

Introduction to Time Series Analysis

Kritchavat Ploddi

Section 1

Introduction to Time Series Analysis

Learning objectives

After completing this tutorials, students should be able to:

- Understand basic concepts of time series analysis
- Apply statistical models to forecast health outcomes with R
- Evaluate predictive performance of time series models

What is a time series?

What is a time series?

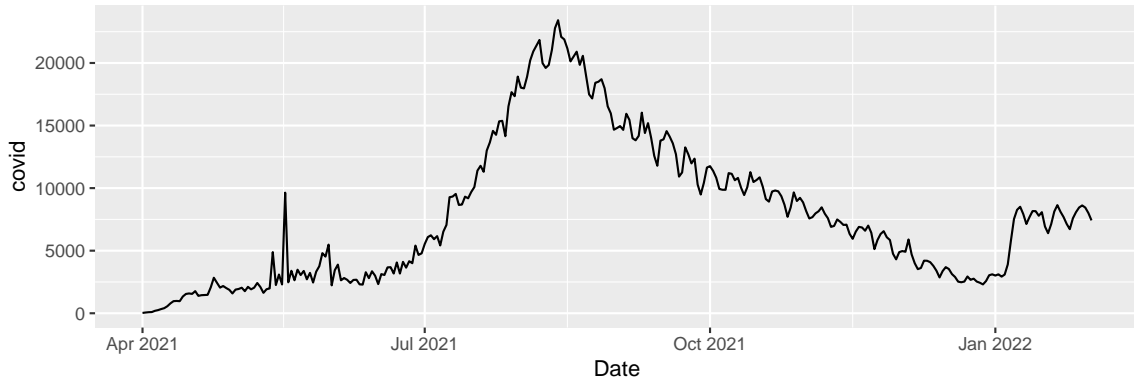
- A time series is a series of data points indexed (or listed or graphed) in time order.
- Most commonly, a time series is a sequence taken at successive equally spaced points in time.

What is a time series? (Examples)

##		date	covid
## 1		2022-01-01	3011
## 2		2022-01-02	3112
## 3		2022-01-03	2927
## 4		2022-01-04	3091
## 5		2022-01-05	3899
## 6		2022-01-06	5775
## 7		2022-01-07	7526
## 8		2022-01-08	8263
## 9		2022-01-09	8511
## 10		2022-01-10	7926
## 11		2022-01-11	7133
## 12		2022-01-12	7681
## 13		2022-01-13	8167
## 14		2022-01-14	8158
## 15		2022-01-15	7793

What is a time series? (Examples)

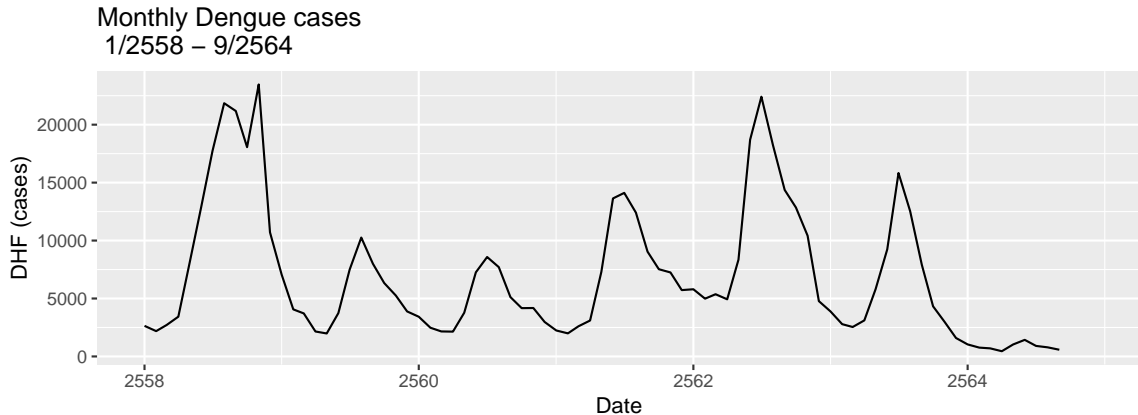
Daily confirmed covid cases (1 April 2021 – 31 January 2022)



What is a time series? (Examples)

##	Year	Month	DHF
## 1	2558	1	2639
## 2	2558	2	2183
## 3	2558	3	2716
## 4	2558	4	3431
## 5	2558	5	8065
## 6	2558	6	12913
## 7	2558	7	17735
## 8	2558	8	21852
## 9	2558	9	21181
## 10	2558	10	18058
## 11	2558	11	23472
## 12	2558	12	10707
## 13	2559	1	7068
## 14	2559	2	4060
## 15	2559	3	3712

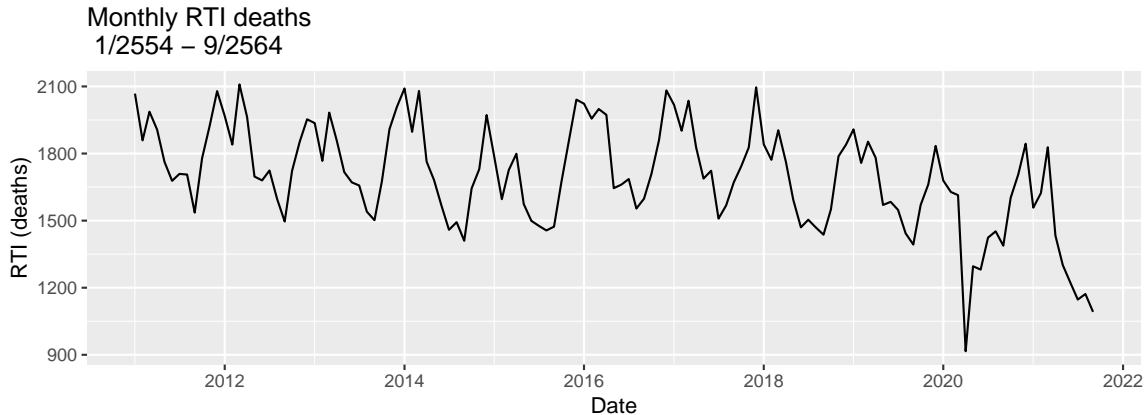
What is a time series? (Examples)



What is a time series? (Examples)

##		date	rti
## 1		2011-01-01	2068
## 2		2011-02-01	1859
## 3		2011-03-01	1987
## 4		2011-04-01	1907
## 5		2011-05-01	1763
## 6		2011-06-01	1678
## 7		2011-07-01	1709
## 8		2011-08-01	1706
## 9		2011-09-01	1536
## 10		2011-10-01	1779
## 11		2011-11-01	1925
## 12		2011-12-01	2079
## 13		2012-01-01	1968
## 14		2012-02-01	1840
## 15		2012-03-01	2109

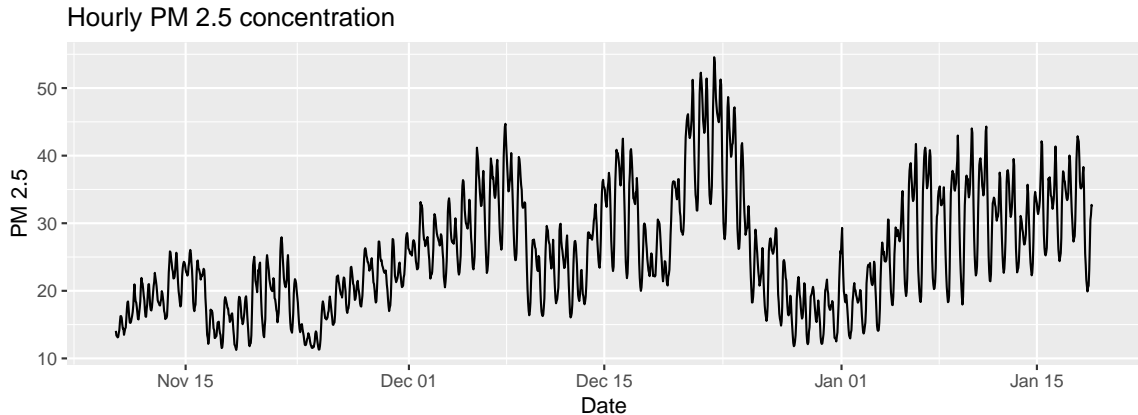
What is a time series? (Examples)



What is a time series? (Examples)

##	DATE	TIME	DATA	PM25
## 1	2021-11-10	00:00:00		14.07865
## 2	2021-11-10	01:00:00		13.37079
## 3	2021-11-10	02:00:00		13.28090
## 4	2021-11-10	03:00:00		13.11236
## 5	2021-11-10	04:00:00		13.16854
## 6	2021-11-10	05:00:00		13.73034
## 7	2021-11-10	06:00:00		14.19101
## 8	2021-11-10	07:00:00		15.35227
## 9	2021-11-10	08:00:00		16.25000
## 10	2021-11-10	09:00:00		16.26136
## 11	2021-11-10	10:00:00		15.98864
## 12	2021-11-10	11:00:00		15.09091
## 13	2021-11-10	12:00:00		14.41573
## 14	2021-11-10	13:00:00		14.39326
## 15	2021-11-10	14:00:00		13.48864

What is a time series? (Examples)



Other time series data?

Other time series data?

- **Medicine:** EKG, Vital signs, Plasma glucose

Other time series data?

- **Medicine:** EKG, Vital signs, Plasma glucose
- **Business and finance:** Sales, Stock prices

Other time series data?

