

Workshop 9 - Time Series

JPS

28/12/2021

Introduction

We are going to practice time-series analysis using ARIMA models. In the first exercise we will go step by step covering all the steps that should be taken into consideration when describing and modelling time-series. We will learn how to plot and decompose time-series into its main different elements (T , S , and I), and how to build ARIMA models by carefully considering the number of *differences* required, and their *AR* and *MA* components. After estimating our ARIMA models we will use them to make future predictions. In the second exercise we will follow an unsupervised (*data driven*) approach to build ARIMA models. This follows the steps undertaken in exercise 1, but does so in an unsupervised way, i.e. without having a person making the modelling decisions. The first exercise will be harder but it is important so you can see everything that goes behind the estimation of even relatively simple ARIMA models. Once we have learnt how the *black box* works we will employ the faster (unsupervised) approach to explore more substantively interesting research questions.

Exercise 1: This first exercise borrows different examples from R. Dalinina and S. Sapegina time-series tutorials, and is based on the same dataset they use. The data comes from a bike sharing company, capturing daily count of bikes rented between years 2011 and 2012 (if you want to know more about this dataset click [here](#)). We are going to explore this time-series and learn from it. Is there a trend? I.e., do more people rent bikes over time? Is there a seasonal effect? Do more people rent bikes certain days of the week? Or different months of the year? Can we learn enough from its past behaviour to be able to forecast bike renting in the future? And how reliably can we do that?

Exercise 2: In this exercise we are going to replicate some of the analyses carried out in Pina-Sánchez et al. (2019) ‘Have The England and Wales Guidelines Affected Sentencing Severity? An Empirical Analysis Using a Scale of Severity and Time-Series Analyses’. As the title indicates in this paper we assess whether the new sentencing guidelines can be blamed for having increased sentence severity in England and Wales. To explore this question we will use data from the Ministry of Justice Quarterly Statistics and ARIMA models built using the command *auto.ARIMA*.

Exercise 1. Bike Share

We start by importing the data, which is available in Minerva as a .csv file.

```
bike = read.csv('day.csv')
```

We can see that every case represents a different day, from the 1st of January of 2011 to the 31st of December of 2012. For each observation we have multiple variables capturing the characteristics of that day, day of the week, whether a working day, and weather conditions such as temperature, windspeed, etc.

```
head(bike)
```

```
##   instant      dteday season yr  mnth holiday weekday workingday weathersit
## 1         1 2011-01-01      1  0    1         0         6           0         2
## 2         2 2011-01-02      1  0    1         0         0           0         2
## 3         3 2011-01-03      1  0    1         0         1           1         1
```

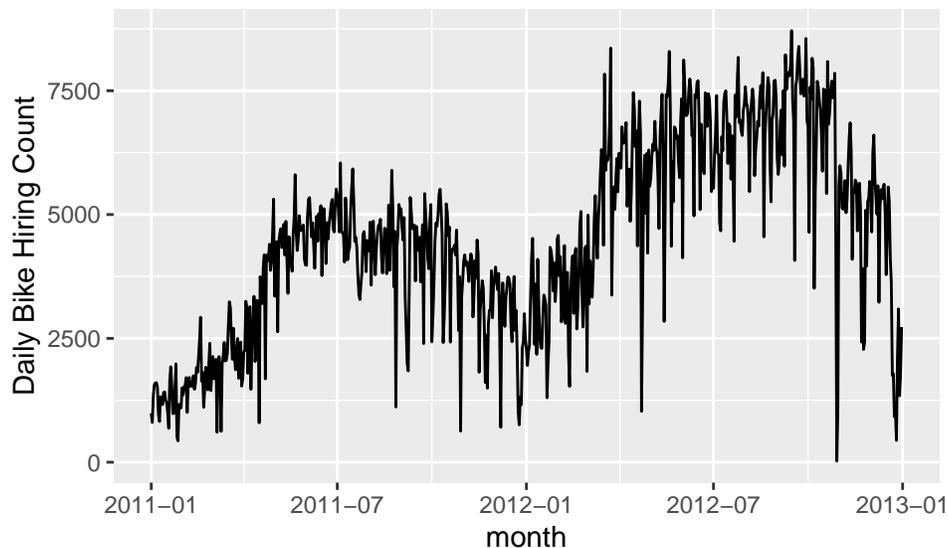
```
## 4      4 2011-01-04      1 0 1      0      2      1      1
## 5      5 2011-01-05      1 0 1      0      3      1      1
## 6      6 2011-01-06      1 0 1      0      4      1      1
##      temp      atemp      hum windspeed casual registered cnt
## 1 0.344167 0.363625 0.805833 0.1604460    331      654 985
## 2 0.363478 0.353739 0.696087 0.2485390    131      670 801
## 3 0.196364 0.189405 0.437273 0.2483090    120     1229 1349
## 4 0.200000 0.212122 0.590435 0.1602960    108     1454 1562
## 5 0.226957 0.229270 0.436957 0.1869000     82     1518 1600
## 6 0.204348 0.233209 0.518261 0.0895652     88     1518 1606
```

```
bike$dtoday[731] #731 is the last observation in the dataset.
```

```
## [1] "2012-12-31"
```

However, for our analysis we will only use two of the variables in the dataset: 'Date' and 'cnt'. Putting these two variables together we will be able to generate our time-series describing the number of bikes that were hired across the window of observation. But before doing that we need to tell R that 'dtoday' needs to be considered a variable of class date, instead of a character type variable.

```
bike$Date = as.Date(bike$dtoday)
library(ggplot2)
ggplot(bike, aes(Date, cnt)) + geom_line() + scale_x_date('month') + ylab("Daily Bike Hiring Count")
```

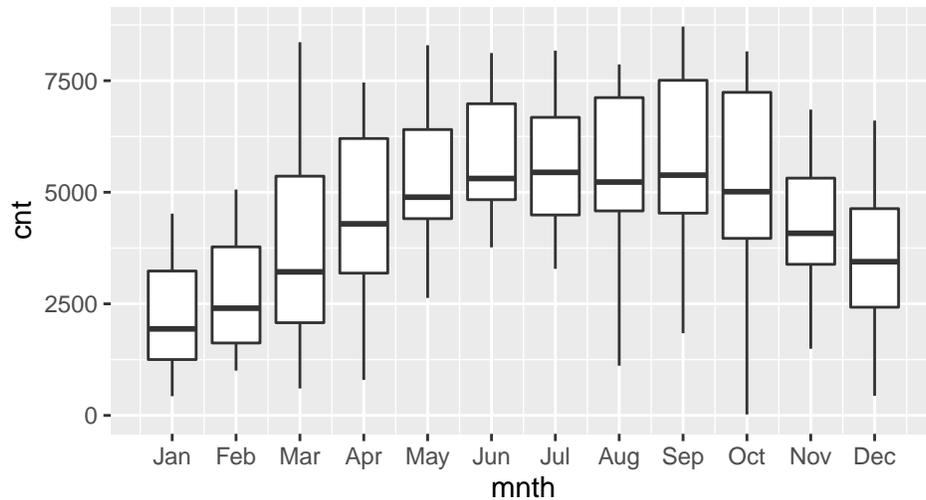


Visualising the time-series in a plot should always be the first step in the analysis of time-series. This will provide us with some initial insights about the patterns and trends in the data, which will help us guide our modelling strategy. From the above plot we can see lots of volatility, with some strong fluctuations from one day to another. However, even with this volatility present, we can already see some patterns emerge. For example, it seems that the use of bicycles decreases in the winter months and increases in the summer months. There also seems to be a trend roughly growing until October'2012 but then it declines rapidly from then. It is not clear whether fluctuations are also due to day of the week since we do not have enough resolution in the graph to identify the specific days of the week, but from a theoretical point of view one might expect that if bike sharing is a recreational service we should see higher usage during the weekend.

In order to assess these potential seasonal effects (*seasonal* understood loosely as any kind of fluctuations observable in the dataset over a regular time pattern) across the months of the year, and the days of the week, more clearly, we can use boxplots.

