# 시계열 forecasting Part II - 시계열 예측 모델

**A future like the past:**

시계열 예측의 가장 기본적인 가정은 과거의 패턴은 미래에도 계속된다는 가정이다. 이 가정에 의한 단기 미래예측에서는 데이터가 발생되는 환경이 현재와 유사하기 때문에 불확실성이 작아 예측이 잘 맞는다. 하지만 장기 미래예측으로 갈수록 데이터가 발생되는 환경이 달라질 가능성이 높아지고 예측하지 못했던 오류들이 계속 쌓이면서 예측 데이터에 대한 불확실성이 높아질 수 밖에 없다.

이러한 이유때문에 시계열 예측은 외부 충격이나 원인모를 이유에 의한 갑작스런 데이터 흐름 변화를 예측해낼 수 없다는 한계를 지닌다. 그런 지점을 전환점(Turning Point)라고 한다. 이 전환점은 시계열 분석에 있어 중요한 부분이지만 전환점을 예측하기 위해서는 전통적인 시계열 분석 기법보다는 다른 기법을 활용해야 할 수 있다. [[1]](#footnote-20)

본 장에서는 시계열 예측에 많이 사용되는 모델들을 설명한다. 과거 오랫동안 사용되었던 전통적(Classical) 방법부터 최근에 개발되어 사용되는 최신 모델까지 각각의 모델의 특성과 구축 방법을 설명한다. 시계열 예측 모델을 설명하기 위해 주로 ts 객체를 사용하는 forecast 패키지에서 제공하는 함수들 위주로 설명하지만 모델에 따라 데이터프레임을 사용하기도 할 것이다.

사실 이번 장의 앞에 소개하는 몇몇 알고리즘을 보면 ’이게 무슨 예측이야?’라고 생각할 수도 있다. 하지만 우리는 실무에서도 3년치 평균, 6개월치 평균과 같이 단순한 모델을 사용해 비지니스의 의사결정을 하는 경우들이 있는 걸 보면 복잡한 모델이라고 꼭 좋은 것은 아닐수 있다는 점을 간과하면 안된다 .

이번 장에서 설명하는 것은 일단 ‘이런 모델들이 있구나’ 하는 모델의 이해 정도만 하고 넘어가도 좋다. 과거에는 forecast와 같은 패키지에서 제공하는 다양한 함수를 사용하여 여러가지 모델을 각각 생성하고 성능평가를 위해서 성능이 좋은 모델을 선택했다. 그러나 최근에는 여러가지 모델을 하나의 프레임워크에서 수행하여 가장 좋은 모델을 선정하는 방법이 사용된다. 따라서 이 장에서는 모델을 이해하고 다음 장에서 시계열 평가, 시각화를 위한 프레임워크를 소개하고자 한다.

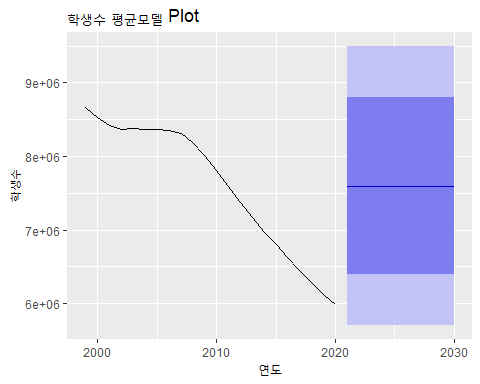
## 평균 모델

평균 모델은 우리가 흔히 알고 있는 평균이 앞으로의 미래에도 적용하는 방법이다. forecast 패키지에서는 meanf() 함수를 통해 시계열 데이터의 평균을 활용한 미래 예측값을 반환하는 함수를 제공한다. 이 함수에서는 평균 모델에 의한 점 예측값(Point Forecast)뿐 아니라 예측구간 80%와 95%를 산출해 준다. 매개변수를 설정함으로써 이 예측구간을 Bootstrap 방법을 통해 산출할 수도 있다. meanf() 함수의 입력 데이터는 숫자 벡터나 ts 클래스 객체를 사용한다. meanf()함수의 결과를 plot하기 위해서는 autoplot() 함수에 meanf() 결과 객체를 전달하여 plotting 할 수 있다.

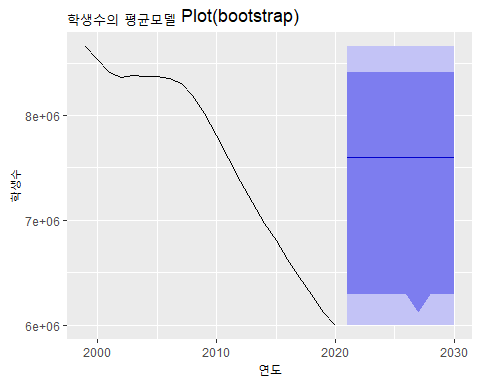
library(forecast)  
# meanf()를 사용하여 ts 객체의 학생수계 열에 대한 평균모델을 생성하고 summary()로 상세내용을 출력  
summary(meanf(students.ts[,2]))

Forecast method: Mean  
  
Model Information:  
$mu  
[1] 7598603  
  
$mu.se  
[1] 189914  
  
$sd  
[1] 890775.4  
  
$bootstrap  
[1] FALSE  
  
$call  
meanf(y = students.ts[, 2])  
  
attr(,"class")  
[1] "meanf"  
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set -1.694451e-10 870295.1 780669 -1.422719 10.7441 6.074563 0.8753895  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 7598603 6393450 8803757 5704501 9492706  
2022 7598603 6393450 8803757 5704501 9492706  
2023 7598603 6393450 8803757 5704501 9492706  
2024 7598603 6393450 8803757 5704501 9492706  
2025 7598603 6393450 8803757 5704501 9492706  
2026 7598603 6393450 8803757 5704501 9492706  
2027 7598603 6393450 8803757 5704501 9492706  
2028 7598603 6393450 8803757 5704501 9492706  
2029 7598603 6393450 8803757 5704501 9492706  
2030 7598603 6393450 8803757 5704501 9492706

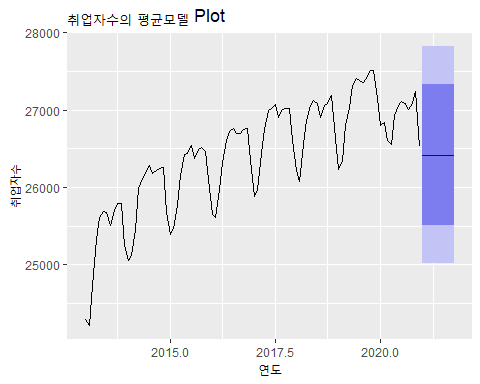
# autoplot()을 사용하여 ts 객체의 학생수계 열에 대한 평균모델을 plotting  
autoplot(meanf(students.ts[,2]), main = '학생수 평균모델 Plot', xlab = '연도', ylab = '학생수')



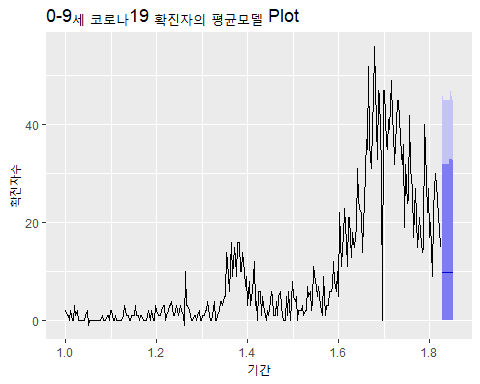
# autoplot()을 사용하여 ts 객체의 학생수계 열에 대한 평균모델(예측구간 산출에 boosted 방법을 사용)을 plotting  
autoplot(meanf(students.ts[,2], bootstrap = TRUE), main = '학생수의 평균모델 Plot(bootstrap)', xlab = '연도', ylab = '학생수')



# 전체 취업자수에 대한 평균모델 plotting  
autoplot(meanf(employees.ts[,2]), main = '취업자수의 평균모델 Plot', xlab = '연도', ylab = '취업자수')



# 코로나 19 확진자에 대한 평균모델(예측구간 산출에 boosted 방법을 사용) plotting  
autoplot(meanf(covid19.ts[,2], bootstrap = TRUE), main = '0-9세 코로나19 확진자의 평균모델 Plot', xlab = '기간', ylab = '확진자수')



## 단순(Naive) 모델

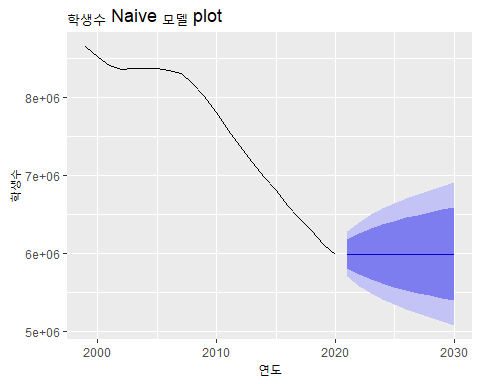
단순(Naive) 모델은 시계열 측정값의 마지막 값이 지속될 것이라는 가정하에 향후 데이터값을 예측하는 모델이다. 이 모델은 경제 금융 시계열 모델에서 많이 사용된다. [[2]](#footnote-23) 앞선 평균 모델의 예측구간은 시간에 따라 일정한 반면 과 달리 Naive 모델의 예측구간은 예측 시간이 늘어날수록 범위가 늘어난다. 아주 가까운 미래의 데이터는 지금 데이터에서 크게 벗어나지 않을것이라는 가정이 깔려있지만 이 가정은 시간이 지날수록 확실치 않다는 모델이다. 일반적으로 다른 시계열 모델들의 예측구간도 Naive 모델 과 같이 예측기간이 늘어날 수록 범위가 넓어지는 경향을 지닌다.

forecast 패키지에서는 Naive 모델을 생성하는 함수로 naive()를 제공한다. naive()의 실행 결과는 ts 객체로 전달된 데이터에 대한 시차 10의 미래 데이터까지 예측치와 80%, 95% 예측구간의 데이터를 산출해준다. 앞선 평균모델의 meanf()와 같이 autoplot()을 사용하여 plot을 만들 수 있다.

# 학생수계 열에 대한 Naive 모델의 상세 정보와 plot  
students.ts[, 2] %>% naive() %>% summary()

Forecast method: Naive method  
  
Model Information:  
Call: naive(y = .)   
  
Residual sd: 77849.7308   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set -126815.2 147831.2 128514.4 -1.771615 1.791893 1 0.8922526  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5995239 5805786 6184692 5705495 6284983  
2022 5995239 5727312 6263166 5585479 6404999  
2023 5995239 5667096 6323382 5493388 6497090  
2024 5995239 5616332 6374146 5415751 6574727  
2025 5995239 5571608 6418870 5347352 6643126  
2026 5995239 5531175 6459303 5285514 6704964  
2027 5995239 5493993 6496485 5228649 6761829  
2028 5995239 5459384 6531094 5175720 6814758  
2029 5995239 5426879 6563599 5126007 6864471  
2030 5995239 5396135 6594343 5078988 6911490

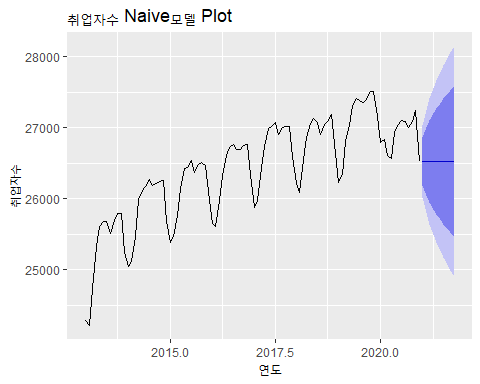
students.ts[, 2] %>% naive() %>% autoplot(main = '학생수 Naive 모델 plot', xlab = '연도', ylab = '학생수')



# 취업자수 열에 대한 Naive 모델의 상세 정보와 plot  
employees.ts[,2] %>% naive() %>% summary()

Forecast method: Naive method  
  
Model Information:  
Call: naive(y = .)   
  
Residual sd: 261.7975   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set 23.56842 261.4803 194.5579 0.08782727 0.7411892 0.6138395  
 ACF1  
Training set 0.4208691  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
Jan 2021 26526 26190.90 26861.10 26013.51 27038.49  
Feb 2021 26526 26052.10 26999.90 25801.23 27250.77  
Mar 2021 26526 25945.59 27106.41 25638.34 27413.66  
Apr 2021 26526 25855.80 27196.20 25501.02 27550.98  
May 2021 26526 25776.69 27275.31 25380.03 27671.97  
Jun 2021 26526 25705.17 27346.83 25270.66 27781.34  
Jul 2021 26526 25639.41 27412.59 25170.07 27881.93  
Aug 2021 26526 25578.19 27473.81 25076.45 27975.55  
Sep 2021 26526 25520.70 27531.30 24988.52 28063.48  
Oct 2021 26526 25466.32 27585.68 24905.36 28146.64

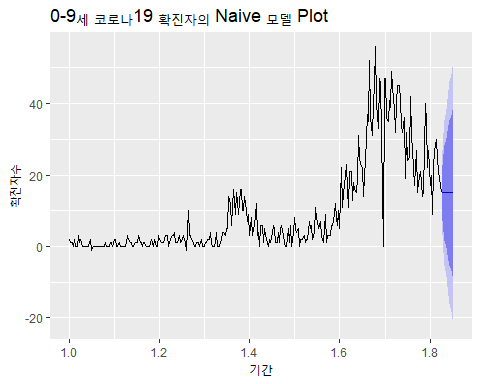
employees.ts[,2] %>% naive() %>% autoplot(main = '취업자수 Naive모델 Plot', xlab = '연도', ylab = '취업자수')



# 0-9세 코로나 확진자수에 대한 Naive 모델의 상세 정보와 plot  
covid19.ts[,2] %>% naive() %>% summary()

Forecast method: Naive method  
  
Model Information:  
Call: naive(y = .)   
  
Residual sd: 6.0029   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set 0.04318937 5.993075 3.465116 NaN Inf NaN -0.3127329  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
1.827397 15 7.3195658 22.68043 3.253790 26.74621  
1.830137 15 4.1382258 25.86177 -1.611650 31.61165  
1.832877 15 1.6970978 28.30290 -5.345033 35.34503  
1.835616 15 -0.3608683 30.36087 -8.492421 38.49242  
1.838356 15 -2.1739729 32.17397 -11.265325 41.26533  
1.841096 15 -3.8131447 33.81314 -13.772222 43.77222  
1.843836 15 -5.3205188 35.32052 -16.077552 46.07755  
1.846575 15 -6.7235483 36.72355 -18.223300 48.22330  
1.849315 15 -8.0413025 38.04130 -20.238631 50.23863  
1.852055 15 -9.2876654 39.28767 -22.144779 52.14478

covid19.ts[,2] %>% naive() %>% autoplot(main = '0-9세 코로나19 확진자의 Naive 모델 Plot', xlab = '기간', ylab = '확진자수')



## 계절성 단순(Seasonal Naive) 모델

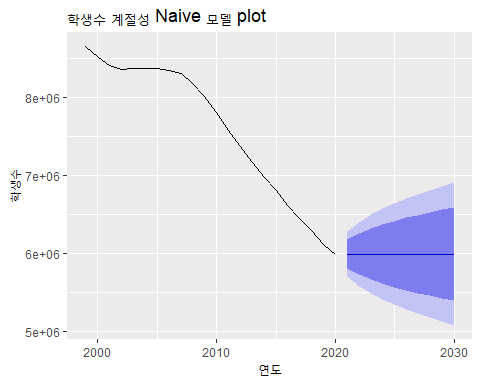
계절성 단순(Seasonal Naive) 모델은 단순(Naive) 모델에서 계절성을 추가한 모델이다. 1장에서 설명했듯이 계절성(Seasonal Pattern)은 주기성(Cyclic Pattern)은 일정한 주기를 갖는가에 따라 다른데 계절성 단순 모델에서 적용하는 것은 일정한 주기를 가지는 계절성에 한정된다. 아래의 예에서 보면 년별 데이터인 전체 학생수계는 계절성을 가지지 않기 때문에 앞선 Naive 모델과 다르지 않다. 하지만 전체 취업자 수는 명확한 계절성을 지니기 때문에 Naive모델과는 달리 곡선형태의 예측 결과를 나타내게 된다.

계절성 단순 모델을 생성하는 함수는 forecast 패키지의 snaive()를 아래와 같이 사용할 수 있다. meanf()와 naive()와 다른 점은 meanf()와 naive()는 기본적으로 10 lag 미래의 데이터를 산출해주지만 snaive()는 기본적으로 2 lag 미래 데이터를 산출해주기 때문에 예측 산출 주기를 다시 설정해 줄 필요가 있다.

# 학생수계 열에 대한 계절성 Naive 모델의 상세 정보와 plot  
students.ts[,2] %>% snaive(10) %>% summary()

Forecast method: Seasonal naive method  
  
Model Information:  
Call: snaive(y = ., h = 10)   
  
Residual sd: 77849.7308   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set -126815.2 147831.2 128514.4 -1.771615 1.791893 1 0.8922526  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5995239 5805786 6184692 5705495 6284983  
2022 5995239 5727312 6263166 5585479 6404999  
2023 5995239 5667096 6323382 5493388 6497090  
2024 5995239 5616332 6374146 5415751 6574727  
2025 5995239 5571608 6418870 5347352 6643126  
2026 5995239 5531175 6459303 5285514 6704964  
2027 5995239 5493993 6496485 5228649 6761829  
2028 5995239 5459384 6531094 5175720 6814758  
2029 5995239 5426879 6563599 5126007 6864471  
2030 5995239 5396135 6594343 5078988 6911490

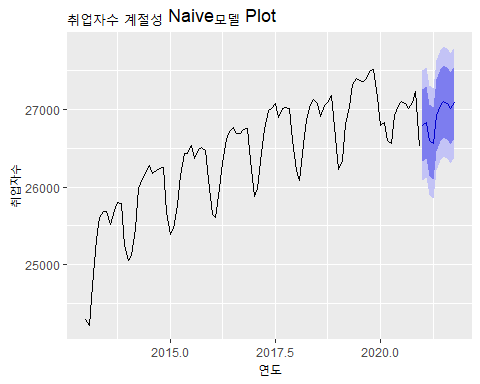
students.ts[,2] %>% snaive(10) %>% autoplot(main = '학생수 계절성 Naive 모델 plot', xlab = '연도', ylab = '학생수')



# 취업자수 열에 대한 계절성 Naive 모델의 상세 정보와 plot  
employees.ts[,2] %>% snaive(10) %>% summary()

Forecast method: Seasonal naive method  
  
Model Information:  
Call: snaive(y = ., h = 10)   
  
Residual sd: 281.6863   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set 229.2857 361.904 316.9524 0.8720369 1.19817 1 0.8173272  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
Jan 2021 26800 26336.2 27263.8 26090.68 27509.32  
Feb 2021 26838 26374.2 27301.8 26128.68 27547.32  
Mar 2021 26609 26145.2 27072.8 25899.68 27318.32  
Apr 2021 26562 26098.2 27025.8 25852.68 27271.32  
May 2021 26930 26466.2 27393.8 26220.68 27639.32  
Jun 2021 27055 26591.2 27518.8 26345.68 27764.32  
Jul 2021 27106 26642.2 27569.8 26396.68 27815.32  
Aug 2021 27085 26621.2 27548.8 26375.68 27794.32  
Sep 2021 27012 26548.2 27475.8 26302.68 27721.32  
Oct 2021 27088 26624.2 27551.8 26378.68 27797.32

employees.ts[,2] %>% snaive(10) %>% autoplot(main = '취업자수 계절성 Naive모델 Plot', xlab = '연도', ylab = '취업자수')



## Random Walk 모델

랜덤 워크는 시간적으로 다음값이 시간적으로 현재값의 영향을 받는다는 모델이다. 앞선 Naive모델과 유사한 형태이지만 마지막 데이터에 예측이 어려운 랜덤 값의 변동이 추가된다는 점에서 차이가 있다. 주식의 예에서 보면 내일의 종가는 오늘의 종가에 오늘의 주가 변동이 반영된 값이다. 오늘의 주가 변동은 사전에 알기 어렵고 예측도 어렵기 때문에 랜덤 값으로 간주한다면 이 경우가 랜덤 워크 모델이다. 랜덤 워크에는 어제값 이외에 의미있는 정보가 없기 때문에 예측이 복잡하지 않다[[3]](#footnote-27).

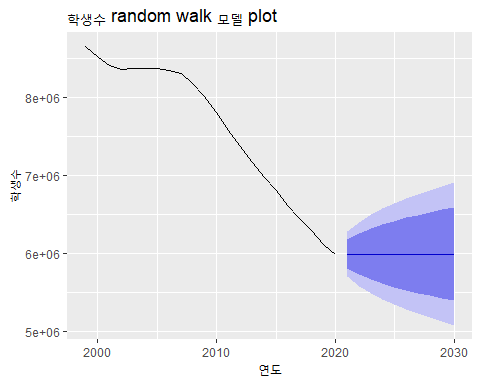
랜덤 워크 모델은 드리프트가 없는 모델과 드리프트가 존재하는 모델의 두가지 종류가 있다. 드리프트가 존재하는 모델은 랜덤워크 모델을 기반으로 하지만 예측값이 시간의 흐름에 따라 증가하거나 감소하는 모델이다.