- **Medicine:** EKG, Vital signs, Plasma glucose
- **Business and finance:** Sales, Stock prices
- **Meteorology:** Temperature, Rainfall levels

Common characteristics of time series data

Common characteristics of time series data

- **Trend** pattern exists when there is a long-term increase or decrease in the data

Common characteristics of time series data

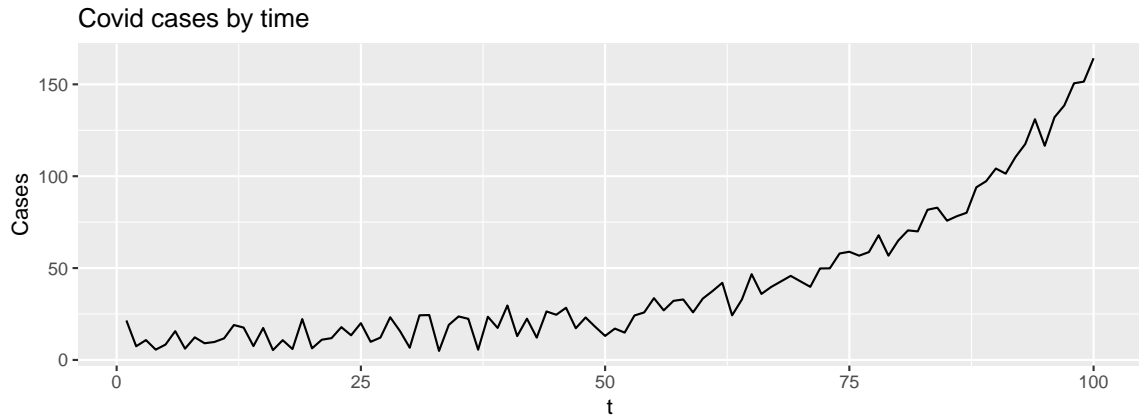
- **Trend** pattern exists when there is a long-term increase or decrease in the data
- **Seasonal** pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

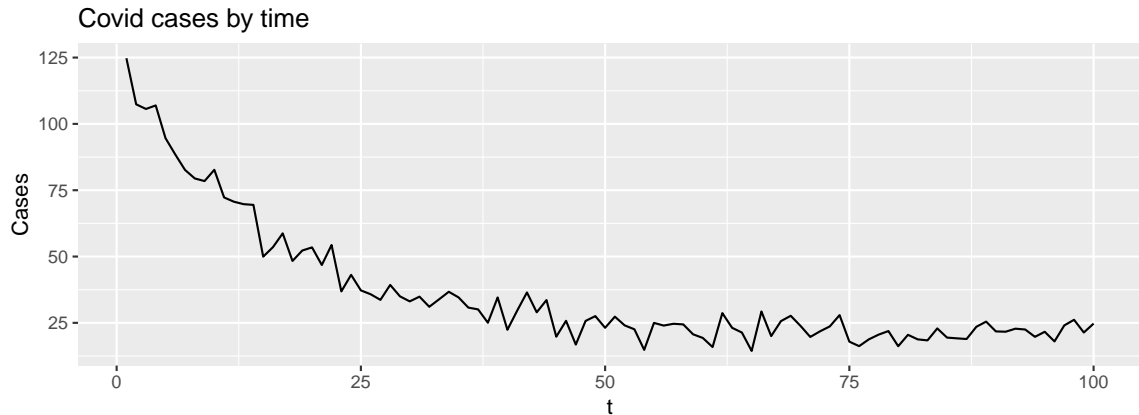
Common characteristics of time series data

- **Trend** pattern exists when there is a long-term increase or decrease in the data
- **Seasonal** pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).
- **Cyclic** pattern exists when data exhibit rises and falls that are not of fixed period (duration usually of at least 2 years).

Comparing seasonality and cycle

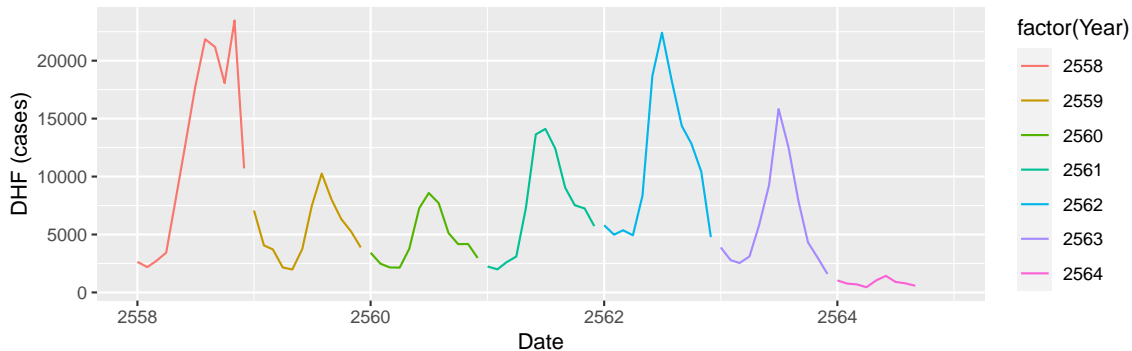
Patterns	Variability	Length	Magnitude
Seasonality	Constant	Shorter	Lower
Cycle	Variable	Longer	Higher





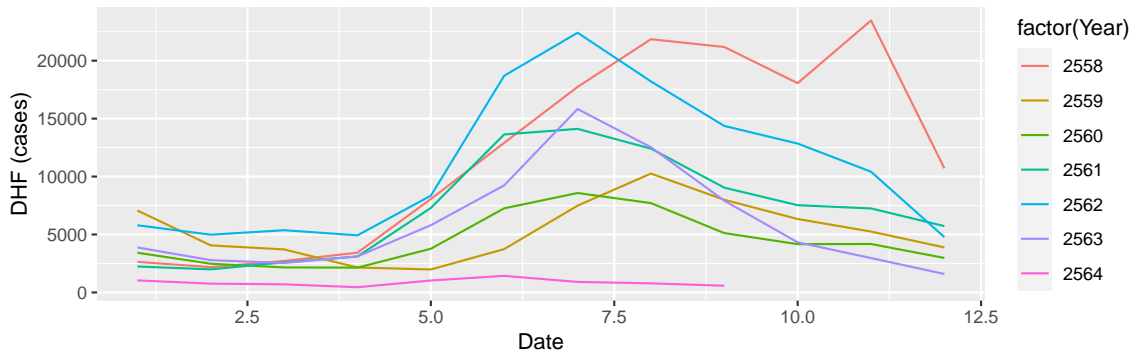
Seasonality

Monthly Dengue cases
1/2558 – 9/2564

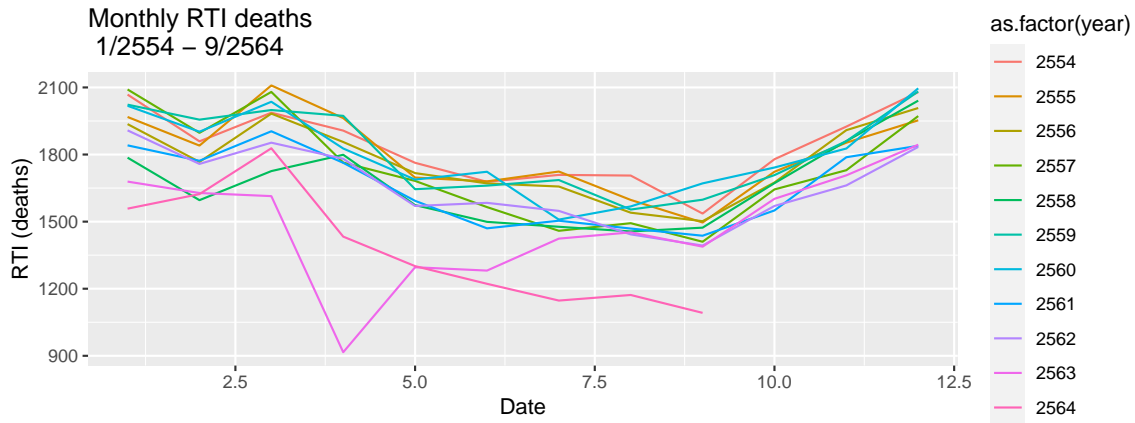


Seasonality

Monthly Dengue cases
1/2558 – 9/2564

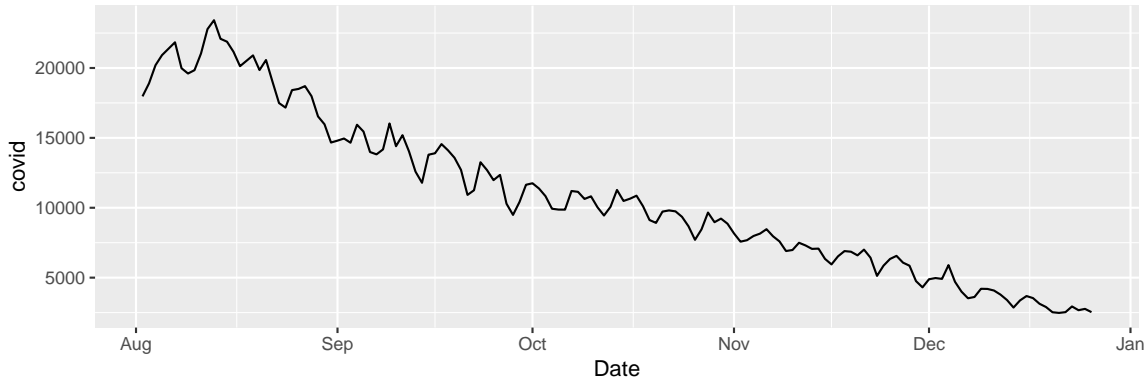


Seasonality



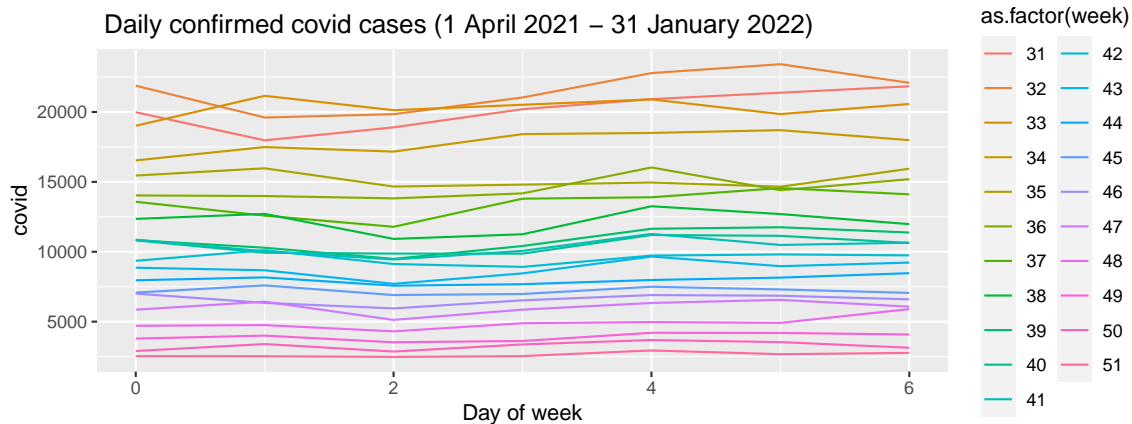
Seasonality

Daily confirmed covid cases (Aug – Dec)



Seasonality

Daily confirmed covid cases (1 April 2021 – 31 January 2022)



Section 2

Time Series Visualization

Plotting time series data with R ggplot2

Before working with RStudio, please set working directory to the workshop folder.

Please install ggplot2 and tidyquant with the following commands:

```
install.packages("ggplot2")
```

```
install.packages("tidyquant")
```

Plotting time series data with R ggplot2

You can load csv data into R workspace with the command `read.csv()`.

```
covid <- read.csv('data/covidts.csv')  
head(covid)
```

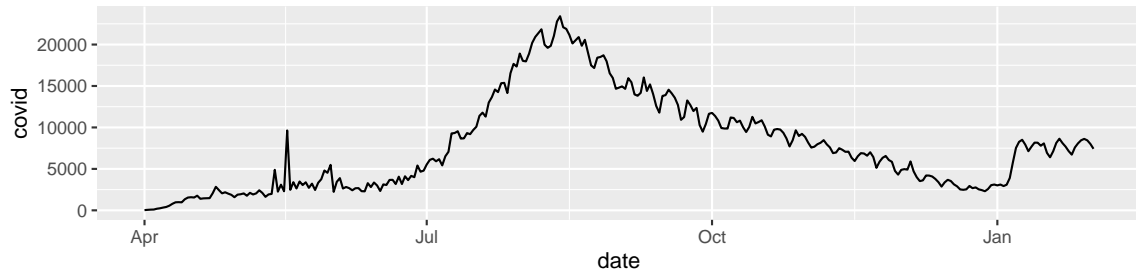
```
##           date day month year dow week covid  
## 1 2021-04-01   1     4 2021   4   13    26  
## 2 2021-04-02   2     4 2021   5   13    58  
## 3 2021-04-03   3     4 2021   6   13    84  
## 4 2021-04-04   4     4 2021   0   13    96  
## 5 2021-04-05   5     4 2021   1   14   194  
## 6 2021-04-06   6     4 2021   2   14   250
```


Plotting time series data with R ggplot2

Once loaded, we create a time series plot having date as X and covid as Y.