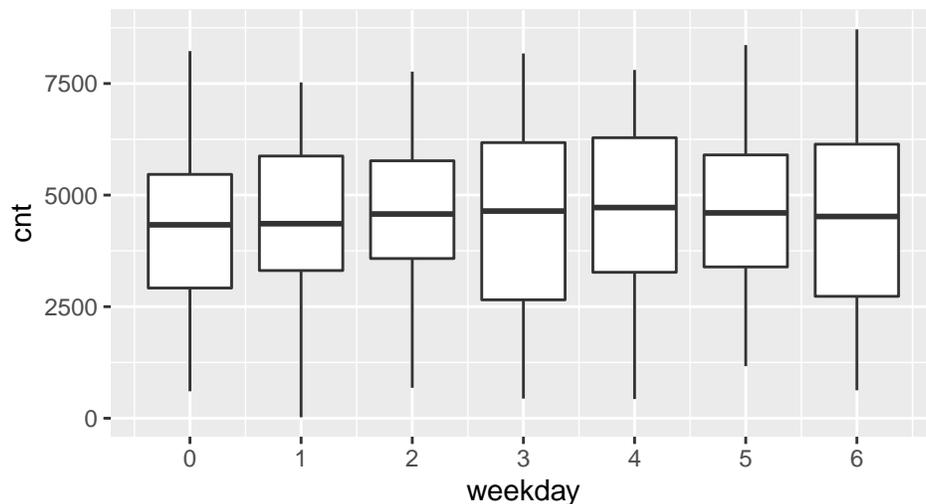
```
ggplot(bike, aes(y=cnt, x = mnth)) + geom_boxplot(aes(group=mnth)) + scale_x_continuous(breaks=seq(0,12),
labels=c("", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
```

Bike Hiring Count by Month



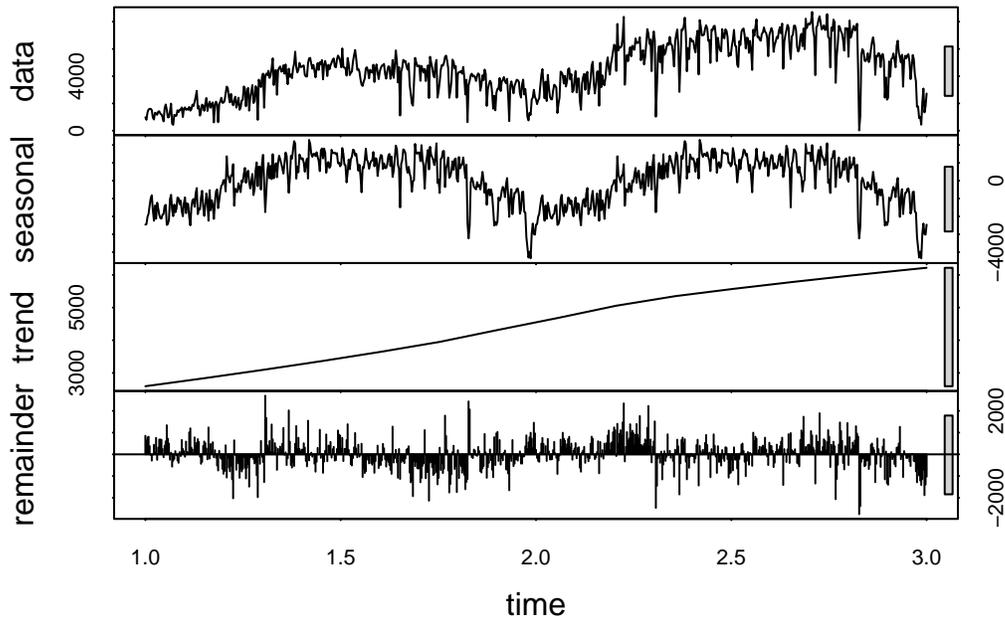
```
ggplot(bike, aes(y=cnt, x = weekday)) + geom_boxplot(aes(group=weekday)) +
scale_x_continuous(breaks=seq(0,7,1)) + ggtitle("Bike Hiring Count by Day")
```

Bike Hiring Count by Day



As suspected we can see some clear seasonal effects across months. However, the day of the week does not seem to influence bike usage at all. This is an important insight, it will help us identify the S component when modelling the time-series. To visualise all the other components defining the time-series we can use the really useful command *stl*, for which we first need to define the time-series itself using *ts*. Notice that in defining our time-series, and in the presence of a seasonal effect, we also need to include the number of observations that need to be contemplated within each *season*. Since we have observations at the day level, and we have identified seasonal effects at the month level that seem to be repeated each year, we can use a frequency of 365 for *ts*. This indicates that the seasonal cycle is repeated every 365 time periods (in our case days).

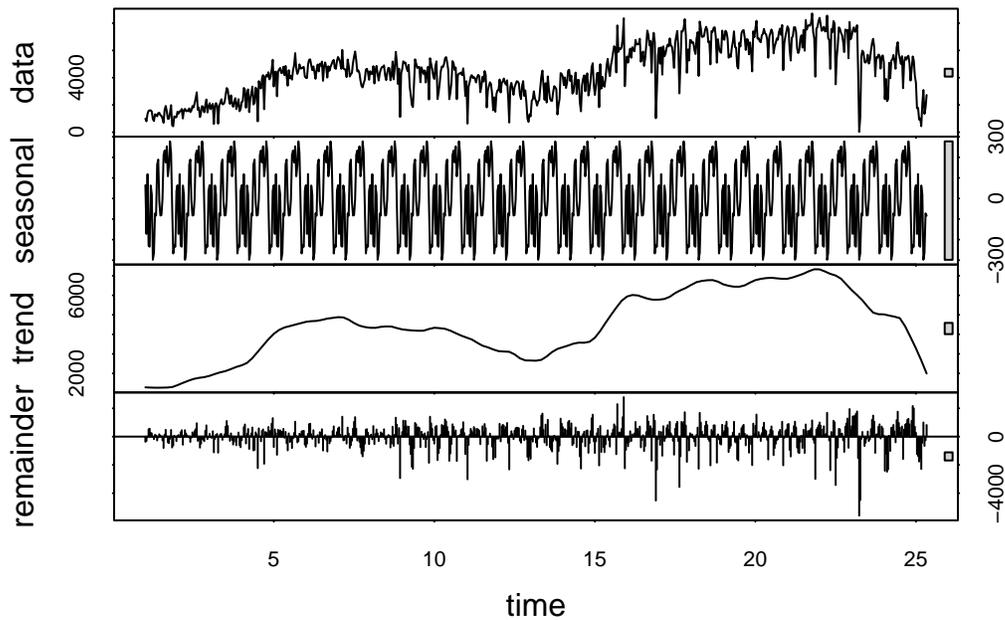
```
count_ts = ts(bike$cnt, frequency=365)
decomp = stl(count_ts, s.window="periodic")
plot(decomp)
```



The first plot (*data*) represents the time-series as it is. The second (*seasonal*) depicts the seasonal effect that we have noticed previously, bike usage goes up in the summer and down in the winter. The third plot is the *trend*, which is obtained after subtracting *seasonal* and *remainder* from *data*. The fourth plot (*remainder*) represents the irregularity component, i.e. what remains unexplained after controlling for the *trend* and *seasonal* effect.

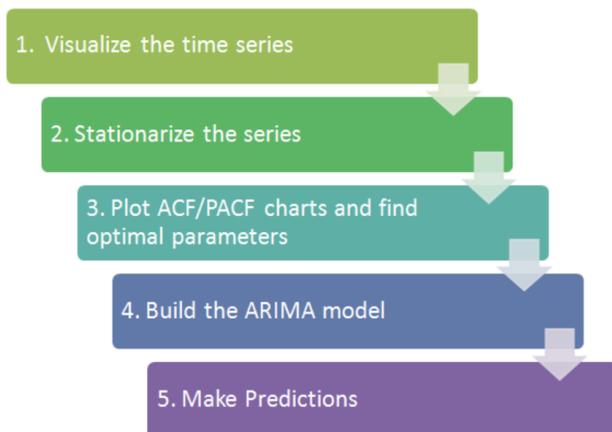
Notice as well that the seasonal effect is only repeated twice since the window of observation used in this dataset only covers two years. If we had say 20 years worth of data it might make sense to keep the formatting of our time-series as we just did, but with only two years in our window of observation, we are not going to be able to model these effects that well. Instead, I would suggest using seasonal effects within each month, that should allow us to break down the timeframe into 24 monthly observations that we can use in our model, rather than two observations at the annual level.

```
count_ts = ts(bike$cnt, frequency=30)
decomp = stl(count_ts, s.window="periodic")
plot(decomp)
```



Ok, this seems to make sense, we can see how the trend is still reflecting the seasonal effects within years, which isn't ideal, but at least we now have broken down the time-series into 24 periods that we can use to build our ARIMA models.

We are now going to move from the descriptive part of the analysis into the modelling part. It is important to understand how when building ARIMA models we will need to follow a specific sequence of steps. These are nicely represented in the following flowchart (source: Tavish Srivastava)



We have just completed step 1. Step 2 involves checking that the time-series is stationary, and if it is not, proceed to differentiate it until it becomes stationary. In the lecture we saw that the main condition for a time-series to be considered stationary is that its properties remain constant across time. Just from the look of the upward trend in our time-series we can see that it does not seem stationary. However, if we want to test this more formally we can use an Augmented Dickey Fuller test, which can be called using the command `adf.test` from the package `tseries`.

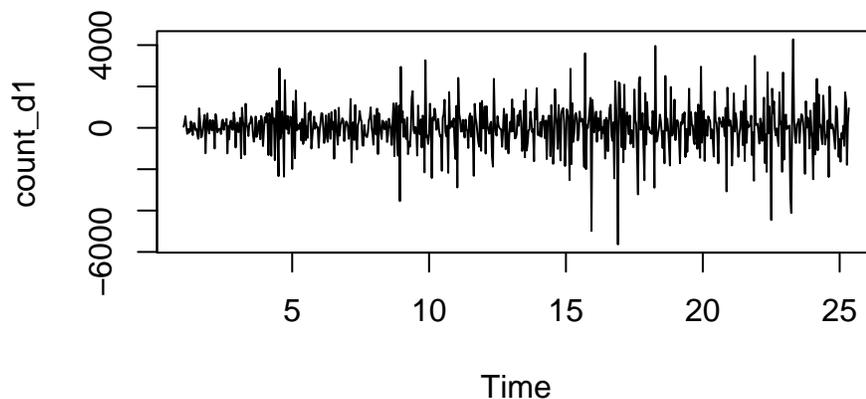
```
library(tseries)
adf.test(count_ts)
```

```
##
## Augmented Dickey-Fuller Test
##
## data: count_ts
## Dickey-Fuller = -1.6351, Lag order = 9, p-value = 0.7327
## alternative hypothesis: stationary
```

We can see that the p-value obtained for ' H_0 : time-series not stationary' is too high, so we cannot reject H_0 , which means we accept the alternative hypothesis pointing at the time-series being stationary.