랜덤 워크 모델은 forecast 패키지의 rwf()를 이용하여 생성할 수 있다. 앞서 설명한 드리프트의 유무는 drift 매개변수를 통해 설정할 수 있다.

# 학생수계 열에 대한 random walk 모델의 상세 정보와 plot  
students.ts[,2] %>% rwf() %>% summary()

Forecast method: Random walk  
  
Model Information:  
Call: rwf(y = .)   
  
Residual sd: 77849.7308   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set -126815.2 147831.2 128514.4 -1.771615 1.791893 1 0.8922526  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5995239 5805786 6184692 5705495 6284983  
2022 5995239 5727312 6263166 5585479 6404999  
2023 5995239 5667096 6323382 5493388 6497090  
2024 5995239 5616332 6374146 5415751 6574727  
2025 5995239 5571608 6418870 5347352 6643126  
2026 5995239 5531175 6459303 5285514 6704964  
2027 5995239 5493993 6496485 5228649 6761829  
2028 5995239 5459384 6531094 5175720 6814758  
2029 5995239 5426879 6563599 5126007 6864471  
2030 5995239 5396135 6594343 5078988 6911490

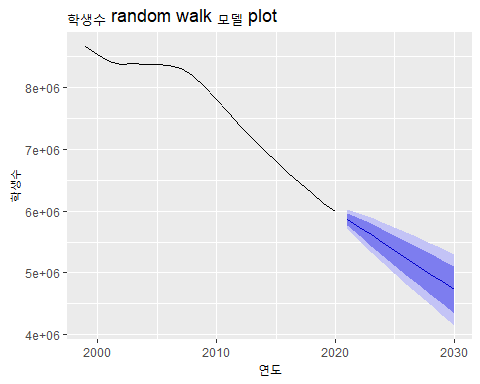
students.ts[,2] %>% rwf() %>% autoplot(main = '학생수 random walk 모델 plot', xlab = '연도', ylab = '학생수')



# 학생수계 열에 대한 드리프르가 있는 random walk 모델의 상세 정보와 plot  
students.ts[,2] %>% rwf(drift = T) %>% summary()

Forecast method: Random walk with drift  
  
Model Information:  
Call: rwf(y = ., drift = T)   
  
Drift: -126815.1905 (se 16988.204)  
Residual sd: 77849.7308   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -1.773887e-10 75973.56 63881.97 -0.06808499 0.8320398 0.4970802  
 ACF1  
Training set 0.8922526  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5868424 5768655 5968192 5715841 6021006  
2022 5741609 5597194 5886023 5520746 5962471  
2023 5614793 5433948 5795639 5338214 5891373  
2024 5487978 5274664 5701292 5161743 5814214  
2025 5361163 5117753 5604573 4988899 5733427  
2026 5234348 4962425 5506271 4818478 5650218  
2027 5107533 4808227 5406838 4649785 5565281  
2028 4980717 4654875 5306560 4482385 5479050  
2029 4853902 4502177 5205628 4315985 5391820  
2030 4727087 4349998 5104176 4150379 5303795

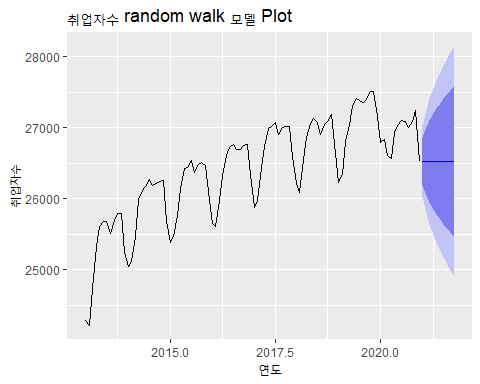
students.ts[,2] %>% rwf(drift = T) %>% autoplot(main = '학생수 random walk 모델 plot', xlab = '연도', ylab = '학생수')



# 취업자수 열에 대한 random walk 모델의 상세 정보와 plot  
employees.ts[,2] %>% rwf() %>% summary()

Forecast method: Random walk  
  
Model Information:  
Call: rwf(y = .)   
  
Residual sd: 261.7975   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set 23.56842 261.4803 194.5579 0.08782727 0.7411892 0.6138395  
 ACF1  
Training set 0.4208691  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
Jan 2021 26526 26190.90 26861.10 26013.51 27038.49  
Feb 2021 26526 26052.10 26999.90 25801.23 27250.77  
Mar 2021 26526 25945.59 27106.41 25638.34 27413.66  
Apr 2021 26526 25855.80 27196.20 25501.02 27550.98  
May 2021 26526 25776.69 27275.31 25380.03 27671.97  
Jun 2021 26526 25705.17 27346.83 25270.66 27781.34  
Jul 2021 26526 25639.41 27412.59 25170.07 27881.93  
Aug 2021 26526 25578.19 27473.81 25076.45 27975.55  
Sep 2021 26526 25520.70 27531.30 24988.52 28063.48  
Oct 2021 26526 25466.32 27585.68 24905.36 28146.64

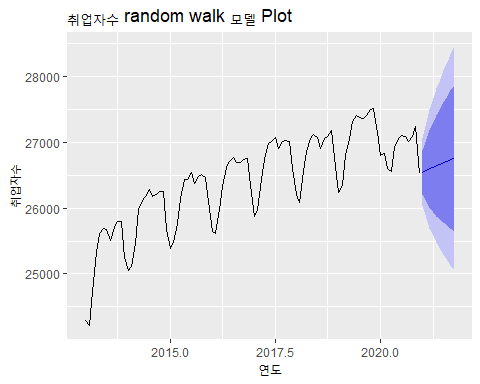
employees.ts[,2] %>% rwf() %>% autoplot(main = '취업자수 random walk 모델 Plot', xlab = '연도', ylab = '취업자수')



# 취업자수 열에 대한 드리프르가 있는 random walk 모델의 상세 정보와 plot  
employees.ts[,2] %>% rwf(drift = T) %>% summary()

Forecast method: Random walk with drift  
  
Model Information:  
Call: rwf(y = ., drift = T)   
  
Drift: 23.5684 (se 26.8599)  
Residual sd: 261.7975   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set 9.956667e-13 260.416 190.0696 -0.0013627 0.7245059 0.5996787  
 ACF1  
Training set 0.4208691  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
Jan 2021 26549.57 26214.06 26885.08 26036.45 27062.68  
Feb 2021 26573.14 26096.17 27050.11 25843.68 27302.60  
Mar 2021 26596.71 26009.50 27183.91 25698.66 27494.75  
Apr 2021 26620.27 25938.75 27301.80 25577.97 27662.58  
May 2021 26643.84 25877.99 27409.69 25472.58 27815.10  
Jun 2021 26667.41 25824.24 27510.58 25377.89 27956.93  
Jul 2021 26690.98 25775.71 27606.25 25291.19 28090.76  
Aug 2021 26714.55 25731.25 27697.84 25210.72 28218.37  
Sep 2021 26738.12 25690.07 27786.16 25135.27 28340.96  
Oct 2021 26761.68 25651.60 27871.77 25063.95 28459.41

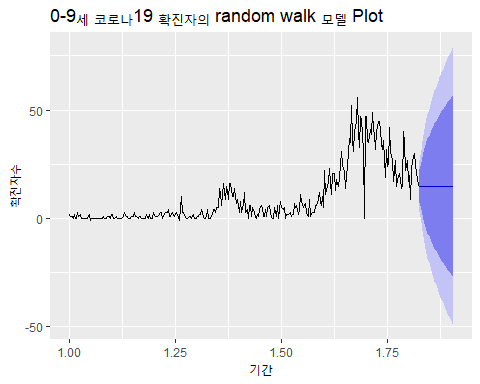
employees.ts[,2] %>% rwf(drift = T) %>% autoplot(main = '취업자수 random walk 모델 Plot', xlab = '연도', ylab = '취업자수')



# 0-9세 코로나 확진자수에 대한 계절성 Naive 모델의 상세 정보와 plot  
covid19.ts[,2] %>% rwf(30) %>% summary()

Forecast method: Random walk  
  
Model Information:  
Call: rwf(y = ., h = 30)   
  
Residual sd: 6.0029   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set 0.04318937 5.993075 3.465116 NaN Inf NaN -0.3127329  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
1.827397 15 7.3195658 22.68043 3.253790 26.74621  
1.830137 15 4.1382258 25.86177 -1.611650 31.61165  
1.832877 15 1.6970978 28.30290 -5.345033 35.34503  
1.835616 15 -0.3608683 30.36087 -8.492421 38.49242  
1.838356 15 -2.1739729 32.17397 -11.265325 41.26533  
1.841096 15 -3.8131447 33.81314 -13.772222 43.77222  
1.843836 15 -5.3205188 35.32052 -16.077552 46.07755  
1.846575 15 -6.7235483 36.72355 -18.223300 48.22330  
1.849315 15 -8.0413025 38.04130 -20.238631 50.23863  
1.852055 15 -9.2876654 39.28767 -22.144779 52.14478  
1.854795 15 -10.4731184 40.47312 -23.957773 53.95777  
1.857534 15 -11.6058044 41.60580 -25.690067 55.69007  
1.860274 15 -12.6921992 42.69220 -27.351564 57.35156  
1.863014 15 -13.7375533 43.73755 -28.950295 58.95030  
1.865753 15 -14.7461936 44.74619 -30.492877 60.49288  
1.868493 15 -15.7217367 45.72174 -31.984842 61.98484  
1.871233 15 -16.6672414 46.66724 -33.430866 63.43087  
1.873973 15 -17.5853225 47.58532 -34.834950 64.83495  
1.876712 15 -18.4782364 48.47824 -36.200544 66.20054  
1.879452 15 -19.3479458 49.34795 -37.530650 67.53065  
1.882192 15 -20.1961710 50.19617 -38.827898 68.82790  
1.884932 15 -21.0244295 51.02443 -40.094611 70.09461  
1.887671 15 -21.8340683 51.83407 -41.332846 71.33285  
1.890411 15 -22.6262895 52.62629 -42.544444 72.54444  
1.893151 15 -23.4021709 53.40217 -43.731052 73.73105  
1.895890 15 -24.1626837 54.16268 -44.894156 74.89416  
1.898630 15 -24.9087066 54.90871 -46.035100 76.03510  
1.901370 15 -25.6410376 55.64104 -47.155103 77.15510  
1.904110 15 -26.3604038 56.36040 -48.255279 78.25528  
1.906849 15 -27.0674705 57.06747 -49.336644 79.33664

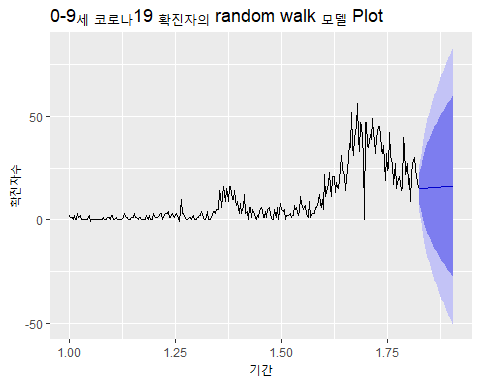
covid19.ts[,2] %>% rwf(30) %>% autoplot(main = '0-9세 코로나19 확진자의 random walk 모델 Plot', xlab = '기간', ylab = '확진자수')



# 취업자수 열에 대한 드리프르가 있는 random walk 모델의 상세 정보와 plot  
covid19.ts[,2] %>% rwf(30, drift = T) %>% summary()

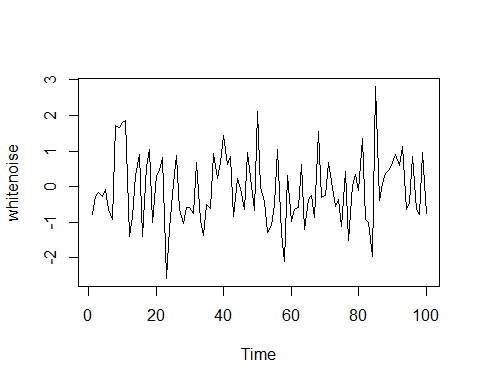
Forecast method: Random walk with drift  
  
Model Information:  
Call: rwf(y = ., h = 30, drift = T)   
  
Drift: 0.0432 (se 0.346)  
Residual sd: 6.0029   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set 1.142727e-16 5.992919 3.475017 NaN Inf NaN -0.3127329  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
1.827397 15.04319 7.350165 22.73621 3.277724 26.80866  
1.830137 15.08638 4.188742 25.98402 -1.580119 31.75288  
1.832877 15.12957 1.760664 28.49847 -5.316406 35.57554  
1.835616 15.17276 -0.289776 30.63529 -8.475147 38.82066  
1.838356 15.21595 -2.100101 32.53200 -11.266664 41.69856  
1.841096 15.25914 -3.740715 34.25899 -13.798629 44.31690  
1.843836 15.30233 -5.253365 35.85802 -16.134890 46.73954  
1.846575 15.34551 -6.665203 37.35623 -18.316974 49.00800  
1.849315 15.38870 -7.995057 38.77247 -20.373672 51.15108  
1.852055 15.43189 -9.256607 40.12039 -22.325910 53.18970  
1.854795 15.47508 -10.460165 41.41033 -24.189456 55.13962  
1.857534 15.51827 -11.613726 42.65027 -25.976539 57.01308  
1.860274 15.56146 -12.723638 43.84656 -27.696865 58.81979  
1.863014 15.60465 -13.795036 45.00434 -29.358290 60.56759  
1.865753 15.64784 -14.832146 46.12783 -30.967276 62.26296  
1.868493 15.69103 -15.838493 47.22055 -32.529214 63.91127  
1.871233 15.73422 -16.817052 48.28549 -34.048655 65.51709  
1.873973 15.77741 -17.770363 49.32518 -35.529481 67.08430  
1.876712 15.82060 -18.700612 50.34181 -36.975037 68.61623  
1.879452 15.86379 -19.609697 51.33727 -38.388226 70.11580  
1.882192 15.90698 -20.499280 52.31323 -39.771588 71.58554  
1.884932 15.95017 -21.370823 53.27115 -41.127361 73.02769  
1.887671 15.99336 -22.225620 54.21233 -42.457524 74.44424  
1.890411 16.03654 -23.064827 55.13792 -43.763843 75.83693  
1.893151 16.07973 -23.889475 56.04894 -45.047896 77.20736  
1.895890 16.12292 -24.700492 56.94634 -46.311103 78.55695  
1.898630 16.16611 -25.498718 57.83094 -47.554747 79.88697  
1.901370 16.20930 -26.284912 58.70352 -48.779990 81.19859  
1.904110 16.25249 -27.059765 59.56475 -49.987889 82.49287  
1.906849 16.29568 -27.823909 60.41527 -51.179408 83.77077

covid19.ts[,2] %>% rwf(30, drift = T) %>% autoplot(main = '0-9세 코로나19 확진자의 random walk 모델 Plot', xlab = '기간', ylab = '확진자수')

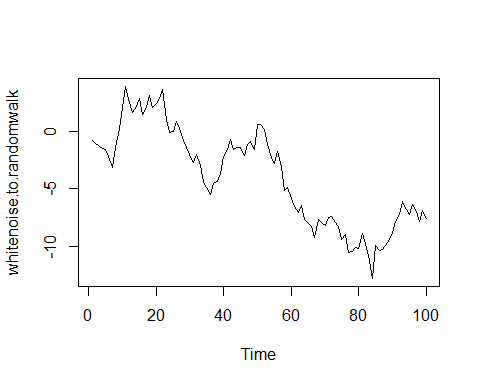


랜덤 워크는 시간적으로 하나 앞선 값과 현재값을 뺀 1차 차분값들이 백색잡음이 되어야한다는 조건이 있다. 따라서 랜덤 워크 모델은 차분을 하면 백색잡음이 되고 백색잡음은 누적합계를 하면 랜덤 워크가 된다.

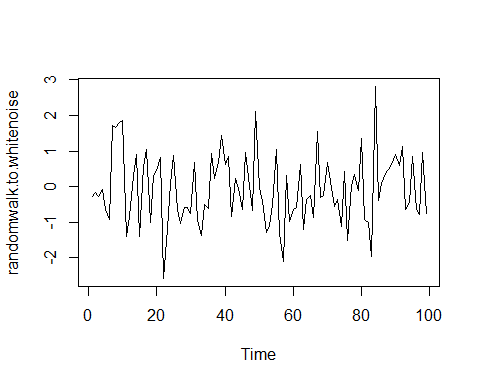
set.seed(345)  
# white noise simulation 데이터 생성  
whitenoise <- ts(rnorm(100), start = 1)   
ts.plot(whitenoise)



# white noise 데이터로 random walk 생성  
whitenoise.to.randomwalk <- cumsum(whitenoise)   
ts.plot(whitenoise.to.randomwalk)

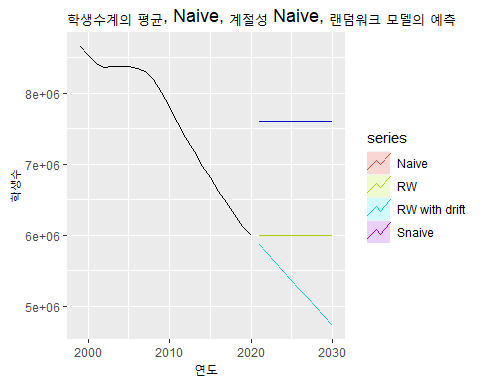


# random walk에서 white noise 생성  
randomwalk.to.whitenoise <- diff(whitenoise.to.randomwalk)   
ts.plot(randomwalk.to.whitenoise)

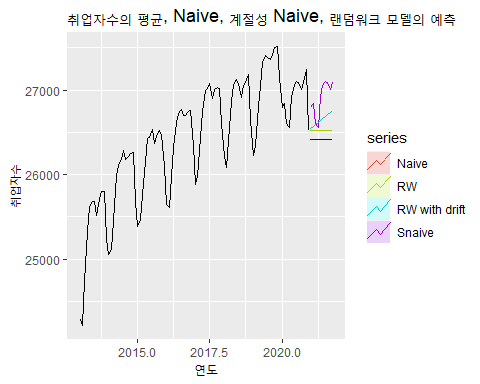


위에서 설명한 평균, Naive, 계절성 Naive, 랜덤 워크 모델들을 하나의 plot으로 그려보면 모델 간의 차이를 살펴볼 수 있다.

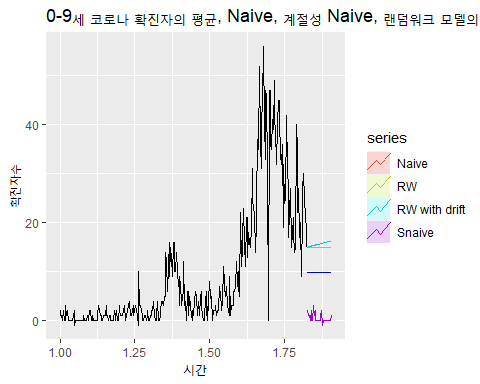
# 학생수계의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측값 plot  
autoplot(meanf(students.ts[,2], h = 10), PI = FALSE, series = 'Mean') +   
 autolayer(naive(students.ts[,2], h = 10), PI = FALSE, series = 'Naive') +  
 autolayer(snaive(students.ts[,2], h = 10), PI = FALSE, series = 'Snaive') +  
 autolayer(rwf(students.ts[,2], h = 10), PI = FALSE, series = 'RW') +  
 autolayer(rwf(students.ts[,2], h = 10, drift = TRUE), PI = FALSE, series = 'RW with drift') +   
 labs(title = '학생수계의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측', x = '연도', y = '학생수')



# 취업자수의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측값 plot  
autoplot(meanf(employees.ts[,2], h = 10), PI = FALSE, series = 'Mean') +   
 autolayer(naive(employees.ts[,2], h = 10), PI = FALSE, series = 'Naive') +  
 autolayer(snaive(employees.ts[,2], h = 10), PI = FALSE, series = 'Snaive') +  
 autolayer(rwf(employees.ts[,2], h = 10), PI = FALSE, series = 'RW') +  
 autolayer(rwf(employees.ts[,2], h = 10, drift = TRUE), PI = FALSE, series = 'RW with drift') +   
 labs(title = '취업자수의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측', x = '연도', y = '취업자수')



# 0-9세 코로나 확진자의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측값 plot  
autoplot(meanf(covid19.ts[,2], h = 30), PI = FALSE, series = 'Mean') +   
 autolayer(naive(covid19.ts[,2], h = 30), PI = FALSE, series = 'Naive') +  
 autolayer(snaive(covid19.ts[,2], h = 30), PI = FALSE, series = 'Snaive') +  
 autolayer(rwf(covid19.ts[,2], h = 30), PI = FALSE, series = 'RW') +  
 autolayer(rwf(covid19.ts[,2], h = 30, drift = TRUE), PI = FALSE, series = 'RW with drift') +   
 labs(title = '0-9세 코로나 확진자의 평균, Naive, 계절성 Naive, 랜덤워크 모델의 예측', x = '시간', y = '확진자수')



## Regression 모델

회귀 모델은 선형 회귀(Linear Regression), 비선형 회귀,(Non-linear Regression) 로지스틱 회귀(Logistic Regression) 등의 방법이 있다. 회귀 모델은 종속변수와 독립변수와의 관계를 가장 잘 나타내는 회귀방정식을 도출하여 미래 데이터에 대한 예측치를 생성하는 방법으로 아직도 머신러닝 알고리즘 중에 가장 많이 사용되는 알고리즘 중에 하나이다.[[4]](#footnote-30)

시계열 데이터의 회귀 모델은 두가지로 구분할 수 있다. 첫번째는 두 가지 이상의 시계열 데이터(Multivariate) 간의 상관 관계를 회귀 모델로 추상화하는 것과 한가지 시계열 데이터(Univariate)의 시간에 따른 회귀모델이다. 사실 첫번째 모델의 경우는 일반적 회귀모델과 큰 차이는 없지만 사용하는 함수는 시계열 패키지에서 따로 제공된다. 두번째 모델의 경우는 시계열 데이터의 특성인 추세(trend)와 계절성(season), 반복성(cycle)을 회귀에 반영한다는 점에서 일반적 회귀와는 차이가 있다.

