; d = 12
3	linear regression	Harmonic Regression; d = {4,12}
4	Take log of data	Differencing; d = 12

```r
# --- Classical Model and Residuals --------- UNIT 1, FOLKS!
M.0 <- c("trend", 1, "season", 12)
M.3 <- c("trend", 1, "hr", c(4,12))
M.4 <- c("log", "diff", 12)

R.0 <- Resid(wine.past, M = M.0)
R.3 <- Resid(wine.past, M = M.3)
R.4 <- Resid(wine.past, M = M.4)

# --- ARMA ---------------------------------- UNIT 2, FOLKS!!
A.0 <- arma(x = R.0, p = 0, q = 0) # But not yet.
A.3 <- arma(x = R.3, p = 0, q = 0) # Just set p = q = 0, for now.
A.4 <- arma(x = R.4, p = 0, q = 0)

# --- Forecast ------------------------------ UNIT 3, FOLKS!!!
fc.0 <- forecast(wine.past, M = M.0, a = A.0, h = 24, opt = 0)
fc.3 <- forecast(wine.past, M = M.3, a = A.3, h = 24, opt = 0)
fc.4 <- forecast(wine.past, M = M.4, a = A.4, h = 24, opt = 0)
```

```r
par(mfrow = c(3,1), mar = c(4,4,4,1))

plot.future(fc.0,
     "Model 0 test data prediction --- Linear regression + S1 method (d=12)")
plot.future(fc.3,
     "Model 3 --- Linear + Harmonic Regression (d = 4,12)")
plot.future(fc.4,
     "Model 4 --- Take log + differencing (d = 12)")
```







Notice something weird happens with Model 4: the prediction interval widens over time! That's because we're finding a *linear* predictor for *logarithmic* data. Interesting phenomenon, but for this workshop, we'll be sticking to models 0 and 3.

AIC Model Selection

Models 0 and 3 are slightly different from how they appeared in Workshop 4: the only difference is that we didn't model any ARMA behaviour, this time.

Question: If we want to model the classical residuals (`R.0` and `R.3`) as an ARMA(p,q) process, *how do we choose p and q?*

AIC is a diagnostic criterion which compares the performances of various versions of the same model, where the versions differ by a choice of parameter (i.e. the p and q of an ARMA). It outputs a numerical value which has little meaning on its own, kind of like how covariance is only a *relative* measurement until it's *standardized.*

Note: we are not comparing Model 0 to Model 3. We are instead

1. comparing different ARMA(p,q) fits for Model 0, alone.
2. comparing different ARMA(p,q) fits for Model 3, alone.

We will consider all ordered pairs (p, q) where $p, q \in \{0, 1, 2, 3, 4, 5\}$.
1 AIC value for each ordered pair $\longrightarrow$ output is a 6×6 matrix.

Computing the AIC across ARMA parameters (p,q)

```
# --- Model Selection ----------- UNIT 4, FOLKS!

## Create empty matrix
aic.0 <- aic.3 <- matrix(data = 0, nrow = 6, ncol = 6) # p,q = 0 to 5

# ---  Double loop
for (p in 0:5) for (q in 0:5) {
  # Create models
  R0.fit <- arima(R.0, order = c(p,0,q))
  R3.fit <- arima(R.3, order = c(p,0,q),
                  optim.control = list(maxit = 1000)) # add this if u get a warning

  # Get AIC values and put inside matrices
  aic.0[p+1,q+1] <- R0.fit$aic
  aic.3[p+1,q+1] <- R3.fit$aic
}

rownames(aic.0) <- rownames(aic.3) <- paste("p =",0:5)
colnames(aic.0) <- colnames(aic.3) <- paste("q =",0:5)
```

The objects `R0.fit` and `R3.fit` are ARMA *models* with all kinds of info (just like when you create a linear *model* object with `lm()`). One of those pieces of info is indeed the AIC value. We pulled them out using the dollar sign, in our function.