What we need to do now is proceed to differentiate the time-series until it becomes stationary. Let's do that but let's also keep in mind that we have identified a seasonal component within months that we would also like to take into account while carrying out the difference. So, we will use the following formula $Y_t^d = (Y_t - Y_{t-s}) - (Y_{t-1} - Y_{t-s-1})$ rather than the simpler difference process when no seasonality is present, $Y_t^d = Y_t - Y_{t-1}$. Specifically, we will remove the seasonal component first using *seasadj* from the package *forecast*, and then will proceed with a simple one lag difference using *diff*.

```
library(forecast)
deseasonal_cnt = seasadj(decomp) #This is to remove the seasonal component
count_d1 = diff(deseasonal_cnt, differences=1) #The difference process
plot(count_d1) #To assess stationarity visually
```



```
adf.test(count_d1)
```

```
##
## Augmented Dickey-Fuller Test
##
## data: count_d1
## Dickey-Fuller = -13.859, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

One difference was enough to turn the time-series stationary. This could be determined visually by plotting the differentiated serie, as we did above, where we cannot detect any trend or seasonality, which is confirmed by the subsequent ADF test.

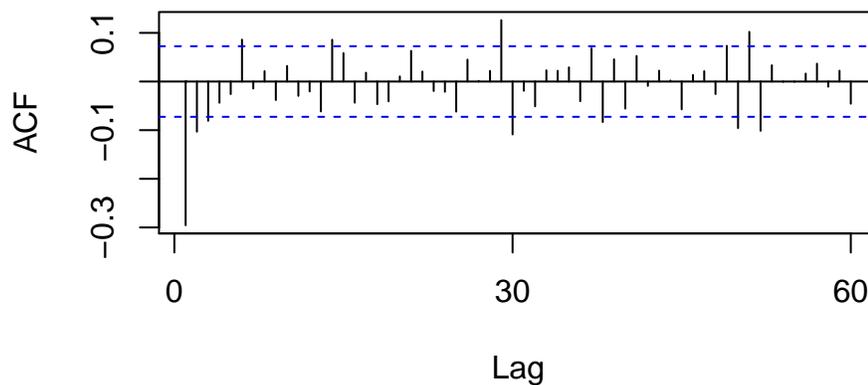
Ok, so we have now completed step 2 of the modelling process. Let's now move to step 3, plotting auto-correlation (ACF) and partial auto-correlation functions (PACF) to help us decide on the number of parameters to be included in our ARIMA model.

The rule of thumb is to define the AR order by looking at how many auto-correlations appear to be significant, starting from the left-hand side of the plot, that is, starting from the auto-correlation of the time-series with its previous lag, then looking at the second lag, third, etc. To define the order of the MA component we will follow the same approach but looking at the PACF.

This process, however, is not deterministic, it involves a bit of comparing and tweaking. This is because, by definition, the more auto-regressive components you include in your model the less unexplained variance there will be, therefore the smaller the partial auto-correlation components and the less of a need for moving averages. And vice versa. In addition, just because one of those components appear significant it does not mean that including it will improve the model. Remember that for a 95% confidence level we should expect 1 in 20 estimates to appear significant due to random variation only (i.e. false positives). Remember as well that we should penalise overfitted models, or put differently, we should aim for simple models. So, we need to use our judgement, consider two or more model specifications and then compare their goodness of fit, for which we will use the AIC.

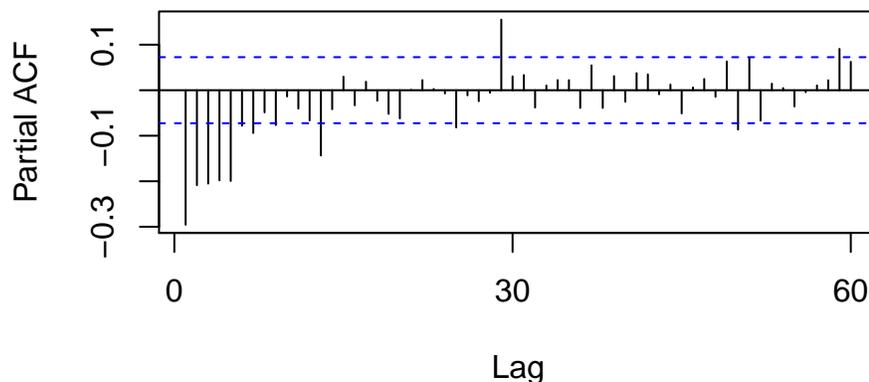
```
Acf(count_d1, main='ACF for Differenced Serie')
```

ACF for Differenced Serie



```
Pacf(count_d1, main='PACF for Differenced Serie')
```

PACF for Differenced Serie



Based on the ACF and PACF plots I would suggest considering two models, one relatively simple, and another one more complex:

1. Model 1, defined as AR(1) and MA(1). This is because the first lags are by far the more prominent in both ACF and PACF plots, so I suspect after controlling for these, the other lags that appear significant in the ACF and PACF plots might become less relevant.
2. Model 2, defined as AR(1) and MA(5). Here I seek to investigate the potential recurrent MA lag for the first five lags. These are all significant in the PACF, and the fact that they do not decay across that interval of time makes me think that those components might be relevant in specifying our time-series.

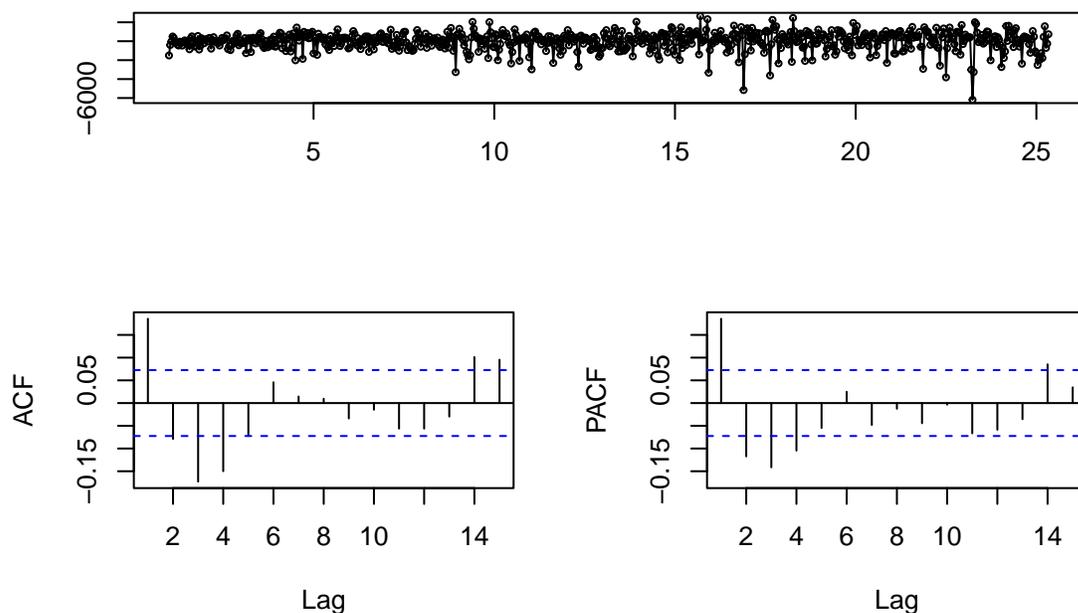
To estimate these two models we are going to use the command *ARIMA*. We can feed the time-series that we have already differentiated, which is what the code below does, or we could simply specify the required difference in the command by changing ‘order=c(1,0,1)’ to ‘order=c(1,1,1)’ in the specification of model 1 and ‘order=c(1,0,5)’ to ‘order=c(1,1,5)’ in model 2. Notice that we have now entered step 4 in the flowchart “Build the ARIMA model”.