시계열 선형 회귀 모델은 데이터간의 관계를 가장 잘 나타내는 직선(Linear)을 산출하여 미래 데이터를 예측하는 방법이다. 선형 회귀 모델은 선형 방정식으로 표현되는데 직선의 기울기(Slope)와 Y 축 절편(Intercept)의 계수(Coefficient)가 산출된다. 시계열 데이터의 선형 회귀를 위해서 forecast 패키지의 tslm()함수, timetk 패키지의 plot\_time\_series\_regression() 함수(stats::lm() 함수를 사용하여 선형회귀 결과를 plotting하는 함수) 등을 사용할 수 있다.

### forecast::tslm()

forecast 패키지에서 제공하는 tslm() 함수는 시계열 선형회귀 모델을 위한 함수이다. tslm() 함수는 lm() 함수의 래퍼 함수(Wrapper)로 사용법은 비슷하다. 시간축이 독립변수에 포함되지 않은 두개 혹은 두개 이상의 시계열 객체에 대한 선형회귀는 lm() 함수의 결과와 같다. 즉 시계열 데이터이지만 독립변수 시간이 포함되지 않는다면 시계열 데이터로써의 특성이 적용되지 않은 양 데이터간의 특성정보만으로 선형회귀 방정식이 얻어진다. 예를 들자면 앞선 학생수 예제에서 유치원 학생수와 초등학교 학생수간의 선형 회귀분석을 실시하거나 하는 경우이다.

반면 시간을 독립변수로 하여 선형회귀 모형을 적용하기 위해서는 ‘trend’, ‘season’ 키워드를 함수식의 ~ 오른쪽의 독립변수에 기재하여 시간에 대한 선형회귀 방정식을 얻을 수 있다. ‘trend’는 시계열적인 추세를 반영하여 선형회귀 모델을 만들고 ’season’은 시계열의 계절성을 반영하여 회귀모델을 만들게 되는데 두가지를 모두 고려할 때는’+’ 기호로 연결하여 사용한다. 이 과정에서 추가적인 독립변수를 추가할 수도 있는데 독립변수를 추가할 때도 ‘+’ 기호를 사용하여 회귀모델을 만들 수 있다.

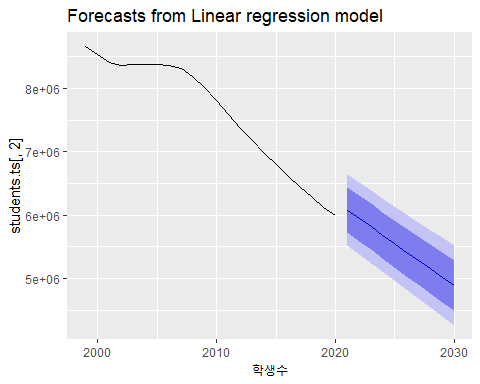
앞선 모델들과 달리 앞으로 설명할 모델들은 모델 자체에서 미래 예측치를 생성하지 않는다. 따라서 모델에 따른 미래 예측치를 산출하기 위해서는 forecast 패키지에서 제공하는 forecast()를 사용해야하고 이 결과값으로 plot을 생성해야 한다.

# 전체 학생수 예측 모델을 추세를 반영하여 생성  
student.ts.lm <- tslm(students.ts[,2] ~ trend, data = students.ts)  
summary(student.ts.lm)

Call:  
tslm(formula = students.ts[, 2] ~ trend, data = students.ts)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-327962 -206697 -48003 183951 390933   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 9118484 107928 84.49 < 2e-16 \*\*\*  
trend -132164 8218 -16.08 6.61e-13 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 244500 on 20 degrees of freedom  
Multiple R-squared: 0.9282, Adjusted R-squared: 0.9246   
F-statistic: 258.7 on 1 and 20 DF, p-value: 6.608e-13

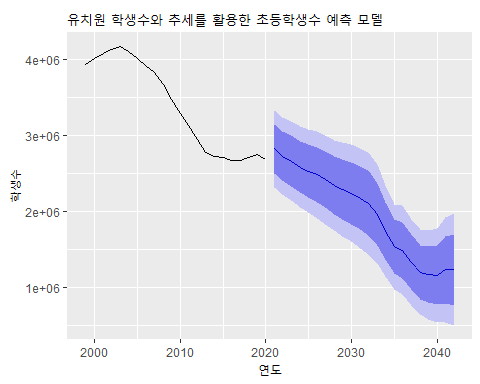
* 결과 설명
  + Residuals : 모델을 적용한 잔차의 최소(Min), 25%(1Q), 중간(Median), 75%(3Q), 최대(Max)값을 산출
  + Coefficients : y 절편(intercept)은 9,118,484이고 p value가 0.05보다 작으니 통계적으로 유의미하고, 선형 회귀 기울기(trend)는 -132164로 1년이 지날 수록 13만명정도가 계속 감소한다는 의미로 p value가 0.05보다 작으니 통계적으로 유의미
  + 이를 선형 방정식으로 표현하면 총학생수 = -132164\*년도 + 9118484 이다.
  + R-squreed : 이 선형모델은 전체 데이터의 92.8%를 설명할 수 있음.

# 전체 학생수 예측 모델을 forecast()를 사용해 예측치를 산출하고 autoplot()으로 plot 생성  
student.ts.lm %>% forecast() %>% autoplot() + labs(x = '학생수')



아래의 모델은 초등학교 학생수 예측모델을 만드는데 트랜드와 유치원 학생수를 독립변수로 하는 모델을 만드는 예이다.

# 초등학생 학생수의 예측모델을 생성하는데 유치원 학생수와 트랜드를 사용하는 선형 모델을 생성  
student.ts.lm <- tslm(students.ts[,4] ~ students.ts[,3] + trend, data = students.ts)   
# forecast()로 생성된 모델에 대한 미래 예측 데이터를 만들고 autoplot()으로 plot 생성  
student.ts.lm %>% forecast(h = 22) %>% autoplot(main = '유치원 학생수와 추세를 활용한 초등학생수 예측 모델', xlab = '연도', ylab = '학생수')



위의 예제에서 사용한 데이터는 연도별 학생수의 합계이다. 이 데이터와 같은 연도별 데이터는 계절성이 존재하지 않기 때문에 season 키워드를 사용하면 아래와 같이 에러를 낸다.

student.ts.lm <- tslm(students.ts[,2] ~ trend + season, data = students.ts)

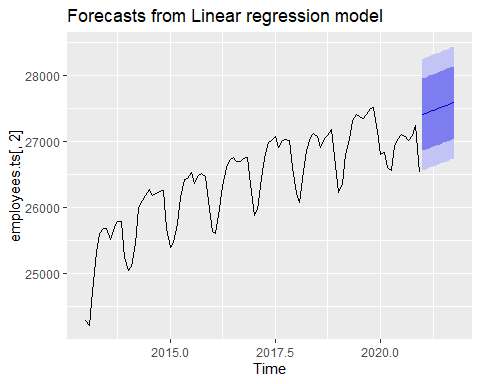
Error in tslm(students.ts[, 2] ~ trend + season, data = students.ts): Non-seasonal data cannot be modelled using a seasonal factor

전체 취업자수와 교육서비스업 취업자수에 대한 선형회귀분석 모델과 plot은 다음과 같다.

# 전체 취업자수를 추세(trend)만으로 선형 회귀분석  
employee.total.ts.lm <- tslm(employees.ts[,2] ~ trend, data = employees.ts)  
# y절편이 25430, 기울기가 20.39인 선형 회귀모델 생성  
summary(employee.total.ts.lm)

Call:  
tslm(formula = employees.ts[, 2] ~ trend, data = employees.ts)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-1256.4 -264.1 119.9 350.6 525.7   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 25430.605 85.545 297.28 <2e-16 \*\*\*  
trend 20.394 1.531 13.32 <2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 415.8 on 94 degrees of freedom  
Multiple R-squared: 0.6536, Adjusted R-squared: 0.6499   
F-statistic: 177.3 on 1 and 94 DF, p-value: < 2.2e-16

# 전체 취업자수에 대한 선형회귀 모델의 예측치에 대한 plot 생성  
employee.total.ts.lm %>% forecast() %>% autoplot()

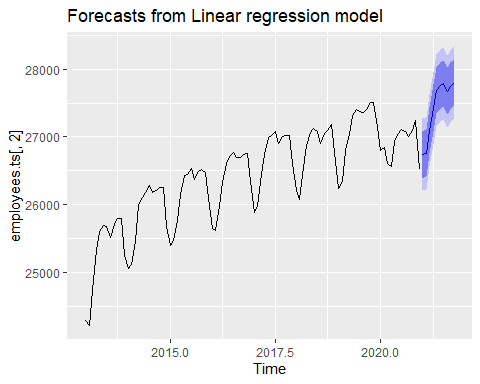


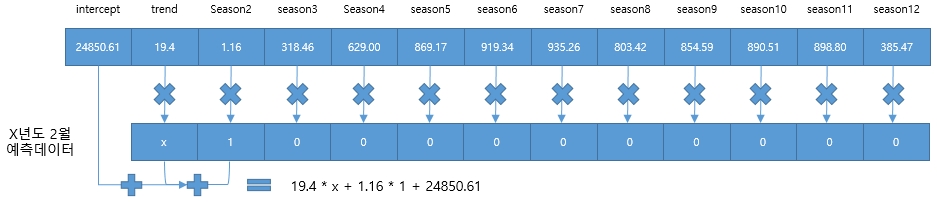
아래의 예는 시계열 선형 회귀 모델에 추세와 계절성까지 반영한 모델을 사용한 경우이다. 계절성을 포함시키기 위해서는 tslm()의 ~의 오른쪽 수식에 season 키워드를 넣어줌으로써 계절성을 반영한 시계열 모델을 생성할 수 있다. 계절성 선형 모델을 summary()를 통해 상세 정보를 보면 앞서 봤던 추세만으로 생성한 모델과는 조금 다른 점이 있다. season2부터 season12까지의 변수가 추가되는데 이를 더미변수(dummy variable)이라고 한다. 2월부터 12월까지에 각각 대응되는 변수인데 이 변수는 1과 0의 값만 가질수 있다. 예를 들어 2월의 데이터를 예측하기 위해서는 season2의 값만 1이고 나머지 season3부터 season12까지는 0으로 설정된다. 따라서 2월의 회귀계수만이 영향을 미치게 되므로 계수값은 많이 나오지만 각각의 예측치를 만들때에는 season2부터 season12중에 하나의 계수값만 더해지게 된다.

# 전체 취업자수를 추세(trend)와 계절성(season)으로 선형 회귀분석  
employee.total.ts.lm <- tslm(employees.ts[,2] ~ trend + season, data = employees.ts)  
summary(employee.total.ts.lm)

Call:  
tslm(formula = employees.ts[, 2] ~ trend + season, data = employees.ts)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-675.70 -42.60 76.66 147.08 312.95   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 24850.616 96.043 258.744 < 2e-16 \*\*\*  
trend 19.457 0.919 21.171 < 2e-16 \*\*\*  
season2 1.168 123.798 0.009 0.99249   
season3 318.462 123.808 2.572 0.01189 \*   
season4 629.005 123.825 5.080 2.29e-06 \*\*\*  
season5 869.174 123.849 7.018 5.63e-10 \*\*\*  
season6 919.342 123.880 7.421 9.13e-11 \*\*\*  
season7 935.260 123.918 7.547 5.14e-11 \*\*\*  
season8 803.429 123.962 6.481 6.13e-09 \*\*\*  
season9 854.597 124.013 6.891 9.94e-10 \*\*\*  
season10 890.516 124.071 7.177 2.75e-10 \*\*\*  
season11 898.809 124.135 7.241 2.07e-10 \*\*\*  
season12 385.477 124.207 3.104 0.00261 \*\*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 247.6 on 83 degrees of freedom  
Multiple R-squared: 0.8915, Adjusted R-squared: 0.8759   
F-statistic: 56.86 on 12 and 83 DF, p-value: < 2.2e-16

employee.total.ts.lm %>% forecast() %>% autoplot()

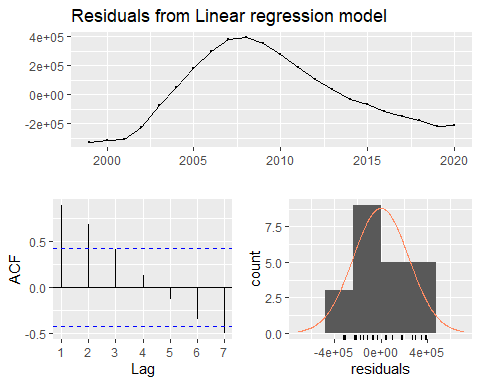




선형 회귀분석을 시행할 때 주의해야 할 점은 회귀모델이 적용 결과의 잔차가 백색 잡음이어야 한다는 점이다. 잔차의 자가회귀성이 존재하는 경우는 시계열적 특성을 여전히 지니고 있기 때문에 이를 제거할 필요가 있다. 다만 이 예측은 ‘잘못된’ 것은 아니나 예측구간이 커지기 때문에 비효율적 예측 모델이 된다.[[5]](#footnote-32) 또한 예측에 적용해야할 시계열적 특성이 남아있다는 것을 잔차에서 확인할 수 있다.

아래의 예에서 보면 전체 학생수를 추세에 의해 선형 회귀분석을 시행한 경우 잔차는 자기 상관성을 지니고 있음을 볼 수 있다. 이는 forecast 패키지의 checkresiduals() 함수를 사용하여 확인하는데 plot만 봐도 백색잡음이 아님을 확인할 수 있지만 백색잡음 테스트인 Breusch-Godfrey 테스트 결과(checkresiduals() 함수는 회귀모델에 대해서는 Breusch-Godfrey 테스트를, 나머지는 Ljung-Box 테스트를 시행한다.)의 p-value가 0.05보다 작기 때문에 자기상관성이 존재하여 백색잡음으로 볼 수 없다.

checkresiduals(tslm(students.ts[,2] ~ trend, data = students.ts))



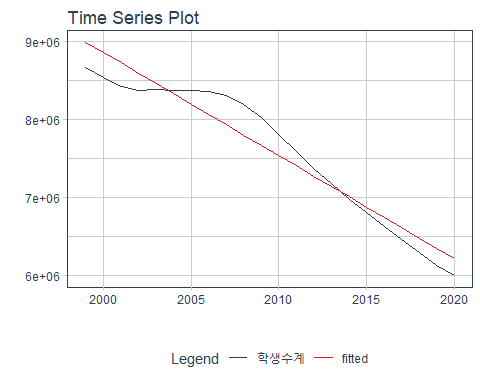
Breusch-Godfrey test for serial correlation of order up to 5  
  
data: Residuals from Linear regression model  
LM test = 19.584, df = 5, p-value = 0.001496

### timetk::plot\_time\_series\_regression

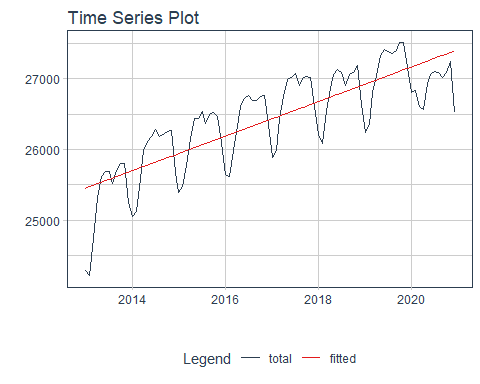
timetk 패키지는 시계열 데이터를 핸들링하고 plotting 하는 데 주로 활용하는 패키지이다. 그래서 모델링을 위한 함수를 바로 제공하지는 않고 plotting 함수에서 회귀 모델을 호출하여 회귀 결과를 plotting 하는 함수를 제공한다. plot\_time\_series\_regression() 함수에서 사용하는 선형회귀 함수에서도 trend, season 을 적용할 수 없다. 다만 ts 객체가 아닌 data.frame 객체를 사용할 수있다는 장점이 있다.

library(timetk)  
library(lubridate)  
plot\_time\_series\_regression(.data = students,   
 .date\_var = 연도,   
 .formula = 학생수계 ~ 연도,   
 .interactive = FALSE,   
 .show\_summary = TRUE)

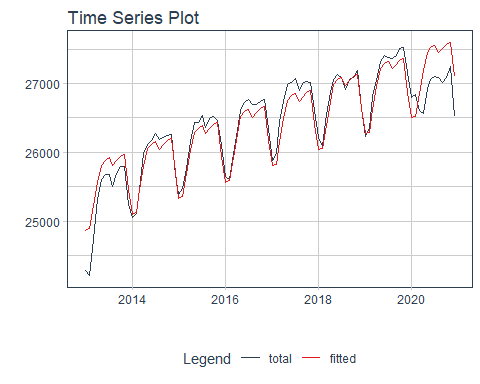
Call:  
stats::lm(formula = .formula, data = .data)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-328001 -206778 -47994 183960 390806   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 12819016.6 328759.6 38.99 < 2e-16 \*\*\*  
연도 -361.8 22.5 -16.08 6.61e-13 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 244500 on 20 degrees of freedom  
Multiple R-squared: 0.9282, Adjusted R-squared: 0.9246   
F-statistic: 258.7 on 1 and 20 DF, p-value: 6.612e-13



# 계절성 반영을 위해   
employees$date <- as.yearmon(employees$time, "%Y. %m")  
# plot\_time\_series\_regression에 trend만 반영시  
plot\_time\_series\_regression(.data = employees,   
 .date\_var = time,   
 .formula = total ~ as.numeric(yearmonth(date)),   
 .interactive = FALSE)



# plot\_time\_series\_regression에 trend, season(월)까지 반영시  
plot\_time\_series\_regression(.data = employees,   
 .date\_var = time,   
 .formula = total ~ year(date) +  
 month(date, label = TRUE),   
 .interactive = FALSE)



## 지수 평활(Exponential Smoothing) 모델

지수 평활 모델은 1950년대에 제안된 모델로 랜덤 워크(Random Walk) 모델과 같이 시계열 데이터는 최근의 값이 현재의 값에 가장 영향을 미친다는 점에 착안하여 나온 모델이다. 다만 랜덤 워크 모델과 같이 마지막 값에 모든 가중치를 둬서 일정하게 유지하는 것이 아닌 현재와 가까운 과거일수록 더 많은 가중치를 주는 방식으로 예측하는 방법이다. [[6]](#footnote-37)

최근의 데이터에 가중치를 높게 주기 때문에 추세, 계절성, 순환성이 심하지 않은 단기 데이터의 모델링에 적합한 방법이다. 추세나 계절성이 없는 데이터에 적합한 ’단순 지수평활 모델(Simple Exponential Smoothign), 추세가 있는 데이터에 적합한 홀트(Holt) 모델, 추세와 계절성이 있는 데이터에 적합한 홀트-윈터(Holt-Winter) 모델 등이 있다.

지수 평활 모델에서 핵심적인 변수는 평활 계수이다. 평활 매개변수는 앞서 설명한 현재와 가까운 과거에 할당하는 가중치를 의미한다. 평활 매개변수은 보통 0에서 1사이의 변수인데 홀트 모델이나 홀트-윈터 모델에서는 추가적인 계수가 추가될 수 있다. 이고 이 가중치를 어떻게 설정하는 가에 따라 예측 모델의 성능도 달라질 수 있다.

필자는 처음 이 모델을 접했을때 왜 지수(Exponential)과 평활(Smoothing) 이라는 이름을 사용했는지 궁금했다. 평활 매개변수를 설명하는 과정에서 명칭에서 지수를 붙인 이유는 설명이 되겠지만 아직도 평활이라는 이름을 붙인 이유는 명쾌하게 이해되지는 않는다. 시계열 데이터에 대한 추세선을 부드럽게, 스무딩하게 만드는 모델이라고 생각이 되나 사실 홀트-윈터 모델을 보면 별로 부드럽지 않다는 느낌이 들어서 이 부분에 대해 잘 아시는 분은 알려주시길 부탁드린다.

