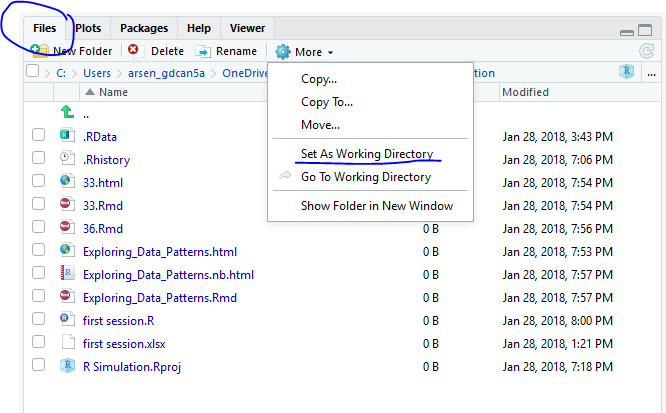
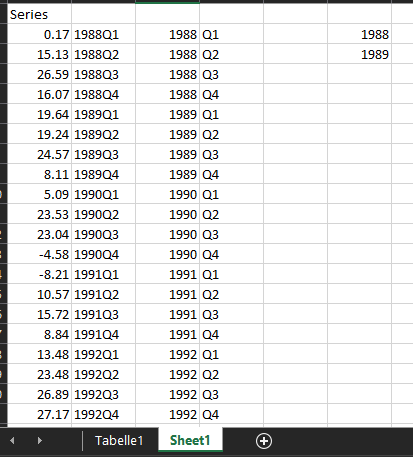
Data Patterns

This section covers sessions 1-2, chapters 2-3 of HW. Covered topics: - getting data into R - making a time series - exploring autocorrelation - how to do formulas from p.41-42 of HW - naive forecasting method - moving averages - exponential smoothing - Holt’s linear smoothing - Winter’s method - Winters-Holt

Load Data into R and transform to Time Series

First, set your working directory to wherever your excel file is, otherwise you’ll have to type in the complete file path.Also save your R project and your R script there. 

Now you need to create an object and read the data from an excel file. If you typed the values from the book, type them as one series (one column). If you are using someone’s file, transform them to one column. You don’t need the years and quarters.Here is how my excel file looks after transforming: 

There are two tabs in the file and I concatenated the Year+Quarter. Now we want to import the data from Sheet1 to R.

#For this, we use the package readxl and its function read\_excel  
#first, apply the function to the file  
#create an object called firstdata0 and apply the function to it  
#we have to specify the file name. This argument is a string, therefore use quotation marks ""  
firstdata0 <- readxl::read\_excel("first session.xlsx")  
firstdata0

## # A tibble: 12 x 18  
## Year `1st` `2nd` `3rd` `4th` X\_\_1 X\_\_2 X\_\_3 X\_\_4 X\_\_5 X\_\_6   
## <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <lgl> <lgl> <lgl> <lgl> <lgl>  
## 1 1988 0.170 15.1 26.6 16.1 1988Q1 NA NA NA NA NA   
## 2 1989 19.6 19.2 24.6 8.11 1988Q2 NA NA NA NA NA   
## 3 1990 5.09 23.5 23.0 - 4.58 1988Q3 NA NA NA NA NA   
## 4 1991 - 8.21 10.6 15.7 8.84 1988Q4 NA NA NA NA NA   
## 5 1992 13.5 23.5 26.9 27.2 <NA> NA NA NA NA NA   
## 6 1993 24.9 42.2 48.8 38.4 <NA> NA NA NA NA NA   
## 7 1994 41.8 58.5 58.6 20.3 <NA> NA NA NA NA NA   
## 8 1995 11.8 59.7 67.7 43.4 <NA> NA NA NA NA NA   
## 9 1996 33.0 85.3 60.9 28.2 <NA> NA NA NA NA NA   
## 10 1997 50.9 93.8 92.5 80.6 <NA> NA NA NA NA NA   
## 11 1998 70.0 133 130 100 <NA> NA NA NA NA NA   
## 12 1999 95.8 158 127 93.8 <NA> NA NA NA NA NA   
## # ... with 7 more variables: X\_\_7 <lgl>, X\_\_8 <lgl>, X\_\_9 <lgl>,  
## # X\_\_10 <dbl>, X\_\_11 <dbl>, X\_\_12 <dbl>, X\_\_13 <dbl>

Clearly that’s not what we want. Our goal is to get the time series in a friendly form.