```
model1 = arima(deseasonal_cnt, order=c(1,0,1))  
model2 = arima(deseasonal_cnt, order=c(1,0,5))
```

One way to assess the goodness of fit of the model is to plot the model’s residuals and any potential remaining auto-correlation and partial auto-correlation. To do so we can use the command *tsdisplay*.

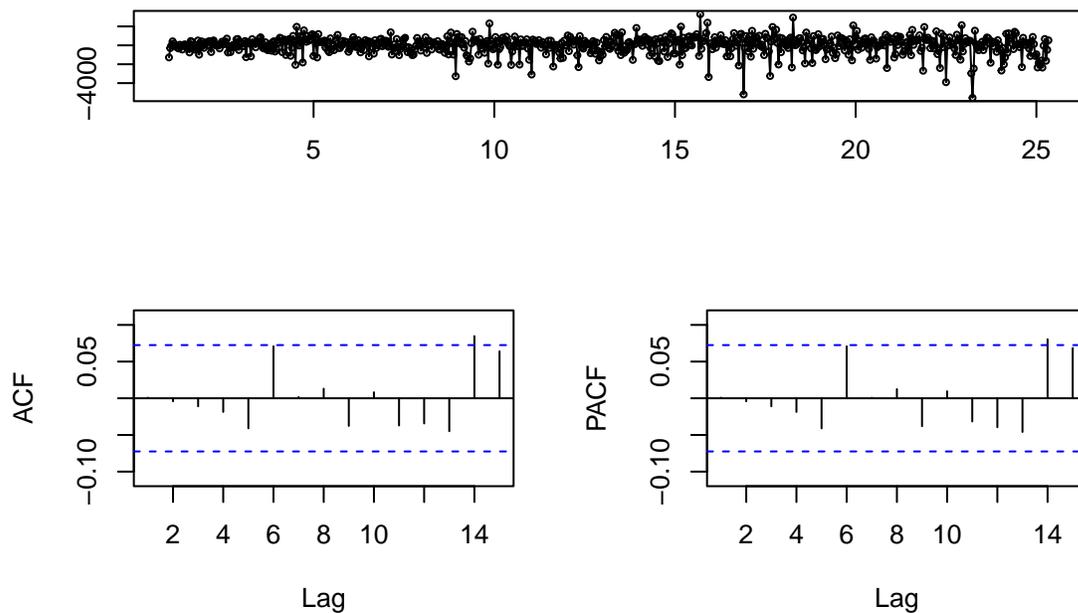
```
tsdisplay(residuals(model1), lag.max=15, main='(1,0,1) Model Residuals')
```

(1,0,1) Model Residuals



```
tsdisplay(residuals(model2), lag.max=15, main='(1,0,5) Model Residuals')
```

(1,0,5) Model Residuals



Notice how the residuals from the two models look kind of similar, but there are some clear differences when we look at the ACF and PACF plots, with many more AC and PAC lags that appear significant in model 1. To assess more clearly which is the better model we can also compare their AICs, which are recorded in the

objects that we have created for each model. So we can simply call those and see.

```
model1

##
## Call:
## arima(x = deseasonal_cnt, order = c(1, 0, 1))
##
## Coefficients:
##      ar1      ma1  intercept
##    0.9871 -0.6865 4127.9943
## s.e. 0.0075  0.0536  794.2119
##
## sigma^2 estimated as 897281:  log likelihood = -6048.22,  aic = 12104.44
```

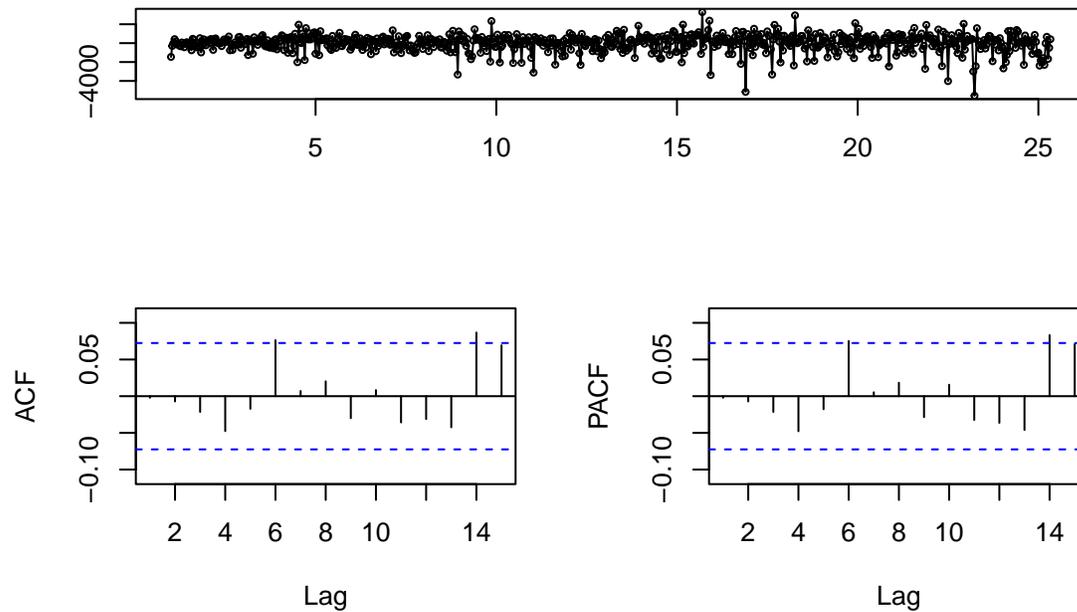
```
model2

##
## Call:
## arima(x = deseasonal_cnt, order = c(1, 0, 5))
##
## Coefficients:
##      ar1      ma1      ma2      ma3      ma4      ma5  intercept
##    0.9959 -0.5309 -0.1960 -0.1058 -0.0219  0.0402  3826.158
## s.e. 0.0032  0.0375  0.0428  0.0391  0.0419  0.0352  1185.598
##
## sigma^2 estimated as 824179:  log likelihood = -6017.35,  aic = 12050.7
```

As suspected, model 2 shows a lower AIC and can therefore be considered the better model. We can get a clue for why that is the case by looking at the four additional MA components included in the model. We do not get p-values for each coefficient, but by comparing their size against that of their standard errors we can deduce that MA2 and MA3 are significant. The rule of thumb is, if the standard errors multiplied by two are larger than the absolute value of the regression coefficient, then you can consider that coefficient is not statistically significant. Similarly, we can also see that MA4 and MA5 are not significant, so it is worth specifying one more model without them and see if it performs better. Remember that we should always uphold the principle of parsimony when building models, that is, aim for simplicity and avoid overfitted models.

```
model3 = arima(deseasonal_cnt, order=c(1,0,3))
tsdisplay(residuals(model3), lag.max=15, main='(1,0,3) Model Residuals')
```

(1,0,3) Model Residuals



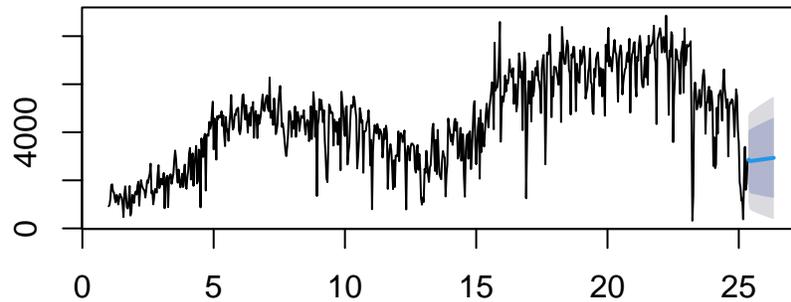
```
model3
##
## Call:
## arima(x = deseasonal_cnt, order = c(1, 0, 3))
##
## Coefficients:
##      ar1      ma1      ma2      ma3  intercept
##  0.9970 -0.5255 -0.1980 -0.0980  4565.805
## s.e.  0.0051  0.0369  0.0418  0.0366  2468.286
##
## sigma^2 estimated as 826167:  log likelihood = -6018.34,  aic = 12048.68
```

Model 3 residuals also seem ok, with no significant AC or PAC, and the AIC of the model is lower than for model 2, which points at model 3 being the best of the three we have estimated, and therefore the one we will end up using.

Ok, so this ends the model building process. At this point we can enter step 5, that is, we can start using our model to forecast future values. To do so we will use the *forecast* command. We will consider a forecast one month into the future, and another one for six months into the future.

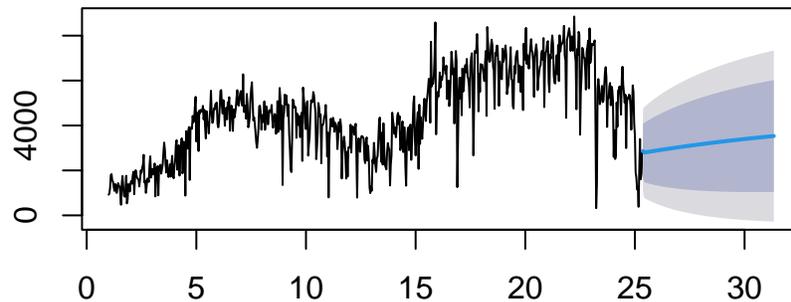
```
fcast1month = forecast(model3, h=30)
plot(fcast1month)
```

Forecasts from ARIMA(1,0,3) with non-zero mean



