# Lecture 12

Julia Schedler

### Announcements

- There will be participation credit today! Make sure you submit.
- Assignment 4 due tonight, extensions through Weds allowed if desired
- Assignment 5 posted soon, will be due **Wednesday**

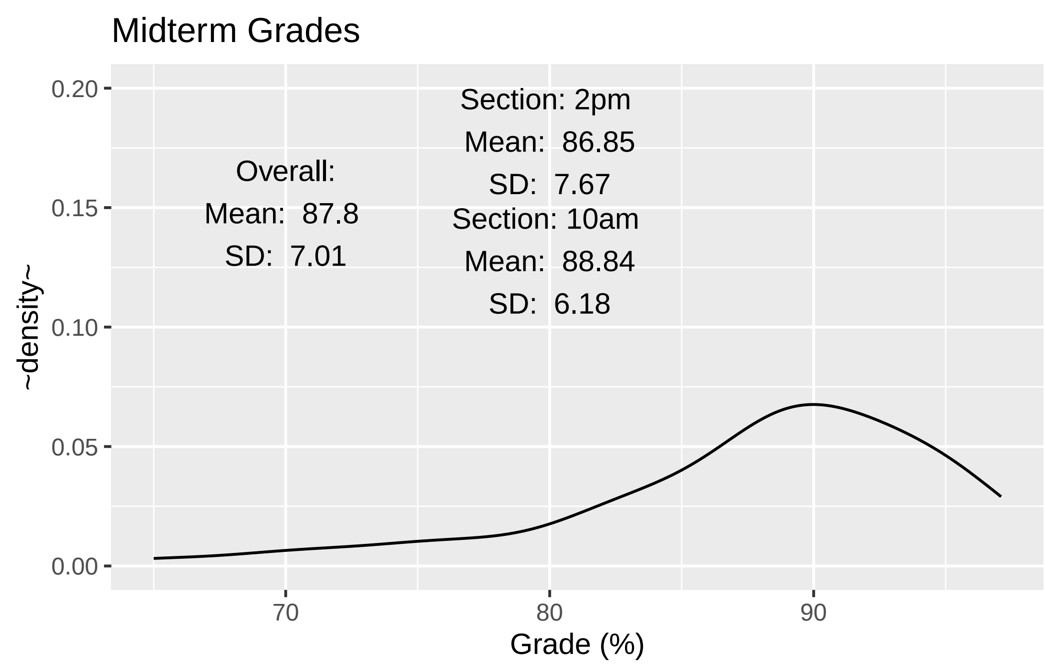
### Midterm Grades

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
midterm_grades <- read.csv("../../Student Data/2024-11-04T0741_Grades-STAT-416-01-2248.csv",
names(midterm_grades) <- c("Section", "Grade")

## convert section to factor
midterm_grades$Section <- factor(as.factor(midterm_grades$Section), labels = c("10am", "2pm")

## density plot add summary statistics to plot as text
```

```
summary_stats <- midterm_grades |>
  group_by(Section) |>
  summarise(mean = mean(Grade), sd = sd(Grade))

summary_stats_overall <- midterm_grades |>
  summarise(mean = mean(Grade), sd = sd(Grade))

ggplot(midterm_grades, aes(x = Grade)) + geom_density() +
  geom_text(data = summary_stats, aes(x = c(80,80), y = c(.15, .20), label = paste("Section:"
  geom_text(data = summary_stats_overall, aes(x = 70, y = .17, label = paste("Overall:\nMean
  ylab("~density~") + xlab("Grade (%)")+ ggtitle("Midterm Grades")
```




**Activity 1: Review your midterm**

- Check the page totals (bottom corner) and exam totals (first page)
- Find where you lost the most points. What did you do incorrectly?
- What was something you did well?
- Fill out the canvas Quiz

## R code for SARIMA models

- `sarima` function from the `astsa` package
- `auto.arima` function from the `forecast` package
- `model` function from the `fable` package

Which to use??? They all are fine, but output the model object differently, so the model output won't work with functions from other packages.

## Example: Trying to use `sarima` output with `fable` diagnostics

You get an error!

