# Prophet 라이브러리

메타(페이스북)에서 발표한 시계열 분석 머신러닝 라이브러리

`pip install prophet`

> https://facebook.github.io/prophet/

> https://peerj.com/preprints/3190.pdf

## #01. 패키지 참조

```
from prophet import Prophet
from prophet.plot import add_changepoints_to_plot
from pandas import read_excel, DataFrame, concat, Series, date_range
from pandas import to_datetime
from matplotlib import pyplot as plt
```

## #02. 데이터 가져오기

```
#origin = read_excel("https://data.hossam.kr/G02/daily_mini_temp.xlsx")
origin = read_excel("daily_mini_temp.xlsx")
print(origin.info())
origin.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3650 entries, 0 to 3649
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   날짜      3650 non-null   object
 1   최저기온    3650 non-null   float64
dtypes: float64(1), object(1)
memory usage: 57.2+ KB
None
```

| | 날짜 | 최저기온 |
|---|---|---|
| 0 | 1/1/1981 | 20.7 |
| 1 | 1/2/1981 | 17.9 |
| 2 | 1/3/1981 | 18.8 |
| 3 | 1/4/1981 | 14.6 |
| 4 | 1/5/1981 | 15.8 |

## #03. 데이터 전처리

### 날짜 컬럼의 데이터타입 수정

```
df = origin.copy()
df['날짜'] = to_datetime(df['날짜'], format='%m/%d/%Y')
df.head()
```

| | 날짜 | 최저기온 |
|---|---|---|
| 0 | 1981-01-01 | 20.7 |
| 1 | 1981-01-02 | 17.9 |
| 2 | 1981-01-03 | 18.8 |
| 3 | 1981-01-04 | 14.6 |
| 4 | 1981-01-05 | 15.8 |

### 데이터 프레임의 필드 이름 변경

prophet 라이브러리의 요구사항에 따라 날짜 필드의 이름을 `ds`로, 데이터의 필드 이름을 `y`로 변경해야 한다.

```
df.rename(columns={'날짜': 'ds', '최저기온': 'y'}, inplace=True)
df.head()
```

| | ds | y |
|---|---|---|
| 0 | 1981-01-01 | 20.7 |
| 1 | 1981-01-02 | 17.9 |
| 2 | 1981-01-03 | 18.8 |
| 3 | 1981-01-04 | 14.6 |
| 4 | 1981-01-05 | 15.8 |

## #04. 시계열 분석 모델 구현

### 모델 정의 및 학습

```
m = Prophet()
m.fit(df)
```

```
10:03:26 - cmdstanpy - INFO - Chain [1] start processing
10:03:27 - cmdstanpy - INFO - Chain [1] done processing
```

```
<prophet.forecaster.Prophet at 0×198bf2f9be0>
```

### 학습결과를 활용하여 예측

`periods`는 예측할 기간 (일(`D`), 월(`M`) 단위 설정 가능 / 기본은 일단위)

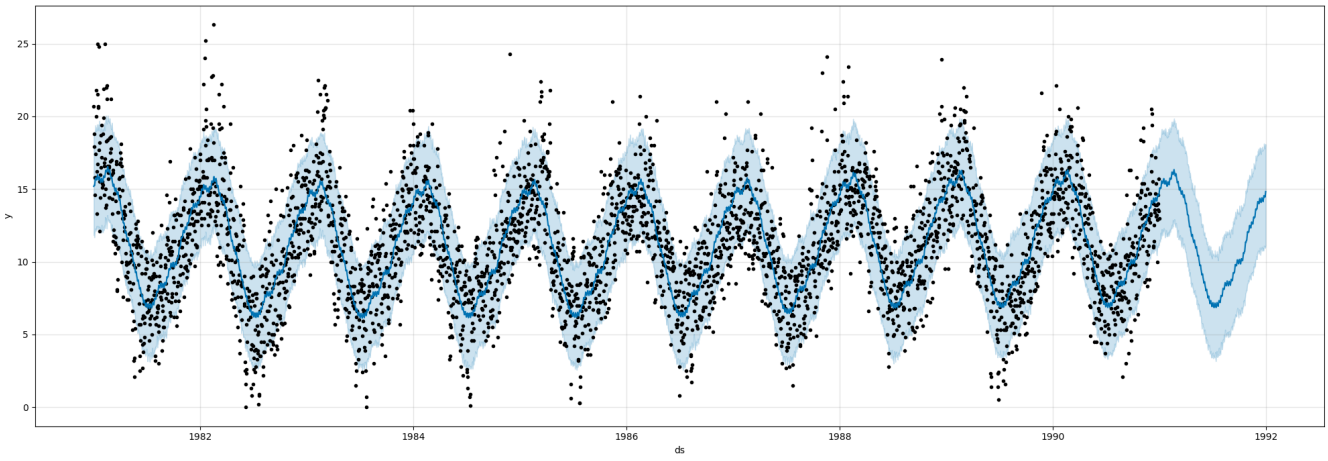
```
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)
forecast.head()
```