```
fcast6months = forecast(model3, h=180)
plot(fcast6months)
```

Forecasts from ARIMA(1,0,3) with non-zero mean



The blue line represents the forecasted values, the dark blue shaded area represents the 80% confidence interval for those forecasts, and the lighter grey area represents the 95% confidence interval. Notice how both of those intervals expand as we move away from the last time point in our time-series, this is particularly clear when we expand the forecast region to six months.

This is just a rudimentary example using a ‘toy’ dataset. Yet, it represents the essence of a large part of the research undertaken in fields like macroeconomics, or business analytics, where their research questions often involve trying to anticipate the future values (e.g. prices, quantities, rates) of a given commodity or metric (w.g. airline passengers, unemployment, GDP, etc.). Of course, things get much more complicated than what we have seen today, but in essence time-series analyses tend to follow a similar sequence of steps to that we have practiced here.

Exercise 2. Sentence Severity

We can also use time-series analysis to assess the causal effect of specific interventions. The idea is to learn from the time-series by modelling it up to the point when a given intervention took place, and forecast from that point on. Then, look at the time-series after the intervention took place and assess whether it falls within our forecast region (within the 95% confidence interval) or not. If it moves away from the forecast region then we can take that as evidence that the intervention might have had a causal effect on the outcome variable, by either shifting the time-series up or down, or changing its slope.

We are going to practice this type of analysis by looking at whether the sentencing guidelines have had an effect on sentence severity. The research question to be explored is, in my view, quite important. Just to provide some context, crime has been declining for decades, yet, prisons are becoming ever more overcrowded. This paradox is explained by sentencing becoming more punitive over time. This is a real problem that has led different researchers to investigate the causes that could be behind such sentence inflation. Recently, a number of researchers (Allen, 2016; Padfield, 2016) have pointed at the new sentencing guidelines (imposed in England and Wales from 2011 to 2020), as a factor contributing to the increase in severity. However, the evidence used is mostly qualitative or based on simple descriptive statistics. Here, we will use time-series to look into this question more proficiently.

Before we get into the analysis of the impact of the sentencing guidelines using ARIMA models, it is important to visualise the phenomenon of sentence inflation experienced in England and Wales. To do so, we are going to access two datasets capturing the different types of sentences that have been imposed in this jurisdictions over the last couple of decades. The first dataset captures all offences imposed in the jurisdiction, the second is restricted to indictable offences, which represent the most serious offences.

```
MoJ = read.csv("Tables_Final.csv")
Indict = read.csv("Indictable.csv")
```

We have one observation per year, and variables capturing the total of cases imposed aggregated and broken down by sentence type. In order of severity (from high to low), these can be ranked as: immediate custodial sentences (for which we also have their length), suspended custodial sentences, community orders, fines, and others (mainly conditional and absolute discharges).

```
summary(MoJ)
```

##	Year	Total	Immediate	Suspended
##	Min. :1999	Min. :1177133	Min. : 86275	Min. : 2519
##	1st Qu.:2004	1st Qu.:1242492	1st Qu.: 94086	1st Qu.: 3116
##	Median :2008	Median :1365347	Median :100231	Median :41151
##	Mean :2008	Mean :1353255	Mean : 99581	Mean :31392
##	3rd Qu.:2012	3rd Qu.:1420043	3rd Qu.:106179	3rd Qu.:48459
##	Max. :2017	Max. :1547352	Max. :111607	Max. :57072
##	Community	Fine	Other	Length
##	Min. : 95112	Min. : 799747	Min. : 70733	Min. :11.4
##	1st Qu.:138859	1st Qu.: 891109	1st Qu.:110884	1st Qu.:12.5
##	Median :177603	Median : 930121	Median :138665	Median :13.3
##	Mean :163099	Mean : 932410	Mean :126774	Mean :13.7
##	3rd Qu.:191120	3rd Qu.: 982579	3rd Qu.:145277	3rd Qu.:15.0
##	Max. :204247	Max. :1082690	Max. :155461	Max. :16.9

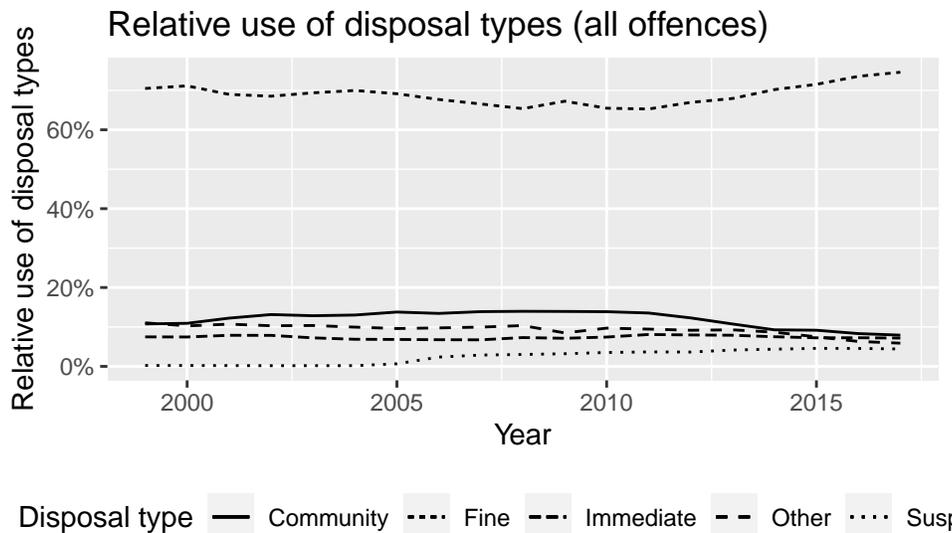
```
summary(Indict)
```

##	Year	Total	Immediate	Suspended
##	Min. :1999	Min. :223699	Min. :71182	Min. : 1963
##	1st Qu.:2004	1st Qu.:296347	1st Qu.:75068	1st Qu.: 2500
##	Median :2008	Median :314941	Median :79788	Median :28455
##	Mean :2008	Mean :306952	Mean :78665	Mean :22152
##	3rd Qu.:2012	3rd Qu.:329529	3rd Qu.:80789	3rd Qu.:34977

```
## Max. :2017 Max. :347422 Max. :87507 Max. :41569
## Community Fine Other Length
## Min. : 45063 Min. :38285 Min. :31402 Min. :14.00
## 1st Qu.: 77162 1st Qu.:49670 1st Qu.:53605 1st Qu.:15.40
## Median :102186 Median :55351 Median :55909 Median :16.10
## Mean : 91362 Mean :59802 Mean :54971 Mean :16.52
## 3rd Qu.:108251 3rd Qu.:70548 3rd Qu.:60827 3rd Qu.:17.50
## Max. :111724 Max. :92121 Max. :69460 Max. :20.00
```

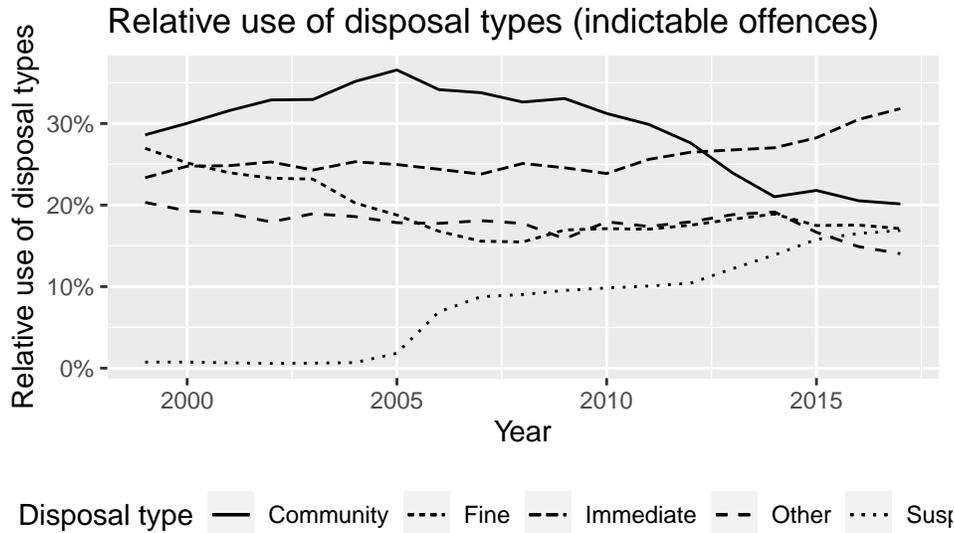
Looking at specific sentence types divided by the total number of sentences imposed that same year we can plot the relative frequency of each sentence type. We can do this first using 'MoJ', the dataset that captures all the offences passes in England and Wales.