### 단순 지수 평활 모델(Simple Exponentail Smoothing Model)

단순 지수 평활 모델은 추세, 계절성 등의 시계열적 특성이 비교적 약한 데이터에 적합한 예측 모델이다. 단순 지수 평활 모델을 사용하기 위해서는 우선 평활 계수를 설정해아한다.

평활 계수는 과거치에 대한 가중치로 현재에 가장 가까운 첫번째 과거 데이터에 대한 가중치이다. 이후 가중치 들은 1에서 가중치를 뺀 가중치(1-평활계수)로 설정하고 과거로 계속 갈수록 (1-평활계수)를 계속 곱해서 가중치를 할당한다. 평활 계수를 0.5로 가정하고 학생수 예에 적용해 보면 다음의 표와 같이 가중치가 설정되게 된다. 평활 계수가 지수형태로 계산되기 때문에 평활계수의 합은 1이 될 수 없다.

|  |  |
| --- | --- |
| 연도 | 가중치(평활계수 = 0.5) |
| 2020 | 0.5 |
| 2019 | 0.25 = (1 - 0.5) |
| 2018 | 0.125 = (1 - 0.5) \* (1 - 0.5) |
| 2017 | 0.0625 = (1 - 0.5) \* (1 - 0.5) \* (1 - 0.5) |
| … | … |

#### forecast::ses() 함수

단순 지수 평활 모델을 적용하는 방법은 forecast 패키지의 ses() 함수를 사용하면 모델을 구축할 수 있다. ses() 함수에서는 평활 계수를 alpha 매개변수를 통해 설정할 수 있지만 설정하지 않으면 자동으로 계산하여 설정해준다. ses() 함수는 뒤에서 설명하는 홀트 모델, 홀트-윈터 모델도 beta와 gamma 매개변수를 통해 생성할 수 있다.

ses() 함수를 통해 생성되는 모델은 몇가지 특성값을 가지는데 앞에서 설명한 평활계수가 alpha로 표현되고 초기상태값인 l 값이 나타난다. 단순 지수 평활 모델은 한단계씩 예측해나가는 방법(One Step Forecast)이기 때문에 초기값인 l값부터 시작하여 한단계 앞 값을 예측하고 또 다음 단계를 예측하는 방법으로 수행된다. l 값을 산출하는 방법은 Rob Hyndman의 저서에서 확인할 수 있다.[[7]](#footnote-39) 이는 ses() 모델을 summary() 함수를 통해 실행시키면 확인이 가능하다.

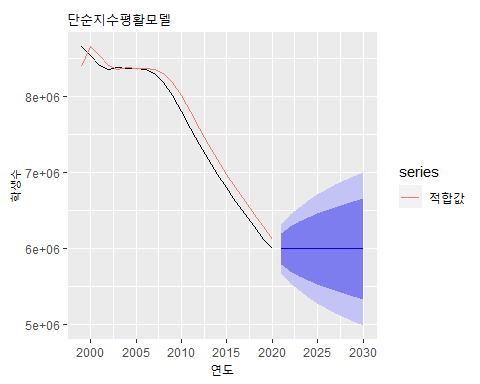
아래의 총학생수에 대한 ses()의 alpha 계수는 0.999로 계산되었다. 이는 거의 1과 가깝기 때문에 바로 이전 데이터에 거의 모든 가중치를 다 준 것이고 이로 인해 결국 Naive 모델과 거의 같은 모델이 생성되었다. 만약 alpha값을 강제로 낮추어 준다면 어떻게 될까? 아래의 코드를 잘 살펴보길 바란다.

# 전체 학생수에 대한 Simple Exponential Smoothing  
ses(students.ts[,2]) %>% summary()

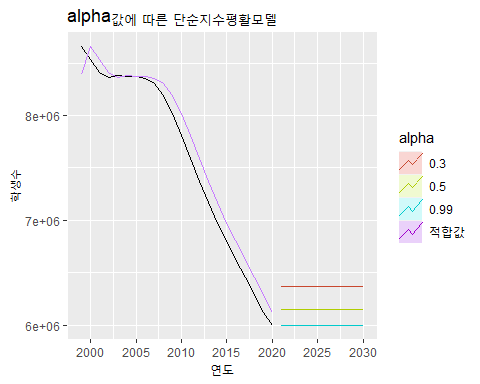
Forecast method: Simple exponential smoothing  
  
Model Information:  
Simple exponential smoothing   
  
Call:  
 ses(y = students.ts[, 2])   
  
 Smoothing parameters:  
 alpha = 0.9999   
  
 Initial states:  
 l = 8394600.8229   
  
 sigma: 162570.5  
  
 AIC AICc BIC   
599.8562 601.1896 603.1294   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE ACF1  
Training set -109072.2 155004.8 134671.7 -1.552769 1.849053 1.047911 0.4105529  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5995252 5786909 6203594 5676619 6313884  
2022 5995252 5700626 6289878 5544660 6445843  
2023 5995252 5634416 6356087 5443401 6547102  
2024 5995252 5578598 6411905 5358035 6632468  
2025 5995252 5529421 6461082 5282825 6707678  
2026 5995252 5484962 6505542 5214830 6775673  
2027 5995252 5444077 6546427 5152302 6838201  
2028 5995252 5406022 6584481 5094103 6896401  
2029 5995252 5370280 6620223 5039440 6951063  
2030 5995252 5336474 6654029 4987739 7002765

* 결과 설명
  + Smoothing parameters: 지수평활의 가중치로 사용할 alpha값이 0.9999
  + Initial states : 지수평활의 시작점 위치

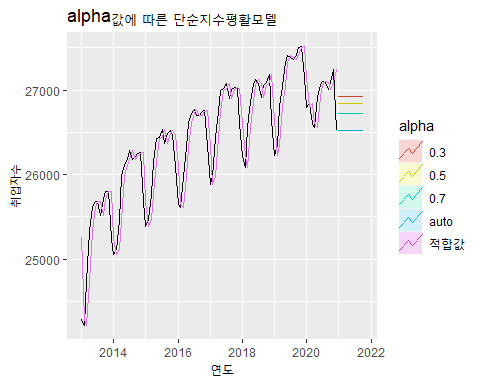
autoplot(students.ts[,2]) +   
 autolayer(fitted(ses(students.ts[,2])), series = '적합값') +  
 autolayer(ses(students.ts[,2])) +   
 labs(title = '단순지수평활모델', x = '연도', y = '학생수')



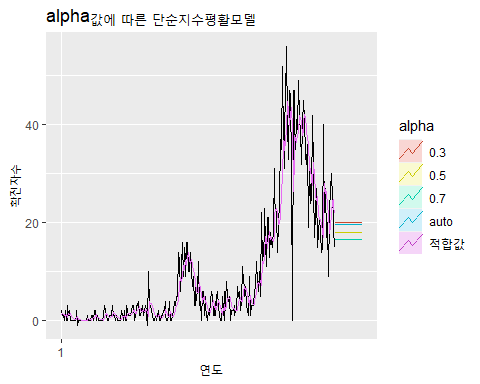
autoplot(students.ts[,2]) +   
 autolayer(fitted(ses(students.ts[,2])), series = '적합값') +  
 autolayer(ses(students.ts[,2]), PI = F, series = '0.99') +   
 autolayer(ses(students.ts[,2], alpha = 0.5), PI = F, series = '0.5') +   
 autolayer(ses(students.ts[,2], alpha = 0.3), PI = F, series = '0.3') +   
 labs(title = 'alpha값에 따른 단순지수평활모델', x = '연도', y = '학생수', color = 'alpha')



# 전체 취업자수에 대한 Simple Exponential Smoothing  
autoplot(employees.ts[,2]) +   
 autolayer(fitted(ses(employees.ts[,2])), series = '적합값') +  
 autolayer(ses(employees.ts[,2]), series = 'auto', PI = F) +   
 autolayer(ses(employees.ts[,2], alpha = 0.3), series = '0.3', PI = F) +   
 autolayer(ses(employees.ts[,2], alpha = 0.5), series = '0.5', PI = F) +   
 autolayer(ses(employees.ts[,2], alpha = 0.7), series = '0.7', PI = F) +   
 labs(title = 'alpha값에 따른 단순지수평활모델', x = '연도', y = '취업자수', color = 'alpha')

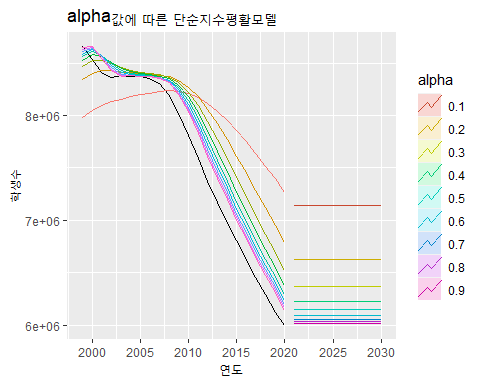


# 코로나 신규확진자수(0-9세)에 대한 Simple Exponential Smoothing  
autoplot(covid19.ts[,2]) +   
 autolayer(fitted(ses(covid19.ts[,2])), series = '적합값') +  
 autolayer(ses(covid19.ts[,2], h = 30), series = 'auto', PI = F) +   
 autolayer(ses(covid19.ts[,2], alpha = 0.3, h = 30), series = '0.3', PI = F) +   
 autolayer(ses(covid19.ts[,2], alpha = 0.5, h = 30), series = '0.5', PI = F) +   
 autolayer(ses(covid19.ts[,2], alpha = 0.7, h = 30), series = '0.7', PI = F) +   
 labs(title = 'alpha값에 따른 단순지수평활모델', x = '연도', y = '확진자수', color = 'alpha')



평활 계수가 클수록 데이터 변화에 빠르게 반응하여 예측의 감응도가 높지만 평활계수가 작으면 데이터의 변화에 느리게 반응하여 예측의 안정성이 높아진다 [[8]](#footnote-41)

# 전체 학생수의 alpha 값에 따른 적합치와 예측치의 변화  
autoplot(students.ts[,2], color = 'black') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.1)), series = '0.1') +  
 autolayer(ses(students.ts[,2], alpha = 0.1, PI = FALSE), series = '0.1') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.2)), series = '0.2') +   
 autolayer(ses(students.ts[,2], alpha = 0.2, PI = FALSE), series = '0.2') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.3)), series = '0.3') +  
 autolayer(ses(students.ts[,2], alpha = 0.3, PI = FALSE), series = '0.3') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.4)), series = '0.4') +  
 autolayer(ses(students.ts[,2], alpha = 0.4, PI = FALSE), series = '0.4') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.5)), series = '0.5') +  
 autolayer(ses(students.ts[,2], alpha = 0.5, PI = FALSE), series = '0.5') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.6)), series = '0.6') +  
 autolayer(ses(students.ts[,2], alpha = 0.6, PI = FALSE), series = '0.6') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.7)), series = '0.7') +  
 autolayer(ses(students.ts[,2], alpha = 0.7, PI = FALSE), series = '0.7') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.8)), series = '0.8') +  
 autolayer(ses(students.ts[,2], alpha = 0.8, PI = FALSE), series = '0.8') +  
 autolayer(fitted(ses(students.ts[,2], alpha = 0.9)), series = '0.9') +  
 autolayer(ses(students.ts[,2], alpha = 0.9, PI = FALSE), series = '0.9') +   
 labs(title = 'alpha값에 따른 단순지수평활모델', x = '연도', y = '학생수', color = 'alpha')



### 홀트(Holt) 모델

홀트(holt) 모델은 지수 평활 모델을 확장하여 추세를 반영하는 모델이다. 홀트 모델은 추세조정 지수평활법이나 이중 지수 평활법으로 불리기도 한다. 추세를 반영하기 위해 평활 계수인 alpha와 추세 기울기 값 beta를 사용하여 모델을 세운다. 평활 계수 값과 마찬가지로 추세 기울기 값도 0과 1사이의 값을 지니고 이 값을 반영한 이동평균을 통해 예측값을 산출한다.

#### forecast::holt()

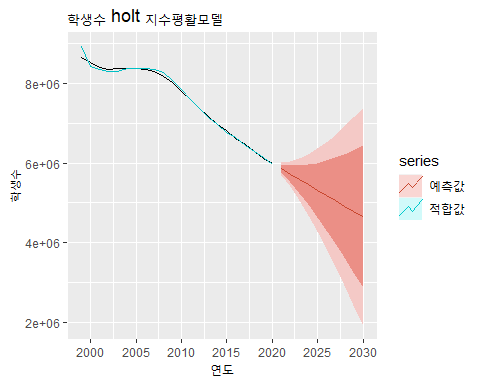
홀트 모델은 앞선 단순 지수 평활 모델에 사용했던 ses() 함수에 추세 기울기 값인 beta 값을 추가하여 사용할 수도 있고 forecast 패키지에서 제공하는 holt() 함수를 사용하여 모델을 생성할 수도 있다. 다만 alpha, beta 등의 매개변수를 지정하지 않으면 자동 계산되어 적절한 값이 사용된다.

# 전체 학생수에 대한 Holt modeling  
summary(holt(students.ts[,2]))

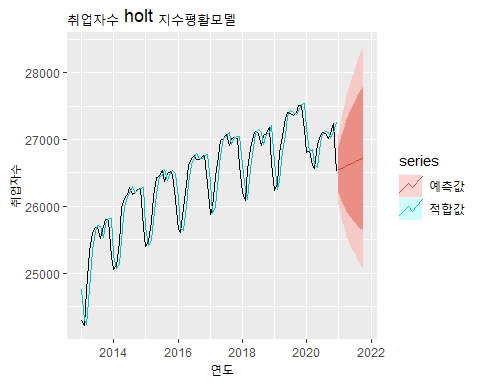
Forecast method: Holt's method  
  
Model Information:  
Holt's method   
  
Call:  
 holt(y = students.ts[, 2])   
  
 Smoothing parameters:  
 alpha = 0.8654   
 beta = 0.8654   
  
 Initial states:  
 l = 8966192.6427   
 b = -36934.0002   
  
 sigma: 82114.94  
  
 AIC AICc BIC   
571.4867 575.2367 576.9419   
  
Error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -5106.807 74275.76 46971.89 -0.0472354 0.5836661 0.365499  
 ACF1  
Training set -0.00685211  
  
Forecasts:  
 Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
2021 5855531 5750297 5960766 5694589 6016474  
2022 5721369 5511013 5931725 5399658 6043081  
2023 5587207 5242397 5932018 5059866 6114549  
2024 5453045 4951452 5954638 4685925 6220166  
2025 5318883 4641431 5996336 4282810 6354957  
2026 5184721 4314366 6055077 3853627 6515816  
2027 5050559 3971709 6129409 3400601 6700518  
2028 4916397 3614582 6218212 2925443 6907352  
2029 4782235 3243885 6320586 2429531 7134940  
2030 4648073 2860364 6435783 1914007 7382139

* 결과 설명
  + Smoothing parameter : 예측 수준(level)을 결정하는 alpha 평활지수는 0.8654이고 추세를 나타내는 beta 평활지수는 0.8654로 계산됨
  + Initial states : 예측수준(level)의 초기치는 8966192이고 추세의 초기치는 -36934로 계산됨

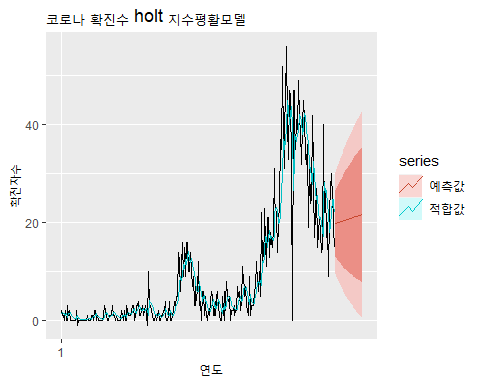
# 전체 학생수에 대한 Holt modeling  
autoplot(students.ts[,2]) +   
 autolayer(fitted(holt(students.ts[,2])), series = '적합값') +  
 autolayer(holt(students.ts[,2]), series = '예측값') +   
 labs(title = '학생수 holt 지수평활모델', x = '연도', y = '학생수')



# 전체 취업자수에 대한 Holt modeling  
autoplot(employees.ts[,2]) +   
 autolayer(fitted(holt(employees.ts[,2])), series = '적합값') +  
 autolayer(holt(employees.ts[,2]), series = '예측값') +   
 labs(title = '취업자수 holt 지수평활모델', x = '연도', y = '취업자수')

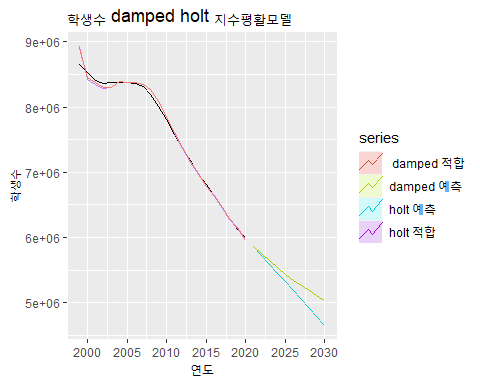


# 코로나 신규확진자수(0-9세)에 대한 Holt modeling  
autoplot(covid19.ts[,2]) +   
 autolayer(fitted(holt(covid19.ts[,2])), series = '적합값') +  
 autolayer(holt(covid19.ts[,2], h = 30), series = '예측값') +   
 labs(title = '코로나 확진수 holt 지수평활모델', x = '연도', y = '확진자수')

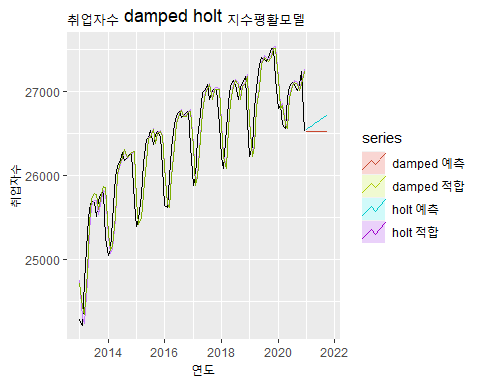


홀트 모델은 위와 같이 일정한 증가, 감소가 나타난다. 추세가 증가 추세이면 예측값이 계속 증가하게 되고 감소추세이면 계속 감소하게 된다. 특히 감소추세 일때 계속 감소하면 어느 순간 음수값을 가질수도 있다. 따라서 감소 추세가 있는 경우 어느정도에서 감소추세를 지연시킬 필요가 있다. holt() 함수에서는 damped 매개변수를 통해 감소추세를 지연시킬 수 있다.

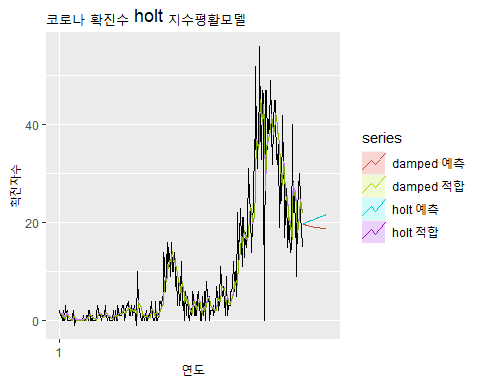
# 전체 학생수에 대한 Holt modeling 비교  
autoplot(students.ts[,2]) +   
 autolayer(fitted(holt(students.ts[,2])), series = 'holt 적합') +  
 autolayer(fitted(holt(students.ts[,2], damped = TRUE)), series = ' damped 적합') +  
 autolayer(holt(students.ts[,2]), series = 'holt 예측', PI = FALSE) +   
 autolayer(holt(students.ts[,2], damped = TRUE), series = 'damped 예측', PI = FALSE) +   
 labs(title = '학생수 damped holt 지수평활모델', x = '연도', y = '학생수')



# 전체 취업자수에 대한 Holt modeling 비교  
autoplot(employees.ts[,2]) +   
 autolayer(fitted(holt(employees.ts[,2])), series = 'holt 적합') +  
 autolayer(fitted(holt(employees.ts[,2], damped = TRUE)), series = 'damped 적합') +  
 autolayer(holt(employees.ts[,2]), series = 'holt 예측', PI = FALSE) +   
 autolayer(holt(employees.ts[,2], damped = TRUE), series = 'damped 예측', PI = FALSE) +   
 labs(title = '취업자수 damped holt 지수평활모델', x = '연도', y = '취업자수')

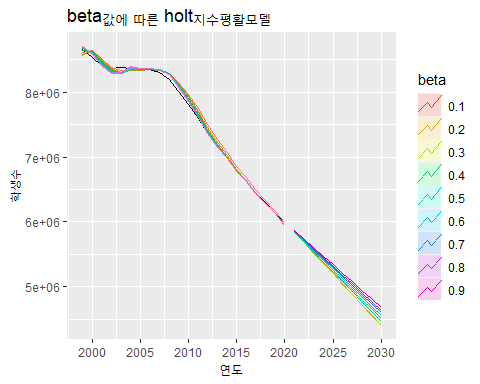


# 코로나 신규확진자수(0-9세)에 대한 Holt modeling 비교  
autoplot(covid19.ts[,2]) +   
 autolayer(fitted(holt(covid19.ts[,2])), series = 'holt 적합') +  
 autolayer(fitted(holt(covid19.ts[,2], damped = TRUE)), series = 'damped 적합') +  
 autolayer(holt(covid19.ts[,2], h = 30), series = 'holt 예측', PI = FALSE) +   
 autolayer(holt(covid19.ts[,2], h = 30, damped = TRUE), series = 'damped 예측', PI = FALSE) +   
 labs(title = '코로나 확진수 holt 지수평활모델', x = '연도', y = '확진자수')



추세 기울기 값에 따른 변동은 아래 plot과 같이 나타난다. 사실상 추세 기울기 값은 결과값에 큰 차이를 나타내지 않는다.

autoplot(students.ts[,2]) +   
 autolayer(fitted(holt(students.ts[,2], beta = 0.1)), series = '0.1') +   
 autolayer(holt(students.ts[,2], beta = 0.1, PI = F), series = '0.1') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.2)), series = '0.2')+   
 autolayer(holt(students.ts[,2], beta = 0.2, PI = F), series = '0.2') +   
 autolayer(fitted(holt(students.ts[,2], beta = 0.3)), series = '0.3')+   
 autolayer(holt(students.ts[,2], beta = 0.3, PI = F), series = '0.3') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.4)), series = '0.4')+   
 autolayer(holt(students.ts[,2], beta = 0.4, PI = F), series = '0.4') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.5)), series = '0.5')+   
 autolayer(holt(students.ts[,2], beta = 0.5, PI = F), series = '0.5') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.6)), series = '0.6')+   
 autolayer(holt(students.ts[,2], beta = 0.6, PI = F), series = '0.6') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.7)), series = '0.7')+   
 autolayer(holt(students.ts[,2], beta = 0.7, PI = F), series = '0.7') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.8)), series = '0.8')+   
 autolayer(holt(students.ts[,2], beta = 0.8, PI = F), series = '0.8') +  
 autolayer(fitted(holt(students.ts[,2], beta = 0.9)), series = '0.9')+   
 autolayer(holt(students.ts[,2], beta = 0.9, PI = F), series = '0.9') +   
 labs(title = 'beta값에 따른 holt지수평활모델', x = '연도', y = '학생수', color = 'beta')



### 홀트-윈터(Holt-Winter) 모델

홀트-윈터(Holt-Winter) 모델은 단순 지수 평활 모델을 확장하여 추세와 계절성을 반영하는 모델이다. 홀트-윈터 모델은 ses 모델의 alpha, 홀트 모델의 beta에 계절 매개변수인 gamma까지 포함하기 때문에 삼중 지수 평활법이나 계절조정 지수 평활법이라고도 한다.