#remove firstdata0  
remove(firstdata0)  
#add some inputs to the function  
#We additionally specify the file name, which sheet to take the data from, and the range. Why the range? I only want to get the first column.  
firstdata <- readxl::read\_excel("first session.xlsx", sheet='Sheet1', range = "A1:A48")  
#this displays the first five entries  
head(firstdata, n=5)

## # A tibble: 5 x 1  
## Series  
## <dbl>  
## 1 0.170  
## 2 15.1   
## 3 26.6   
## 4 16.1   
## 5 19.6

Now the data is loaded into R. Let’s check what object type we have

str(firstdata)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 47 obs. of 1 variable:  
## $ Series: num 0.17 15.13 26.59 16.07 19.64 ...

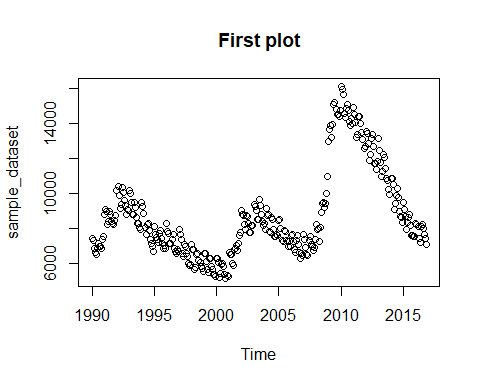
This is a data frame. We however want a timeseries because then we can use a number of specific functions.

#to make a time series, create a new object and apply the ts function to it like this:  
series <- ts(firstdata$Series, frequency=4, start=1988)  
#as you can see, we defined the frequency of observations as 4 per year and the starting year as 1988.   
#let's display the object  
series

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1988 0.17 15.13 26.59 16.07  
## 1989 19.64 19.24 24.57 8.11  
## 1990 5.09 23.53 23.04 -4.58  
## 1991 -8.21 10.57 15.72 8.84  
## 1992 13.48 23.48 26.89 27.17  
## 1993 24.93 42.15 48.83 38.37  
## 1994 41.85 58.52 58.62 20.34  
## 1995 11.83 59.72 67.72 43.36  
## 1996 33.00 85.32 60.86 28.16  
## 1997 50.87 93.83 92.51 80.55  
## 1998 70.01 133.39 129.64 100.38  
## 1999 95.85 157.76 126.98

Let’s plot a series (standard series of R-base).

#we use a generic plot function, there are many more.  
#we use type = "p" for points in the scatterplot. Other types are available, use Help in Rstudio(F1) to find out more.  
#main is the title of the plot  
library(seasonal)  
sample\_dataset <- unemp  
plot(sample\_dataset, type="p",main="First plot")



Autocorrelation

To do this, we use the acf function from the stats package, which is pre-loaded.

# Load packages  
library(zoo)

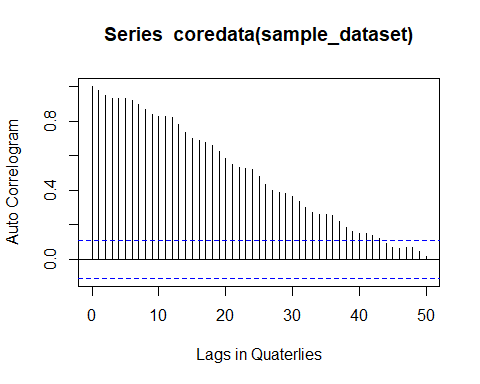
##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(forecast)  
# We specify the series, then we want to first look at the ACF, then we want to see it plotted. All until lag 10. Coredata is used to extract the timeseries plainly.   
acf(coredata(sample\_dataset), plot=FALSE, lag.max =10)