| | ds | trend | yhat_lower | yhat_upper | trend_lower | trend_upper | additive_terms | additi |
|---|---|---|---|---|---|---|---|---|
| 0 | 1981-01-01 | 11.858854 | 11.834831 | 18.831322 | 11.858854 | 11.858854 | 3.328315 | 3.3283 |
| 1 | 1981-01-02 | 11.857015 | 12.047962 | 18.792508 | 11.857015 | 11.857015 | 3.375099 | 3.3750 |
| 2 | 1981-01-03 | 11.855177 | 11.674667 | 18.817311 | 11.855177 | 11.855177 | 3.411920 | 3.4119 |
| 3 | 1981-01-04 | 11.853338 | 11.905813 | 18.935321 | 11.853338 | 11.853338 | 3.394334 | 3.3943 |
| 4 | 1981-01-05 | 11.851499 | 12.204500 | 18.765032 | 11.851499 | 11.851499 | 3.600142 | 3.6001 |

## 예측 결과에 대한 시각화

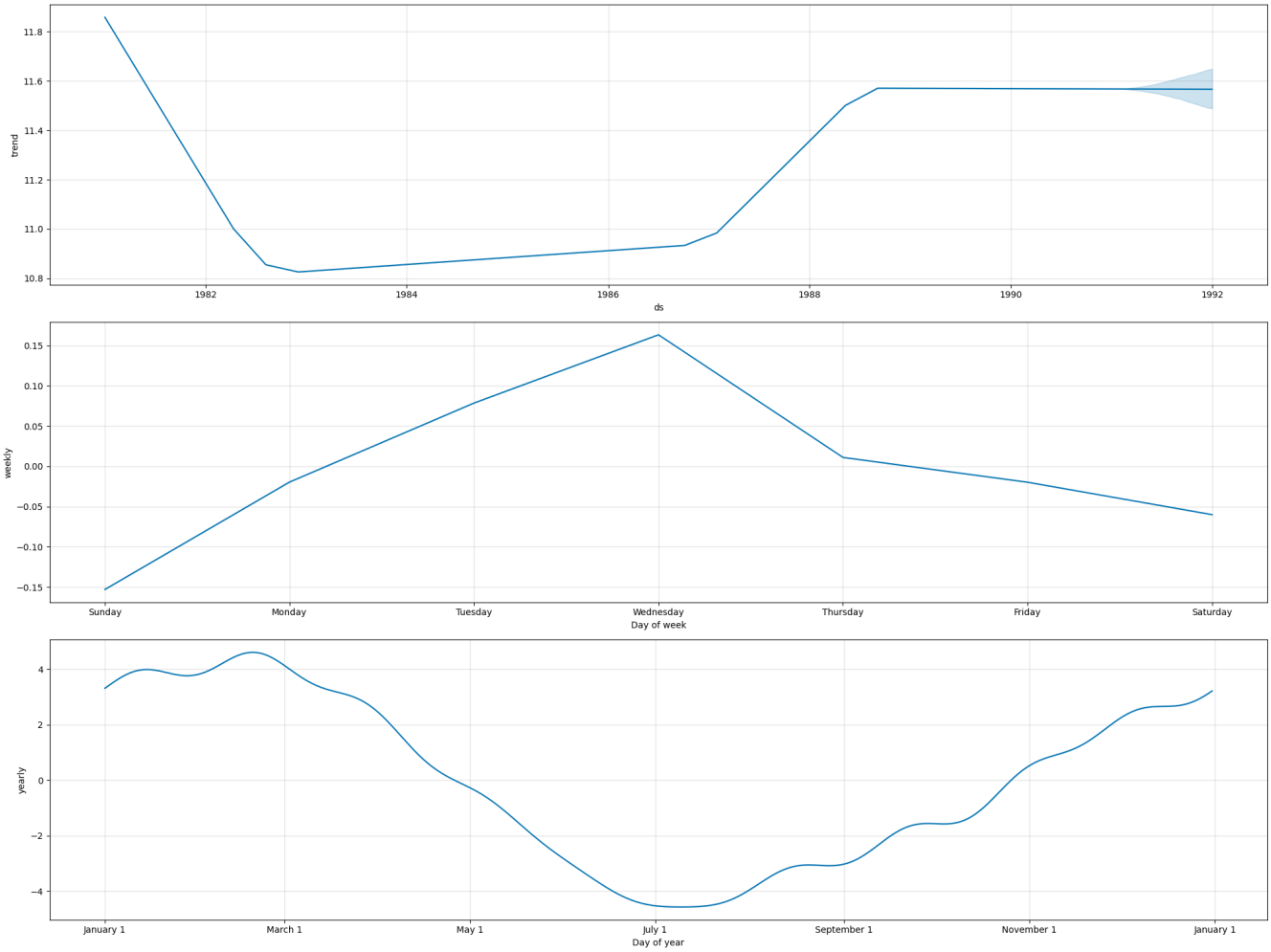
- 검은색 점 : 실 데이터
- 파란색 선 : 모델이 예측한 데이터

```
fig = m.plot(forecast, figsize=(20, 7))
```