홀트-윈터 모델은 덧셈 방법(additive)와 곱셈 방법(multiplicative)의 두가지 방법이 있다.

앞서 시계열 분해에서 설명했듯히 덧셈 방법은 계절성의 변화 비교적 일정하게 나타날때 사용하는 방법이고 곱셈 방법은 계절성의 진폭이 추세에 비례하여 변동이 있을때 사용하는 방법이다. 따라서 미래 예측치를 산출할 때도 덧셈 방법은 예측치가 크게 변동되지 않지만 곱셈 방법은 먼 미래일 수록 예측값이 커지게 된다.

#### forecast::hw() 함수

홀트-윈터 모델을 생성하기 위해서는 forecast 패키지의 hw()를 사용한다. 홀트윈터모델도 앞선 홀트 모델과 같이 ses() 함수에 추세 기울기 값인 beta 값과 계절 변수인 gamma 값을 추가하여 사용할 수도 있고 hw() 함수를 사용하여 모델을 생성할 수도 있다. 다만 alpha, beta 등의 매개변수를 지정하지 않으면 자동 계산되어 적절한 값이 사용된다.

덧셈 방법과 곱셈 방법을 사용하는 방법은 hw() 함수에서 seasonal 매개변수를 통해 설정할 수 있다. seasonal 매개변수를 ‘additive’로 설정하면 덧셈 방법,’multiplicative’를 설정하면 곱셈 방법을 사용하여 모델이 구축된다.

사실상 뎃셈 방법과 곱셈 방법 중 어느 방법을 선택해야하는 지를 선택하는 방법은 시각적으로 확인하거나 다음 장에서 설명할 모델 성능 비교 방법을 통해 선택해야한다.

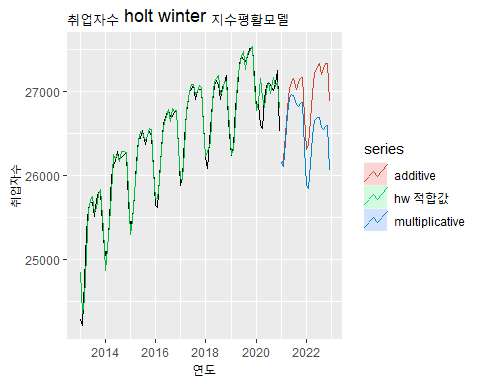
연간 데이터인 전체 학생수는 계절성을 찾을수 없기 때문에 hw()를 적용하면 다음과 같은 오류를 발생시킨다.

hw(students.ts[,2])

Error in hw(students.ts[, 2]): The time series should have frequency greater than 1.

아래는 총 취업자수에 대한 홀트윈터 모델 plot이다. 덧셈방법과 곱셈방법의 두가지 plot을 보여주고 있다.

autoplot(employees.ts[,2]) +   
 autolayer(fitted(hw(employees.ts[,2])), series = 'hw 적합값') +  
 autolayer(hw(employees.ts[,2], seasonal = 'additive'), PI = FALSE, series = 'additive') +   
 autolayer(hw(employees.ts[,2], seasonal = 'multiplicative'), PI = FALSE, series = 'multiplicative') +   
 labs(title = '취업자수 holt winter 지수평활모델', x = '연도', y = '취업자수')



### ETS 모델

ETS 모델은 Error, Trend, Season의 앞글자만 따서 만들었다고도 하고 ExponenTial Smoothing의 약자라고도 한다. 앞에 설명한 지수 평활 모델을 잔차(실제값 - 적합값)로 보정한 모델이다.

ETS 모델은 모델을 구성하는 방법에 따라 여러가지 모델로 구분될 수 있다. 각 모델은 데이터의 측정식(measurement equation)과 측정식에서 추출된 추세, 계절성 등을 통해 추정되는 미래 예측값에 대한 상태식(state equation)에 따라 결정되는데 이를 상태공간모델(status space model)이라고 한다.[[9]](#footnote-49)

상태공간모델에 의한 ETS 모델은 Error에 의한 덧셈 보정(A), 곱셈 보정(M), 추세에 따른 덧셈 방법(A), 감쇄 덧셈 방법(Ad), 추세 없음(N), 계절성에 따른 계절성 없음(N), 덧셈 계절성(A), 곱셈 계절성(M)으로 구분된다. 이 8가지 방법을 활용하여 최종 ETS 모델을 결정할 수 있는데 모든 조합이 모델이 되지는 않는다. 다음은 조합이 가능한 모든 상태공간모델이다. 이 중 일부는 덧셈 혹은 곱셈 보정에 따라 앞서 설명한 단순지수평활모델, 홀트모델, damped 홀트모델, 홀트윈터모델과 동일한 모델도 있다.

* 보정방법 : 덧셈 보정(A)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 덧셈 추세(A) | 감쇄 덧셈 추세(Ad) | 추세없음(N) |
| **덧셈 계절성(A)** | (A, A, A) = 덧셈 홀트윈터 | (A, Ad, A) | (A, N, A) |
| **곱셈 계절성(M)** | (A, A, M) = 곱셈 홀트윈터 | (A, Ad, M) = 감쇄 홀트 윈터 | (A, N, M) |
| **계절성 없음(N)** | (A, A, N) = 홀트 선형 | (A, Ad, N) = 감쇄 추세 | (A, N, N) = 단순평활 |

* 보정방법 : 곱셈 보정(M)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 덧셈 추세(A) | 감쇄 덧셈 추세(Ad) | 추세없음(N) |
| **덧셈 계절성(A)** | (M, A, A) = 덧셈 홀트윈터 | (M, Ad, A) | (M, N, A) |
| **곱셈 계절성(M)** | (M, A, M)= 곱셈 홀트윈터 | (M, Ad, M) = 감쇄 홀트 윈터 | (M, N, M) |
| **계절성 없음(N)** | (M, A, N)= 홀트 선형 | (M, Ad, N) = 감쇄 추세 | (M, N, N) = 단순평활 |

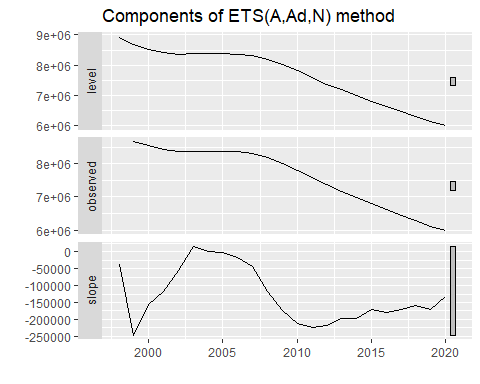
forecast 패키지의 ets() 함수에는 상태공간모델을 지정할 수 있지만 ets() 함수에서 자동적으로 선정해준다. ets 모델을 autoplot()에 적용시키면 각각의 성분별로 plot을 확인할 수 있다.

# 총학생수에 대한 ets 모델  
ets(students.ts[,2]) %>% summary

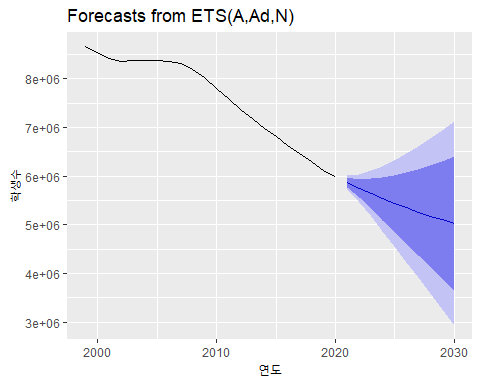
ETS(A,Ad,N)   
  
Call:  
 ets(y = students.ts[, 2])   
  
 Smoothing parameters:  
 alpha = 0.8929   
 beta = 0.8929   
 phi = 0.9401   
  
 Initial states:  
 l = 8930071.6714   
 b = -36934.5936   
  
 sigma: 74403.08  
  
 AIC AICc BIC   
567.8898 573.4898 574.4361   
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -13547.67 65403.96 42008.54 -0.1655554 0.523083 0.326878  
 ACF1  
Training set 0.007506032

* 결과 설명
  + ETS(A, Ad N) : ETS모델로 덧셈 보정, 감쇄 추세, 계절성 없음 모델이 선정됨 - 덧셈 보정 감쇄 추세 모델
  + Smooting parameters : 레벨 평활 계수는 0.8929, 추세 평활 계수는 0,8929, 감쇄(damped) 평활 계수는 0.9401로 설정
  + Initial states : 레벨 초기 상태와 추세 초기상태 설정

# 전체 학생수에 대한 ets 모델 ploting  
ets(students.ts[,2]) %>% autoplot()



# 전체 학생수에 대한 예측치 ploting  
ets(students.ts[,2]) %>% forecast() %>%   
 autoplot() +   
 labs(x = '연도', y = '학생수')

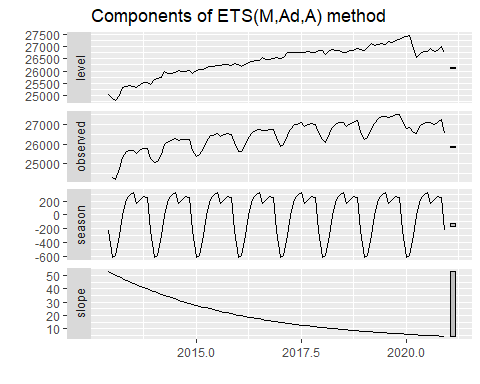


# ETS(M,Ad,A)로 모델 선정  
ets(employees.ts[,2])

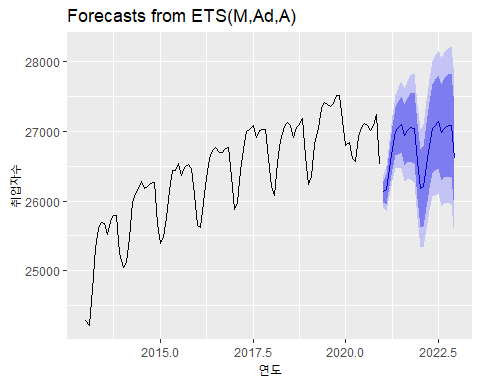
ETS(M,Ad,A)   
  
Call:  
 ets(y = employees.ts[, 2])   
  
 Smoothing parameters:  
 alpha = 0.9989   
 beta = 1e-04   
 gamma = 2e-04   
 phi = 0.9744   
  
 Initial states:  
 l = 25055.6086   
 b = 53.1073   
 s = -222.3076 255.8486 261.7129 227.3178 163.5351 319.1916  
 274.8844 226.215 -5.605 -294.4144 -588.041 -618.3374  
  
 sigma: 0.0045  
  
 AIC AICc BIC   
1372.968 1381.851 1419.126

* 결과 설명
  + ETS(M, Ad A) : ETS모델로 곱셈 보정, 감쇄 덧셈 추세, 덧셈 계절성 모델이 선정됨
  + Smooting parameters : 레벨 평활 계수는 0.9989, 추세 평활 계수는 0.0001, 계절 평활계수는 0.0002, 감쇄(damped) 평활 계수는 0.9744로 설정
  + Initial states : 레벨 초기 상태와 추세, 계절성 초기상태 설정

ets(employees.ts[,2]) %>% autoplot()



ets(employees.ts[,2]) %>% forecast() %>%   
 autoplot() +   
 labs(x = '연도', y = '취업자수')



## ARIMA 모델

이번에 설명하는 ARIMA 모델은 앞의 지수 평활 모델과 함께 시계열 예측 모델에서 가장 많이 사용되어온 모델이다. ARIMA는 AutoRegression Integrated Moving Average의 앞글자만 딴 이름이다. 지수 평활 모델은 주어진 데이터에서 추세와 계절성을 계량화한 모델이지만 ARIMA 모델은 이름에서도 나타나듯이 자기상관과 이동평균을 수식화한 모델이다.

사실 ARIMA 모델은 ARMA(AutoRegression Moving Average)을 기반으로 한 모델이지만 ARMA 모델은 정상성 시계열에 한한 모델이기 때문에 비정상성 데이터를 정상화하는 단계를 포함하여 ARIMA 모델로 구현된다.[[10]](#footnote-53)

ARIMA 모델은 arima(p, d, q)로 표현되는데 p는 AR모델의 차수, d는 비정상성 시계열을 정상성 시계열로 변환하기 위한 차분 차수, q는 이동평균 차수를 의미한다. 결국 ARIMA(p, d, q)모델은 데이터를 정상성 시계열로 만들기 위해 d번 차분한 데이터에 ARMA(p, q)모델을 적용하는 것과 동일하다.

### 자기회귀모델(AutoRegressive Model)

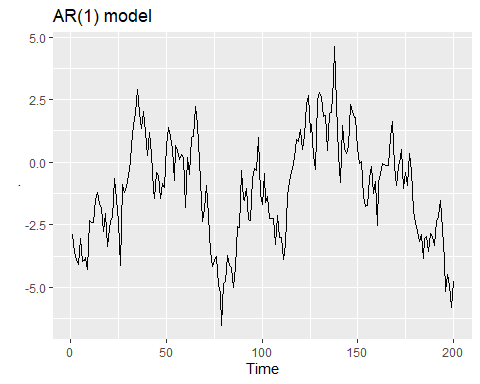
머신러닝을 공부할 때 대부분 선형 회귀모델을 가장 먼저 배운다. 선형회귀모델은 독립변수와 종속변수를 분리하고 두 변수간의 상관관계를 분석하여 모델링을 한다. 앞선 장에서 시계열 선형회귀분석을 설명하였는데 ~를 사이에 두고 오른쪽의 독립변수는 시간이었고 왼쪽의 종속변수는 예측을 원하는 변수를 설정하였다.

그러나 AR모델은 독립변수와 종속변수가를모두 자기 자신을 사용한다는 점에서 일반 회귀모델과 다르다. 과거의 자기 자신의 데이터와 현재의 자기 자신 데이터간의 상관관계를 분석하여 회귀모델을 세우는 것이 AR 모델이다.

AR(p) 모델은 자기 상관관계가 lag p까지 영향을 미치는 모델이다. AR(1)은 자신의 데이터에 lag 1을 취한 데이터간의 회귀분석 모델이고 AR(2)는 자신의 데이터에 lag 1과 lag 2 데이터 간의 다중 회귀분석 모델이다. 따라서 AR(1) 모델은 회귀계수가 하나이고 AR(2) 모델은 회귀계수가 2개이므로 AR(p) 모델은 회귀계수가 p개 존재하게 된다.

다음의 예는 자기회귀모델을 생성하고 ploting하는 예를 보이고 있다. arima.sim() 은 tseries 패키지에서 제공하는 함수로 ARIMA 모형에 따른 랜덤 데이터를 생성하는 함수이고 auto.arima()는 forecast 패키지에서 제공하는 함수로 ARIMA 모델을 자동으로 결정해 주는 함수이다.

library(tseries)  
set.seed(345)   
arima100 <- arima.sim(model = list(order = c(1, 0, 0), ar = 0.9), n = 200)  
arima100 %>% autoplot(main = 'AR(1) model')



urca::ur.kpss(arima100) %>% urca::summary()

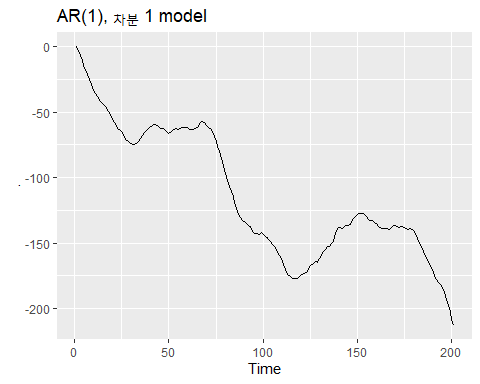
#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 0.2684   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

ndiffs(arima100, test = 'kpss')

[1] 0

* 코드 설명
  + 항상 동일한 랜덤 변수가 나오도록 seed를 345로 설정(set.seed(345))
  + arima.sim()을 사용하여 1차 AR 회귀계수가 09(ar = 0.9)인 ARIMA(1, 0, 0) 모델(order = c(1, 0, 0))에 적합한 데이터 200개(n = 200) 생성해서 arima100에 저장
  + autoplot()을 사용하여 arima100을 plotting
  + kpss.test()를 통해 정상성을 검사하는데 검정통계량이(0.2684)가 5% 임계치(0.463)보다 작으므로 이미 정상성을 만족
  + ndiffs()를 사용하여 arima100에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 kpss 테스트에서 이미 정상성이라고 판단하였으므로 차분이 불필요하다는 0를 출력

set.seed(345)  
arima110 <- arima.sim(model = list(order = c(1, 1, 0), ar = 0.9), n = 200)   
arima110 %>% autoplot(main = 'AR(1), 차분 1 model')



urca::ur.kpss(arima110) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 3.2912   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

ndiffs(arima110, test = 'kpss')

[1] 1

urca::ur.kpss(diff(arima110)) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 0.2684   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

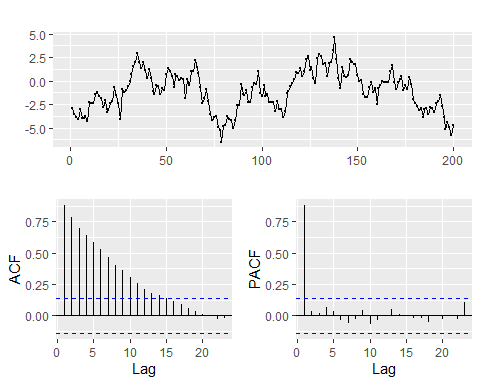
ndiffs(diff(arima110), test = 'kpss')

[1] 0

* 코드 설명
  + 항상 동일한 랜덤 변수가 나오도록 seed를 345로 설정(set.seed(345))
  + arima.sim()을 사용하여 1차 AR 회귀계수가 0.9(ar = 0.9)인 ARIMA(1, 1, 0) 모델(order = c(1, 1, 0))에 적합한 데이터 200개(n = 200) 생성해서 arima110에 저장
  + autoplot()을 사용하여 arima110을 plotting
  + ur.kpss()를 통해 정상성을 검사하는데 검정통계량이(3.2912) 5% 임계치(0.463)보다 크므로 비정상성임
  + ndiffs()를 사용하여 arima110에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 1를 출력하므로 1차 차분이 필요
  + 1차 차분 데이터를 ur.kpss()를 통해 정상성을 검사하는데 검정통계량이(0.2684) 5% 임계치(0.463)보다 작으므로 이미 정상성을 만족
  + ndiffs()를 사용하여 arima110의 1차차분 데이터(diff(arima110))에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 kpss 테스트에서 이미 정상성이라고 판단하였으므로 차분이 불필요하다는 0를 출력

ARIMA(1, 0, 0)의 ACF, PACF plot은 다음과 같이 나타난다. acf plot을 보면 자기 상관성이 전반적으로 높고 천천히 감소하는 형태(Tail off)를 보인다. 그리고 pacf plot은 lag 1에서 매우 높지만 2에서부터는 끊어진다.(cut off) ARIMA(1, 1, 0)도 유사한 형태를 나타낸다.

arima100 %>% ggtsdisplay()



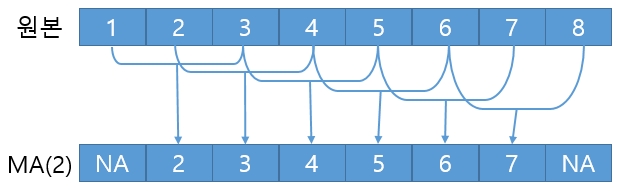
### 이동평균 모델(Moving Average Model)

보통 우리는 데이터의 전체합을 데이터의 갯수로 나눈 것을 평균이라 한다 . 평균은 데이터 전체를 설명하는 가장 간편하고 널리 사용되는 모델이고 시계열데이터에서도 많이 사용된다.

