Data 624: Predictive Analytics HW 1

Group 2: Alice Friedman, Kayleah Griffen, Josh Iden, Michael Ippolito

6/18/2023

## Introduction

This homework assignment includes problems from:

1. Hyndman & Athanasopoulos. “Forecasting: Principles and Practice”
2. Kuhn & Johnson. “Applied Predictive Modeling”

This accompanies readings from KJ 1,2 and 3 and HA 1,2,6,7 and 8.

## Week 1 Homework Solutions

### HA 2.1

Using the help function (?) to explore what the series gold, woolyrnq and gas represent, we get the following results in the description:

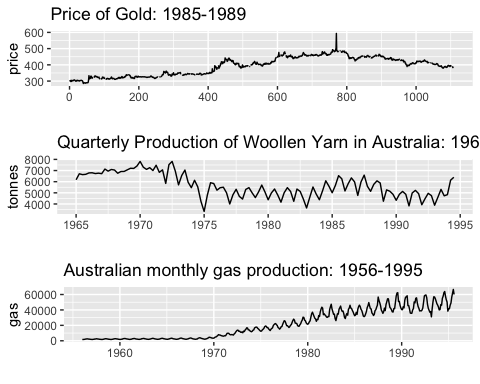
?gold “Daily morning gold prices in US dollars. 1 January 1985 – 31 March 1989”

?woolyrnq “Quarterly production of woollen yarn in Australia: tonnes. Mar 1965 – Sep 1994”

?gas Australian monthly gas production: 1956–1995.

#### a. Use autoplot() to plot each of these in separate plots.

library(fpp2)  
library(gridExtra)  
library(tidyverse)  
  
  
  
p1 = autoplot(gold) + ggtitle('Price of Gold: 1985-1989') + ylab('price') + xlab('')  
p2 = autoplot(woolyrnq) + ggtitle('Quarterly Production of Woollen Yarn in Australia: 1965 - 1994') + ylab('tonnes') + xlab('')  
p3 = autoplot(gas) + ggtitle('Australian monthly gas production: 1956-1995') + ylab('gas') + xlab('')  
  
grid.arrange(p1, p2, p3, nrow=3)



#### b. What is the frequency of each series?

Gold <- frequency(gold)  
  
Wool <- frequency(woolyrnq)  
  
Gas <- frequency(gas)  
  
Freqs <- data.frame(  
 Series = c("Gold", "Wool", "Gas"),  
 Frequency = c(Gold, Wool, Gas),  
 Seasonality = c("Daily", "Quaterly", "Monthly")  
 )   
Freqs %>% print()

## Series Frequency Seasonality  
## 1 Gold 1 Daily  
## 2 Wool 4 Quaterly  
## 3 Gas 12 Monthly

Based on frequency and the help function you can tell the gold observations are taken daily, the woolyrnq is quarterly, and the gas is monthly.

#### c. Use which.max() to spot the outlier in the gold series. Which observation was it?

which.max determines the index of the maximum of the numeric vector. This occurs at day 770 in the gold series. We can check what date this is using lubridate package.

library(lubridate)  
  
d <- ymd("1985/01/01") + which.max(gold)  
  
paste0('day:', which.max(gold) , ' date:', d, ' price: $', gold[which.max(gold)])

## [1] "day:770 date:1987-02-10 price: $593.7"

The max price of gold occurred on February 10, 1987.

### HA 2.3

#### a. Download some monthly Australian retail data from the book website (<https://otexts.com/fpp2/extrafiles/retail.xlsx>). These represent retail sales in various categories for different Australian states, and are stored in a MS-Excel file.