```
library(fpp3)
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
## -- Attaching packages -------------------------------------------- fpp3 1.0.1 --
## v tsibble     1.1.5      v feasts      0.4.1
## v tsibbledata 0.4.1      v fable       0.4.0
## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
library(astsa)

log_gnp <- log(gnp)

sarima_model <- sarima(log_gnp, p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, details = FALSE)
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1        0.3467 0.0627  5.5255       0
## constant   0.0083 0.0010  8.5398       0
##
## sigma^2 estimated as 9.029576e-05 on 220 degrees of freedom
##
```

```
## AIC = -6.446939  AICc = -6.446692  BIC = -6.400957
##

diagnostics <- sarima_model |>
  residuals() |>
  ACF() |>
  gg_tsdisplay()
## Error in UseMethod("measured_vars"): no applicable method for 'measured_vars' applied to 
```

#### Example: trying to use `fable` output with `sarima` diagnostics

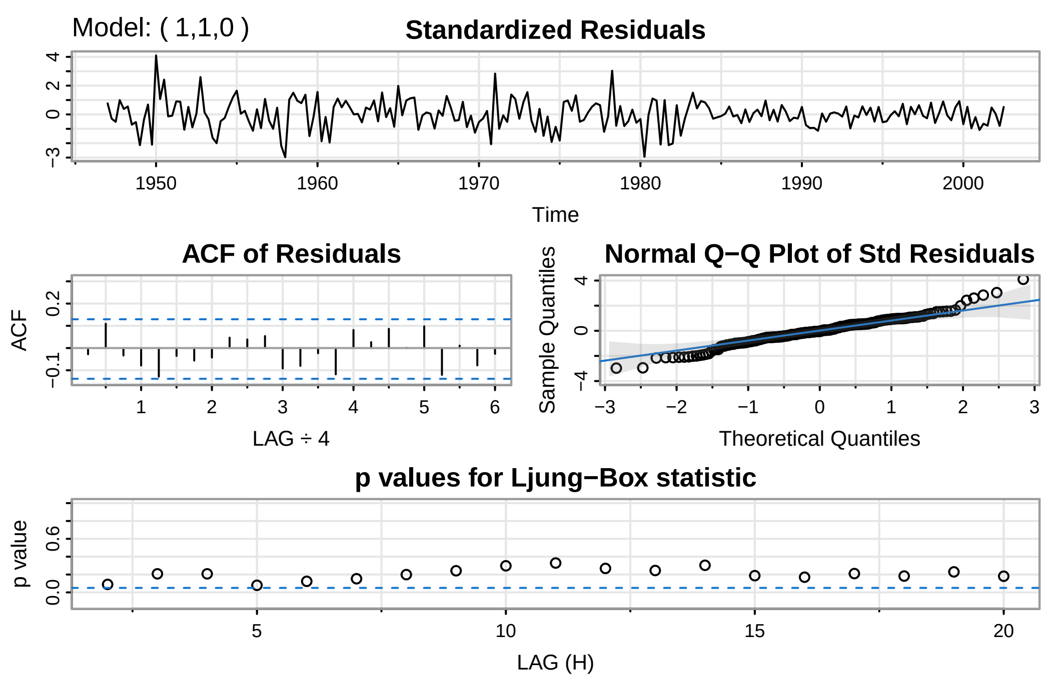
Since `sarima` outputs diagnostics when you fit the model, you could just re-fit the model with `sarima` and then use the diagnostics.

```
fable_model <- log_gnp |>
  as_tsibble() |>
  model(ARIMA(value ~ pdq(1,1,0) + PDQ(0,0,0))) |> report()
## Series: value
## Model: ARIMA(1,1,0) w/ drift
##
## Coefficients:
##          ar1  constant
##       0.3467    0.0054
## s.e.  0.0627    0.0006
##
## sigma^2 estimated as 9.136e-05:  log likelihood=718.61
## AIC=-1431.22   AICc=-1431.11   BIC=-1421.01

sarima(fable_model) ## doesn't work
## Error in sarima(fable_model): argument "d" is missing, with no default
```

```
## fit the same model, but with default details = TRUE.
sarima(log_gnp, p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0)
```

```
initial  value -4.589567
iter   2 value -4.654150
iter   3 value -4.654150
iter   4 value -4.654151
iter   4 value -4.654151
iter   4 value -4.654151
```

```
final  value -4.654151
converged
initial  value -4.655919
iter   2 value -4.655921
iter   3 value -4.655921
iter   4 value -4.655922
iter   5 value -4.655922
iter   5 value -4.655922
iter   5 value -4.655922
final  value -4.655922
converged
<><><><><><><><><><><><><><>