## 모델이 갖는 컴포넌트 확인

```
fig = m.plot_components(forecast, figsize=(20, 15))
```



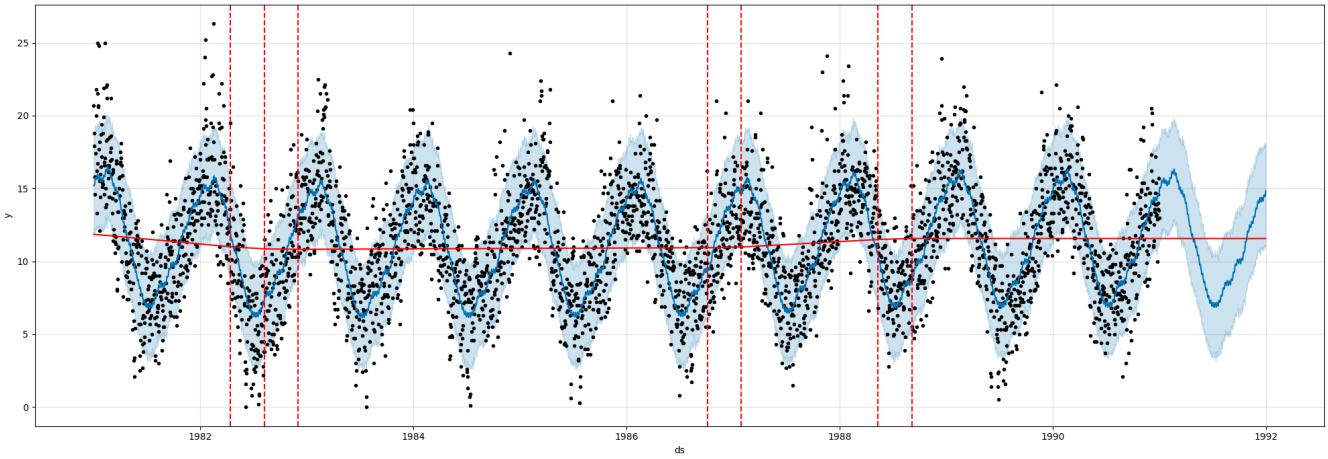

## changepoint 를 추가한 시각화

changepoint - Trend가 변화하는 지점

- fig.gca() : 시계열 그래프
- m : 미리 만들어준 Prophet 객체
- forecast : predict 결과

```
fig = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)
```

```
[[<matplotlib.lines.Line2D at 0×198c6688490>],
 <matplotlib.lines.Line2D at 0×198c6688c70>,
 <matplotlib.lines.Line2D at 0×198c668f040>,
 <matplotlib.lines.Line2D at 0×198c668f6a0>,
 <matplotlib.lines.Line2D at 0×198c668f9d0>,
 <matplotlib.lines.Line2D at 0×198c66885e0>,
 <matplotlib.lines.Line2D at 0×198c4f06760>,
 <matplotlib.lines.Line2D at 0×198c667f5e0>]
```