그러나 평균은 데이터의 분포(분산)이 크고 이상치(Outlier)가 많을때는 데이터를 제대로 설명해내지 못한다. 시계열에서도 비슷한 경향이 있는데 일반 데이터와 달리 시계열 데이터에서는 장긱간의 시계열 데이터일수록 평균을 사용하여 데이터를 설명하기 힘들어진다.

예를 들어 우리나라의 1인당 GDP를 분석하고자 할 때 1960년대부터 2020년까지의 GDP를 평균을 낸다면 GDP가 낮았던 1970년대까지의 데이터로 인해 전체 평균이 최근의 GDP에 비해 낮게 나타날 것이다. 경우에 따라 이 데이터가 의미있을 수도 있지만 최근의 경향에 맞지 않은 데이터로 간주될 것이다. 따라서 이런 경우는 시계열 데이터를 최근 3년, 5년 등으로 한정하여 평균을 내는 방법을 사용한다. 이렇게 시계열 데이터에 대한 전체 평균이 아닌 특정 기간간의 평균을 지속적으로 내는 것을 이동평균이라고 한다.

이동 평균을 가장 많이 볼 수 있는 응용은 주식 plot이다. 주식 plot에서는 3일 이동평균, 5일 이동평균, 10일 이동평균등 다양한 이동평균을 통해 해당 주식 주가의 전반적인 흐름을 파악한다.



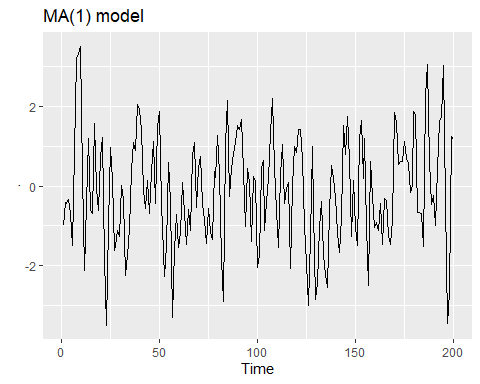
이동 평균 모델은 MA(q)로 표현하는데 q는 이동평균을 산출하는 차수를 의미한다. q가 1인 경우는 기준 시점값과 바로 이전값의 평균, 2인경우는 기준 시점값과 이전, 차이전 값과의 평균을 의미한다. 기준 시점값은 시계열 적으로 과거값으로 하나씩 이동할 수 있고 시계열 초기값까지 이동하면 이동평균 산출은 끝난다.

ARIMA모델에서도 이동평균을 사용한다. 앞에서 설명한 바와 같이 ARIMA(p, d, q)에서 q에 해당하는 값이 이동평균의 차수를 가리킨다.

ARIMA 모델에서 이동 평균을 사용할 때 주의해야 할점은 이동 평균 모델의 계수를 적용하는 독립 변수가 오차항이라는 점이다. 앞선 자기회귀모델에서는 자기회귀계수를 자기자신의 lag 차수값에 적용하였지만 이동 평균 모델은 이동평균 계수를 오차항에 적용시킨다. (이해가 어렵지만 원리를 알고싶다면 참고문헌[[11]](#footnote-57)을 참조하라)

아래와 같이 MA(1)모델을 생성시켜 차이를 살펴보자.

set.seed(345)   
# ARIMA(1,0,0)에 AR(1)의 회귀계수가 0.9인 데이터 200개 생성  
arima001 <- arima.sim(model = list(order = c(0, 0, 1), ma = 0.9), n = 200)   
arima001 %>% autoplot(main = 'MA(1) model')



# kpss 테스트를 통해 생성된 데이터가 정상성인지 테스트 - 0.05보다 크므로 정상성, 차분 불필요  
urca::ur.kpss(arima001) %>% urca::summary()

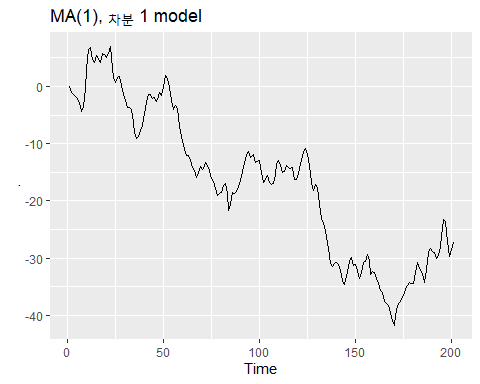
#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 0.0906   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

# 비정상 제거를 위한 차분수 - 0이 나오므로 차분 불필요  
ndiffs(arima001, test = 'kpss')

[1] 0

* 코드 설명
  + 항상 동일한 랜덤 변수가 나오도록 seed를 345로 설정(set.seed(345))
  + arima.sim()을 사용하여 1차 MA 회귀계수가 0.9(ma = 0.9)인 ARIMA(0, 0, 1) 모델(order = c(0, 0, 1))에 적합한 데이터 200개(n = 200) 생성해서 arima001에 저장
  + autoplot()을 사용하여 arima001을 plotting
  + ur.kpss()를 통해 정상성을 검사하는데 검정통계량이(0.0906) 5% 임계치(0.463)보다 작으므로 정상성임
  + ndiffs()를 사용하여 arima001에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 0를 출력하므로 1차 차분이 불필요

set.seed(345)  
arima011 <- arima.sim(model = list(order = c(0, 1, 1), ma = 0.9), n = 200)   
arima011 %>% autoplot(main = 'MA(1), 차분 1 model')



# kpss 테스트를 통해 생성된 데이터가 정상성인지 테스트 - 0.05보다 작으므로 정상성, 차분 필요  
urca::ur.kpss(arima011) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 3.5785   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

# 비정상성을 제거하기 위해 필요한 차분수  
ndiffs(arima011, test = 'kpss')

[1] 1

urca::ur.kpss(diff(arima011)) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 4 lags.   
  
Value of test-statistic is: 0.0906   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

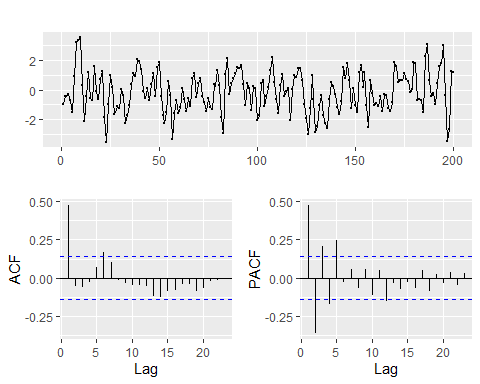
ndiffs(diff(arima011), test = 'kpss')

[1] 0

* 코드 설명
  + 항상 동일한 랜덤 변수가 나오도록 seed를 345로 설정(set.seed(345))
  + arima.sim()을 사용하여 1차 MA 회귀계수가 0.9(ma = 0.9)인 ARIMA(0, 1, 1) 모델(order = c(0, 1, 1))에 적합한 데이터 200개(n = 200) 생성해서 arima011에 저장
  + autoplot()을 사용하여 arima011을 plotting
  + ur.kpss()를 통해 정상성을 검사하는데 검정통계량이(3.5785) 5% 임계치(0.463)보다 크므로 비정상성임
  + ndiffs()를 사용하여 arima011에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 1를 출력하므로 1차 차분이 필요
  + 1차 차분 데이터를 ur.kpss()를 통해 정상성을 검사하는데 검정통계량이(0.0906) 5% 임계치(0.463)보다 작으므로 이미 정상성을 만족
  + ndiffs()를 사용하여 arima011의 1차차분 데이터(diff(arima011))에 필요한 차분수를 kpss 테스트를 기반으로 산출하는데 kpss 테스트에서 이미 정상성이라고 판단하였으므로 차분이 불필요하다는 0를 출력

ARIMA(0, 0, 1)의 ACF, PACF plot은 다음과 같이 나타난다. AR(1)모델의 acf는 점차 감소(tail off)하였고 pacf는 1에서 절단(cut off)되었지만 MA(1)모델에서는 acf plot이 1에서 절단(cut off)되고 pacf plot은 +와 -를 반복하지만 전반적으로 감소(tail off)하고 있다.

arima001 %>% ggtsdisplay()



### ARIMA 모델 결정

ARIMA 모델을 사용하기 위해서는 p, d, q의 차수를 결정하는 것이 매우 중요하다. 차수를 결정하는 방법은 앞서 설명한 acf, pacf를 보고 판단할 수 있는데 forecast 패키지의 auto.arima() 함수에서는 자동으로 p, d, q의 차수를 결정해주기 때문에 편리사게 사용할 수 있다.

하지만 ARIMA 모델의 전반적인 동작을 이해하기 위해서는 acf와 pacf를 확인하여 모델을 결정하는 방법을 알아둘 필요가 있다. 우선 간략하게 ARIMA 모델을 결정하는 방법을 설명하자면 다음과 같다.

1. 차분을 통한 정상화
   1. ur.kpss()를 사용하여 검정통계량이 5% 임계치보다 크면 비정상으로 ndiffs() 를 사용하여 차분수를 구함
   2. ur.kpss()의 검정통계량이 5% 임계치보다 작을때까지 a를 반복
2. ACF, PACF를 확인
   1. ACF가 점차 감소이고 PACF의 p차에서 절단값이 있다면 AR(p)
   2. PACF가 점차 감소이고 ACF가 q차에서 절단값이 있다면 MA(q)
   3. ACF와 PACF가 모두 점차감소라면 ARMA 모델(육안으로는 선택이 어려움)
3. 잔차를 확인
   1. 각 시차의 잔차 Ljung-box 테스트 값이 임계치 이상이어서 정상성을 만족하는지 확인

#### 차분을 통한 정상화

앞에서 설명했다시피 ARIMA모델은 비정상 데이터를 정상화로 만들기 위해 d 차분한 데이터에 ARMA(p, q)를 적용한 것과 같다고 했다. 따라서 ARIMA 모델을 만들기 위해서는 먼저 정상데이터이지를 검사하고 비정상일 경우 차분을 통해 정상데이터로 만들어야 한다. 이 과정은 앞의 예제에서 ur.kpss()와 ndiffs()를 이용하는 방법을 설명했다.

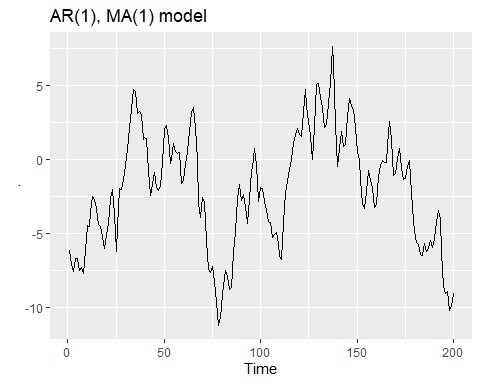
#### ACF, PACF 확인

앞의 예제에서 AR(1)과 MA(1)의 ACF와 PACF plot을 보았는데 두 경우가 비슷하지만 다른 특성이 있다. AR(1) 모델의 경우는 ACF plot이 점차 감소(tail off)하고 PACF plot의 절단(cut off) 차수가 1이었다. 반면 MA(1)의 경우는 ACF plot의 절단(cut off) 차수가 1이었고 PACF plot이 점차 감소(tail off)하였다. 이를 정리하면 다음의 표와 같이 나타낼 수 있다.

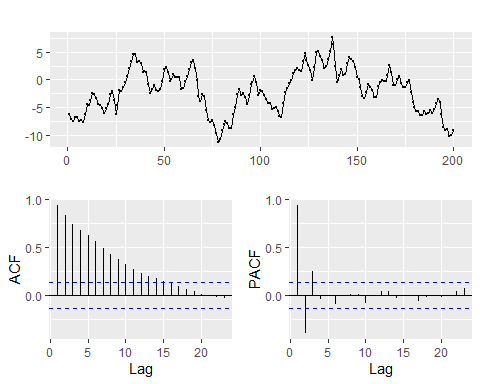
|  |  |  |  |
| --- | --- | --- | --- |
|  | AR(p) | MA(q) | ARMA(p, q) |
| ACF | 점차 감소(tail off) | q차에서 절단(cut off) | 점차 감소(tail off) |
| PACF | p차에서 절단(cut off) | 점차 감소(tail off) | 점차 감소(tail off) |

위의 표에서 보듯이 AR모델은 ACF plot이 점차 감소하며 PACF plot의 절단 차수가 p일때 AR(p)모델로 결정되고, MA 모델은 PACF plot이 점차 감소하며 ACF plot의 절단 차수가 q일때 MA(q)로 결정한다. 만약 AR과 MA가 동시에 나타나는 모델의 경우는 다음의 예에서 보듯이 ACF와 PACF가 모두 점차 감소하는 형태를 보인다.

set.seed(345)  
# ARIMA(1,0,0)에 AR(1)의 회귀계수가 0.9인 데이터 200개 생성  
arima101 <- arima.sim(model = list(order = c(1, 0, 1), ar = 0.9, ma = 0.9), n = 200)   
arima101 %>% autoplot(main = 'AR(1), MA(1) model')



arima101 %>% ggtsdisplay()



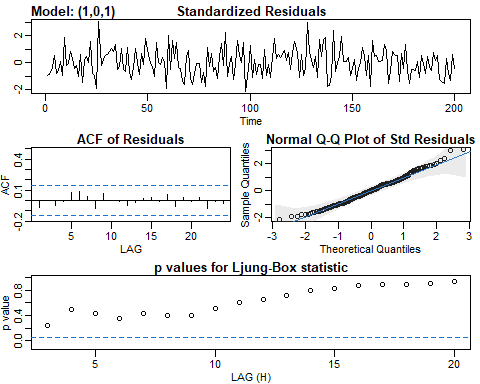
위의 예와 같이 p와 q가 모두 0보다 큰 경우는 ACF와 PACF plot으로 모델을 결정하는 것이 적절하지 않을 수 있다. [[12]](#footnote-61) 이 경우에는 여러가지 경우수를 설정하고 AIC, BIC, RMSE 등의 성능 분석 수치를 통해 가장 좋은 모델을 선정해야 한다. 이와 관련된 내용은 다음 장에서 다루겠다.

#### 잔차 확인

ARIMA 모형에 적용할 차수를 모두 결정한 후에는 결정한 모델을 통해 생성된 잔차가 백색 잡음인지를 확인해야 한다.

백색 잡음인지를 검사하는 방법으로 자기상관관계를 사용한 Ljung-box test와 단위근 검정을 통한 KPSS test를 소개하였다. ARIMA모델을 결정하기 위해 가장 먼저 했던 작업이 KPSS test를 통한 단위근 검정이었기 때문에 이미 단위근 검정은 시행했다. 따라서 모델이 생성한 잔차가 백색잡음인지를 검사하기 위해서는 자기 상관관계를 활용해 백색잡음 여부를 판단하는 Ljung-box test를 추가로 시행한다. ARIMA 모델의 잔차를 검사할 때는 astsa 패키지에서 제공하는 sarima()를 사용하면 편리하다.

library(astsa)  
sarima(arima101, p = 1, d = 0, q = 1)



위 plot의 가장 위의 표준화 잔차 plot(Standardized Residuals)은 잔차의 분포를 보여준다. 전반적으로 평균 0의 레벨에서 분산이 크게 벗어나고 있지 않음을 볼 수 있다.

두번째 열의 왼쪽 잔차의 ACF plot을 보면 잔차의 자기 상관이 거의 없음이 나타난다. 오른쪽의 Q-Q plot은 잔차가 정규분포를 하는지를 보여주는 plot인데 정규분포 데이터일수록 대각선에 포인트들이 몰려있게 된다. 양쪽 끝단의 데이터는 큰 의미가 없다. Q-Q plot을 볼 때도 정규분포에서 크게 벗어나지 않는 것으로 보인다.

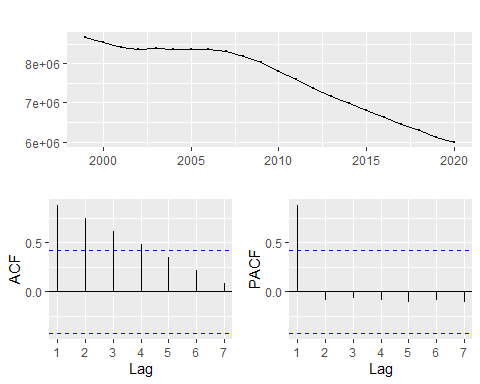
마지막으로 Ljung-Box의 p-value plot이 보이는데 모든 lag의 p-value가 95%임계선 위쪽으로 나타나야 백색 잡음으로 간주할 수 있는데 여기서는 모두 95% 임계선 위에 있으므로 모든 조건이 백색잡음을 만족한다.

#### ARIMA 모델 실습

ARIMA 모델을 학생수 시계열 데이터, 취업자수 시계열 데이터, 코로나 확진자 시계열 데이터에 적용해보면 다음과 같다. 앞서 설명한 바와 같이 forecast 패키지에서는 ARIMA모델을 자동으로 결정해주는 함수인 auto.arima()를 제공한다.

학생수 데이터를 ARIMA 모델에 적용하면 ACF와 PACF로 비교적 확실하게 확인이 가능한 모델을 선정할 수 있다.

students.ts[,2] %>% ggtsdisplay()



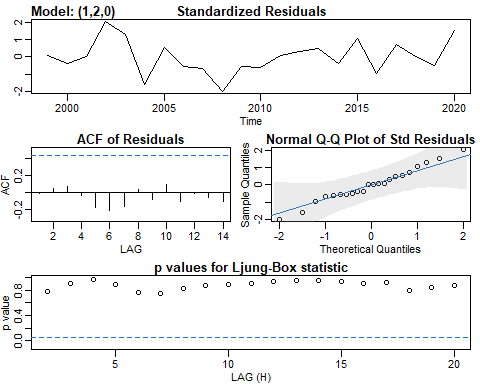
urca::ur.kpss(students.ts[,2]) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 2 lags.   
  
Value of test-statistic is: 0.7939   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

ndiffs(students.ts[,2], test = 'kpss')

[1] 2

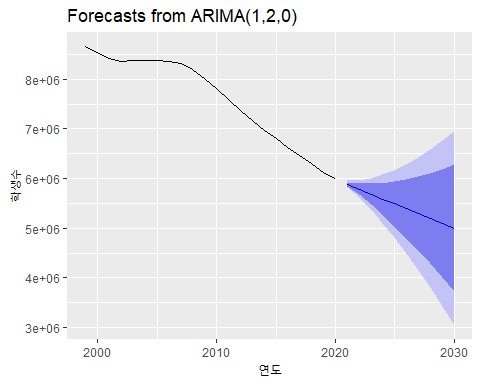
sarima(students.ts[,2], p = 1, d = 2, q = 0)



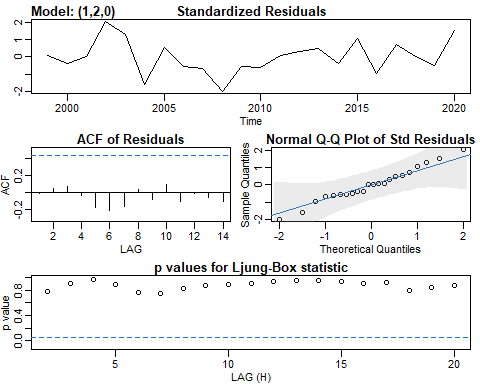
auto.arima(students.ts[,2])

Series: students.ts[, 2]   
ARIMA(1,2,0)   
  
Coefficients:  
 ar1  
 0.3861  
s.e. 0.2075  
  
sigma^2 estimated as 1.174e+09: log likelihood=-236.71  
AIC=477.42 AICc=478.12 BIC=479.41

auto.arima(students.ts[,2]) %>% forecast() %>%   
 autoplot() +   
 labs(x = '연도', y = '학생수')



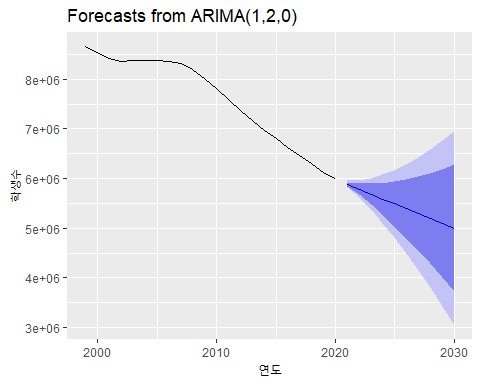
sarima(students.ts[,2], p = 1, d = 2, q = 0)



auto.arima(students.ts[,2])

Series: students.ts[, 2]   
ARIMA(1,2,0)   
  
Coefficients:  
 ar1  
 0.3861  
s.e. 0.2075  
  
sigma^2 estimated as 1.174e+09: log likelihood=-236.71  
AIC=477.42 AICc=478.12 BIC=479.41

auto.arima(students.ts[,2]) %>% forecast() %>%   
 autoplot() +   
 labs(x = '연도', y = '학생수')



* 코드 설명
  + ACF가 점차 감소(tail off)하고 PACF가 lag 1에서 절단(cut off)되었으므로 AR(1) 모델로 결정할 수 있음
  + KPSS 테스트결과 검정통계량이 95% 임계치보다 크기 때문에 차분이 필요함.
  + ndiffs()를 사용해 적정 차분수를 알아봄. 결과가 2로 나와 2차 차분이 필요
  + sarima()로 확인한 잔차의 Ljung-box의 p-value는 모두 임계치보다 크므로 잔차는 백색잡음으로 볼 수 있음
  + auto.arima()를 사용하여 결정된 ARIMA 모형도 ARIMA(1, 2, 0)이며 1차 AR계수는 0.3861로 계산됨
  + auto.arima() 모델을 forecast()를 사용하여 예측치를 생성하고 autoplot()으로 plot 생성

전체 취업자 수는 계절성을 지니기 때문에 계절성 ARIMA를 사용하여야 한다. 다음의 절에서 설명한다.