Coefficients:
         Estimate     SE t.value p.value
ar1        0.3467 0.0627  5.5255       0
constant   0.0083 0.0010  8.5398       0

sigma^2 estimated as 9.029576e-05 on 220 degrees of freedom

AIC = -6.446939  AICc = -6.446692  BIC = -6.400957
```



### Activity 2: Putting it all together

- Download "ARIMA code cheat sheet.docx" from Canvas
- Pick one column
- Fill in the blanks with the correct functions

### Forecasting

Given the data and a model that fits the data, we want to predict future values

How can we do this - by "hand" (or just "manually" using code) - using `fable` or `astsa` functions

Also, how do we plot these forecasts?

### Forecasting using `fable` or `astsa` functions

`fable` functions

1. Fit model using `model()` and `ARIMA()`
2. Forecast using `forecast()` specifying `h`

```
fit <- log_gnp |>
  as_tsibble() |>
  model(ARIMA(value ~ pdq(1,1,0) + PDQ(0,0,0))) ## force nonseasonal

fit |> forecast(h = 10)
```

```
# A fable: 10 x 4 [1Q]
# Key:     .model [1]
   .model                                      index
   <chr>                                       <qtr>
 1 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2002 Q4
 2 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2003 Q1
 3 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2003 Q2
 4 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2003 Q3
 5 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2003 Q4
 6 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2004 Q1
 7 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2004 Q2
 8 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2004 Q3
 9 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2004 Q4
```

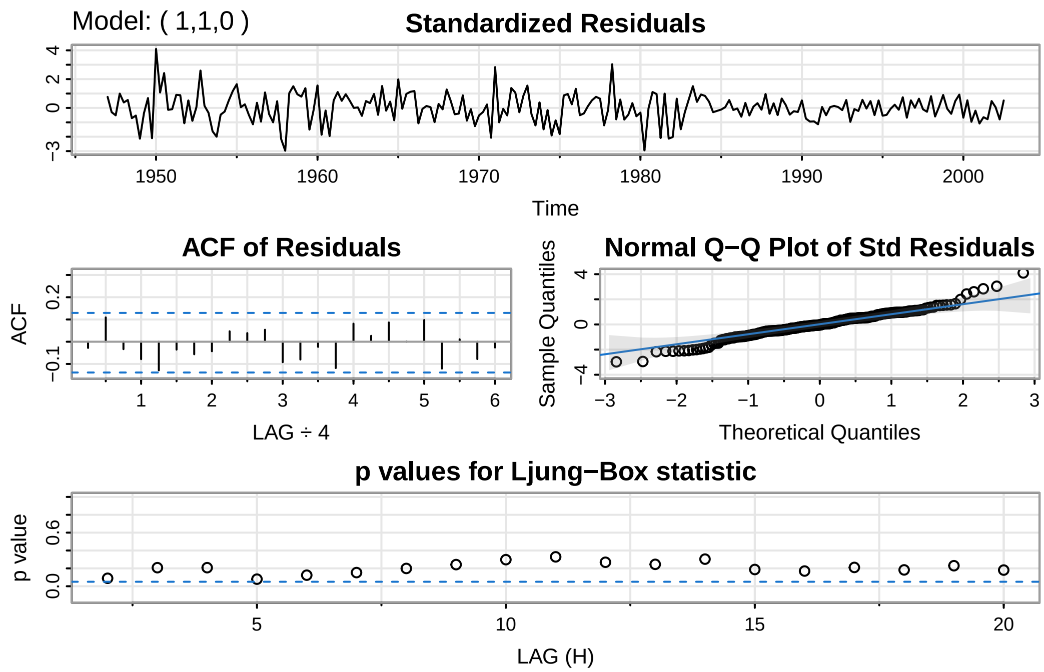
```
10 ARIMA(value ~ pdq(1, 1, 0) + PDQ(0, 0, 0)) 2005 Q1
# i 2 more variables: value <dist>, .mean <dbl>
```

- `astsa` functions

  1. Determine model order using acf/pacf, check model fit using `sarima`
  2. Use `sarima.for` to forecast (with order you chose– re-fits model)

```r
sarima_model <- sarima(log_gnp, p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0)
```

```
initial  value -4.589567
iter   2 value -4.654150
iter   3 value -4.654150
iter   4 value -4.654151
iter   4 value -4.654151
iter   4 value -4.654151
final  value -4.654151
converged
initial  value -4.655919
iter   2 value -4.655921
iter   3 value -4.655921
iter   4 value -4.655922
iter   5 value -4.655922
iter   5 value -4.655922
iter   5 value -4.655922
final  value -4.655922
converged
<><><><><><><><><><><><><><>