Table 3: AIC for ARMA(p,q) models; Classical Model 0

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5
p = 0	1546.673	1544.407	1541.850	1543.768	1543.833	1541.605
p = 1	1542.668	1536.771	1538.739	1540.595	1541.516	1543.469
p = 2	1540.025	1538.736	1540.647	1542.425	1543.503	1544.343
p = 3	1541.442	1540.595	1542.502	1541.212	1542.069	1542.781
p = 4	1542.160	1542.183	1544.168	1539.882	1542.288	1545.998
p = 5	1541.010	1541.908	1541.793	1542.670	1544.609	1540.235

Table 4: AIC for ARMA(p,q) models; Classical Model 3

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5
p = 0	1586.968	1588.925	1587.710	1589.344	1591.344	1590.298
p = 1	1588.912	1586.151	1586.270	1588.156	1589.804	1590.097
p = 2	1587.642	1586.162	1588.134	1590.116	1591.985	1591.790
p = 3	1588.314	1589.709	1590.136	1584.555	1586.470	1590.109
p = 4	1590.313	1591.709	1591.481	1586.470	1588.470	1585.310
p = 5	1588.973	1589.873	1591.670	1593.645	1592.400	1583.522

```
[1] "Index of Smallest AIC (Model 0): 8"
```

```
[1] "Index of Smallest AIC (Model 3): 36"
```

R interprets the i^{th} entry of a 2 dimensional matrix by counting down each column, from the left-most column to the right-most column (like the vertical version of a calendar). So the best choices of parameters for each model, according to the AIC, is

Model	p	q	AIC	Noise variance σ^2
0	1	1	1536.771	2.134385×10^4
3	5	5	1583.522	3.0037787×10^4

The AIC values, above, are not necessarily relative, since they were calculated for distinct sets of residuals (Model 0 has different residuals than Model 3). Remember: what AIC compares is the performance of different possible *parameters,* or (p, q) combinations, for a single model.

Regardless, I'm convinced **Model 0 will be best to work with:**

1. Its classical prediction **better captured the rugged terrain** of the data than the harmonic regression model did.
2. Model 0 is an ARMA(1,1), which is **simpler** than Model 3's ARMA(5,5) structure. According to the principle of parsimony (a.k.a. Occam's Razor), the simpler model is scientifically desirable. Moreover, there are less things for you to have to juggle with, and it's easier to express mathematically.
3. The innovations (the white noise part of the ARMA model) in Model 0 are estimated to have a **smaller variance** than those of Model 3, which is desirable.

Estimating ARMA Coefficients

```
fit.past.0.aic  <- arima(R.0, order = c(1,0,1))
```

Our parameter estimates and their standard errors are:

Table 6: Model 0 - ARMA cofficients (from AIC selection)

	Estimated Coefficient	Standard Error
ar1	0.9149259	0.0736503
ma1	-0.7627690	0.1114317
intercept	10.9078166	39.2048776

Notice you can recreate this table in the white (text, not r-chunk) space of your Quarto script. The following is the table for Model 0, copy-pasted from the console, except I manually put Greek letters in with LaTeX.

	Estimated Coefficient	Standard Error
ϕ_1	0.9149259	0.0736503
θ_1	-0.7627690	0.1114317
intercept	10.9078166	39.2048776

BIC Model Selection

This diagnostic has some advantages over AIC, the details of which are beyond the scope of the course. The downside to BIC, in the specific context of *our* code, is that it's slightly computationally slower (see code comments). It's up to you whether to use AIC or BIC in your project: if their selections disagree, compare the *forecasting* performance of the selected models on the test data.