Head of the downdloaded dataset is below.

library(readxl)  
retaildata <- readxl::read\_excel("retail.xlsx", skip=1)  
head(retaildata)

## # A tibble: 6 × 190  
## `Series ID` A3349335T A3349627V A3349338X A3349398A A3349468W  
## <dttm> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1982-04-01 00:00:00 303. 41.7 63.9 409. 65.8  
## 2 1982-05-01 00:00:00 298. 43.1 64 405. 65.8  
## 3 1982-06-01 00:00:00 298 40.3 62.7 401 62.3  
## 4 1982-07-01 00:00:00 308. 40.9 65.6 414. 68.2  
## 5 1982-08-01 00:00:00 299. 42.1 62.6 404. 66   
## 6 1982-09-01 00:00:00 305. 42 64.4 412. 62.3  
## # ℹ 184 more variables: A3349336V <dbl>, A3349337W <dbl>, A3349397X <dbl>,  
## # A3349399C <dbl>, A3349874C <dbl>, A3349871W <dbl>, A3349790V <dbl>,  
## # A3349556W <dbl>, A3349791W <dbl>, A3349401C <dbl>, A3349873A <dbl>,  
## # A3349872X <dbl>, A3349709X <dbl>, A3349792X <dbl>, A3349789K <dbl>,  
## # A3349555V <dbl>, A3349565X <dbl>, A3349414R <dbl>, A3349799R <dbl>,  
## # A3349642T <dbl>, A3349413L <dbl>, A3349564W <dbl>, A3349416V <dbl>,  
## # A3349643V <dbl>, A3349483V <dbl>, A3349722T <dbl>, A3349727C <dbl>, …

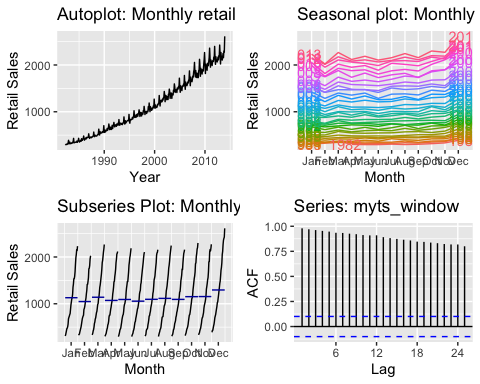
#### b. Select one of the time series as follows (but replace the column name with your own chosen column):

myts <- ts(retaildata[,"A3349335T"],  
 frequency=12, start=c(1982,4))

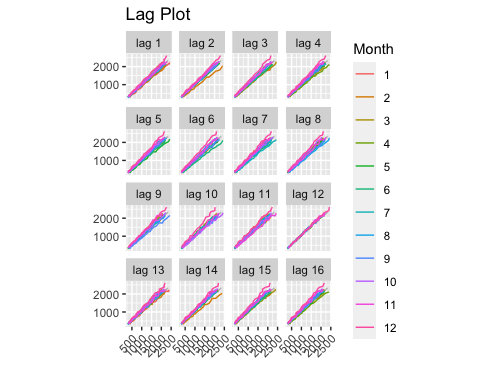
#### c. Explore your chosen retail time series using the following functions: autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf()

Below we plotted autoplot(), ggseasonplot(), ggsubseriesplot(), & ggAcf() in a grid. The lag plot requires a lot of space so it is plotted seperately!

ap <- autoplot(myts) +  
 ggtitle("Autoplot: Monthly retail sales in various categories for different Australian states") +  
 xlab("Year") +  
 ylab("Retail Sales")  
  
sp <- ggseasonplot(myts, year.labels=TRUE, year.labels.left=TRUE) +  
 ylab("Retail Sales") +  
 ggtitle("Seasonal plot: Monthly retail sales")  
  
gsp <- ggsubseriesplot(myts) +  
 ylab("Retail Sales") +  
 ggtitle("Subseries Plot: Monthly retail sales")  
  
myts\_window <- window(myts, start = c(1982, 4))  
win <- ggAcf(myts\_window)  
  
  
grid.arrange(ap, sp, gsp, win, ncol=2)



lagged <- gglagplot(myts\_window) +   
 theme(axis.text.x = element\_text(angle = 45, vjust = 1, hjust=1)) +   
 ggtitle("Lag Plot")  
print(lagged)



#### Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

* From the autoplot, we can see from the that there is a clear, increasing trend, as well as seasonality.
* From the seasonal trend, it is easier to see that that the seasonality has an annual frequency which is additionally following a steady upward trend over time.
* The subseries plot again shows the trend clearly – within each month there is a steady upward trend that appears to have roughly the same slope every month indicating that the trend is consistent across years.
* The window plot shows a clear trend (rather than white noise) which indicates that there is strong autocorrelation in the series.
* Finally, the lagged scatterplots show that the relationships are strongly positive across all lags, but dramatically positive at lag 12. This indicates a high annual correlation. We can see this in the ACF plot as well. We also notice that the correlation, while remaining positive, becomes less strongly positive as time goes by, and then peaks again every 12 months, again pointing to the 12 monthly frequency.

### HA 6.2

The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

#### a. Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle?

# ?plastics  
# "Monthly sales of product A for a plastics manufacturer."  
  
autoplot(plastics) +  
 ggtitle("Monthly sales of product A for a plastics manufacturer") +  
 xlab("Year") +  
 ylab("Sales")

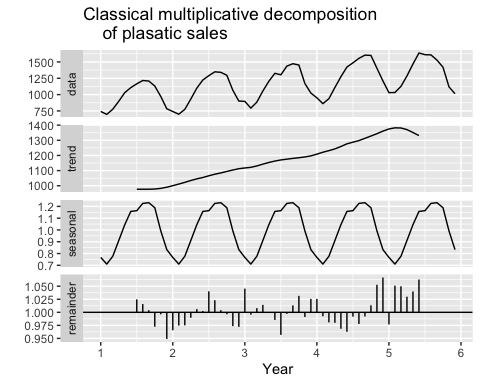


Based on the autoplot there does appear to be seasonality to the data, the plastic sales are highest May - October (peaking in August usually) and lower November - April (lowest in February). There is a consistent upward trend-cycle as well.

#### b. Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal indices.

We use the decompose function with type="mutliplicative" and then autoplot to decompose the series and then evalaute the results.

plastics %>% decompose(type="multiplicative") %>%  
 autoplot() + xlab("Year") +  
 ggtitle("Classical multiplicative decomposition  
 of plasatic sales")



The trend cycle shows a strong seasonal component, with a yearly frequency, and has an increasing trend up until just past year 5 when it begins decreasing. There is some remainder as well. The remainder values below 1 indicate that there is some “leakage” of the trend-cycle component into the remainder component - the trend-cycle estimate has over-smoothed the drop in data.

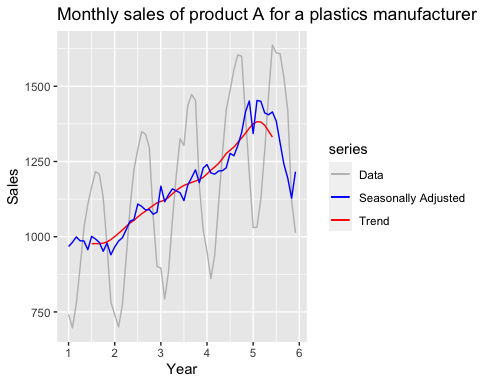
#### c. Do the results support the graphical interpretation from part a?

Yes, the graphical interpretation from part A aligns with the multiplicative decomposition. The yearly seasonal pattern was noted in both a and b. The only part that was not captured well is the drop off in the trend after year 5.