Coefficients:
         Estimate     SE t.value p.value
ar1        0.3467 0.0627  5.5255       0
constant   0.0083 0.0010  8.5398       0

sigma^2 estimated as 9.029576e-05 on 220 degrees of freedom

AIC = -6.446939  AICc = -6.446692  BIC = -6.400957
```




```
sarima.for(log_gnp,p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, n.ahead = 10, plot = F)
```

```
$pred
         Qtr1     Qtr2     Qtr3     Qtr4
2002                            9.165886
2003 9.174510 9.182946 9.191316 9.199664
2004 9.208004 9.216342 9.224678 9.233014
2005 9.241350

$se
            Qtr1        Qtr2        Qtr3        Qtr4
2002                                     0.009502408
2003 0.015938787 0.021173624 0.025569341 0.029380482
2004 0.032772142 0.035850982 0.038687708 0.041330887
2005 0.043815131
```

## Compare predictions

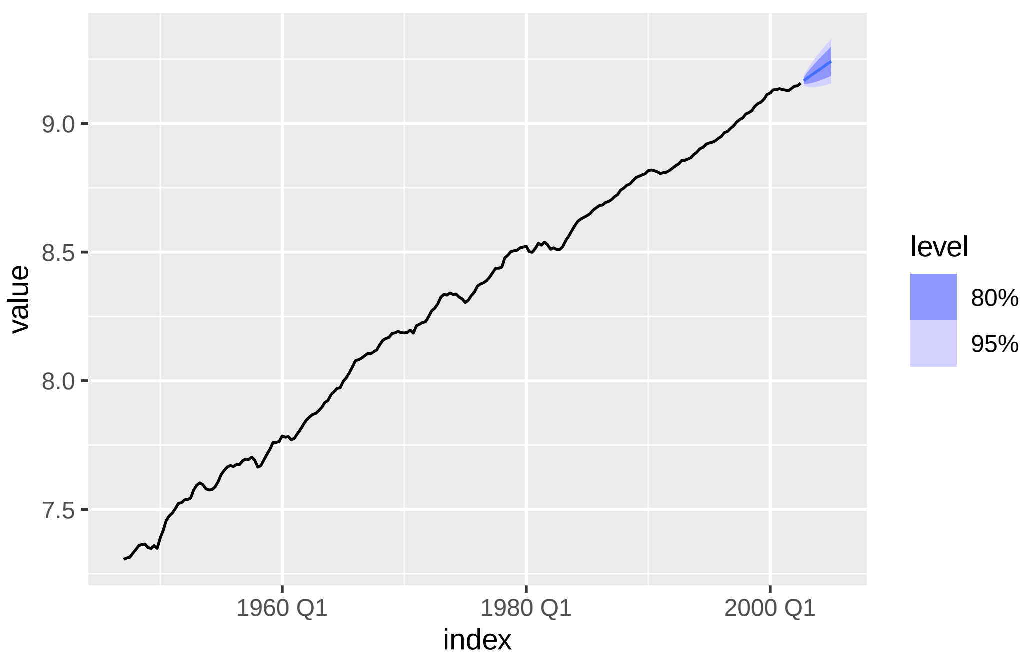
Yay! They're the same :)

```r
fable_forecast <- fit |> forecast(h = 10)
astsa_forecast <- sarima.for(log_gnp,p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, n.ahead = 10,