```
ggplot(MoJ, aes(x=Year, y=Other/Total)) + geom_line(aes(linetype="Other")) +
  geom_line(aes(x=Year, y=Fine/Total, linetype="Fine")) +
  geom_line(aes(x=Year, y=Community/Total, linetype="Community")) +
  geom_line(aes(x=Year, y=Suspended/Total, linetype="Suspended")) +
  geom_line(aes(x=Year, y=Immediate/Total, linetype="Immediate")) +
  ylab("Relative use of disposal types") +
  ggtitle("Relative use of disposal types (all offences)") +
  labs(linetype="Disposal type") +
  scale_y_continuous(labels = scales::percent) + theme(legend.position="bottom")
```



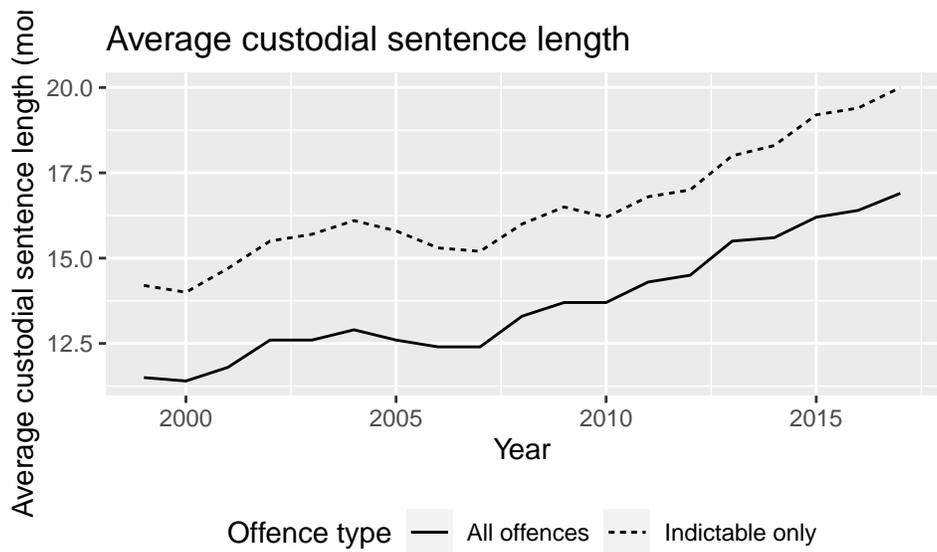
Looking at all offences it is hard to spot a specific pattern. Community orders seem to be on the decline from 2010 while fines are going up. The change in sentence severity is much clearer once we focus on indictable offences.

```
ggplot(Indict, aes(x=Year, y=Other/Total)) + geom_line(aes(linetype="Other")) +
  geom_line(aes(x=Year, y=Fine/Total, linetype="Fine")) +
  geom_line(aes(x=Year, y=Community/Total, linetype="Community")) +
  geom_line(aes(x=Year, y=Suspended/Total, linetype="Suspended")) +
  geom_line(aes(x=Year, y=Immediate/Total, linetype="Immediate")) +
  ylab("Relative use of disposal types") +
  ggtitle("Relative use of disposal types (indictable offences)") +
  labs(linetype="Disposal type") +
  scale_y_continuous(labels = scales::percent) + theme(legend.position="bottom")
```



Suspended sentences (which are very close to prison time in terms of severity) are clearly on the way up while community orders are in decline. Immediate custodial sentences also seem to be on the way up from 2010, but more worryingly is not just the increase in how often those sentences they are employed, but the longer duration imposed, which can be observed in the following plot.

```
ggplot(MoJ, aes(x=Year, y=Length, linetype="All offences")) + geom_line() +
  ylab("Average custodial sentence length (months)") +
  ggtitle("Average custodial sentence length") +
  geom_line(data=Indict, aes(x=Year, y=Length, linetype="Indictable only")) +
  labs(linetype="Offence type") + theme(legend.position="bottom")
```



Putting all of that together, it seems that the problem of increasing sentence severity is definitely real, in particular for the case of indictable offences. It makes sense then to question whether the sentencing guidelines that were introduced over the last decade have contributed to this process. To explore that research question we are going to look at three offence-specific guidelines, the sex, robbery and theft guidelines. We can explore together the analysis of sex offences, then you can try exploring the other two on your own by replicating the following analysis.

The data to be used for each of the offence-specific is available on Minerva. Let's start by opening the dataset capturing sex offences.

```
sex = read.csv("MoJ_sex.csv")
```

The first thing to notice is that the time periods used are now quarters, which generally means four observations per year.

```
head(sex)
```

```
##   quarter avgleng cust susp  com fine absdis condis
## 1 Sep-17   60.60 4206 1104 1318   83    25    79
## 2 Jun-17   59.39 4318 1137 1423   76    23    86
## 3 Mar-17   60.10 4447 1093 1430   80    19    93
## 4 Dec-16   60.02 4423 1127 1486   81    16    88
## 5 Sep-16   60.55 4365 1046 1553   83     9    83
## 6 Jun-16   61.55 4257  977 1601   98     9    84
```

```
sex$quarter
```

```
## [1] "Sep-17" "Jun-17" "Mar-17" "Dec-16" "Sep-16" "Jun-16" "Mar-16" "Dec-15"
## [9] "Sep-15" "Jun-15" "Mar-15" "Dec-14" "Sep-14" "Jun-14" "Mar-14" "Dec-13"
## [17] "Sep-13" "Jun-13" "Mar-13" "Dec-12" "Sep-12" "Jun-12" "Mar-12" "Dec-11"
## [25] "Sep-11" "Jun-11" "Mar-11" "Dec-10" "Sep-10" "Jun-10" "Mar-10" "Dec-09"
## [33] "Sep-09" "Jun-09" "Mar-09" "Dec-08" "Sep-08" "Jun-08" "Mar-08" "Dec-07"
## [41] "Sep-07" "Jun-07" "Mar-07" "Dec-06"
```

We have a similar set of variables covering sentence types to those we have used in the introduction of the exercise. However, the variable capturing the date for each case ('quarter') needs to be reformatted. We need to turn 'quarter' into a variable that R can recognise as a date. To do so, we can use the command *as.Date*, with specific format "%d-%b-%y", which represents: days expressed as two numbers (%d), months expressed as a three letter word (%b), and years expressed as the last two digits of the year (%y). In order to fully match that particular format we are going to add "15-" to include the day part (%d) in our variable 'quarter' using the command *paste*. The number 15 is an arbitrary choice just to be able to match the format required, but it is a harmless choice since we use the same figure across all our observations, so we are not really modifying the time-series. If you want to know more about what kind of date formats can be used in R this is a good reference from Quick-R. In addition, we will also sort the dataset chronologically using *order*, which will help us write more intuitive commands when we get to the forecast stage.

```
sex$date = as.Date(paste("15-", sex$quarter, sep = ""), format = "%d-%b-%y")
sex = sex[order(sex$date),]
```

We can now plot the relative frequency for different sentence types and the average sentence length of immediate custodial sentences, as we did before. However, that would give us multiple time-series to monitor and we would not be able to determine exactly whether a decrease in one sentence type is offset by an increase in another. In sum, we need to find a way to aggregate all these different sentence types into a single scale of sentence severity. In Pina-Sánchez et al. (2019)) we did just that, created a ranking of all the main sentence types by their estimated severity based on a survey conducted with magistrates. Here we are going to borrow that scale of severity to aggregate all the different sentence types.

In the following piece of code we provide values of severity for the counts of each sentence type and divide them by the total number of sentences imposed in that period. We will use 1.32 for fines, 2.12 for community orders, 3.88 for suspended sentences, and a range from 5.05 to 120 for custodial sentences depending on their length.

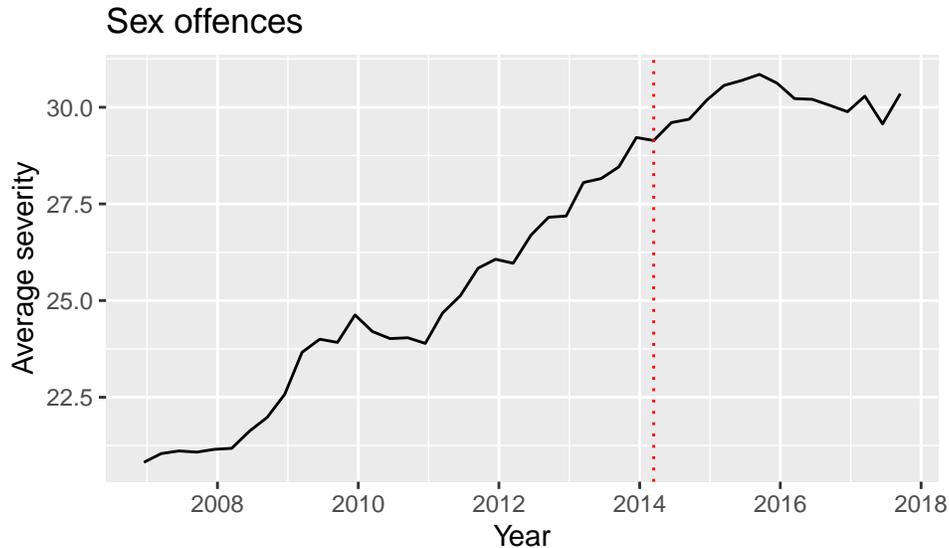
```
sex$severity = (sex$fine*1.32 + sex$com*2.12 + sex$susp*3.88 + sex$cust * (5.05+sex$avgleng*(6.45-5.05)) /
               (sex$cust + sex$susp + sex$com + sex$fine + sex$absdis + sex$condis)
```

Now we can plot sentence severity as a single time-serie. This is done below including a vertical line indicating

the time when the sex guidelines came into force (more information about the guidelines available here).

```
#First plot
```

```
ggplot(sex, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity") +  
  ggtitle("Sex offences") + geom_vline(xintercept=as.numeric(sex$date[30]), linetype=3, colour="red")
```



We can see how sentence severity was higher after than before the sex guidelines came into force. However, we can also see that the rate of increase flattened approximately a year after the guidelines were introduced. To assess the effect of the guidelines more proficiently we can model the time-series behaviour up to the point the guidelines were introduced, and compare forecasts based on that model to what we observed in the two years that followed their introduction.

To do so we are first going to use *ts* to tell R that 'sex\$severity' should be treated as a time-series with a seasonality of frequency 4 (representing the four quarters within a year), and then, rather than follow the whole model building process that we carried out in the previous exercise, we are going to rely on an unsupervised modelling approach using *auto.arima*. This command basically follows a stepwise process (like the one we saw in Workshop 2) to decide the number of integration (I), auto-regressive (AR), and moving average (MA) components to be used in the model. It does that by estimating different model configurations and comparing them using AIC. If you want to learn more about this process see here.