#### d. Compute and plot the seasonally adjusted data.

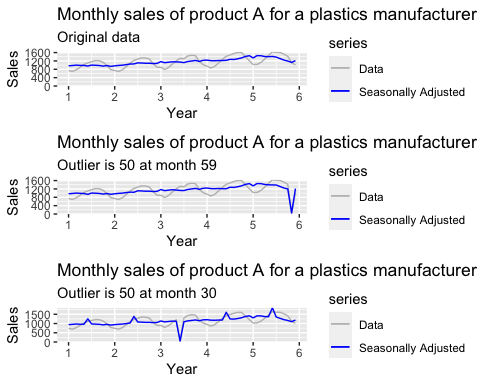
We can compute the seasonally adjusted fit by applying the seasadj function to the decomposed time series.

fit <- plastics %>%  
 decompose(type="multiplicative")   
  
autoplot(plastics, series="Data") +  
 autolayer(trendcycle(fit), series="Trend") +  
 autolayer(seasadj(fit), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Sales") +  
 ggtitle("Monthly sales of product A for a plastics manufacturer") +  
 scale\_colour\_manual(values=c("gray","blue","red"),  
 breaks=c("Data","Seasonally Adjusted","Trend"))



#### e. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

outlier <- function(n, v, outlier){  
 if(outlier==TRUE){  
   
 plastics[n] <- v  
 subtitle <- paste("Outlier is", v, "at month", n)  
  
   
 } else {  
 subtitle <- "Original data"  
  
 }  
   
 fit <- plastics %>% decompose(type="multiplicative")   
  
  
 autoplot(plastics, series="Data") +  
 autolayer(seasadj(fit), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Sales") +  
 labs(  
 title = "Monthly sales of product A for a plastics manufacturer",  
 subtitle = subtitle  
 ) +  
 scale\_colour\_manual(values=c("gray","blue"),  
 breaks=c("Data","Seasonally Adjusted","Trend")) +   
 scale\_y\_continuous(expand = c(0, 0), limits = c(0, NA))  
  
   
}  
  
grid.arrange(  
 outlier(1, 1, FALSE),  
 outlier(59, 50, TRUE),  
 outlier(30, 50, TRUE),   
 ncol=1  
)



The outlier in the middle causes the seasonally adjusted fit to hallucinate seasons that don’t exist! At or near the end, there is no discernible effect.

#### f. Does it make any difference if the outlier is near the end rather than in the middle of the time series?

Yes – the outlier in the middle has a much more pronounced affect than an outlier at the end.

If the outlier is at the end vs in the middle, there is more of an effect on the seasonally adjusted data. The estimate of the trend cycle is unavailable for the first and last few observations - without this there is also no estimate of the remainder component. Due to this, all of the outlier is passed on to the seasonally adjusted rather than having some of it put in the remainder or in the trend.

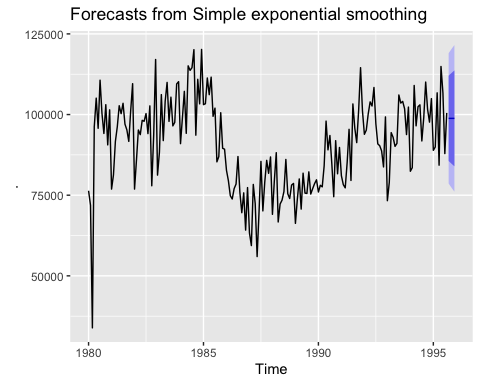
## Week 2 Homework Solutions

### HA 7.1

Consider the pigs series – the number of pigs slaughtered in Victoria each month.

### 7.1 a Use the ses function to find the optimal levels of alpha and l-zero, and generate forecasts for the next 4 months.

fit\_pigs <- pigs %>% ses(h=4)  
  
fit\_pigs %>% autoplot()



summary(fit\_pigs)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = ., h = 4)   
##   
## Smoothing parameters:  
## alpha = 0.2971   
##   
## Initial states:  
## l = 77260.0561   
##   
## sigma: 10308.58  
##   
## AIC AICc BIC   
## 4462.955 4463.086 4472.665   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249 0.01282239  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
## Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
## Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
## Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

By callng summary on the results of the ses object, we can see that the optimal values of alpha and l-zero are 0.2971 and 77260.0561, respectively.