cbind(fable_forecast$.mean,astsa_forecast$pred)
```

```
        fable_forecast$.mean astsa_forecast$pred
2002 Q4             9.165886            9.165886
2003 Q1             9.174510            9.174510
2003 Q2             9.182946            9.182946
2003 Q3             9.191316            9.191316
2003 Q4             9.199664            9.199664
2004 Q1             9.208004            9.208004
2004 Q2             9.216342            9.216342
2004 Q3             9.224678            9.224678
2004 Q4             9.233014            9.233014
2005 Q1             9.241350            9.241350
```

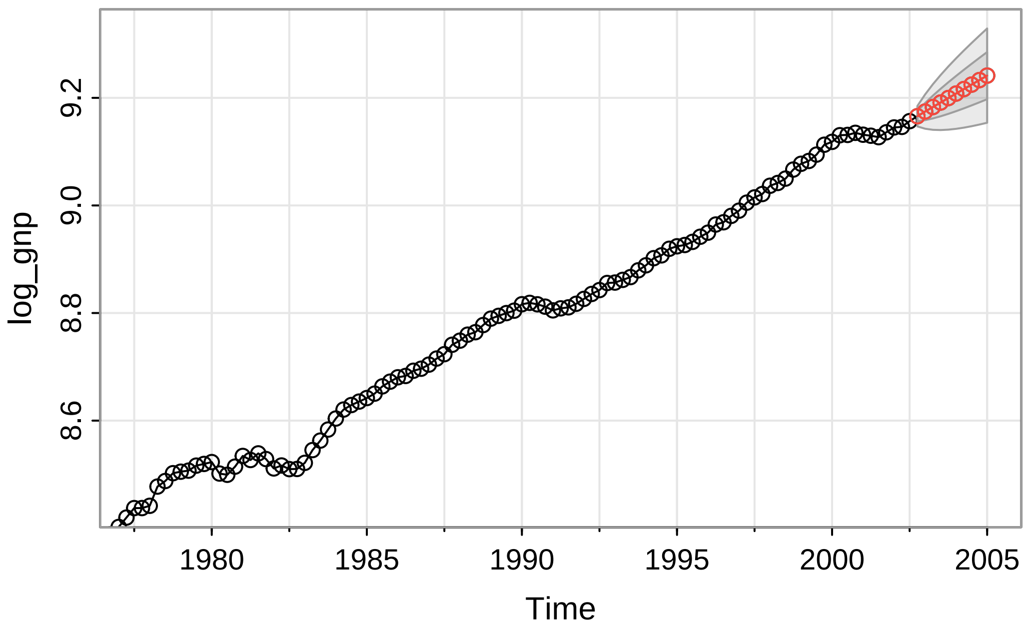
### Plot the forecast (`fable`)

```r
fable_forecast |> autoplot(log_gnp)
```



### Plot the forecast (astsa)

```r
sarima.for(log_gnp,p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, n.ahead = 10, plot = T)
```




```
$pred
         Qtr1     Qtr2     Qtr3     Qtr4
2002                            9.165886
2003 9.174510 9.182946 9.191316 9.199664
2004 9.208004 9.216342 9.224678 9.233014
2005 9.241350

$se
            Qtr1        Qtr2        Qtr3        Qtr4
2002                                     0.009502408
2003 0.015938787 0.021173624 0.025569341 0.029380482
2004 0.032772142 0.035850982 0.038687708 0.041330887
2005 0.043815131
```

### Forecasting "by hand"

1. Write down the equation of your model based on the output

2. Get the observations you need to populate the forecast
3. Calculate the forecast, pushing forward your next forecasted value through the equation

### Forecasting "by hand"

```
astsa_model <- sarima(log_gnp, p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, details = FALSE)
```

```
<><><><><><><><><><><><><><>

Coefficients:
         Estimate     SE t.value p.value
ar1        0.3467 0.0627  5.5255       0
constant   0.0083 0.0010  8.5398       0

sigma^2 estimated as 9.029576e-05 on 220 degrees of freedom

AIC = -6.446939  AICc = -6.446692  BIC = -6.400957
```

(apparent) `fable` forecast equation: $\hat{x}_t = 0.0054 + 0.3467 \cdot x_{t-1}$

(apparent) `astsa` forecast equation: $\hat{x}_t = 0.0083 + 0.3467 \cdot x_{t-1}$

### Activity: Forecasting "by hand"

`fable` forecast equation: $\hat{x}_t = 0.0054 + 0.3467 \cdot x_{t-1}$

`astsa` forecast equation: $\hat{x}_t = 0.0083 + 0.3467 \cdot x_{t-1}$

### Activity Solution: Forecasting "by hand"

```
data.frame(time = tail(time(log_gnp)), log_gnp = tail(log_gnp), diff_log_gnp = tail(diff(log_
```

```
     time  log_gnp  diff_log_gnp
1 2001.25 9.129597 -0.0018845451
2 2001.50 9.126937 -0.0026595616
3 2001.75 9.135994  0.0090568862
4 2002.00 9.145002  0.0090076208
```