```
sexsev = ts(sex$severity, frequency = 4)  
fitpre = auto.arima(sexsev[-c(30:44)])  
#[-c(30:44)] in the above line is used to indicate that we only want to model the time-series up to the  
fitpre
```

```
## Series: sexsev[-c(30:44)]  
## ARIMA(0,1,0) with drift  
##  
## Coefficients:  
##      drift  
##      0.3001  
## s.e.  0.0699  
##  
## sigma^2 = 0.1419: log likelihood = -11.88  
## AIC=27.77  AICc=28.25  BIC=30.43
```

The best model selected by *auto.arima* for the sentence severity of sex offences up to the time the guidelines came into force is a simple ARIMA(0,1,0), so no AR or MA components, just one difference to make the

time-series stationary (you can see clearly in our last plot the strong trend that makes the time-series non-stationary). In addition, *auto.arima* has estimated a *drift* coefficient. This in essence represents the possibility of modifying the constant of the model. It is a modelling choice that we have not covered in the workshop that is nonetheless available in *auto.arima*. If you want to learn more about drifts in time series see [here](#).

Ok, now that we have our model we can carry out the forecast for the post guidelines period. We can decide for how long we want to extend the forecast using the option *h*. Here we use 'h=15', which covers the whole period since the guidelines came into force up to the end of our window of observation. We have chosen that rather long period to facilitate the integration of our forecast into a dataframe that can be used to plot the forecast together with our observed time-series. However, to assess the potential causal impact that could be attributed to the guidelines we will focus on the two-year period after the guidelines came into force. The choice of such interval of time can be deemed arbitrary since it is nothing more than an educated guess, but we need to set it at some point, and two years makes sense to us.

```
fcast_fitpre = forecast(fitpre, h=15)
fcast_fitpre
```

##	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 30	29.51813	29.03533	30.00092	28.77976	30.25649
## 31	29.81818	29.13541	30.50095	28.77397	30.86239
## 32	30.11824	29.28202	30.95446	28.83935	31.39712
## 33	30.41829	29.45271	31.38388	28.94156	31.89502
## 34	30.71835	29.63879	31.79790	29.06731	32.36938
## 35	31.01840	29.83581	32.20100	29.20978	32.82702
## 36	31.31846	30.04111	32.59580	29.36492	33.27199
## 37	31.61851	30.25297	32.98405	29.53010	33.70693
## 38	31.91857	30.47019	33.36694	29.70347	34.13367
## 39	32.21862	30.69190	33.74534	29.88370	34.55354
## 40	32.51868	30.91744	34.11992	30.06979	34.96756
## 41	32.81873	31.14629	34.49117	30.26095	35.37651
## 42	33.11879	31.37806	34.85952	30.45657	35.78101
## 43	33.41884	31.61240	35.22528	30.65613	36.18156
## 44	33.71890	31.84905	35.58874	30.85922	36.57858

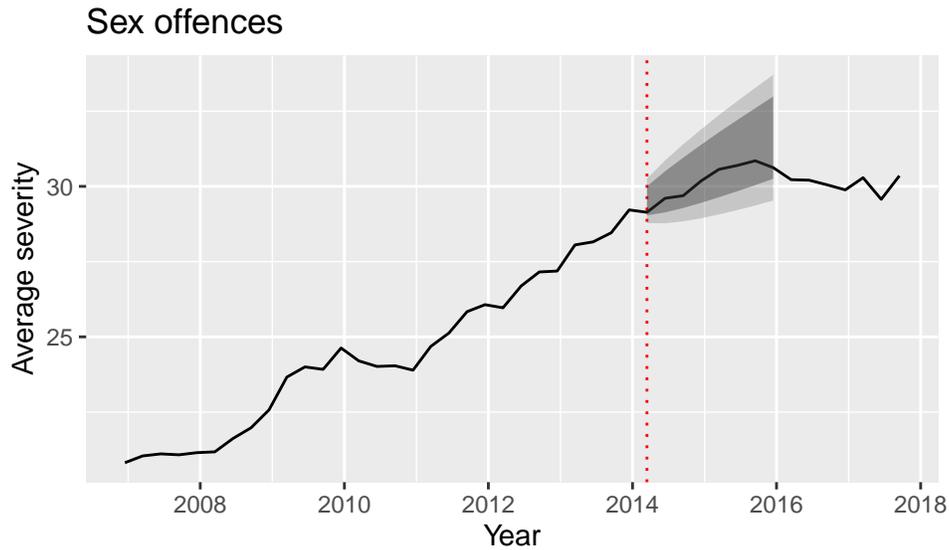
To plot these forecasted values and their 80% and 95% confidence intervals, on top of the ggplot that we produced before, we can use the following code, which will create a specific dataframe for each of the lower and upper bounds of the 80% and 95% intervals of the forecast separately, and set their values for any period different from the two years following the guidelines coming into force as NAs.

```
x = rep(NA,29)
slo95 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$lower[,2]))
slo80 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$lower[,1]))
sup95 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,2]))
sup80 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,1]))
slo95[38:44,] = NA
slo80[38:44,] = NA
sup95[38:44,] = NA
sup80[38:44,] = NA
```

Below is the code to plot the observed time-series with our forecast region (the space confined within the confidence interval of the forecasted values). Notice that this is identical to the code we used for the previous plot plus an added line indicating the variables to be used to define the confidence intervals.

```
ggplot(sex, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity") +
  ggtitle("Sex offences") + geom_vline(aes(xintercept=as.numeric(sex$date[30])), linetype=3, colour=
  geom_ribbon(aes(ymax=sup95$x, ymin=slo95$x), alpha=0.2) + geom_ribbon(aes(ymax=sup80$x, ymin=slo80
```

alpha=0.4)



So, we can see that the level of sentence severity used for sex offences following the introduction of the guidelines falls right within the forecast region. Therefore, we cannot really claim that, in this case, the guidelines are to be blamed for the observed increase in severity. In fact, it seems that the trend of increased severity started much earlier, and if something, the guidelines seem to have stabilised that trend.

For the rest of the exercise you are requested to explore the effect of the robbery (which came into force 1st of April of 2016, so you can consider the first quarter of 2016) and theft guidelines (into force since the 1st of February of 2016, also the first quarter of the year). You can do so by replicating the analysis that we have just conducted for the sex guidelines.

- Open the data for one of those offence types.
- Make sure that you set the date variable as a date that R understands (*as.Date*), for which you might need to provide an arbitrary value for days, if you plan to replicate the code we used above.
- Sort the dataset in chronological order.
- Turn the relative frequency of each sentence type into values of severity using the suggested scale of severity.
- Create the first plot for the time-serie, including a vertical line indicating the robbery guidelines coming into force.
- Formatting severity as a time-serie (*ts*)
- Estimate an ARIMA model (*auto.arima*) considering only the period covered before the guideline came into force.
- Forecast the level of severity for the X number of quarters following the time when the guideline came into force.
- Prepare the datasets to be used to include the lower and upper confidence interval bounds.
- Draw the plot and assess whether the severity level stood within or out of the forecast region.

Remove from the handout#####

Looking at robbery first.

Opening the data.

```
rob = read.csv("MoJ_robbery.csv")
```

Setting the date.

```
rob$date = as.Date(paste("15-", rob$quarter, sep=""), format="%d-%b-%y")
```

Sorting the dataset in increasing chronological order to facilitate making sense of the forecasts, i.e. the further down the value in the dataset the further away in time.

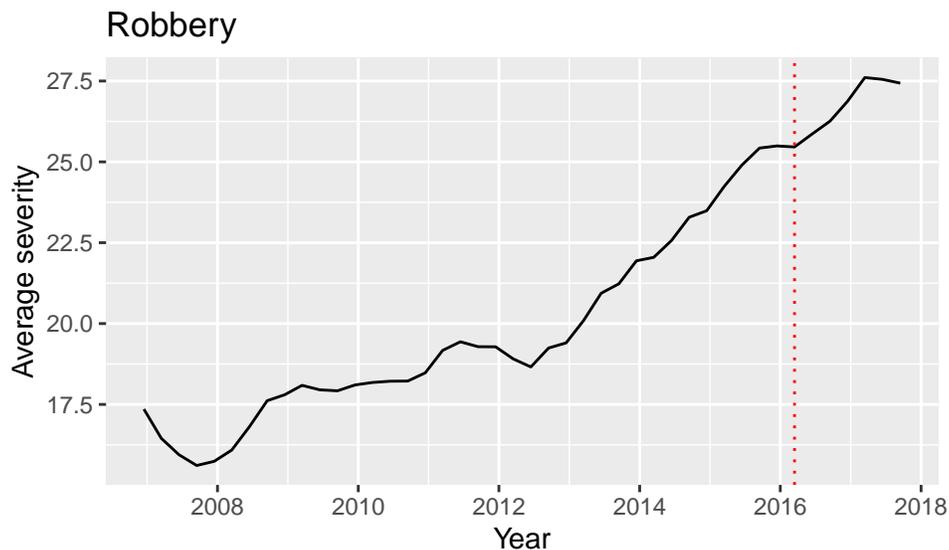
```
rob = rob[order(rob$date),]
```

We now turn the relative frequency for different sentence types into values of severity.