With ggplot, we use `geom_line` to create a line plot.

```
# convert date as date variable  
covid$date = as.Date(covid$date)  
ggplot(covid) +  
  geom_line(aes(x=date, y=covid))
```



Plotting time series data with R ggplot2

Now let's create a time series plot of RTI death.

Plotting time series data with R ggplot2

Also, load the data with `read.csv()` and convert date to date variable.

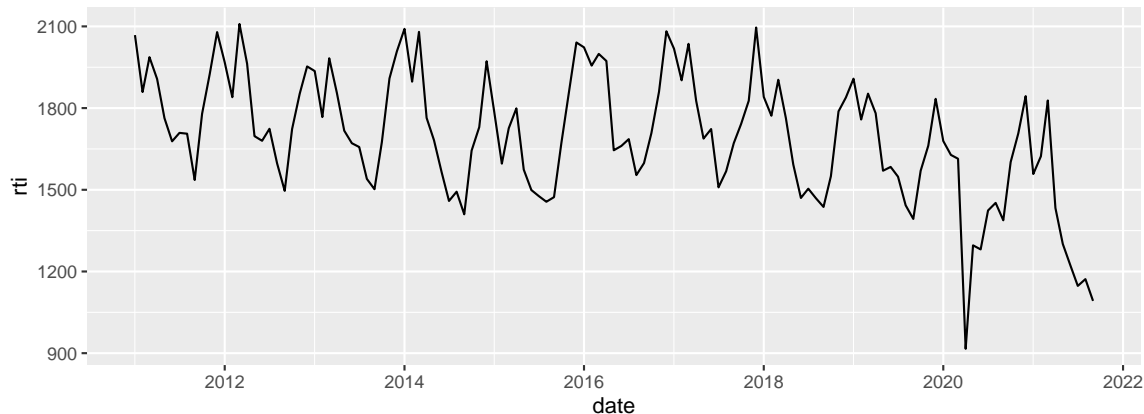
```
rti <- read.csv("data/rti.csv")
rti$date <- as.Date(rti$date)
head(rti)
```

```
##           date  rti year month
## 1 2011-01-01 2068 2554      1
## 2 2011-02-01 1859 2554      2
## 3 2011-03-01 1987 2554      3
## 4 2011-04-01 1907 2554      4
## 5 2011-05-01 1763 2554      5
## 6 2011-06-01 1678 2554      6
```

Plotting time series data with R ggplot2

Plotting the time series plot is also straightforward.

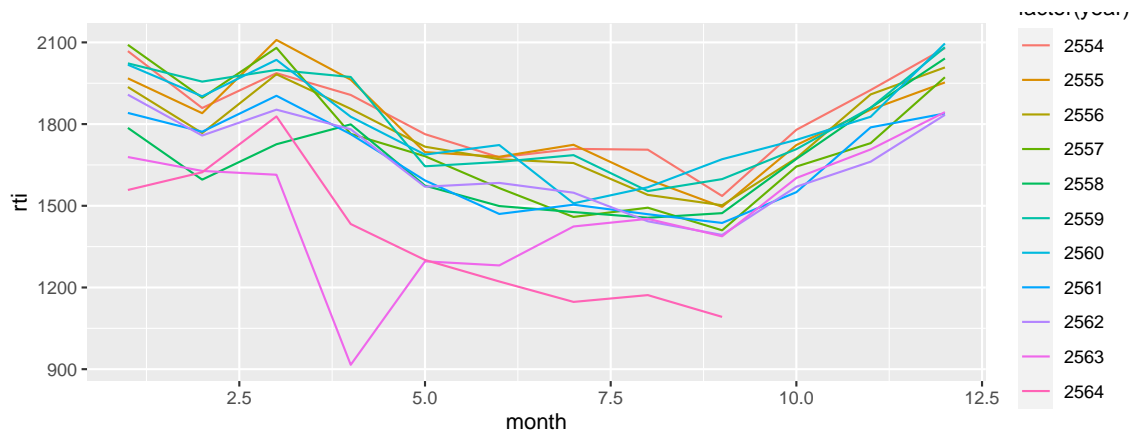
```
ggplot(rti) +  
  geom_line(aes(x=date, y=rti))
```



Plotting time series data with R ggplot2

Next, plotting monthly RTI each year is recommended to examine seasonality.

```
ggplot(rti) +  
  geom_line(aes(x=month, y=rti, color=factor(year)))
```

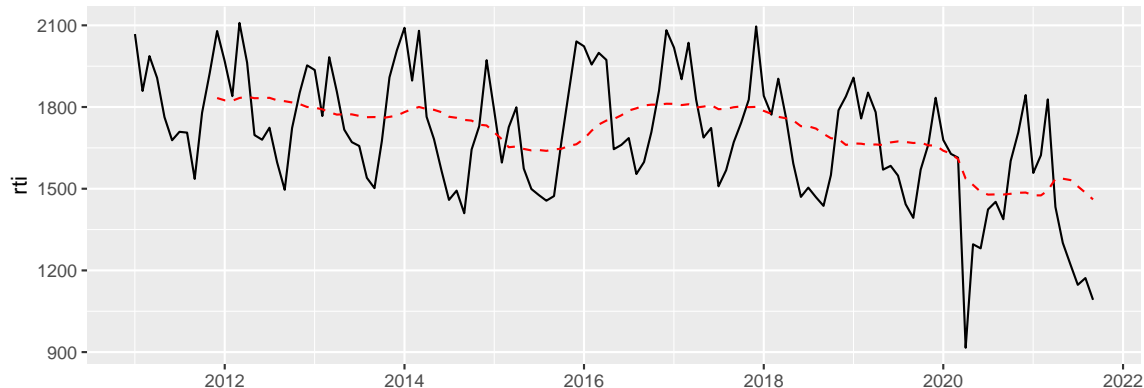


Plotting time series data with R ggplot2

To examine trend pattern, a moving average is recommended.

In this exercise, we use `geom_ma` of `tidyquant` package to create the moving average (of 12 months) plot.

```
ggplot(rti) +  
  geom_line(aes(x=date, y=rti)) +  
  tidyquant::geom_ma(aes(x=date, y=rti), ma_fun=SMA, n=12, color='red', type='l')
```



STL Decomposition

Another way of visualizing time series components is by decomposing them.

Additive: $Y_t = Trend_t + Seasonal_t + Cyclic_t + Others_t$

Multiplicative: $Y_t = Trend_t \times Seasonal_t \times Cyclic_t \times Others_t$

There are methods we used to decompose the time series, e.g. X11, STL.

STL Decomposition

Now we'll use the method called STL decomposition to decompose the RTI death.

Firstly, we have to create a time series object from existing data.

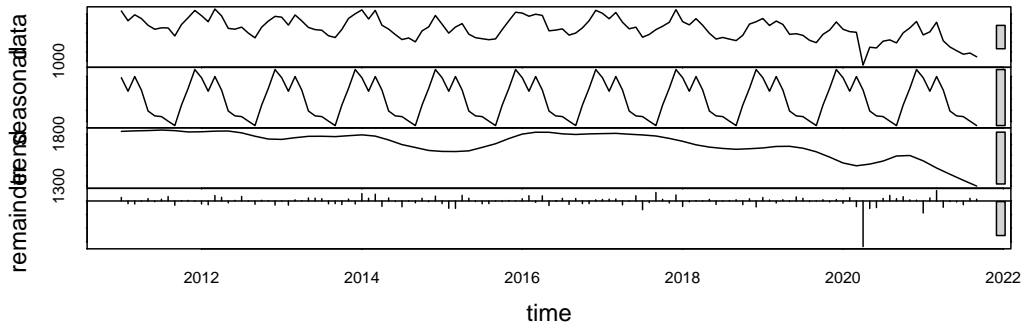
```
ts_rti <- ts(rti$rti, start=c(2011,1), freq=12) # create a time series object  
ts_rti
```

##		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
##	2011	2068	1859	1987	1907	1763	1678	1709	1706	1536	1779	1925	2079
##	2012	1968	1840	2109	1963	1697	1680	1724	1597	1496	1723	1853	1953
##	2013	1936	1767	1983	1856	1717	1671	1657	1540	1502	1675	1909	2008
##	2014	2091	1897	2080	1764	1682	1565	1459	1493	1410	1644	1730	1972
##	2015	1786	1596	1726	1799	1574	1499	1477	1456	1473	1672	1860	2041
##	2016	2023	1956	1999	1973	1645	1661	1686	1554	1598	1708	1860	2082
##	2017	2018	1902	2036	1827	1688	1723	1509	1568	1671	1742	1827	2096
##	2018	1841	1772	1904	1763	1593	1470	1504	1469	1437	1550	1788	1839
##	2019	1908	1758	1853	1781	1570	1584	1548	1443	1393	1570	1662	1834

STL Decomposition

We use the command `stl` to decompose the time series RTI data.

```
decomposition <- stl(ts_rti, t.window=12, s.window="periodic", robust=TRUE) #  
plot(decomposition)
```



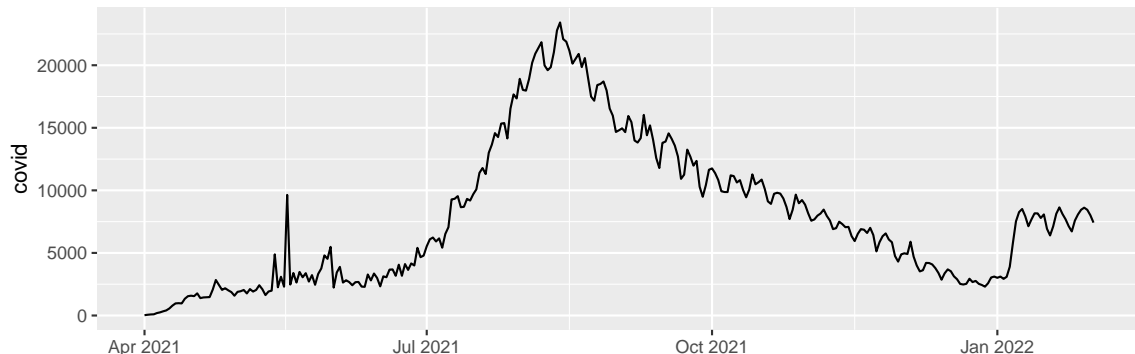
Time series correlation

Most time series data are correlated with themselves.

In statistics, this is called autocorrelation.