The first four predicted values are given by calling the mean operator on the summary object, as shown below.

summary(fit\_pigs)$mean

## Sep Oct Nov Dec  
## 1995 98816.41 98816.41 98816.41 98816.41

### 7.1 b

Compute a 95% prediction interval for the first forecast using y-hat is plus or minus 1.96s where s is the standard deviation of the residuals. Compare your interval with the interval produced by R.

me <- 1.96\*sd(fit\_pigs$residuals)  
  
f1 <- fit\_pigs$mean[1]

The first forecast is 9.8816406^{4} with a 95% margin of error plus or minus 2.0136439^{4}.

library(dplyr)  
labels\_7.1b <- c("My Results", "R")  
my\_pigs <- c(f1-me, f1+me)  
R\_pigs <- c(fit\_pigs$lower[5], fit\_pigs$upper[5])  
df\_pigs <- data.frame(my\_pigs, R\_pigs)  
colnames(df\_pigs) <- labels\_7.1b  
row.names(df\_pigs) <- c("Lower 95% Bound", "Upper 95% Bound")  
df\_pigs <- df\_pigs %>% mutate(  
 `% Diff` = (`My Results` - R)/R\*100  
)  
  
df\_pigs %>% t()

## Lower 95% Bound Upper 95% Bound  
## My Results 7.867997e+04 1.189528e+05  
## R 7.861197e+04 1.190208e+05  
## % Diff 8.649929e-02 -5.713184e-02

### HA 7.3

(In the previous exercise, we we asked to write our own function to implement simple exponential smoothing.)

Modify your function from the previous exercise to return the sum of the squared errors rather than the forecast of the next observations. The use the optim function to find the optimal methods of alpha and l-0. Do you get the same values as the ses() function?

There is not difference depending on the level for moderate values of alpha, but for small alpha it does make a difference.

T <- length(pigs) %>% as.numeric()  
my\_smooth <- function(y, alpha, level) {  
   
 T <- length(y) %>% as.numeric()  
   
 my\_fitted <- c()  
   
 #set first value   
 my\_fitted[1] <- level  
  
 # I really wish we could do this wihout a for loop!!!  
 # Also I don't understand why this doesn't have the exponential or summation??  
 for (i in 2:(T+1)) {  
 my\_fitted[i] <- alpha \* y[i-1] + (1 - alpha) \* my\_fitted[i-1]  
 }  
   
 #return last value  
 return(my\_fitted)  
 }  
  
alpha <- 0.291  
level <- pigs[1]  
  
my\_smooth(pigs, alpha=alpha, level=level)[T+1]

## [1] 98822.32

my\_smooth(pigs, alpha=alpha, level=level/2)[T+1]

## [1] 98822.32

my\_smooth(pigs, alpha=1/T, level=level)[T+1]

## [1] 85451.49

my\_smooth(pigs, alpha=1/T, level=level/2)[T+1]

## [1] 71439.99

ses(pigs, alpha=alpha)$mean[1]

## [1] 98822.32

my\_sse <- function(y, alpha, level) {  
   
 y\_hats <- my\_smooth(y, alpha, level)  
 resid <- y\_hats - y  
 sse <- mapply(function(x) x^2, resid) %>% sum()  
  
 return(sse)  
  
}

Then use the optim function to find the optimal methods of alpha and l-0. ## NOT DONE!

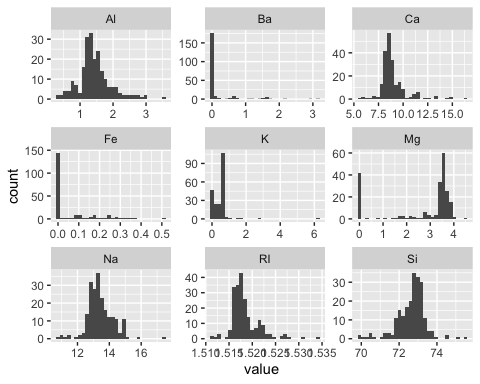
## KJ 3.1

library(mlbench)  
data(Glass)