코로나 확진자 데이터의 자동 ARIMA 모형은 (2, 1, 1)로 나타난다. kpss test와 ndiffs를 통해 차분이 1인 경우 정상성이 되는 것을 확인할 수 있으나 p, q값이 모두 0보다 큰 수이기 때문에 ACF와 PACF plot으로 차수를 결정하는 것이 쉽지 않다. 또한 ARIMA(2, 1, 1)을 통한 잔차의 결과도 Ljung-box 결과가 모두 95% 임계치보다 큰 것이 아니기 때문에 백색잡음이라고 단정할 수 없다. 이럴 경우는 p값과 q값의 주위값들에 대한 ARIMA모델들을 확인하여 결정하는 것이 좋다. auto.arima()는 AIC값이 작은 값을 모델로 선택한다. AIC(Akaike’s Information Criterion)는 모델의 품질을 결정할 때 참고하는 값이다. 일반적으로 과대적합(Overfitting) 또는 과소적합(Underfitting)되는 모델은 효율적이지 않다고 평가한다. 결국 AIC가 높을수록 모형의 적합도가 떨어진다는 것을 의미한다. 여기서 주의해야할 것은 AIC는 모델의 성능을 평가하는 지수가 아닌 모델의 적합도를 평가하는 지수라는 것이다.

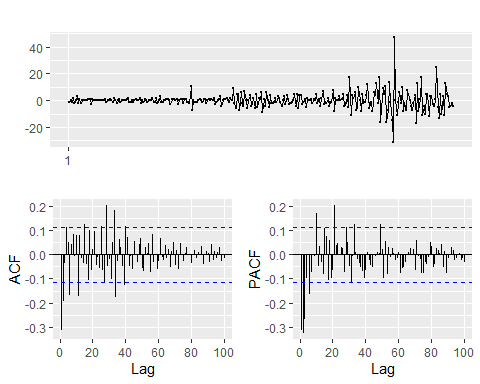
urca::ur.kpss(covid19.ts[,2]) %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 5 lags.   
  
Value of test-statistic is: 3.2656   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

# 비정상성을 제거하기 위해 필요한 차분수가 1  
ndiffs(covid19.ts[,2], test = 'kpss')

[1] 1

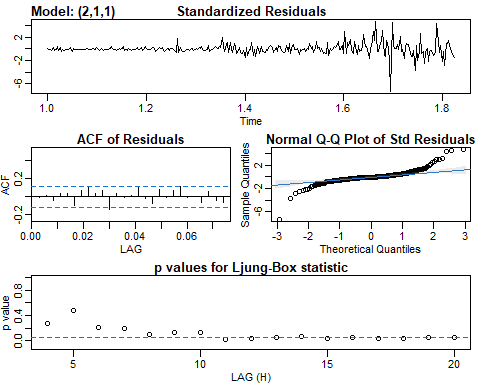
# ACF, PACF 모두 절단(Cut off)이므로 ARMA(p, q) 모델   
diff(covid19.ts[,2]) %>% ggtsdisplay()



# 0-9세 코로나 확진자수의 ARIMA모형은 ARIMA(2, 1, 1)으로 선정됨  
auto.arima(covid19.ts[,2])

Series: covid19.ts[, 2]   
ARIMA(2,1,1)   
  
Coefficients:  
 ar1 ar2 ma1  
 0.1126 -0.1775 -0.6535  
s.e. 0.0868 0.0688 0.0726  
  
sigma^2 estimated as 26.74: log likelihood=-920.47  
AIC=1848.93 AICc=1849.07 BIC=1863.76

sarima(covid19.ts[,2], 2, 1, 1)



### Seasonal ARIMA 모델

위에서 살펴본 모델은 비계절성 ARIMA 모델이었다. 계절성을 지니는 데이터는 ARIMA 모델로는 적절히 모델링이 어렵기 때문에 계절성 ARIMA 모델에 적합시켜야 한다. 계절성 ARIMA는 비계절성 ARIMA의 p, d, q 차수 외에 계절성 차수인 P, D, Q와 관측주기 m을 추가적으로 결정해야 하고 ARIMA(p, d, q)(P, D, Q)m으로 표기한다. 아래의 코드는 전체 취업자수 주기별 합계의 계절성 ARIMA 모델을 산출하는 코드이다. 계절성의 주기가 클수록 ACF와 PACF로 계절성 ARIMA차수를 찾아내는게 어려워지기 때문에 좀 간단한 예로 연습해보자.

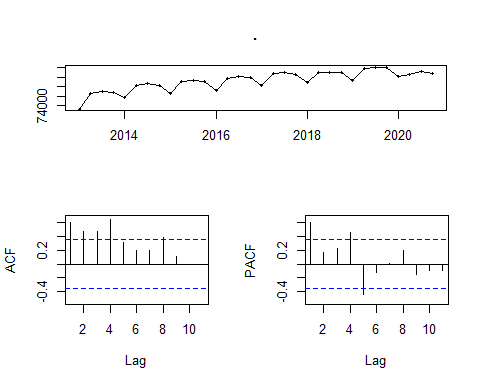
# 전체 취업자수 데이터를 분기별 합계제이터로 변환  
employees %>% mutate(year = lubridate::year(time),   
 qtr = lubridate::quarter(time)) %>%  
 group\_by(year, qtr) %>%  
 summarise(sum = sum(total)) %>%   
 ts(frequency = 4, start = c(2013,1)) -> qtr.employees.ts

`summarise()` regrouping output by 'year' (override with `.groups` argument)

# auto.arima로 일단 ARIMA모형을 검토 - ARIMA(0,1,0)(0,1,0)[4]로 제안됨  
auto.arima(qtr.employees.ts[,3]) %>% summary()

Series: qtr.employees.ts[, 3]   
ARIMA(0,1,0)(0,1,0)[4]   
  
sigma^2 estimated as 240342: log likelihood=-205.52  
AIC=413.04 AICc=413.2 BIC=414.33  
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -119.056 450.3206 266.8772 -0.1506314 0.3348234 0.2848644  
 ACF1  
Training set 0.02400751

# ggtsdisplay()로 ACF plot을 볼때 4주기마다 계절성이 있는듯 보임  
qtr.employees.ts[,3] %>% tsdisplay()



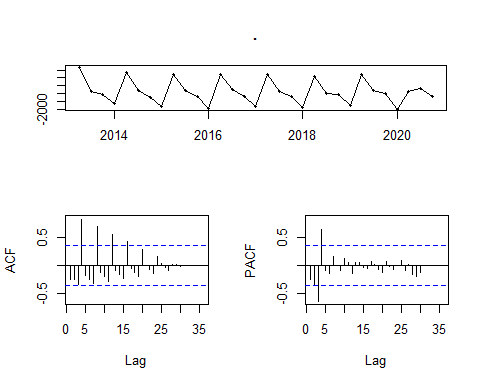
# KPSS 검정결과 단위근이 존재하는 비정상성 데이터  
qtr.employees.ts[,3] %>% urca::ur.kpss() %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 3 lags.   
  
Value of test-statistic is: 0.8663   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

# ndiffs()에 의하면 1차 차분 필요  
qtr.employees.ts[,3] %>% ndiffs()

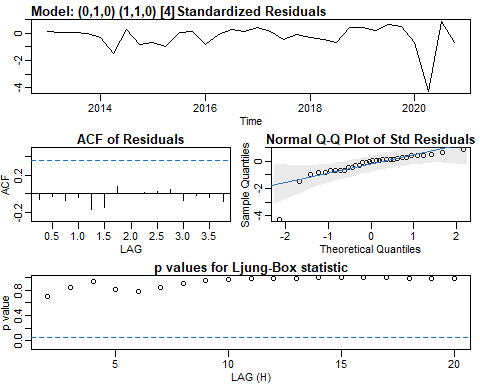
[1] 1

# 1차 차분 결과 plot()  
qtr.employees.ts[,3] %>% diff() %>% tsdisplay(lag.max = 36)

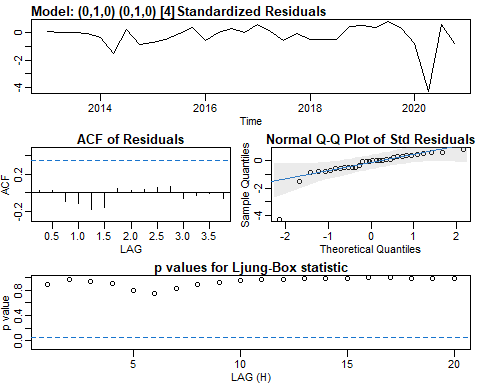


1차 차분된 데이터의 ACF와 PACF를 확인해보면 비계절성 lag(1, 2, 3 등)는 대부분 임계치 아래로 내려가 있기 때문에 ARMA(0,0)으로 결정이 가능하다. 그런데 위의 ACF와 PACF는 앞에서 본 plot과는 좀 다른 점이 있는데 ACF의 선이 4주기별로 점차 감소하는 형태를 보이고 있다는 점이다. 이것을 보고 계절성 ARIMA모형을 결정할 수 있는데 위에서는 ACF에서 4주기별로 점차 감소하고 PACF에서는 lag 4에서 절단 되기때문에 ARMA(1, 0)[4]로 볼 수 있다. 따라서 ACF, PACF를 보고 결정할 수 있는 계절성 ARIMA 모형은 ARIMA(0,1,0)(1,1,0)[4]이다.

# 잔차의 분포, ACF, Q-Q plot, Ljung-box test 모두 백색잡음으로 나타남  
sarima(qtr.employees.ts[,3], p = 0, d = 1, q = 0, P = 1, D = 1, Q = 0, S = 4)



sarima(qtr.employees.ts[,3], p = 0, d = 1, q = 0, P = 0, D = 1, Q = 0, S = 4)



위의 두개의 plot은 ACF, PACF를 통해 결정한 모델과 auto.arima()를 통해 결정된 모델의 잔차 정보 plot이다. 두 모델 모두 잔차를 백색잡음으로 볼 수 있을것으로 보인다. 아래는 위에서 선택한 모델을 Arima()를 사용해 생성하고 두 모델의 미래 예측치를 만들어 plotting하였다.

그럼 auto.arima()가 결정한 ARIMA(0,1,0)(0,1,0)[4]모델과 수작업으로 결정한 ARIMA(0,1,0)(1,1,0)을 비교해보자. 모델을 비교하는 데에는 몇가지 방법이 있고 다음장에서 비교하는 지수에 대해 자세히 설명하겠다.

auto.arima()에서는 모델을 선택하는데 사용하는 지수로 AIC를 사용한다고 하였다. ARIMA 모델 차수 간의 비교에는 AIC가 유용한 판단 지수이지만 다른 전혀 다른 모델과 비교할 때는 잔차에 관련된 성능지수를 사용하는 것이 일반적이다. 따라서 모델간의 성능 비교에서 많이 사용하는 RMSE 지수를 사용하여 비교해 보았다.(RMSE는 다음장에서 상세히 설명한다. 일단 낮은 값이 좋은 값이다.)

수작업으로 선택한 ARIMA(0,1,0)(1,1,0)[4] 모델의 RMSE는 441.4946이고 auto.arima()로 선택된 ARIMA(0,1,0)(0,1,0)[4] 모델의 RMSE는 450.3206으로 산출되었다. RMSE상으로는 수작업으로 선택한 모델이 auto.arima() 선택 모델보다 우수해보인다. 아래 plot으로 두 모델을 확인해보자. 이제 선택은 여러분의 몫이다.

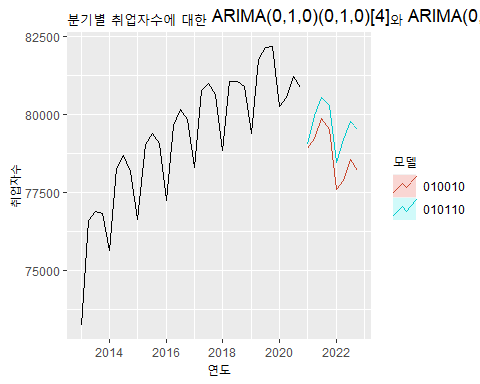
# ARIMA(0,1,0)(0,1,1)[12] 모델 생성  
arima010110 <- Arima(qtr.employees.ts[,3], order = c(0,1,0), seasonal = c(1,1,0))  
arima010010 <- Arima(qtr.employees.ts[,3], order = c(0,1,0), seasonal = c(0,1,0))  
# ARIMA(0,1,0)(0,1,0)[12]모델의 회귀계수 및 다양한 정보  
summary(arima010110)

Series: qtr.employees.ts[, 3]   
ARIMA(0,1,0)(1,1,0)[4]   
  
Coefficients:  
 sar1  
 -0.2955  
s.e. 0.3359  
  
sigma^2 estimated as 239898: log likelihood=-205.16  
AIC=414.33 AICc=414.83 BIC=416.92  
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -127.5149 441.4946 260.5119 -0.1615043 0.3268788 0.27807  
 ACF1  
Training set -0.05923251

summary(arima010010)

Series: qtr.employees.ts[, 3]   
ARIMA(0,1,0)(0,1,0)[4]   
  
sigma^2 estimated as 240342: log likelihood=-205.52  
AIC=413.04 AICc=413.2 BIC=414.33  
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -119.056 450.3206 266.8772 -0.1506314 0.3348234 0.2848644  
 ACF1  
Training set 0.02400751

# ARIMA(0,1,0)(0,1,1)[12] 모델의 예측치 산출  
forecast010110 <- arima010110 %>% forecast()  
forecast010010 <- arima010010 %>% forecast()  
# ARIMA(0,1,0)(0,1,1)[12]의 plot  
autoplot(qtr.employees.ts[,3]) +   
 autolayer(forecast010110, PI = F, series = '010110') +  
 autolayer(forecast010010, PI = F, series = '010010') +  
 labs(title = '분기별 취업자수에 대한 ARIMA(0,1,0)(0,1,0)[4]와 ARIMA(0,1,0)(1,1,0)[4]', x = '연도', y = '취업자수', color = '모델')

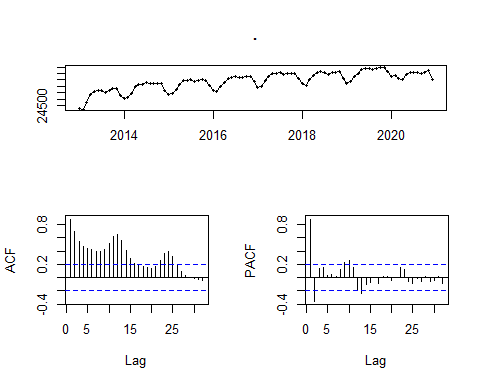


그러면 이제 좀 취업자수의 월별 데이터로 확인해보자.

# auto.arima()는 ARIMA(0,1,0)(0,1,1)[12] 모델 제안  
employees.ts[,2] %>% auto.arima()

Series: .   
ARIMA(0,1,0)(0,1,1)[12]   
  
Coefficients:  
 sma1  
 -0.4246  
s.e. 0.1648  
  
sigma^2 estimated as 15320: log likelihood=-518.14  
AIC=1040.27 AICc=1040.42 BIC=1045.11

# ggtsdisplay()로 ACF plot을 볼때 12주기마다 계절성이 있는듯 보임  
employees.ts[,2] %>% tsdisplay()



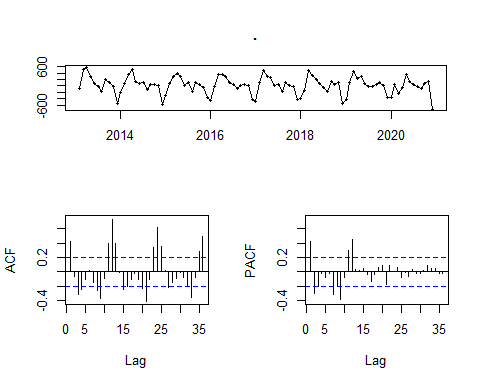
# KPSS 검정결과 단위근이 존재하는 비정상성 데이터  
employees.ts[,2] %>% urca::ur.kpss() %>% urca::summary()

#######################   
# KPSS Unit Root Test #   
#######################   
  
Test is of type: mu with 3 lags.   
  
Value of test-statistic is: 1.9226   
  
Critical value for a significance level of:   
 10pct 5pct 2.5pct 1pct  
critical values 0.347 0.463 0.574 0.739

# ndiffs()에 의하면 1차 차분 필요  
employees.ts[,2] %>% ndiffs()

[1] 1

# 1차 차분 결과 plot()  
# ACF, PACF를 확인해서 비계절성 모델은 (0,1,1) 계절성 모델은 (1,1,0)으로 결정   
employees.ts[,2] %>% diff() %>% tsdisplay(lag.max = 36)



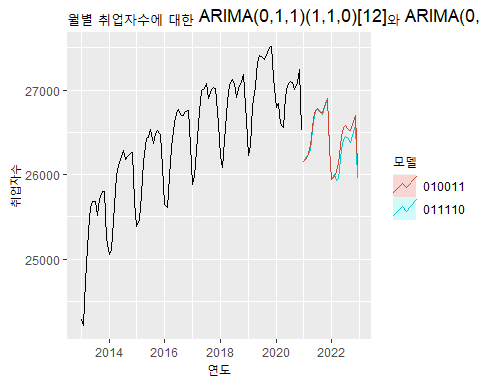
# ARIMA(0,1,1)(1,1,0)[12]모델 생성  
arima011110 <- employees.ts[,2] %>% Arima(order = c(0,1,1), seasonal = c(1,1,0))  
# auto.arima()가 제안한 ARIMA(0,1,0)(0,1,1)[12]모델 생성  
arima010011 <- employees.ts[,2] %>% Arima(order = c(0,1,0), seasonal = c(0,1,1))  
# 각 모델의 정보 확인  
summary(arima011110)

Series: .   
ARIMA(0,1,1)(1,1,0)[12]   
  
Coefficients:  
 ma1 sar1  
 0.0087 -0.3649  
s.e. 0.1385 0.1395  
  
sigma^2 estimated as 15667: log likelihood=-518.22  
AIC=1042.45 AICc=1042.75 BIC=1049.7  
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -15.58899 114.974 74.83796 -0.05996543 0.2828592 0.2361174  
 ACF1  
Training set -0.02801497

summary(arima010011)

Series: .   
ARIMA(0,1,0)(0,1,1)[12]   
  
Coefficients:  
 sma1  
 -0.4246  
s.e. 0.1648  
  
sigma^2 estimated as 15320: log likelihood=-518.14  
AIC=1040.27 AICc=1040.42 BIC=1045.11  
  
Training set error measures:  
 ME RMSE MAE MPE MAPE MASE  
Training set -17.24831 114.3934 73.75538 -0.06622242 0.2787589 0.2327018  
 ACF1  
Training set -0.01264952

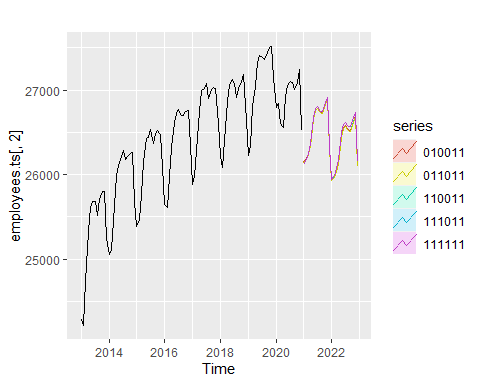
# 두 모델의 예측치 산출  
forecast011110 <- arima011110 %>% forecast()  
forecast010011 <- arima010011 %>% forecast()  
# 두 모델의 plot  
autoplot(employees.ts[,2]) +   
 autolayer(forecast011110, PI = F, series = '011110') +  
 autolayer(forecast010011, PI = F, series = '010011') +  
 labs(title = '월별 취업자수에 대한 ARIMA(0,1,1)(1,1,0)[12]와 ARIMA(0,1,0)(0,1,1)[12]', x = '연도', y = '취업자수', color = '모델')



위의 auto.arima()가 결정한 ARIMA(0,1,0)(0,1,1)[12]모델과 수작업으로 결정한 ARIMA(0,1,1)(1,1,0)을 비교해보자. auto.plot()의 ARIMA(0,1,0)(0,1,1)[12]모델의 RMSE값은 114.39이고 수작업으로 결정한 ARIMA(0,1,1)(1,1,0)의 RMSE값은 114.97이다. 일반적으로 RMSE값이 작은 모델이 성능이 좋은 모델인데 이 두 모델의 중에 성능이 좋은 모델은 auto.arima()의 ARIMA(0,1,0)(0,1,1)[12]모델이다. 또 아래의 표는 ARIMA(0,1,0)(0,1,1)모델 부근의 모델들의 RMSE값이다.

|  |  |
| --- | --- |
| 모델차수 | RMSE |
| ARIMA(0,1,0)(0,1,1) | 114.39 |
| ARIMA(0,1,0)(0,1,0) | 120.51 |
| ARIMA(0,1,0)(1,1,1) | 120.41 |
| ARIMA(0,1,0)(0,1,0) | 120.51 |
| ARIMA(1,1,0)(0,1,1) | **114.35** |
| ARIMA(1,1,0)(0,1,0) | 120.51 |
| ARIMA(1,1,0)(1,1,0) | 114.97 |
| ARIMA(1,1,0)(0,1,0) | 120.51 |
| ARIMA(0,1,1)(0,1,1) | **114.32** |
| ARIMA(0,1,1)(0,1,0) | 120.51 |
| ARIMA(0,1,1)(1,1,1) | 114.32 |
| ARIMA(0,1,1)(0,1,0) | 120.51 |
| ARIMA(1,1,1)(0,1,1) | **113.23** |
| ARIMA(1,1,1)(0,1,0) | 117.42 |
| ARIMA(1,1,1)(1,1,1) | **113.23** |
| ARIMA(1,1,1)(0,1,0) | 117.42 |

auto.arima()와 수동으로 선택한 모델인 ARIMA(0,1,0)(0,1,1)[12]의 RMSE보다 낮은 RMSE값을 갖는 모델이 눈에 보인다. 그렇다면 이 모델이 더 좋은 모델인가? ARIMA(0,1,1)(1,1,1)모델은 잔차의 Ljung-Box 테스트에서 lag 1의 p-value가 임계치보다 낮아 잔차의 정상성을 확보하지 못한다. 다음의 plot은 auto.arima()가 선택한 RMSE보다 낮은 모델의 예측치 plot이다.