Computing the BIC across ARMA parameters (p,q)

```
## Create empty matrix
bic.0 <- matrix(data = 0, nrow = 6, ncol = 6) # p,q = 0 to 5

# ---  Double loop
for (p in 0:5) for (q in 0:5) {
  # Create models
  R0.fit <- arima(R.0, order=c(p,0,q))

  # Get bic values and put inside matrices
      # notice there is no R0.fit$bic produced by the arima() function.
      # -> we need to use the BIC() function. That's why it's slower.
      # R has to make an extra 36 complex calculations compared to AIC
  bic.0[p+1,q+1] <- BIC(R0.fit)
}

rownames(bic.0) <- paste("p =",0:5)
colnames(bic.0) <- paste("q =",0:5)
```

Table 8: BIC for ARMA(p,q) models; Classical Model 0

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5
p = 0	1552.215	1552.719	1552.933	1557.621	1560.457	1561.000
p = 1	1550.981	1547.853	1552.592	1557.219	1560.910	1565.635
p = 2	1551.108	1552.589	1557.271	1561.820	1565.668	1569.279
p = 3	1555.295	1557.219	1561.897	1563.377	1567.005	1570.488
p = 4	1558.784	1561.577	1566.334	1564.818	1569.995	1576.476
p = 5	1560.405	1564.073	1566.730	1570.377	1575.086	1573.483

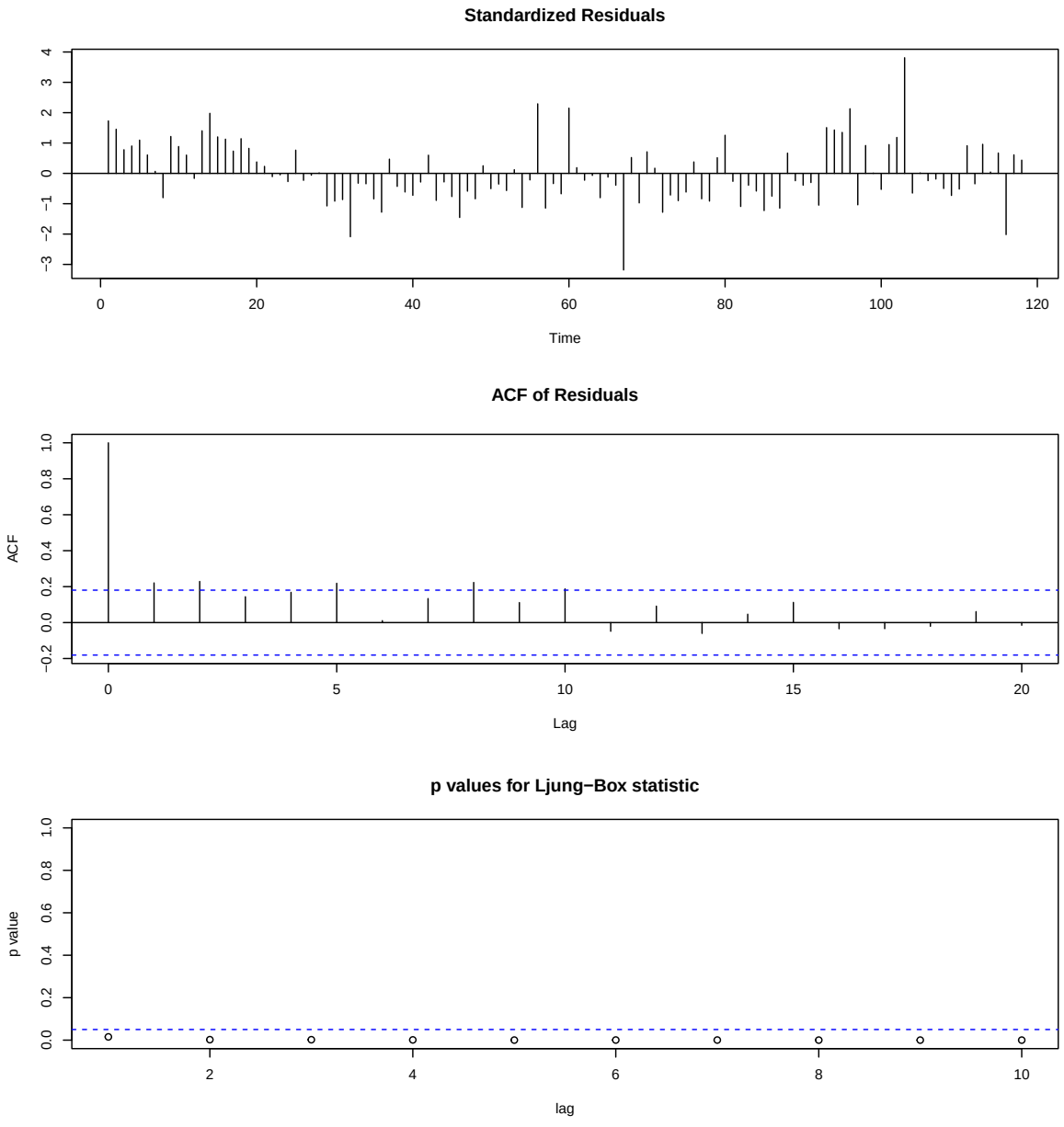
```
[1] "Index of Smallest BIC (Model 0): 8"
```