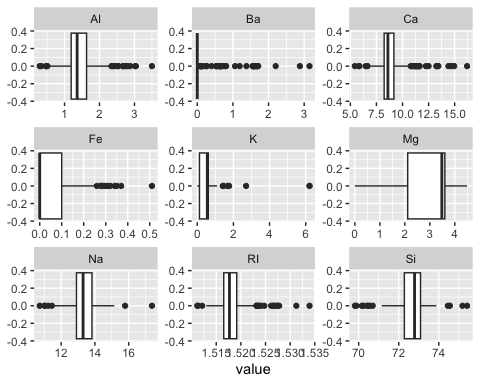
### 3.1a

Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

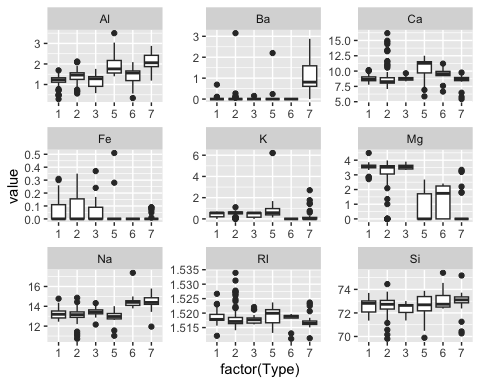
Glass %>% select(-Type) %>% gather() %>% ggplot(aes(value)) + geom\_histogram() + facet\_wrap(~key, scales = "free")



Glass %>% select(-Type) %>% gather() %>% ggplot(aes(value)) + geom\_boxplot() + facet\_wrap(~key, scales = "free")



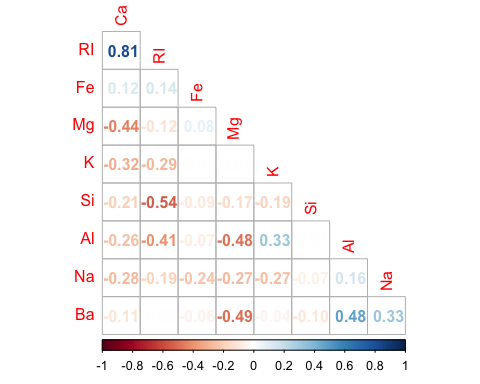
Glass %>% gather("key", "value", -Type) %>%   
 ggplot(aes(factor(Type), value)) +   
 geom\_boxplot() +   
 facet\_wrap(~key, scales = "free")



library(corrplot)

## corrplot 0.92 loaded

# Filter for just quantitative vars  
glassQuant <- Glass %>% select (-Type)  
  
# Corr chart  
corrplot(cor(glassQuant), method="number", order = "AOE", type = "lower", diag = FALSE)



### 3.1b

Do there appear to be any outliers? Are any predictors skewed?

library(psych)

##   
## Attaching package: 'psych'

## The following object is masked \_by\_ '.GlobalEnv':  
##   
## outlier

## The following objects are masked from 'package:ggplot2':  
##   
## %+%, alpha

describe(Glass)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## RI 1 214 1.52 0.00 1.52 1.52 0.00 1.51 1.53 0.02 1.60 4.72  
## Na 2 214 13.41 0.82 13.30 13.38 0.64 10.73 17.38 6.65 0.45 2.90  
## Mg 3 214 2.68 1.44 3.48 2.87 0.30 0.00 4.49 4.49 -1.14 -0.45  
## Al 4 214 1.44 0.50 1.36 1.41 0.31 0.29 3.50 3.21 0.89 1.94  
## Si 5 214 72.65 0.77 72.79 72.71 0.57 69.81 75.41 5.60 -0.72 2.82  
## K 6 214 0.50 0.65 0.56 0.43 0.17 0.00 6.21 6.21 6.46 52.87  
## Ca 7 214 8.96 1.42 8.60 8.74 0.66 5.43 16.19 10.76 2.02 6.41  
## Ba 8 214 0.18 0.50 0.00 0.03 0.00 0.00 3.15 3.15 3.37 12.08  
## Fe 9 214 0.06 0.10 0.00 0.04 0.00 0.00 0.51 0.51 1.73 2.52  
## Type\* 10 214 2.54 1.71 2.00 2.31 1.48 1.00 6.00 5.00 1.04 -0.29  
## se  
## RI 0.00  
## Na 0.06  
## Mg 0.10  
## Al 0.03  
## Si 0.05  
## K 0.04  
## Ca 0.10  
## Ba 0.03  
## Fe 0.01  
## Type\* 0.12