```
rob$severity = (rob$fine*1.32 + rob$com*2.12 + rob$susp*3.88 + rob$cust * (5.05+rob$avgleng*(6.45-5.05))
               (rob$cust + rob$susp + rob$com + rob$fine + rob$absdis + rob$condis)
```

Create the first plot including a vertical line indicating the robbery guidelines coming into force.

```
ggplot(rob, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity") +
  ggtitle("Robbery") + geom_vline(aes(xintercept=as.numeric(rob$date[38])), linetype=3, colour="red")
```



Formatting severity as a time-series.

```
robsev = ts(rob$severity, frequency = 4)
```

Estimating an ARIMA model for the before period.

```
fitpre = auto.arima(robsev[-c(38:44)])
fitpre
```

We will be using an ARIMA(1,2,0).

Undertaking the forecast for the 7 quarters following the robbery guidelines coming into force.

```
fcst_fitpre = forecast(fitpre, h=7)
```

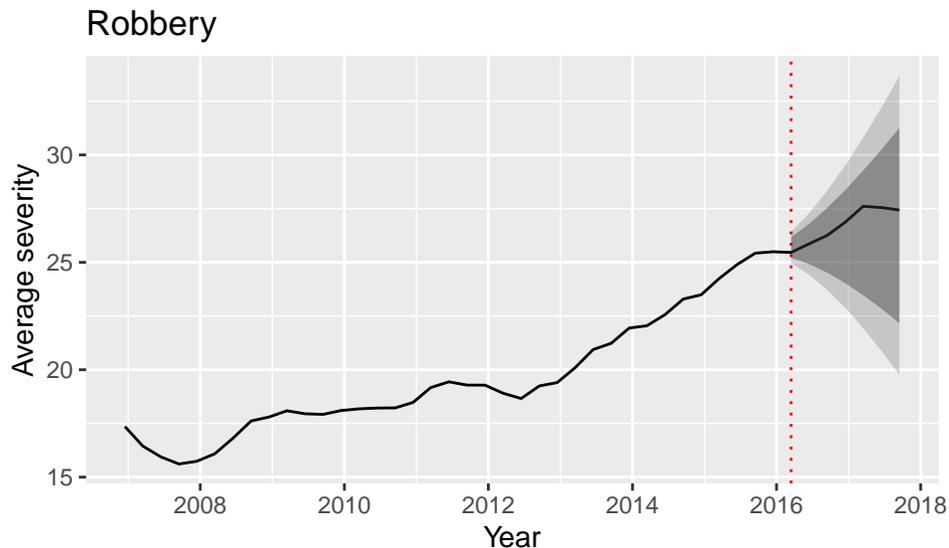
Preparing the data for the next plot including the forecast region.

```
x = rep(NA,37)
rlo95 = rbind(as.data.frame(x),as.data.frame(fcst_fitpre$lower[,2]))
rlo80 = rbind(as.data.frame(x),as.data.frame(fcst_fitpre$lower[,1]))
```

```
rup95 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,2]))
rup80 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,1]))
```

The final plot including the forecast region.

```
ggplot(rob, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity") +
  ggtitle("Robbery") + geom_vline(aes(xintercept=as.numeric(rob$date[38])), linetype=3, colour="red") +
  geom_ribbon(aes(ymax=rup95$x, ymin=rlo95$x), alpha=0.2) + geom_ribbon(aes(ymax=rup80$x, ymin=rlo80$x),
  alpha=0.4)
```



As for the case of sex offences, the sentence severity applied to robbery offences is higher after the guidelines came into force. However, the trend started before the guidelines were introduced, and their behaviour in the post guidelines period falls within the expected forecast region. Hence, again, we cannot conclude that the guidelines have contributed to the increase in sentence severity.

Let's look now into the theft guidelines.

Opening the data.

```
theft = read.csv("MoJ_theft.csv")
```

Setting the date.

```
theft$date = as.Date(paste("15-", theft$quarter, sep=""), format="%d-%b-%y")
```

Sorting the dataset in increasing chronological order

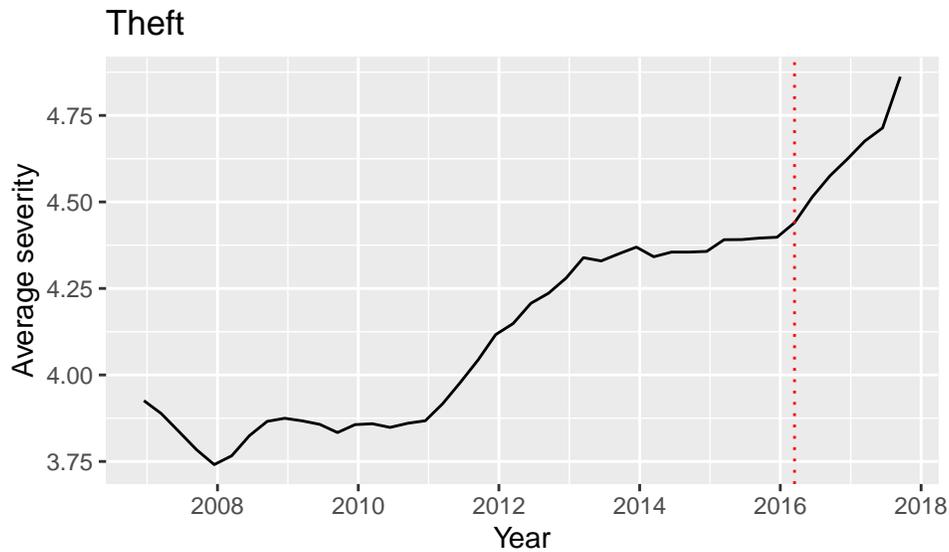
```
theft = theft[order(theft$date),]
```

Turning sentences into values of severity

```
theft$severity = (theft$fine*1.32 + theft$com*2.12 + theft$susp*3.88 + theft$cust*(5.05+theft$avgleng*(
```

Plot including the vertical line at the time the theft guidelines came into force.

```
ggplot(theft, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity") +
  ggtitle("Theft") + geom_vline(aes(xintercept=as.numeric(theft$date[38])), linetype=3, colour="red")
```



Formatting severity as a time-serie.

```
theftsev = ts(theft$severity, frequency = 4)
```

The model.

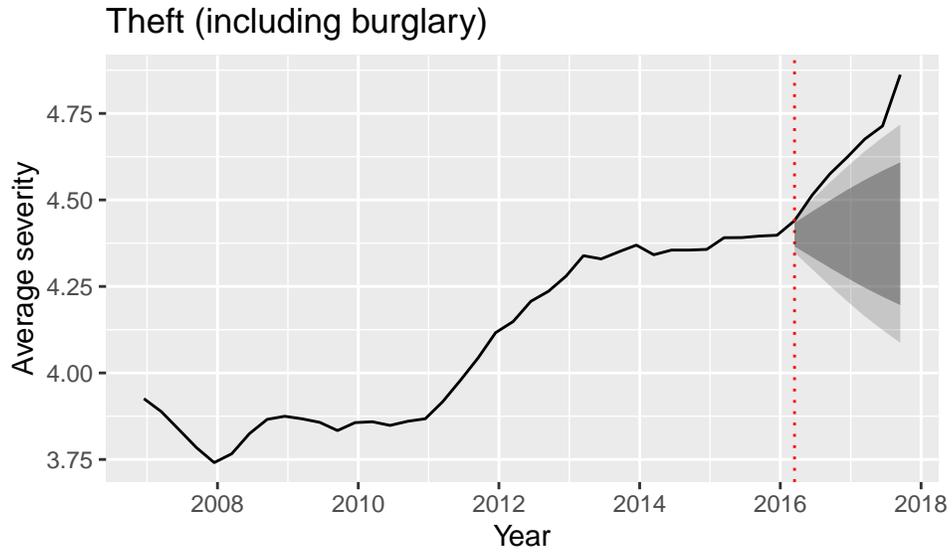
```
fitpre = auto.arima(theftsev[-c(38:44)])
fitpre
```

The forecast.

```
fcast_fitpre = forecast(fitpre, h=7)
x = rep(NA,37)
tlo95 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$lower[,2]))
tlo80 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$lower[,1]))
tup95 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,2]))
tup80 = rbind(as.data.frame(x),as.data.frame(fcast_fitpre$upper[,1]))
```

The second plot including the forecast region.

```
ggplot(theft, aes(x=date, y=severity)) + geom_line() + scale_x_date('Year') + ylab("Average severity")
```



Here we have a very different situation. The increase in severity following the introduction of the theft guidelines is much sharper than what could have been anticipated, which points at a likely causal effect of the guidelines. The reassuring part of this analysis is that the Sentencing Council is already working on modifying those guidelines that have been identified to have had an effect on sentence severity.