This agrees with the result of our AIC output! We can stick with the model we chose before.

Residual Analysis

We can check whether our fitted model is appropriate by examining its residuals. To do this, use the function `tsdiag()` (the input is a time series model object). So, let's do a before & after. Here's the residual analysis of Model 0 *before* modelling Y_t as ARMA(1,1):

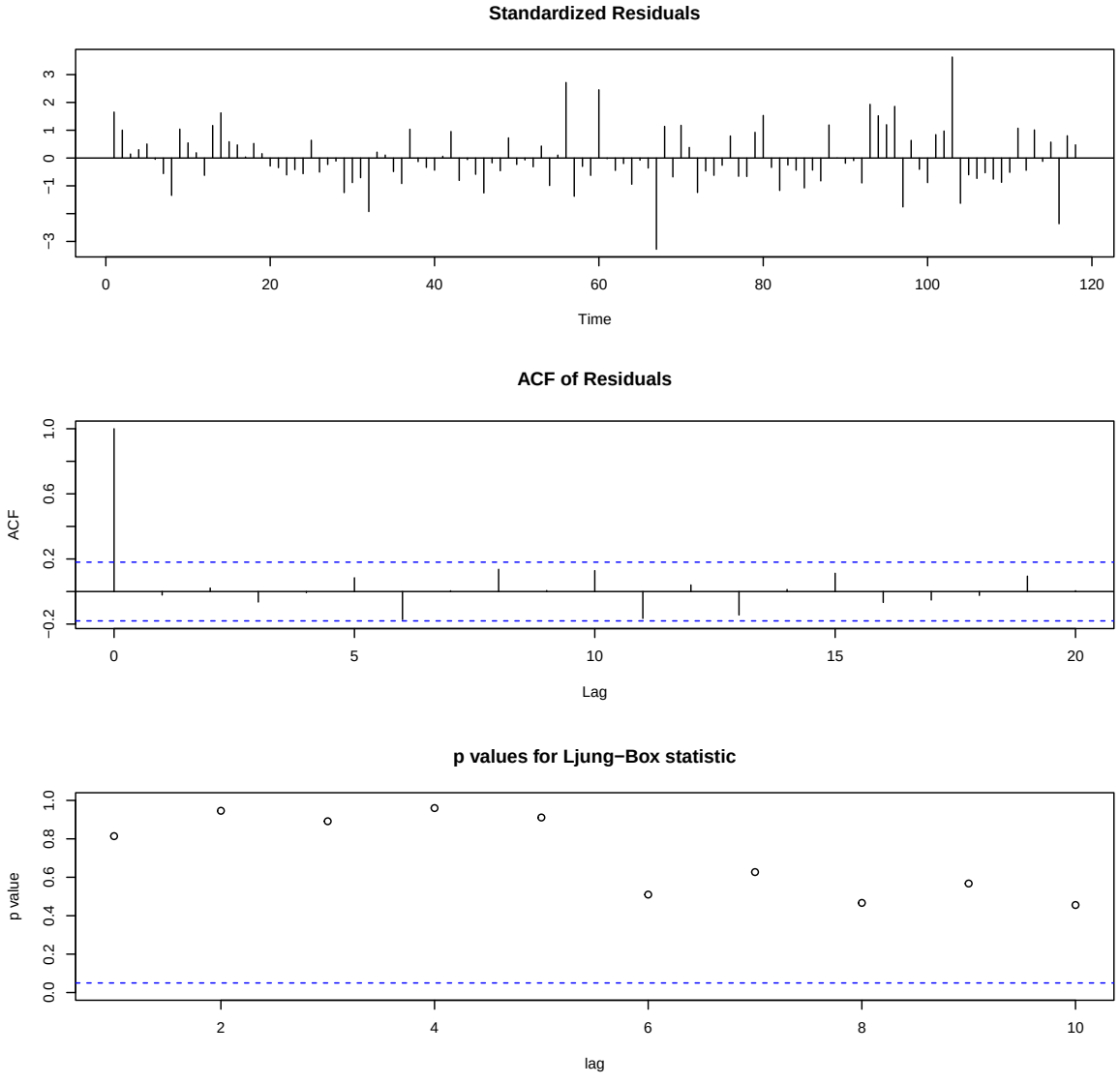
```
tsdiag(arima(R.0, order = c(0,0,0))) # no ARMA: we're setting p = q = 0.
```