Cases tend to be correlated (similar) with the others, especially at the nearest time point.

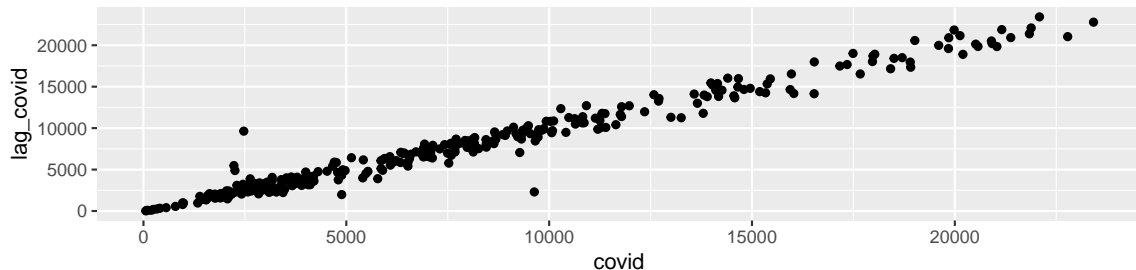
Daily confirmed covid cases (1 April 2021 – 31 January 2022)



Time series correlation

We could use a scatter plot to visually examine the correlation of the time series with its lag (yesterday cases).

```
covid$lag_covid <- lag(covid$covid, 1)
ggplot(covid) +
  geom_point(aes(x=covid, y=lag_covid))
```



Time series correlation

We could also use the Pearson's correlation coefficient.

```
with(covid, cor(covid, lag_covid, use='complete.obs'))
```

```
## [1] 0.985227
```

The value 0.98 suggested a strong correlation between today Covid cases and yesterday cases.

Autocorrelation function (ACF)

The autocorrelation function (ACF) is essentially a function to determine the Pearson's correlation of a time series data and its previous lag at any time points.

In R, the ACF is called by the `acf` command.

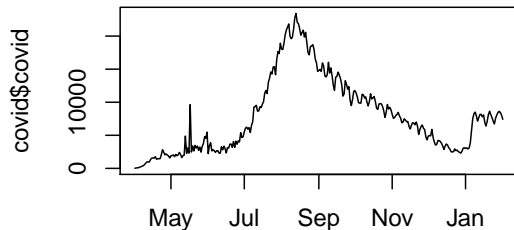
Autocorrelation function (ACF)

```
# format subplots having 1 row and 2 columns
```

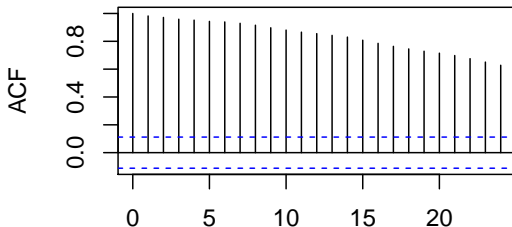
```
par(mfrow=c(1,2))
```

```
plot(covid$date, covid$covid, type='l')
```

```
acf(covid$covid)
```



Series covid\$covid



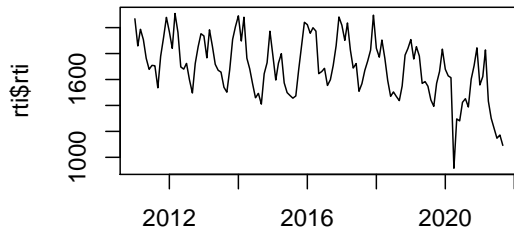
Autocorrelation function (ACF)

format subplots having 1 row and 2 columns

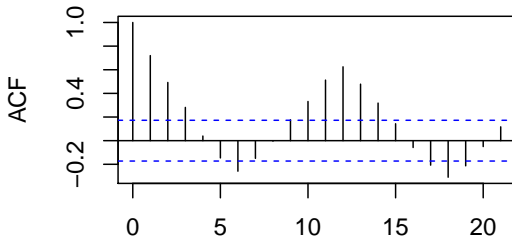
```
par(mfrow=c(1,2))
```

```
plot(rti$date, rti$rti, type='l')
```

```
acf(rti$rti)
```



Series `rti$rti`



Autocorrelation function (ACF)

After examining the ACF, it is obvious that most time series data are correlated with its lag.

However, the ACF cannot exactly determined the correlation with it, say 7th lag, given all other lags.

This is because the correlation with the 7th lag may be confounded with other lags.

Partial Autocorrelation Function (PACF)

The Partial Autocorrelation Function (PACF) also shows the Pearson Correlation Coefficient of the time series and its lags.

It is different from the ACF in that the function is given by all other lag (meaning that there would be no confounding effects from other lags).

In R, we use the command, `pacf`.

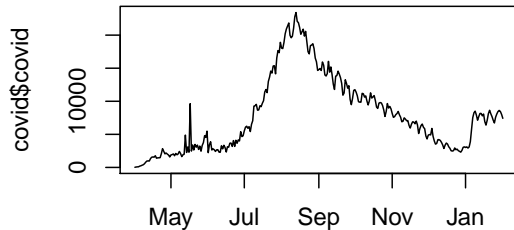
Partial Autocorrelation function (PACF)

```
# format subplots having 1 row and 2 columns
```

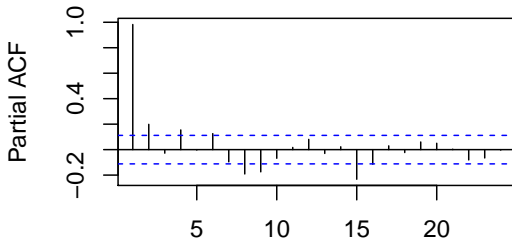
```
par(mfrow=c(1,2))
```

```
plot(covid$date, covid$covid, type='l')
```

```
pacf(covid$covid)
```



Series covid\$covid



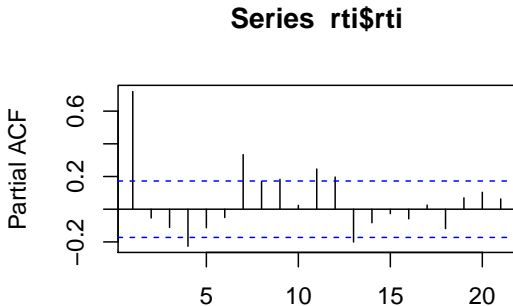
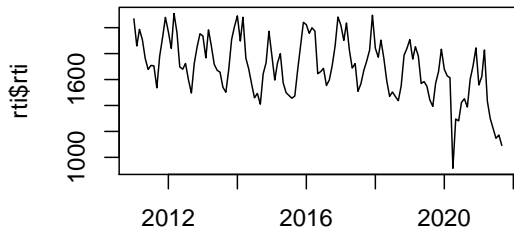
Partial Autocorrelation function (PACF)

```
# format subplots having 1 row and 2 columns
```

```
par(mfrow=c(1,2))
```

```
plot(rti$date, rti$rti, type='l')
```

```
pacf(rti$rti)
```



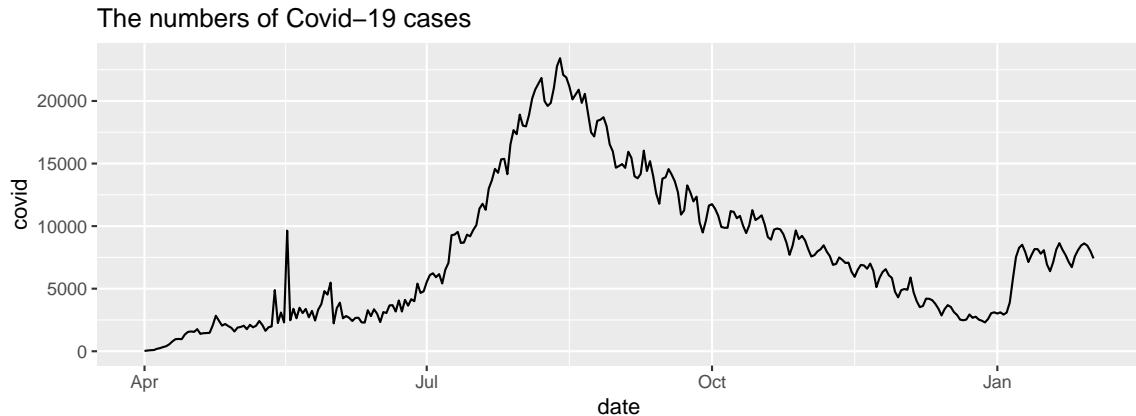
Another important concept in time series analysis is stationary.

This is because most time series forecasting models require stationary assumptions.

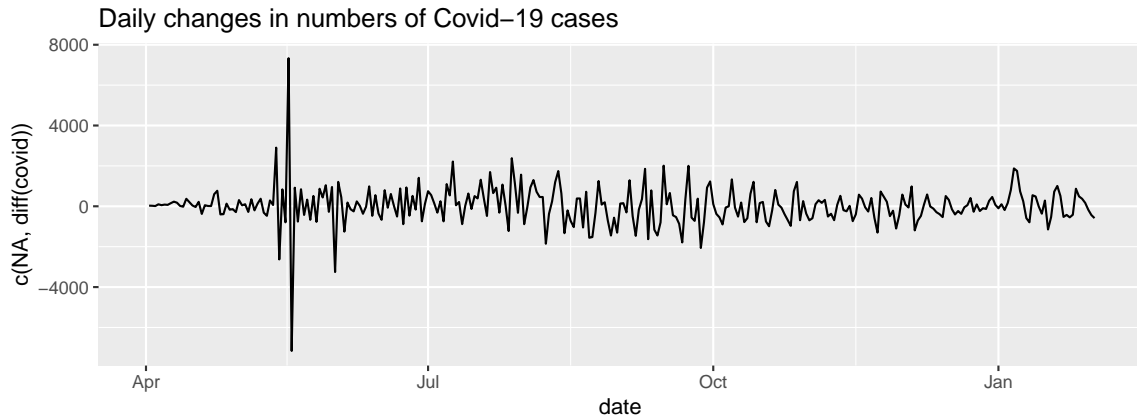
Stationary

A time series Y is said to be stationary if all of its values y_t do not depend on time t .
In other words, the distribution of y_t has constant mean and variance.

Stationary ?



Stationary ?



How to transform non-stationary time series to be stationary

Stationary is characterized by constant mean and variance.

Transformations help to stabilize the variance.