## KJ 3.2

knitr::opts\_chunk$set(echo = TRUE, cache = TRUE)  
library(fpp2)  
library(gridExtra)  
library(tidyverse)  
  
  
  
p1 = autoplot(gold) + ggtitle('Price of Gold: 1985-1989') + ylab('price') + xlab('')  
p2 = autoplot(woolyrnq) + ggtitle('Quarterly Production of Woollen Yarn in Australia: 1965 - 1994') + ylab('tonnes') + xlab('')  
p3 = autoplot(gas) + ggtitle('Australian monthly gas production: 1956-1995') + ylab('gas') + xlab('')  
  
grid.arrange(p1, p2, p3, nrow=3)  
  
Gold <- frequency(gold)  
  
Wool <- frequency(woolyrnq)  
  
Gas <- frequency(gas)  
  
Freqs <- data.frame(  
 Series = c("Gold", "Wool", "Gas"),  
 Frequency = c(Gold, Wool, Gas),  
 Seasonality = c("Daily", "Quaterly", "Monthly")  
 )   
Freqs %>% print()   
library(lubridate)  
  
d <- ymd("1985/01/01") + which.max(gold)  
  
paste0('day:', which.max(gold) , ' date:', d, ' price: $', gold[which.max(gold)])  
  
library(readxl)  
retaildata <- readxl::read\_excel("retail.xlsx", skip=1)  
head(retaildata)  
myts <- ts(retaildata[,"A3349335T"],  
 frequency=12, start=c(1982,4))  
ap <- autoplot(myts) +  
 ggtitle("Autoplot: Monthly retail sales in various categories for different Australian states") +  
 xlab("Year") +  
 ylab("Retail Sales")  
  
sp <- ggseasonplot(myts, year.labels=TRUE, year.labels.left=TRUE) +  
 ylab("Retail Sales") +  
 ggtitle("Seasonal plot: Monthly retail sales")  
  
gsp <- ggsubseriesplot(myts) +  
 ylab("Retail Sales") +  
 ggtitle("Subseries Plot: Monthly retail sales")  
  
myts\_window <- window(myts, start = c(1982, 4))  
win <- ggAcf(myts\_window)  
  
  
grid.arrange(ap, sp, gsp, win, ncol=2)  
lagged <- gglagplot(myts\_window) +   
 theme(axis.text.x = element\_text(angle = 45, vjust = 1, hjust=1)) +   
 ggtitle("Lag Plot")  
print(lagged)  
# ?plastics  
# "Monthly sales of product A for a plastics manufacturer."  
  
autoplot(plastics) +  
 ggtitle("Monthly sales of product A for a plastics manufacturer") +  
 xlab("Year") +  
 ylab("Sales")  
plastics %>% decompose(type="multiplicative") %>%  
 autoplot() + xlab("Year") +  
 ggtitle("Classical multiplicative decomposition  
 of plasatic sales")  
fit <- plastics %>%  
 decompose(type="multiplicative")   
  
autoplot(plastics, series="Data") +  
 autolayer(trendcycle(fit), series="Trend") +  
 autolayer(seasadj(fit), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Sales") +  
 ggtitle("Monthly sales of product A for a plastics manufacturer") +  
 scale\_colour\_manual(values=c("gray","blue","red"),  
 breaks=c("Data","Seasonally Adjusted","Trend"))  
  
outlier <- function(n, v, outlier){  
 if(outlier==TRUE){  
   
 plastics[n] <- v  
 subtitle <- paste("Outlier is", v, "at month", n)  
  