위의 plot을 보면 예측치가 미세하게 차이난다. 어떻게 보면 큰 차이가 아닐수도 있고 어찌보면 auto.plot()의 예측력이 떨어진다고 볼수도 있다. 따라서 육안으로 확인하거나 auto.arima 모델의 p, q값의 주위 모델을 반드시 확인하여 성능 수치가 더 우수한 모델을 찾는 과정을 거쳐야 한다.

## TBATS 모델

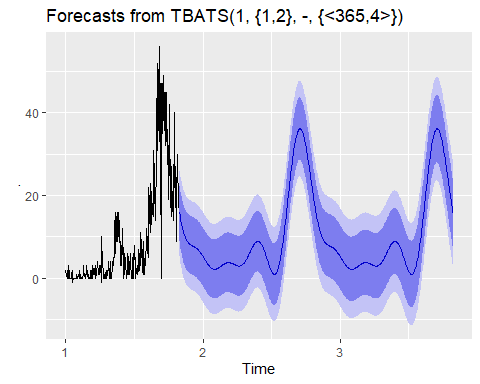
앞선 ETS 모델과 ARIMA 모델은 계절성을 처리할 수 있는 방법을 제공하지만 문제는 다중 계절성을 지는 경우이다. 예를 들어 시간별 데이터는 하루내에서도 계절성을 가질 수 있고 주간적으로도 계절성을 가질 수 있으며 월간적으로, 연간적으로도 계절성을 가질 수 있다. ETS 모델과 ARIMA 모델은 연간(frequency = 1), 분기간(frequency = 4), 월간(frequency = 12) 까지의 계절성을 지원하고 주간(frequency = 52) 이상의 계절성을 지원하지 못한다.

또한 계절성이 동적으로 변동되는 경우는 ETS, ARIMA 모델이 지원하지 못한다. 따라서 이렇게 긴 주기의 계절성을 찾아내는 모델이나 변동성을 지니는 계절성 데이터에 대한 모델을 구축해야 할 때 사용하는 모델이 TBATS모델이다. TBATS 모델은 계절성의 삼각함수(**T**rigonometric), **B**ox-Cox 변환[[13]](#footnote-69), **A**RMA 에러, 추세(**T**rend)와 계절성 컴포넌트(**S**easonal Components)의 다섯가지 요소를 사용한 지수평활 상태공간 모델이다.[[14]](#footnote-71)

코로나 확진자 데이터의 경우 2020년 9월 경 한차례 증가가 있었다가 2020년 12월 또 한차례 증가가 발생했다. 이것을 이중 계절성으로 파악한다면 ETS나 ARIMA모형으로는 모델링이 불가하다. 하지만 TBATS 모델은 다음과 같이 모델이 가능하다.

TBATS모델은 forecast() 패키지의 tbats()를 통해 생성할 수 있다. 생성된 모델은 forecast()를 통해 예측치를 산출할 수 있다.

# 코로나 확진자 데이터에 대한 tBats 모델  
covid19.ts[,2] %>% tbats() %>% forecast() %>% autoplot()



## prophet 모델

Prophet 모델은 페이스북에서 자사 데이터를 기반으로 개발한 시계열 모델로 prophet 패키지를 통해 R과 Python에서 사용할 수 있다. 이 모델은 연도별, 주별, 일별 계절성 및 휴일 효과에 적합한 비선형 추세에 알맞은 가법 모델을 통해 시계열 데이터를 예측한다. (무슨 말인지 몰라도 상관없다)

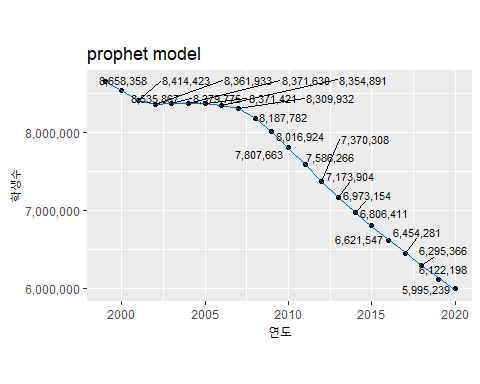
페이스북에서는 prophet 모델이 ‘빠르고 정확하다’, ‘완전 자동화되어 있다’, ‘예측을 조절할수 있다’, ’R과 Python에서 사용할 수 있다’고 홍보하고 있다.[[15]](#footnote-74) 하지만 예측 방법론을 페이스북에서 공개하지 않는 블랙박스 모델이다. 이 부분은 시계열 분석 방법론을 연구하는 연구자들에게는 단점이지만 시계열 분석을 실무에서 사용하는 사람들은 오히려 장점이 될 수 있다.(머리아프게 공부하지 않고 그냥 쓰면 되니까)

prophet 모델은 시계열 데이터를 추세(t), 계절성(s), 휴일(h), 오차의 특성(feature)로 분리한다. 추세(t)는 반복이 되지 않는 추세, 계절성(s)은 계절성과 같은 반복적인 변화, 휴일(h)은 공휴일과 같이 달력상에서의 불규칙한 특성(feature)을 말한다. 여기에 오차가 더해지는데 이 오차는 특정 모델로 적용이 어려운 정규분포를 따르는 오차라고 가정한다.[[16]](#footnote-76)

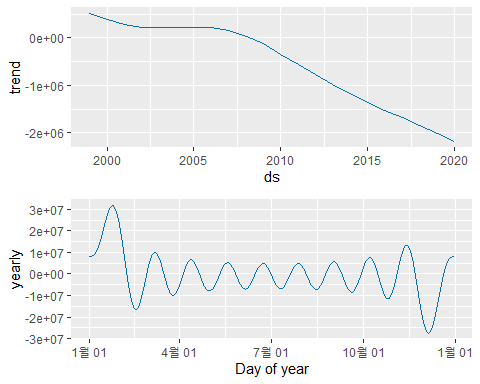
prophet 모델은 ARIMA 모델 같이 시계열 데이터의 구조를 통해 관계를 분석하는 모델과 달리 데이터 구조가 어떻던 데이터에 적합한 커브를 그리는데 목표를 두고 있다. 이렇게 함으로써 여러 주기를 가지는 계절성에 쉽게 사용할 수 있고, ARIMA 모델과 같이 일정한 데이터 주기를 맞추기 위해 데이터를 채워넣는 작업을 피할수 있고, 매우 빠르게 다양한 모델 스펙을 경험할 수 있고, 에 적합시킬 수 있으며, 회귀에 경험이 많은 분석가들의 개념에 쉽게 이해될 수 있는 형태의 매개변수들을 사용한다는 장점이 있다고 한다.

prophet 모델을 적용하기 위해서는 prophet()를 사용한다. prophet()은 data.frame 객체를 대상으로 사용되는데 data.frame에는 컬럼 이름이 ’ds’와 ’y’으로 설정된 두개의 데이터 필드가 필요하다. ds 컬럼은 date 데이터 타입이어야 하며 y 컬럼은 수치 컬럼이어야 한다.

library(prophet)  
students.prophet <- data.frame(ds = students$연도, y = students$학생수계)   
model.prophet.students <- prophet(students.prophet)  
future.students <- make\_future\_dataframe(model.prophet.students, periods = 10, freq = 'year')  
forecast.students <- predict(model.prophet.students, newdata = future.students)  
plot(model.prophet.students, forecast.students) +   
 ggrepel::geom\_text\_repel(aes(label = scales::number(y, big.mark = ',', accuracy = 1)), vjust = 1, size = 3) +  
 labs(title = 'prophet model', x = '연도', y = '학생수') +   
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ','))

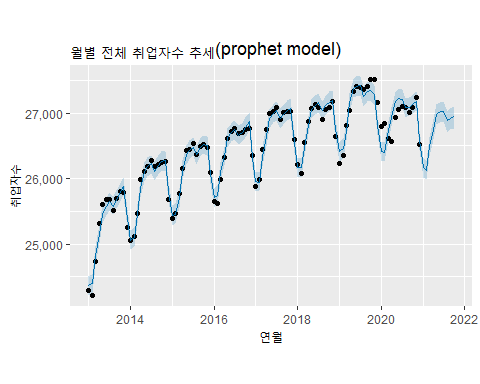


prophet\_plot\_components(model.prophet.students, forecast.students)

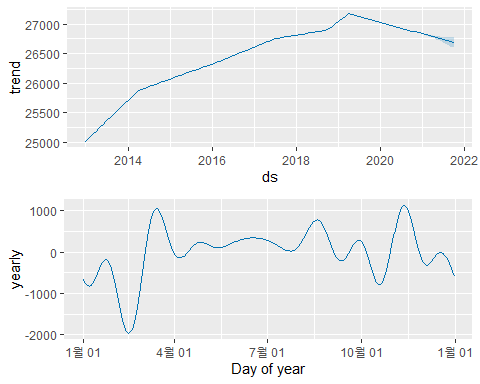


* 코드 설명
  + ds컬럼(students의 연도컬럼(date 클래스로 변환), students의 학생수계)과 y컬럼을 가지는 데이프레임을 생성
  + prophet()을 사용하여 prophet 모델을 생성
  + make\_future\_dataframe()을 사용하여 미래예측 데이터를 담을 빈데이터 프레임(시간 컬럼만 있는) 생성
  + predict()로 미래예측 데이터를 생성하는데 모델로 prophet모델, 예측에 사용될 신규 데이터로는 make\_future\_dataframe()으로 만들어놓은 미래예측 데이터 프레임을 사용
  + plot()을 사용하여 미래 예측 데이터의 plot을 생성, prophet모델이 사용하는 plot()은 ggplot()를 기반으로 작성된 plot()의 랩핑함수이므로 ggplot()의 기능을 이용하여 plot을 변경할 수 있음.
  + prophet\_plot\_components()로 prophet 예측에 사용된 컴포넌트들을 plot

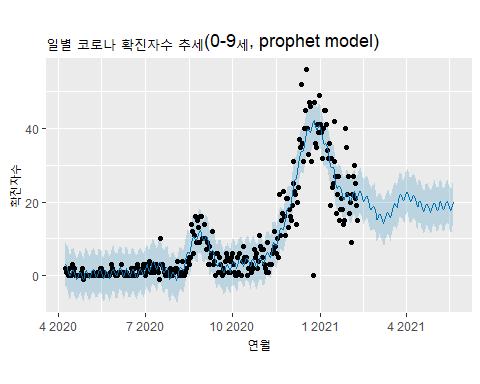
employees.prophet <- data.frame(ds = employees[,1], y = employees[,2])  
model.prophet.employees <- prophet(employees.prophet)  
future.employees <- make\_future\_dataframe(model.prophet.employees, periods = 10, freq = 'month')  
forecast.employees <- predict(model.prophet.employees, future.employees)  
plot(model.prophet.employees, forecast.employees) +  
 labs(title = '월별 전체 취업자수 추세(prophet model)', x = '연월', y = '취업자수') +   
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ','))



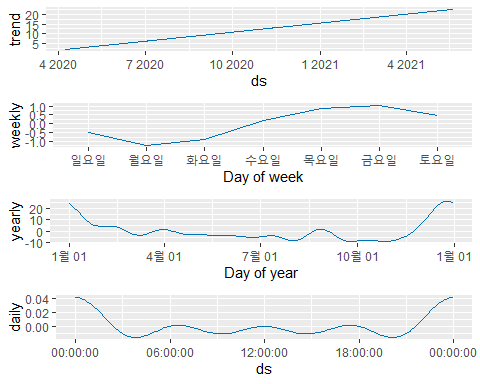
prophet\_plot\_components(model.prophet.employees, forecast.employees)



covid.prophet <- data.frame(ds = covid19$date, y = covid19$`0-9세`)  
model.prophet.covid <- prophet(covid.prophet, yearly.seasonality=TRUE, daily.seasonality=TRUE, weekly.seasonality=TRUE)  
future.covid <- make\_future\_dataframe(model.prophet.covid, periods = 100, freq = 'day')  
forecast.covid <- predict(model.prophet.covid, future.covid)  
plot(model.prophet.covid, forecast.covid) +  
 labs(title = '일별 코로나 확진자수 추세(0-9세, prophet model)', x = '연월', y = '확진자수') +   
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ','))



prophet\_plot\_components(model.prophet.covid, forecast.covid)



## 신경망(Neural Network) 모델

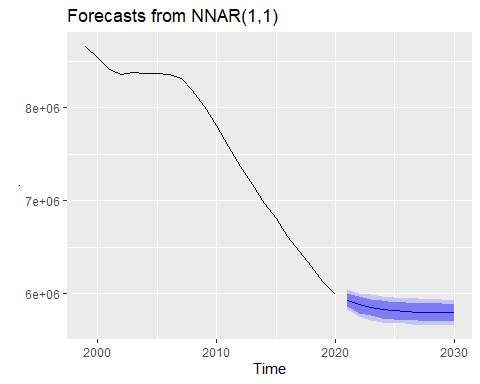
최근 우리는 딥러닝이라는 인공지능 알고리즘을 흔히 듣고 산다. 사실 딥러닝은 다양한 머신러닝 알고리즘 중에 하나로 이번 절에서 설명하려는 신경망(Neural Network) 모델에서 파생된 모델이다. 여기서는 신경망에 대해 깊게 설명하지는 않겠지만 신경망의 기본은 인간의 뇌세포 구조를 본따서 입력 레이어, 히든 레이어, 출력 레이어를 비용함수들로 연결하는 모델인데 히든레이어가 3개 이상 되는 신경망 모델을 딥러닝 모델이라 부른다.

시계열 데이터에 대한 신경망 모델은 forecast 패키지에서 제공하는 nnetar() 함수를 사용하여 구축할 수 있다. nnetar()로 구축되는 모델은 싱글 히든 레이어을 사용한 신경망 모델이다. 따라서 딥러닝이라고까지는 언급할 수 없으나 시계열 데이터를 신경망 이론에 적합하여 만들수 있는 모델이라는데 의미가 있다.

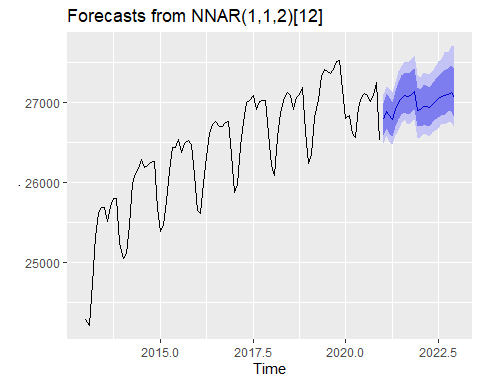
nnetar() 함수에 구축된 모델은 계절성이 없는 경우는 NNAR(p, k)이며 계절성이 없는 경우는 NNAR(p, P, k)[m]으로 표현된다. p값은 히든 레이어에서 예측값을 산출하기 위해 사용하는 과거 데이터의 갯수이고 k값은 히든 레이어에 존재하는 신경세포의 갯수이다. P는 계절성에 대한 lag 값이다. nnetar() 함수는 p와 P 값을 자동으로 산정하고 k 값은 (p + P +1)/2 값으로 설정한다.[[17]](#footnote-79)

NNAR모델의 단점중에 하나는 예측구간의 계산이 원활하지 못하다는 점이다. forecast() 함수에서 ‘PI = TRUE’ 매개변수를 설정하면 예측구간이 계산되지만 계산량이 많아 시간이 걸린다.

# 학생수에 대한 NNAR 모델은 NNAR(1, 1)모델  
students.ts[,2] %>% nnetar() %>% forecast(PI = TRUE) %>% autoplot()

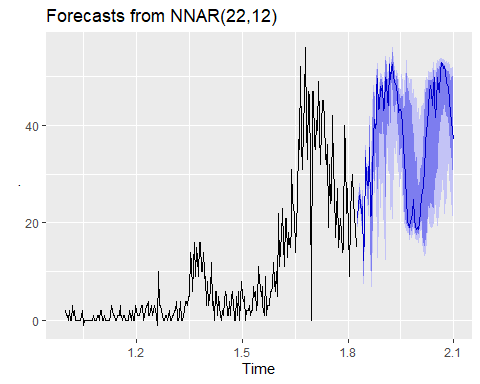


# 전체 취업자수에 대한 모델은 NNAR(1, 1, 2)[12] 모델  
employees.ts[, 2] %>% nnetar() %>% forecast(PI = TRUE) %>% autoplot()



# 코로나 확진자수에 대한 모델은 NNAR(22, 12) 모델(예측값을 위해 22개의 과거 데이터를 활용했고 히든레이어에 12개의 신경세포를 생성 )  
covid19.ts[,2] %>% nnetar() %>% forecast(h = 100, PI = TRUE) %>% autoplot()

Warning in nnetar(.): Series too short for seasonal lags



1. <https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique> [↑](#footnote-ref-20)
2. <https://otexts.com/fpp2/simple-methods.html> [↑](#footnote-ref-23)
3. <https://robjhyndman.com/files/solutions.pdf> [↑](#footnote-ref-27)
4. 이기준 외, 인구지형변화에 따른 머신러닝 기반 고등교육 계열별 수요예측 모형 개발(2020), 한국교육개발원 [↑](#footnote-ref-30)
5. <https://otexts.com/fppkr/regression-evaluation.html> [↑](#footnote-ref-32)
6. <https://otexts.com/fppkr/expsmooth.html> [↑](#footnote-ref-37)
7. <https://otexts.com/fpp2/ses.html> [↑](#footnote-ref-39)
8. <https://m.blog.naver.com/PostView.nhn?blogId=sigmagil&logNo=221502514892&proxyReferer=https:%2F%2Fwww.google.com%2F> [↑](#footnote-ref-41)
9. <https://otexts.com/fppkr/ets.html> [↑](#footnote-ref-49)
10. <https://icim.nims.re.kr/post/easyMath/68> [↑](#footnote-ref-53)
11. <https://otexts.com/fppkr/MA.html> [↑](#footnote-ref-57)
12. <https://otexts.com/fpp2/non-seasonal-arima.html> [↑](#footnote-ref-61)
13. <https://otexts.com/fpp3/ftransformations.html> [↑](#footnote-ref-69)
14. <https://pkg.robjhyndman.com/forecast/reference/tbats.html> [↑](#footnote-ref-71)
15. <https://facebook.github.io/prophet/> [↑](#footnote-ref-74)
16. <https://peerj.com/preprints/3190.pdf> [↑](#footnote-ref-76)
17. <https://otexts.com/fpp3/nnetar.html> [↑](#footnote-ref-79)