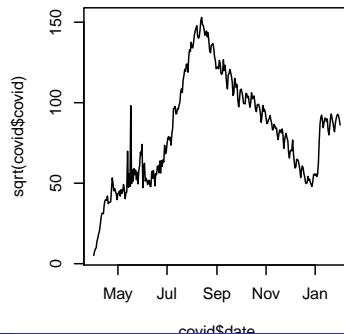
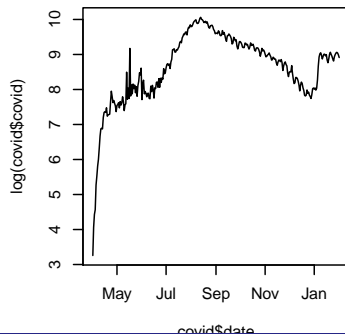
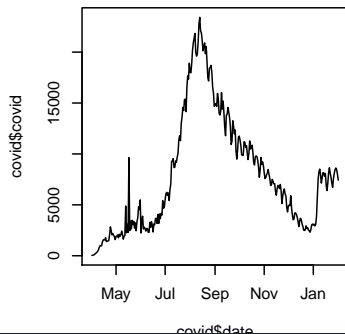
Difference help to stabilize the mean

Variance could be stabilized by taking:

- (Natural) Logarithm (\log)
- Squared root ($\sqrt{}$)

Transformations

```
par(mfrow=c(1,3))  
plot(covid$date, covid$covid, type='l')  
plot(covid$date, log(covid$covid), type='l')  
plot(covid$date, sqrt(covid$covid), type='l')
```



Transformations

We found that taking natural logarithm on Covid cases mostly stabilize the variance.

```
covid$log_covid <- log(covid$covid)
head(covid[, c('date', 'covid', 'log_covid')])
```

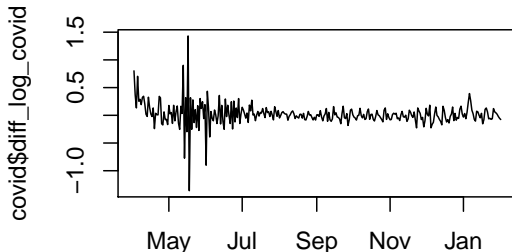
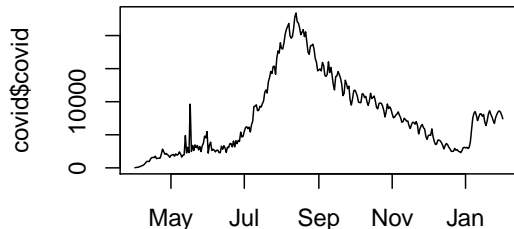
```
##           date covid log_covid
## 1 2021-04-01     26  3.258097
## 2 2021-04-02     58  4.060443
## 3 2021-04-03     84  4.430817
## 4 2021-04-04     96  4.564348
## 5 2021-04-05    194  5.267858
## 6 2021-04-06    250  5.521461
```

Differencing

Difference helps stabilize the mean.

In R, we could use the command `diff()` (plus NA offset at the first index).

```
covid$diff_log_covid <- c(NA, diff(covid$log_covid))  
  
par(mfrow=c(1,2))  
plot(covid$date, covid$covid, type='l')  
plot(covid$date, covid$diff_log_covid, type='l')
```



Section 3

Time Series Regression

Linear regression applied to time series data

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \dots + \beta_n x_{n,t} + \epsilon_t$$

- y_t is the value of a time series Y we want to predict.
- $x_{n,t}$ is the n predictor, which could be any other values from other series, or the time t itself.
- ϵ_t is the error term called the white noise, where

$$\epsilon_t \sim \mathcal{N}(\mu, \sigma^2)$$

Linear regression in R

In R, linear regression is fitted with the command `lm()`.

The formula of the command is `lm(y ~ X, data)`, where y is the predicted variable and X are the predictors.

```
fit <- lm(rti ~ year + factor(month), data = rti)
summary(fit)
```

```
##
## Call:
## lm(formula = rti ~ year + factor(month), data = rti)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -680.54  -58.91   16.22   64.04  249.09
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  84484.761   8888.239   9.505 3.43e-16 ***
## year        -32.273     3.473  -9.292 1.08e-15 ***
## factor(month)2 -116.182    52.099  -2.230 0.02767 *
## factor(month)3   22.091    52.099   0.424 0.67234
## factor(month)4 -172.182    52.099  -3.305 0.00126 **
## factor(month)5 -304.545    52.099  -5.845 4.74e-08 ***
## factor(month)6 -349.273    52.099  -6.704 7.73e-10 ***
## factor(month)7 -366.545    52.099  -7.035 1.48e-10 ***
## factor(month)8 -402.364    52.099  -7.723 4.43e-12 ***
## factor(month)9 -443.636    52.099  -8.515 6.92e-14 ***
## factor(month)10 -247.455    53.414  -4.633 9.50e-06 ***
## factor(month)11 -101.855    53.414  -1.907 0.05901 .
## factor(month)12   60.845    53.414   1.139 0.25700
```

Time series regression

The most common application of time series regression is to predict time series based on another series.

For example, if we wish to predict the number of COPD visits based on asthma visits.

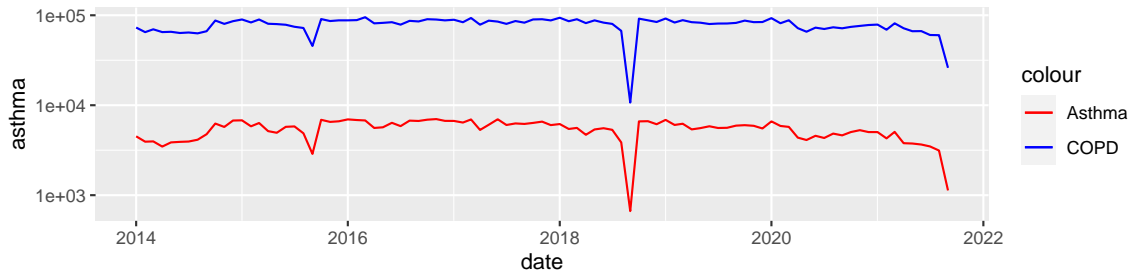
```
ncd <- read.csv('data/aci_month.csv')
ncd$date <- as.Date(ncd$date)
ncd
```

##	date	asthma	copd	ihd	month	year
## 1	2014-01-01	4510	73038	700	1	2014
## 2	2014-02-01	3933	64946	612	2	2014
## 3	2014-03-01	3961	69802	570	3	2014
## 4	2014-04-01	3470	65011	712	4	2014
## 5	2014-05-01	3861	65504	703	5	2014
## 6	2014-06-01	3915	63536	709	6	2014
## 7	2014-07-01	3948	64355	774	7	2014
## 8	2014-08-01	4136	62949	620	8	2014
## 9	2014-09-01	4762	66514	725	9	2014

Time series regression

Before we begin, we should plot both time series (asthma and COPD) to see whether there are patterns associated with both series.

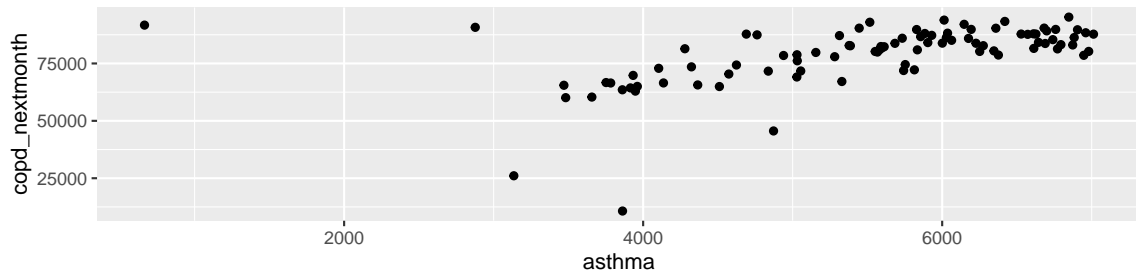
```
ggplot(ncd) +  
  geom_line(aes(x=date, y=asthma, color='Asthma')) +  
  geom_line(aes(x=date, y=copd, color='COPD')) +  
  scale_colour_manual(breaks=c("Asthma", "COPD"), values=c("red", "blue")) +  
  scale_y_log10()
```



Time series regression

To identify any linear relationship between asthma and COPD, we could use the scatter plot.

```
ggplot(ncd) +  
  geom_point(aes(x=asthma, y=copd_nextmonth))
```



It is clearly seen that there is linear relationship between both series.

Time series regression

Now we could fit the linear regression between asthma and COPD.

```
lm_copd_asthma <- lm(copd_nextmonth ~ asthma+factor(month), data=ncd)
summary(lm_copd_asthma)
```

```
##
## Call:
## lm(formula = copd_nextmonth ~ asthma + factor(month), data = ncd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -41462  -3661   -197    3740   28498
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.119e+04  6.547e+03   6.292 1.63e-08 ***
## asthma       6.251e+00  9.268e-01   6.745 2.29e-09 ***
## factor(month)2  1.092e+04  4.463e+03   2.447 0.016620 *
## factor(month)3 -8.054e+02  4.439e+03  -0.181 0.856492
## factor(month)4  6.488e+03  4.635e+03   1.400 0.165464
## factor(month)5  4.716e+03  4.583e+03   1.029 0.306582
## factor(month)6 -7.985e+02  4.499e+03  -0.177 0.859585
## factor(month)7  1.012e+03  4.554e+03   0.222 0.824797
## factor(month)8 -1.317e+04  4.580e+03  -2.875 0.005187 **
## factor(month)9  1.786e+04  4.836e+03   3.694 0.000405 ***
## factor(month)10 4.410e+03  4.581e+03   0.963 0.338656
## factor(month)11 4.879e+03  4.581e+03   1.065 0.290098
## factor(month)12 9.815e+03  4.581e+03   2.142 0.035251 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Time series regression

For the prediction task, we are interested in the coefficients (not the standard error or p-value).

```
coef(lm_copd_asthma)
```

```
##      (Intercept)          asthma factor(month)2 factor(month)3 factor(month)4
##      41193.693018          6.251183    10920.578564      -805.369335      6488.101183
## factor(month)5 factor(month)6 factor(month)7 factor(month)8 factor(month)9
##      4716.490558      -798.496066     1011.528168    -13167.183578     17861.581183
## factor(month)10 factor(month)11 factor(month)12
##      4410.259675      4878.714532      9814.721015
```

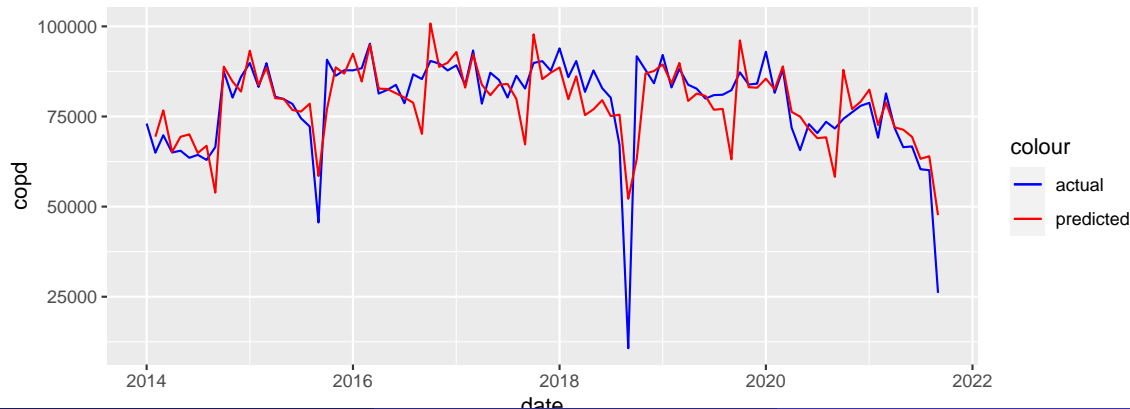
$$E[COPD] = 25966.27844 + 9.75708 \times Asthma$$

Note that we use $E[\dots]$, which is denoted as the expected number of \dots , in other words, we are predicting the mean of COPD visits.

Time series regression

We could use predict function to predict COPD cases using the fitted model.