The standardized residuals flip-flop between prolonged positive/negative regions. They don't look random enough. The sample ACF enters the rejection region way too many times. The Ljung-Box stat tells us there are significant autocorrelations at all lags (more on this, soon). There's no way our Y_t is white noise, dude.

Now compare this to the residual analysis of Model 0 *after* the ARMA(1,1) modelling. To clarify: we're now looking at Z_t from Unit 2, not Y_t from Unit 1.

```
tsdiag(fit.past.0.aic) # this one is ARMA(1,1)
```



Top Plot: These are the innovations Z_t of Model 0, *after* the ARMA(1,1) modelling. They look nice and random: there are no trends, cycles, or major changes in variance.

Middle Plot: According to the ACF, there are no trends significant enough to suggest that the residuals are anything but white noise. (We failed to reject H_0).

Bottom Plot: The Ljung-Box statistic is another diagnostic which, in this context, attempts to detect significant autocorrelations. Think of it as conducting a mini white noise hypothesis test for each individual lag of the ACF. Large p-values tell us that the behaviour of the residuals is statistically insignificant. If they drop below the blue line, you might want to go back and tweak your model.

Final Model

This doesn't have to be a fancy table or anything. Just give the equations in LaTeX.

Model 0		
Classical Model	X_t	$X_t = m_t + s_t + Y_t$
ARMA-modelled Residuals	Y_t	$\phi(B)Y_t = \theta(B)Z_t$
Innovations (noise)	Z_t	$Z_t \sim wn(\sigma^2 = 21343.85)$
Trend estimate	m_t	Line of best fit (linear regression)
Seasonality estimate	s_t	Periodic fn. derived from 12-point moving average (S1 Method)
ARMA(p,q) selection	AIC	
AR polynomial	$\phi(z)$	$1 - 0.9149259z$
MA polynomial	$\theta(z)$	$1 - 0.7627690z$

Interpretation

Our classical model suggests an upward linear trend in wine sales. There is also a 12 month long repeating cycle — we can see that the sales peak each July, and drop drastically each January. However, this pattern's *repeating segment* is not smooth or sinusoidal: there is a plateau of sales in the fall months, where we would expect a more consistent decline if the data were more wave-like. This is likely due to the tight sequence of holidays: Halloween, American Thanksgiving, and then a bunch of winter holidays (including New Year's Eve.)

After having removed these trends, both our AIC and BIC analyses suggested that an ARMA(1,1) model would best capture the behaviour of the residuals. Our residual analysis confirmed the effectiveness of this model, as several diagnostics showed a successful *whitening* of the classical residuals. Thus, accounting for ARMA(1,1) behaviour, we should be able to predict the noisier components of the data with greater accuracy than the classical model.