   
 } else {  
 subtitle <- "Original data"  
  
 }  
   
 fit <- plastics %>% decompose(type="multiplicative")   
  
  
 autoplot(plastics, series="Data") +  
 autolayer(seasadj(fit), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Sales") +  
 labs(  
 title = "Monthly sales of product A for a plastics manufacturer",  
 subtitle = subtitle  
 ) +  
 scale\_colour\_manual(values=c("gray","blue"),  
 breaks=c("Data","Seasonally Adjusted","Trend")) +   
 scale\_y\_continuous(expand = c(0, 0), limits = c(0, NA))  
  
   
}  
  
grid.arrange(  
 outlier(1, 1, FALSE),  
 outlier(59, 50, TRUE),  
 outlier(30, 50, TRUE),   
 ncol=1  
)  
  
fit\_pigs <- pigs %>% ses(h=4)  
  
fit\_pigs %>% autoplot()  
  
summary(fit\_pigs)  
summary(fit\_pigs)$mean  
me <- 1.96\*sd(fit\_pigs$residuals)  
  
f1 <- fit\_pigs$mean[1]  
  
  
library(dplyr)  
labels\_7.1b <- c("My Results", "R")  
my\_pigs <- c(f1-me, f1+me)  
R\_pigs <- c(fit\_pigs$lower[5], fit\_pigs$upper[5])  
df\_pigs <- data.frame(my\_pigs, R\_pigs)  
colnames(df\_pigs) <- labels\_7.1b  
row.names(df\_pigs) <- c("Lower 95% Bound", "Upper 95% Bound")  
df\_pigs <- df\_pigs %>% mutate(  
 `% Diff` = (`My Results` - R)/R\*100  
)  
  
df\_pigs %>% t()  
T <- length(pigs) %>% as.numeric()  
my\_smooth <- function(y, alpha, level) {  
   
 T <- length(y) %>% as.numeric()  
   
 my\_fitted <- c()  
   
 #set first value   
 my\_fitted[1] <- level  
  
 # I really wish we could do this wihout a for loop!!!  
 # Also I don't understand why this doesn't have the exponential or summation??  
 for (i in 2:(T+1)) {  
 my\_fitted[i] <- alpha \* y[i-1] + (1 - alpha) \* my\_fitted[i-1]  
 }  
   
 #return last value  
 return(my\_fitted)  
 }  
  
alpha <- 0.291  
level <- pigs[1]  
  
my\_smooth(pigs, alpha=alpha, level=level)[T+1]  
my\_smooth(pigs, alpha=alpha, level=level/2)[T+1]  
my\_smooth(pigs, alpha=1/T, level=level)[T+1]  
my\_smooth(pigs, alpha=1/T, level=level/2)[T+1]  
  
ses(pigs, alpha=alpha)$mean[1]  
  
my\_sse <- function(y, alpha, level) {  
   
 y\_hats <- my\_smooth(y, alpha, level)  
 resid <- y\_hats - y  
 sse <- mapply(function(x) x^2, resid) %>% sum()  
  
 return(sse)  
  
}  
  
library(mlbench)  
data(Glass)  
  
Glass %>% select(-Type) %>% gather() %>% ggplot(aes(value)) + geom\_histogram() + facet\_wrap(~key, scales = "free")  
  
Glass %>% select(-Type) %>% gather() %>% ggplot(aes(value)) + geom\_boxplot() + facet\_wrap(~key, scales = "free")  
  
Glass %>% gather("key", "value", -Type) %>%   
 ggplot(aes(factor(Type), value)) +   
 geom\_boxplot() +   
 facet\_wrap(~key, scales = "free")   
  
library(corrplot)  
  
# Filter for just quantitative vars  
glassQuant <- Glass %>% select (-Type)  
  
# Corr chart  
corrplot(cor(glassQuant), method="number", order = "AOE", type = "lower", diag = FALSE)  
library(psych)  
describe(Glass)