```
5 2002.25 9.145983  0.0009816371
6 2002.50 9.156718  0.0107348840
```

`fable` forecast equation: $\hat{x}_t = 0.0054 + 0.3467 \cdot x_{t-1}$

- $\hat{x}_{2002Q4} = 0.0054 + 0.3467 \cdot x_{2002Q3} = 0.0054 + 0.3467 \cdot 9.156718 = 3.1800034$

`astsa` forecast equation: $\hat{x}_t = 0.0083 + 0.3467 \cdot x_{t-1}$

- $\hat{x}_{2002Q4} = 0.0083 + 0.3467 \cdot x_{2002Q3} = 0.0083 + 0.3467 \cdot 9.156718 = 3.182934$

These values don't make sense, and don't match?

### Reconciling the by hand difference

For our AR(1) model, we have

$$(1 - \phi B)(1 - B)y_t = c + error$$

Ignoring the error since we want a forecast of the mean, applying the backwards shift operator, F.O.I.L., using our estimate of $\phi$, and solving for $y_t$ we get:

$$\hat{x}_t = x_{t-1} + \hat{\phi}(x_{t-1} - x_{t-2})) + c$$

### Let's try this equation!

$$\hat{x}_{2002Q4} = x_{2002Q3} + \hat{\phi}(x_{2002Q3} - x_{2002Q2})) + c$$

```r
## for fable
9.156718 + 0.3467*(9.156718 - 9.145983) + 0.0054
```

```
[1] 9.16584
```

```r
## for sarima
9.156718 + 0.3467*(9.156718 - 9.145983) + 0.0083
```

```
[1] 9.16874
```

That's better, but only `fable` matches the forecast functions?

### Comparing constants

Takeaway: Different packages estimate the "constant" differently!

See https://otexts.com/fpp3/arima-r.html#understanding-constants-in-r.

```
library(fpp3)
library(astsa)
log_gnp <- log(gnp)

fable_model <- log_gnp |>
  as_tsibble() |>
  model(ARIMA(value ~ pdq(1,1,0) + PDQ(0,0,0))) ## force nonseasonal

fable_model |> report()
## Series: value
## Model: ARIMA(1,1,0) w/ drift
##
## Coefficients:
##          ar1  constant
##       0.3467    0.0054
## s.e.  0.0627    0.0006
##
## sigma^2 estimated as 9.136e-05:  log likelihood=718.61
## AIC=-1431.22   AICc=-1431.11   BIC=-1421.01
sarima_model <- sarima(log_gnp, p = 1, d = 1, q = 0, P = 0, D = 0, Q = 0, details = F)
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1        0.3467 0.0627  5.5255       0
## constant   0.0083 0.0010  8.5398       0
##
## sigma^2 estimated as 9.029576e-05 on 220 degrees of freedom
##
## AIC = -6.446939  AICc = -6.446692  BIC = -6.400957
##

mu = mean(diff(log_gnp))

mu*(1-sarima_model$fit$coef[1])
##         ar1
## 0.005447277
```

### Actual by-hand forecast equations:

For `fable`, where $c$ is the constant outputted.

$$\hat{x}_{2002Q4} = x_{2002Q3} + \hat{\phi}(x_{2002Q3} - x_{2002Q2})) + c_{fable}$$

For `astsa`, where $c$ is the constant outputted.

$$\hat{x}_{2002Q4} = x_{2002Q3} + \phi(x_{2002Q3} - x_{2002Q2})) + mean(diff(x_t))(1 - \hat{\phi})$$

```r
## for astsa

9.156718 + 0.3467*(9.156718 - 9.145983) + mean(diff(log_gnp))*(1- 0.3467)
```

```
[1] 9.165887
```

### More on the Ljung-box statistic

- "another way to view the ACF of the residuals"
- "not a bunch of highly dependent tests"
- "accumulation of autocorrelation"
- "considers the magnitudes" of the autocorrelations all together

$$Q = n(n+2) + \sum_{h=1}^{H} \frac{\hat{\rho}_{resid}(h)^2}{n-h}$$

Test statistic used to calcualte p-values in the `sarima` output– the $\hat{\rho}_{resid}(h)$ is the sample acf we plot using `acf` or `acf1`.

### ETS Models

Stepping away from ARIMA for the time being…

### Smoothing

We have seen several smoothers:

- Moving Average
- Loess
- Kernel

We have also seen the `decompose` function, but emphasized it is an exploratory data analysis tool.

What if we wanted to use these equations for forecasting?

### Simple exponential smoothing

- useful for forecasting data with no trend or seasonal component
- Predicts the future as a weighted average of the past, where the weights decrease exponentially the further back in time you go

### Example: Algerian Exports (FPP 8.1)

```r
algeria_economy <- global_economy |>
  filter(Country == "Algeria")
algeria_economy |>
  autoplot(Exports) +
  labs(y = "% of GDP", title = "Exports: Algeria")
```