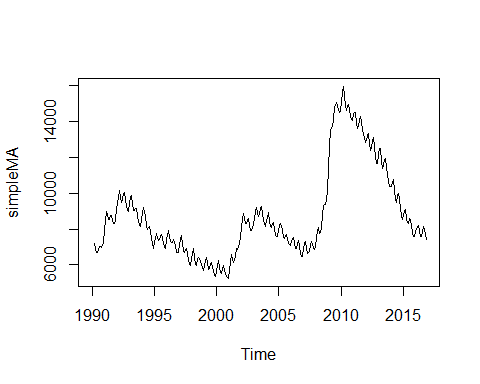
##   
## Autocorrelations of series 'coredata(sample\_dataset)', by lag  
##   
## 0 1 2 3 4 5 6 7 8 9 10   
## 1.000 0.976 0.947 0.928 0.927 0.930 0.919 0.898 0.865 0.835 0.824

acf(coredata(sample\_dataset), plot=TRUE, lag.max=50, xlim=c(0,50), ylab="Auto Correlogram", xlab="Lags in Quaterlies")



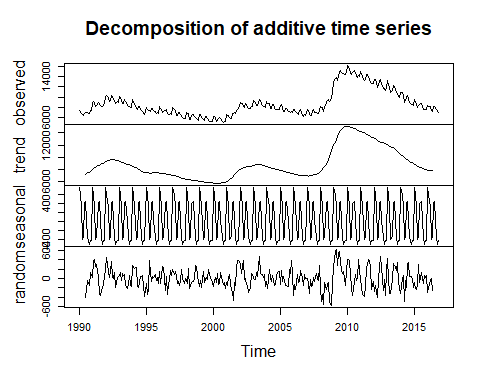
For forecasting we use the “TTR” package. Make sure to load this package. For the example we will be using the dataset “austres” that is preloaded in R. Fromt here you can use the package as shown below. Note that we are using an moving average of 3.

library(TTR)  
simpleMA <- SMA(sample\_dataset, n=3)  
plot.ts(simpleMA)



For decomposition of your forecast, it is important to analyse the seasonality, trend and randomness of your historic data. This can easily be done in R by calling the decompose function.

decomposition\_of\_historic\_data <- decompose(sample\_dataset)  
  
plot(decomposition\_of\_historic\_data)



In order to create predictions we can use the HoltWinters Method. This is the most evolved version of the exponential smoothing methods. For simplicity, realize that beta and gamma can be set to zero, disabling the seasonal and/or trend component, reducing it to simpler technique.

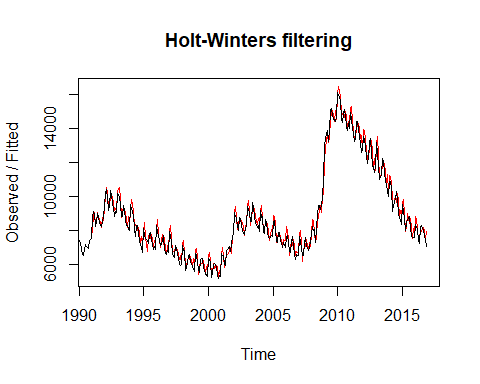
exp <- HoltWinters(sample\_dataset, beta = 0, gamma = 0)  
exp\_seasonal <- HoltWinters(sample\_dataset, gamma = 0)  
exp\_seasonal\_trend <- HoltWinters(sample\_dataset)  
  
print(exp)

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = sample\_dataset, beta = 0, gamma = 0)  
##   
## Smoothing parameters:  
## alpha: 0.9999498  
## beta : 0  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 7437.15162  
## b 137.37602  
## s1 -409.19097  
## s2 696.01736  
## s3 901.43403  
## s4 663.01736  
## s5 -214.60764  
## s6 -107.19097  
## s7 352.80903  
## s8 -29.10764  
## s9 -291.39931  
## s10 -461.73264  
## s11 -728.94097  
## s12 -371.10764