```
predicted_copd <- c(rep(NA, 1), predict(lm_copd_asthma))  
ggplot(ncd) +  
  geom_line(aes(x=date, y=copd, color='actual')) +  
  geom_line(aes(x=date, y=predicted_copd, color='predicted')) +  
  scale_colour_manual(breaks=c("actual", "predicted"), values=c("blue", "red"))
```



Time series regression

However, it is assumed that we have already know the actual asthma visits, in order to forecast the COPD visits in the future.

If it is not the case, we have to forecast the COPD visits based on its trend and seasonality.

Assuming that the trend is continuing downward further from 2019.

Time series regression

Firstly, we split the training data only on and after 2019.

```
ncd2019 <- ncd[ncd$year >= 2019, ]  
ncd2019
```

```
##           date  asthma  copd  ihd month  year  copd_nextmonth  
## 61 2019-01-01   6873 92060 1389     1 2019           83059  
## 62 2019-02-01   6036 83059 1209     2 2019           88208  
## 63 2019-03-01   6226 88208 1287     3 2019           83807  
## 64 2019-04-01   5386 83807 1283     4 2019           82714  
## 65 2019-05-01   5567 82714 1188     5 2019           79974  
## 66 2019-06-01   5834 79974 1230     6 2019           80929  
## 67 2019-07-01   5582 80929 1157     7 2019           81027  
## 68 2019-08-01   5612 81027 1084     8 2019           82285  
## 69 2019-09-01   5930 82285 1072     9 2019           87250  
## 70 2019-10-01   6001 87250 1320    10 2019           83857  
## 71 2019-11-01   5904 83857 1116    11 2019           84100
```

Time series regression

Next, we create a new forecasting horizon (2021 to 2022).

```
newdata2021 <- data.frame(year=2021, month=1:12) # 2021
newdata2022 <- data.frame(year=2022, month=1:12) # 2022
newdata <- rbind(newdata2021, newdata2022) # Appending 2021 and 2022
newdata$date <- as.Date(ISDate(newdata$year, newdata$month, 1)) # create a new date variable
newdata
```

```
##   year month      date
## 1  2021     1 2021-01-01
## 2  2021     2 2021-02-01
## 3  2021     3 2021-03-01
## 4  2021     4 2021-04-01
## 5  2021     5 2021-05-01
## 6  2021     6 2021-06-01
## 7  2021     7 2021-07-01
## 8  2021     8 2021-08-01
## 9  2021     9 2021-09-01
## 10 2021    10 2021-10-01
## 11 2021    11 2021-11-01
## 12 2021    12 2021-12-01
## 13 2022     1 2022-01-01
## 14 2022     2 2022-02-01
## 15 2022     3 2022-03-01
## 16 2022     4 2022-04-01
## 17 2022     5 2022-05-01
## 18 2022     6 2022-06-01
## 19 2022     7 2022-07-01
## 20 2022     8 2022-08-01
## 21 2022     9 2022-09-01
```


Time series regression

Next, the linear model is fitted to the training data (ncd2019) with the following formula:

$$E[COPD] = Date + Jan + Feb + Mar + \dots + Dec$$

```
lm2 <- lm(copd ~ date + factor(month), data=ncd2019)
summary(lm2)
```

```
##
## Call:
## lm(formula = copd ~ date + factor(month), data = ncd2019)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24369.6  -1716.1   433.8   2176.7  12705.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   566311.672   88024.475     6.434 2.83e-06 ***
## date          -26.195     4.814    -5.442 2.51e-05 ***
## factor(month)2  -9213.302   6346.108    -1.452  0.16206
## factor(month)3  -489.455   6350.779    -0.077  0.93933
## factor(month)4  -9751.757   6359.237    -1.533  0.14083
## factor(month)5 -13169.587   6370.742    -2.067  0.05190 .
## factor(month)6 -10790.221   6386.039    -1.690  0.10662
## factor(month)7 -12624.384   6404.120    -1.971  0.06269 .
## factor(month)8 -10825.353   6426.160    -1.685  0.10761
## factor(month)9 -21562.321   6451.578    -3.342  0.00325 **
## factor(month)10 -4773.192   7106.617    -0.672  0.50949
## factor(month)11 -4737.660   7117.340    -0.666  0.51324
```

Time series regression

We could use the fitted model to predict COPD cases in 2021 and 2022.

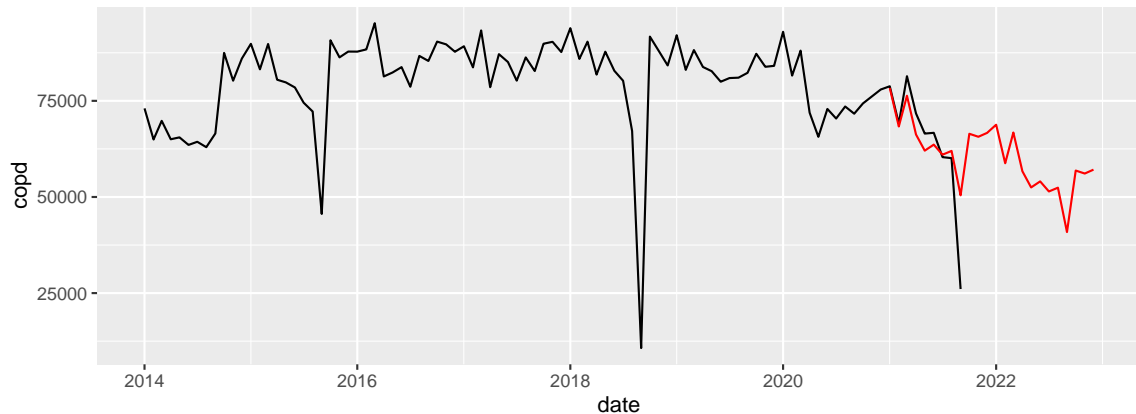
```
newdata$forecast <- predict(lm2, newdata=newdata)
newdata
```

```
##      year month      date forecast
## 1  2021      1 2021-01-01 78359.18
## 2  2021      2 2021-02-01 68333.85
## 3  2021      3 2021-03-01 76324.25
## 4  2021      4 2021-04-01 66249.92
## 5  2021      5 2021-05-01 62046.25
## 6  2021      6 2021-06-01 63613.58
## 7  2021      7 2021-07-01 60993.58
## 8  2021      8 2021-08-01 61980.58
## 9  2021      9 2021-09-01 50431.58
## 10 2021     10 2021-10-01 66434.87
## 11 2021     11 2021-11-01 65658.37
## 12 2021     12 2021-12-01 66667.87
## 13 2022      1 2022-01-01 68798.17
## 14 2022      2 2022-02-01 58772.83
## 15 2022      3 2022-03-01 66763.23
## 16 2022      4 2022-04-01 56688.90
## 17 2022      5 2022-05-01 52485.23
## 18 2022      6 2022-06-01 54052.56
## 19 2022      7 2022-07-01 51432.56
## 20 2022      8 2022-08-01 52419.56
## 21 2022      9 2022-09-01 40870.56
## 22 2022     10 2022-10-01 56873.86
## 23 2022     11 2022-11-01 56097.36
## 24 2022     12 2022-12-01 57106.86
```

Time series regression

We could also visualize the forecast.

```
ggplot() +  
  geom_line(data=ncd, aes(x=date, y=copd)) +  
  geom_line(data=newdata, aes(x=date, y=forecast), color='red')
```



Time series regression

We finally summarise predicted cases in 2022.

```
newdata[newdata$year==2022, ]
```

##	year	month	date	forecast
## 13	2022	1	2022-01-01	68798.17
## 14	2022	2	2022-02-01	58772.83
## 15	2022	3	2022-03-01	66763.23
## 16	2022	4	2022-04-01	56688.90
## 17	2022	5	2022-05-01	52485.23
## 18	2022	6	2022-06-01	54052.56
## 19	2022	7	2022-07-01	51432.56
## 20	2022	8	2022-08-01	52419.56
## 21	2022	9	2022-09-01	40870.56
## 22	2022	10	2022-10-01	56873.86
## 23	2022	11	2022-11-01	56097.36
## 24	2022	12	2022-12-01	57106.86

Time series regression: diagnostic

Residual plots

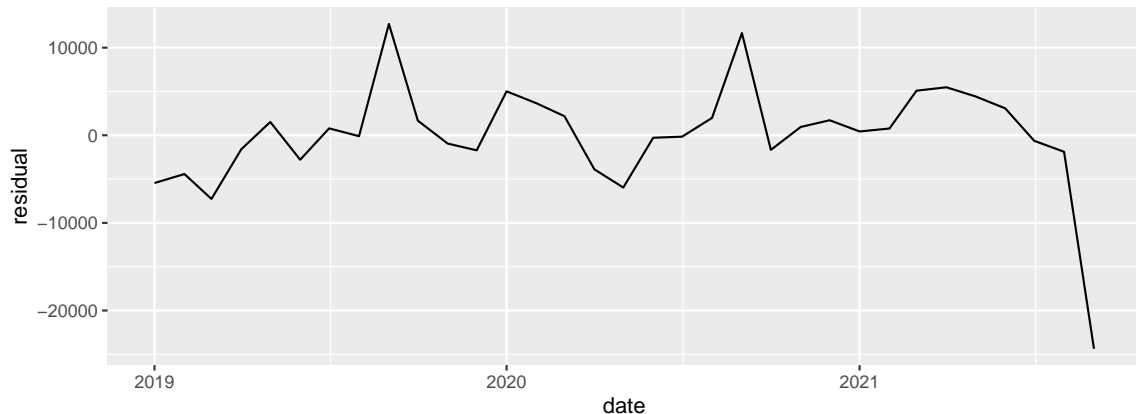
```
ncd2019$fit <- lm2$fit
ncd2019$residual <- lm2$residual
ncd2019[, c("date", "copd", "fit", "residual")]
```

##		date	copd	fit	residual
## 61	2019-01-01	92060	97507.42	-5447.4170	
## 62	2019-02-01	83059	87482.08	-4423.0836	
## 63	2019-03-01	88208	95472.48	-7264.4818	
## 64	2019-04-01	83807	85398.15	-1591.1485	
## 65	2019-05-01	82714	81194.48	1519.5182	
## 66	2019-06-01	79974	82761.82	-2787.8152	
## 67	2019-07-01	80929	80141.82	787.1848	
## 68	2019-08-01	81027	81128.82	-101.8152	
## 69	2019-09-01	82285	69579.82	12705.1848	
## 70	2019-10-01	87250	85582.11	1666.8022	

Time series regression: Diagnostics

Residual plots

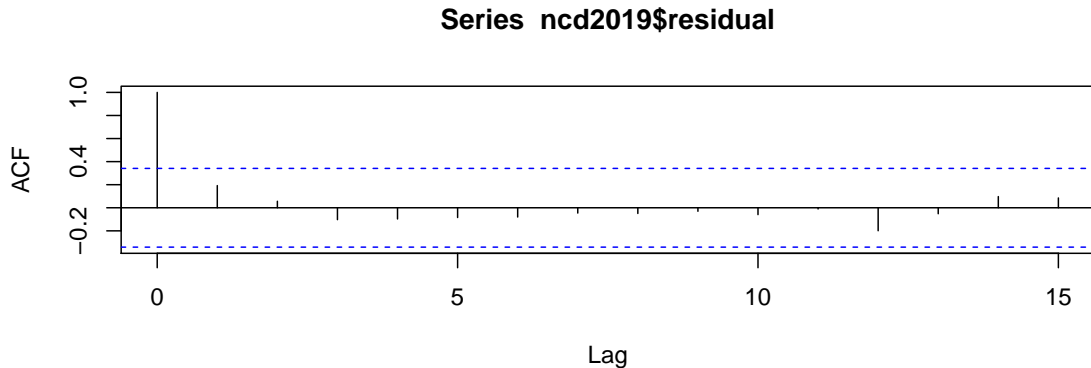
```
ggplot(ncd2019) +  
  geom_line(aes(x=date, y=residual))
```



Time series regression: Diagnostics

ACF plot of the residuals