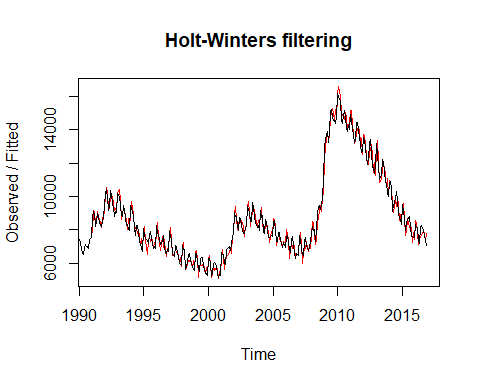
predictions <- exp$fitted  
head(predictions)

## xhat level trend season  
## Jan 1991 7837.925 7004.532 137.376 696.0174  
## Feb 1991 9129.745 8090.935 137.376 901.4340  
## Mar 1991 9019.960 8219.566 137.376 663.0174  
## Apr 1991 8254.752 8331.984 137.376 -214.6076  
## May 1991 8470.794 8440.609 137.376 -107.1910  
## Jun 1991 9032.378 8542.193 137.376 352.8090

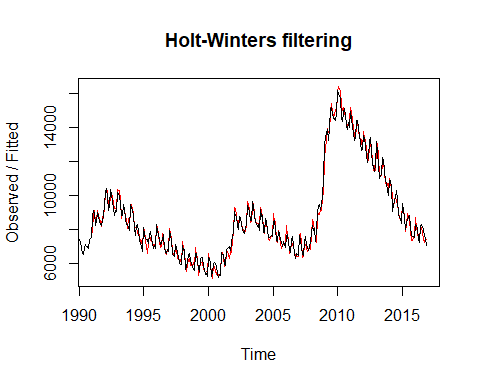
plot(exp)



plot(exp\_seasonal)



plot(exp\_seasonal\_trend)



cat("SSE exp", exp$SSE, "\n", "SSE exp Seasonality", exp\_seasonal$SSE, "\n", "SSE exp Seasonality and Trend", exp\_seasonal\_trend$SSE, "\n")

## SSE exp 41957605   
## SSE exp Seasonality 35865368   
## SSE exp Seasonality and Trend 25231887

In order calculate all the performance measures for the prediction a couple of numbers need to be retrieved. The formula v = n - m relates to the R objects as n = the amoun of values, m = the amount of coefficient. v represents the residual degrees of freedom. In order to calculate everthing the following steps need to be conducted:

n <- length(exp$fitted)  
m <- length(exp$coefficients)  
v <- n - m   
  
cat("With this formula we have established the values for n and m, those being", n, "and", m, "therefore we can calculate v by substraction.", "V is", v)

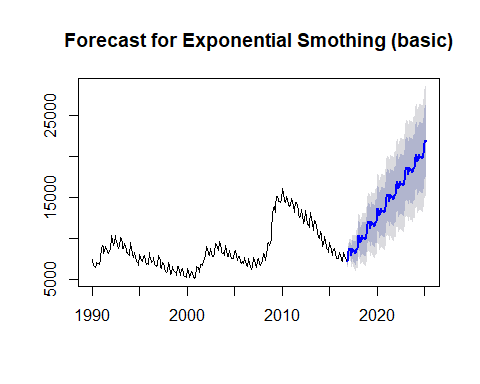
## With this formula we have established the values for n and m, those being 1244 and 14 therefore we can calculate v by substraction. V is 1230

# Calculate MSE  
  
MSE <- exp$SSE / v  
cat("MSE =", MSE)

## MSE = 34111.87

In order to forecast the data for hundred periods, one can use the forecast package. The syntax works as follows:

exp\_fore <- forecast::forecast(exp, h = 100)  
exp\_fore\_trend\_seasonal <- forecast::forecast(exp\_seasonal\_trend, h = 100)  
  
plot(exp\_fore, main="Forecast for Exponential Smothing (basic)")



plot(exp\_fore\_trend\_seasonal, main = "Forecast for HoltWinters method with Trend and Seasonality")