```
acf(ncd2019$residual)
```



Forecasting PM 2.5

Forecasting PM 2.5 concentration capturing trend and seasonality (hour and day-of-week)

```
pm <- read.csv("data/pm_ts.csv")
pm$DATETIMEDATA <- as.POSIXct(pm$DATETIMEDATA)
pm$hour <- format(pm$DATETIMEDATA, "%H")
pm$dow <- weekdays(pm$DATETIMEDATA)
pm
```

##	DATETIMEDATA	PM25	hour	dow
## 1	2021-11-10 00:00:00	14.07865	00	Wednesday
## 2	2021-11-10 01:00:00	13.37079	01	Wednesday
## 3	2021-11-10 02:00:00	13.28090	02	Wednesday
## 4	2021-11-10 03:00:00	13.11236	03	Wednesday
## 5	2021-11-10 04:00:00	13.16854	04	Wednesday
## 6	2021-11-10 05:00:00	13.73034	05	Wednesday
## 7	2021-11-10 06:00:00	14.19101	06	Wednesday
## 8	2021-11-10 07:00:00	15.35227	07	Wednesday
## 9	2021-11-10 08:00:00	16.25000	08	Wednesday
## 10	2021-11-10 09:00:00	16.26136	09	Wednesday
## 11	2021-11-10 10:00:00	15.98864	10	Wednesday
## 12	2021-11-10 11:00:00	15.09091	11	Wednesday
## 13	2021-11-10 12:00:00	14.41573	12	Wednesday
## 14	2021-11-10 13:00:00	14.39326	13	Wednesday
## 15	2021-11-10 14:00:00	13.48864	14	Wednesday
## 16	2021-11-10 15:00:00	13.94318	15	Wednesday
## 17	2021-11-10 16:00:00	14.39326	16	Wednesday
## 18	2021-11-10 17:00:00	14.56180	17	Wednesday
## 19	2021-11-10 18:00:00	15.87640	18	Wednesday
## 20	2021-11-10 19:00:00	17.30337	19	Wednesday
## 21	2021-11-10 20:00:00	18.31461	20	Wednesday

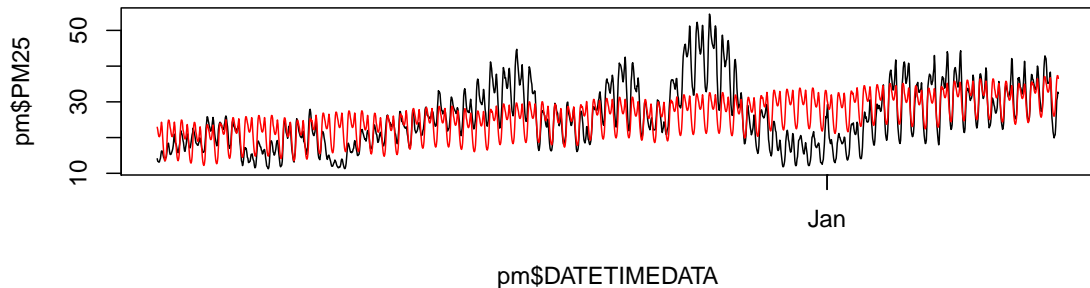
Forecasting PM 2.5

```
lm_pm <- lm(PM25 ~ DATETIMEDATA+hour+dow, pm)
```

```
pm$fit <- lm_pm$fit
```

```
plot(pm$DATETIMEDATA, pm$PM25, type='l')
```

```
lines(pm$DATETIMEDATA, pm$fit, col='red')
```



Case study: Forecasting Dengue with rainfall data

In this tutorial, we will predict Dengue cases, based on rainfall data.

Case study: Forecasting Dengue with rainfall data

Firstly, loaded the data containing dengue cases and rainfall data.

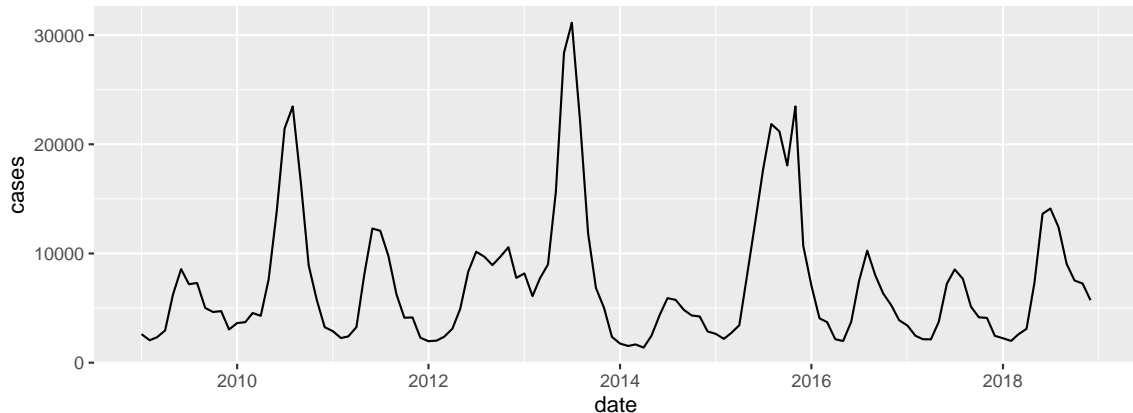
```
# read the data  
dhfrain <- read.csv('data/dhfrain.csv')  
dhfrain$date <- as.Date(dhfrain$date)  
head(dhfrain)
```

##		date	year	month	cases	mean_rain	max_rain	year_from_last_epidemic
## 1		2009-01-01	2009	1	2614	0.6542994	176.2	2
## 2		2009-02-01	2009	2	2057	0.2529173	46.4	2
## 3		2009-03-01	2009	3	2324	2.8785509	132.0	2
## 4		2009-04-01	2009	4	2947	4.5172608	216.8	2
## 5		2009-05-01	2009	5	6234	7.4758991	178.7	2
## 6		2009-06-01	2009	6	8569	5.2782202	157.1	2

Case study: Forecasting Dengue with rainfall data

We visualize time series of monthly dengue cases.

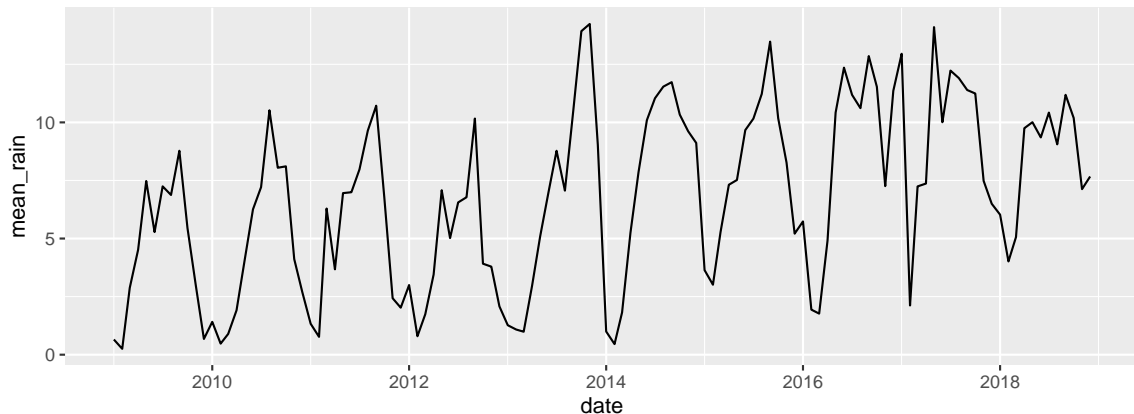
```
ggplot(dhfrain) + geom_line(aes(x=date, y=cases))
```



Case study: Forecasting Dengue with rainfall data

Next, mean monthly rainfall is shown.

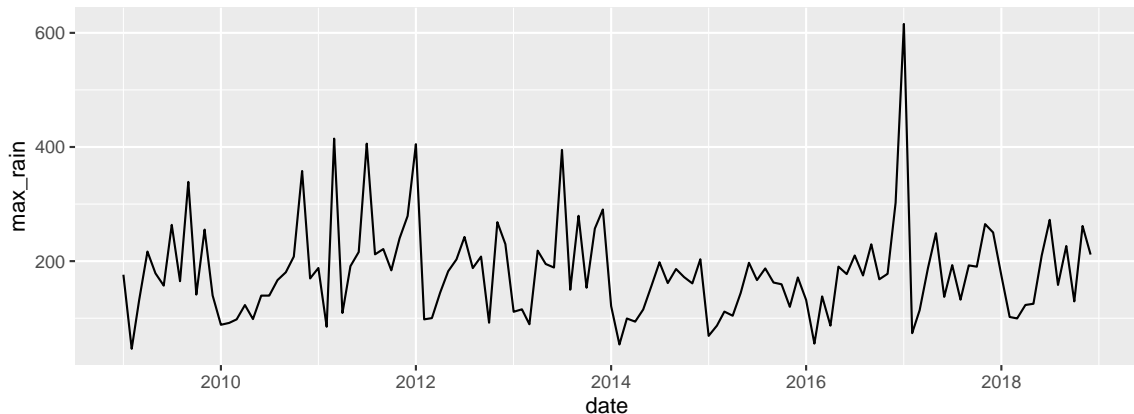
```
ggplot(dhfrain) +  
  geom_line(aes(x=date, y=mean_rain))
```



Case study: Forecasting Dengue with rainfall data

Next, visualizing max rainfall each month

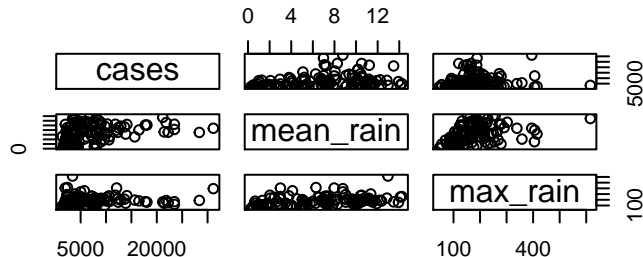
```
ggplot(dhfrain) +  
  geom_line(aes(x=date, y=max_rain))
```



Case study: Forecasting Dengue with rainfall data

We could also examine the relationship between DHF cases and rainfalls.

```
pairs(dhfrain[, c('cases', 'mean_rain', 'max_rain')])
```



Case study: Forecasting Dengue with rainfall data

Now, use `lm()` to fit linear regression.

Firstly, having only rainfall parameters as predictors

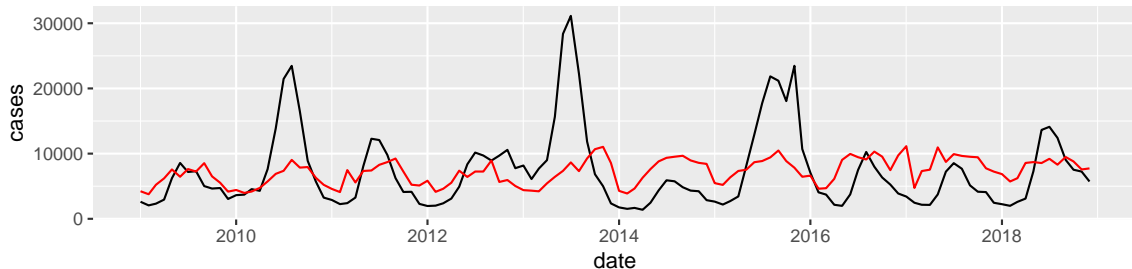
```
model_dhfrain_1 <- lm(cases ~ max_rain+mean_rain, dhfrain)
summary(model_dhfrain_1)
```

```
##
## Call:
## lm(formula = cases ~ max_rain + mean_rain, data = dhfrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7730  -3633  -1654   1984  22483
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3549.278   1349.418   2.630  0.00968 **
## max_rain       2.035     6.608   0.308  0.75861
## mean_rain    489.410    146.535   3.340  0.00113 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5655 on 117 degrees of freedom
## Multiple R-squared:  0.1057, Adjusted R-squared:  0.09043
## F-statistic: 6.916 on 2 and 117 DF,  p-value: 0.00145
```


Case study: Forecasting Dengue with rainfall data

Use fitted model to produce fitted data.

```
dhfrain$fit_dhfrain_1 <- predict(model_dhfrain_1)
ggplot(dhfrain) + geom_line(aes(x=date, y=cases)) + geom_line(aes(x=date, y=fit_dhfrain_1), col='red')
```

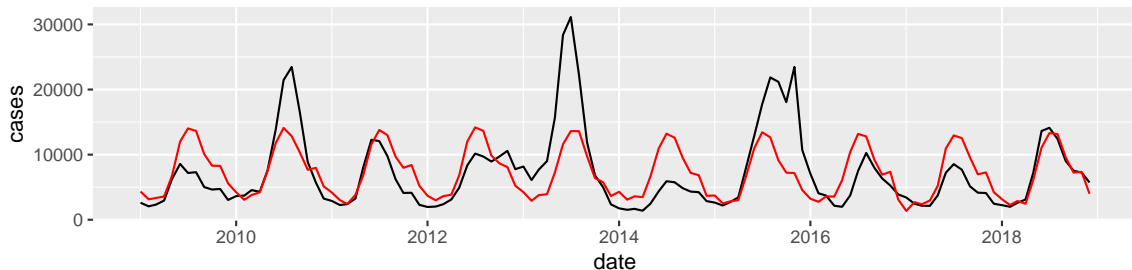


It is clearly seen that this forecast isn't adequate.

Case study: Forecasting Dengue with rainfall data

Next, we could add month-of-the-year to capture seasonal pattern of the disease.

```
model_dhfrain_2 <- lm(cases ~ max_rain+mean_rain+factor(month), dhfrain)
dhfrain$fit_dhfrain_2 <- predict(model_dhfrain_2)
ggplot(dhfrain) + geom_line(aes(x=date, y=cases)) + geom_line(aes(x=date, y=fit_dhfrain_2), col='red')
```

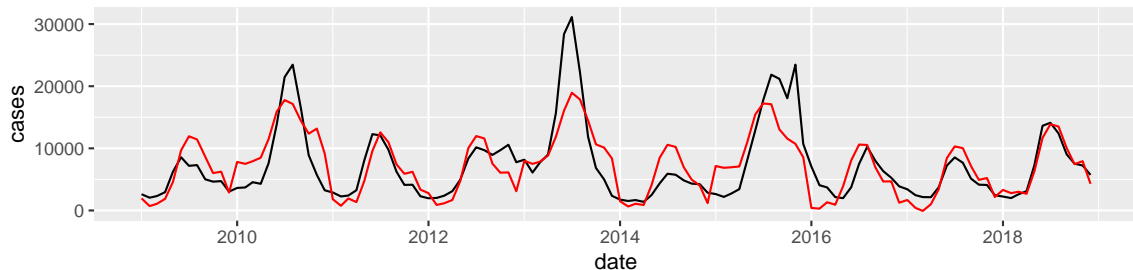


From this plot, it is clear that adding months improve the forecast but still not capture cyclical pattern.

Case study: Forecasting Dengue with rainfall data

Next, we use the number of years from last epidemic as proxy indicator for the cyclical pattern

```
model_dhfrain_3 <- lm(cases ~ max_rain+mean_rain+factor(month)+factor(year_from_last_epidemic), dhfrain)
dhfrain$fit_dhfrain_3 <- predict(model_dhfrain_3)
ggplot(dhfrain) + geom_line(aes(x=date, y=cases)) + geom_line(aes(x=date, y=fit_dhfrain_3), col='red')
```



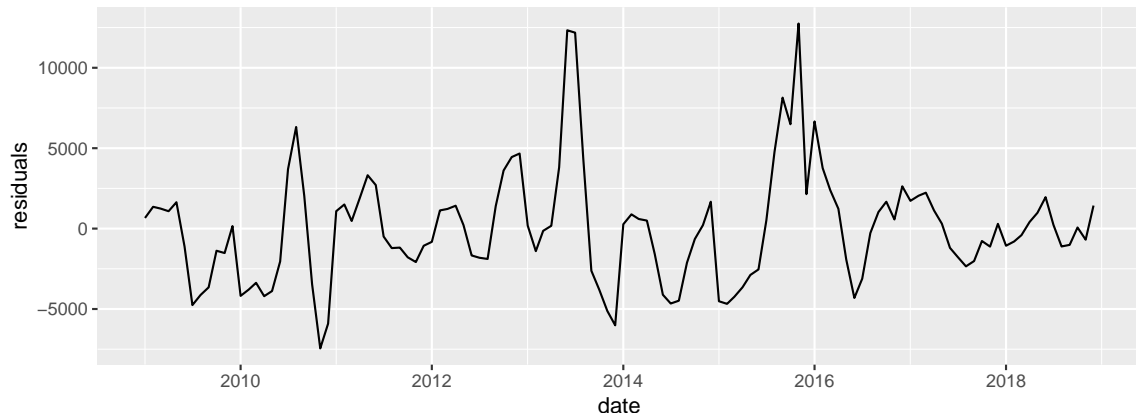
Including the number of years partially capture cyclical pattern.

Case study: Forecasting Dengue with rainfall data

Now, we should diagnose if the model violate linear regression assumptions.

We calculate residuals, which is unlikely normally distributed.

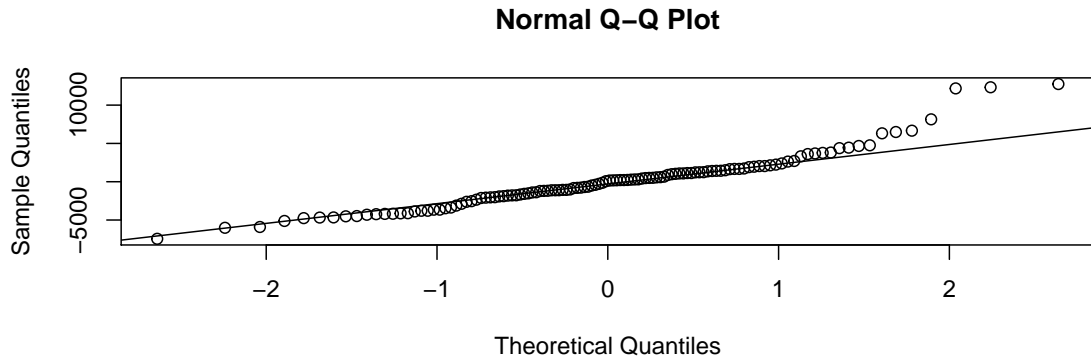
```
dhfrain$residuals <- model_dhfrain_3$residuals  
ggplot(dhfrain) + geom_line(aes(x=date, y=residuals))
```



Case study: Forecasting Dengue with rainfall data

Q-Q plot also suggest that the normality assumption is violated.

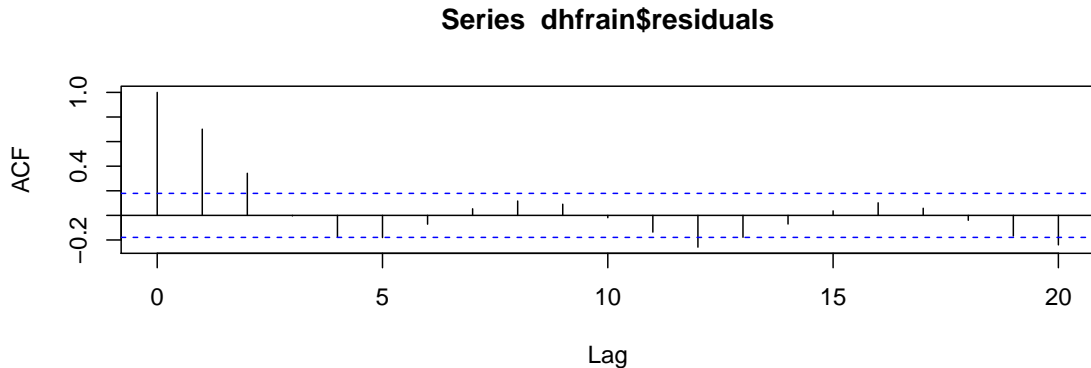
```
qqnorm(dhfrain$residuals)  
qqline(dhfrain$residuals)
```



Case study: Forecasting Dengue with rainfall data

ACF plot reveals autocorrelation at lag 1 and 2. Thereby independence is also violated.

```
acf(dhfrain$residuals)
```



Case study: Forecasting Dengue with rainfall data

Final remarks:

- Linear regression is a great method to forecast, especially with incorporating external information.
- However, most of the time series data violate linear regression assumption so that there would be some information that didn't be captured by the model.
- There are several methods addressing these problems. The most common ways to handle this is to also predict the residuals themselves with other methods, such as ARIMA.