> 빨간색 실선 : 트랜드 / 빨간색 점선 : 트랜드가 변화하는 changepoint

## #05. 모델의 Trend 조절

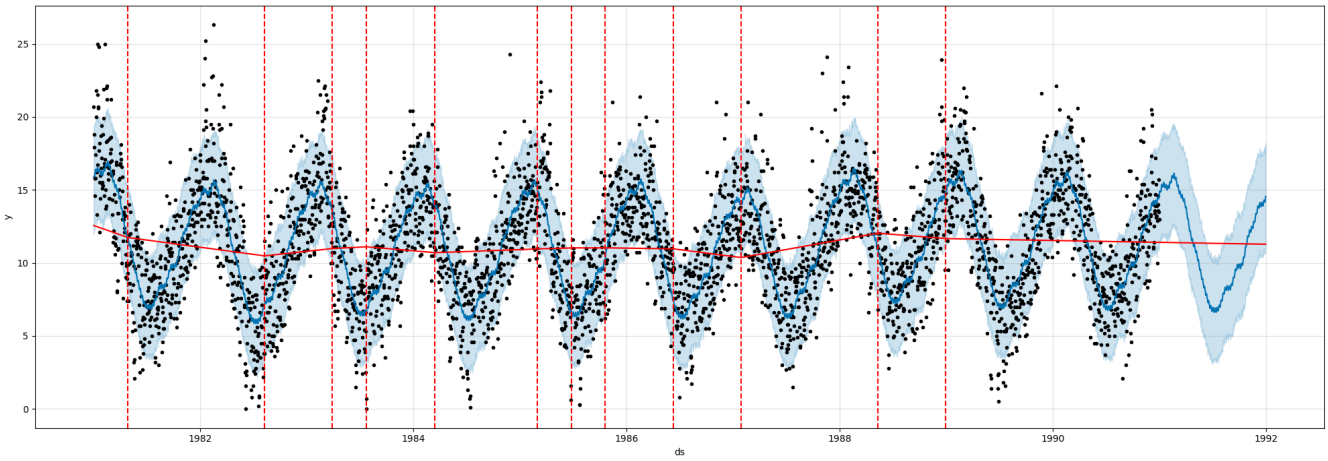
| 파라미터 | 설명 |
|---|---|
| `changepoints` | 트렌드 변화시점을 명시한 리스트 |
| `changepoint_prior_scale` | trend의 유연성 조절(기본값=0.05) |
| `n_changepoints` | chagepoint의 개수 |
| `changepoint_range` | chagepoint 설정 가능 범위 (기본적으로 데이터 중 `80%` 범위 내에서 changepoint를 설정한다) |

```
# 모델 구성
m = Prophet(changepoint_prior_scale=0.3)
m.fit(df)

# 예측
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)

# 시각화
fig = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)
```

```
10:52:30 - cmdstanpy - INFO - Chain [1] start processing
10:52:31 - cmdstanpy - INFO - Chain [1] done processing
```

```
[[<matplotlib.lines.Line2D at 0×198c6688f70>],
 <matplotlib.lines.Line2D at 0×198c4ebb970>,
 <matplotlib.lines.Line2D at 0×198c0c21760>,
 <matplotlib.lines.Line2D at 0×198c0cdc4f0>,
 <matplotlib.lines.Line2D at 0×198c66cb790>,
 <matplotlib.lines.Line2D at 0×198c66cb6a0>,
 <matplotlib.lines.Line2D at 0×198c66cb2e0>,
 <matplotlib.lines.Line2D at 0×198c10db100>,
 <matplotlib.lines.Line2D at 0×198c10db430>,
 <matplotlib.lines.Line2D at 0×198c0bf5ac0>,
 <matplotlib.lines.Line2D at 0×198c10db730>,
```

```
 <matplotlib.lines.Line2D at 0x198c10dba60>,
 <matplotlib.lines.Line2D at 0x198c10dbd90>]
```



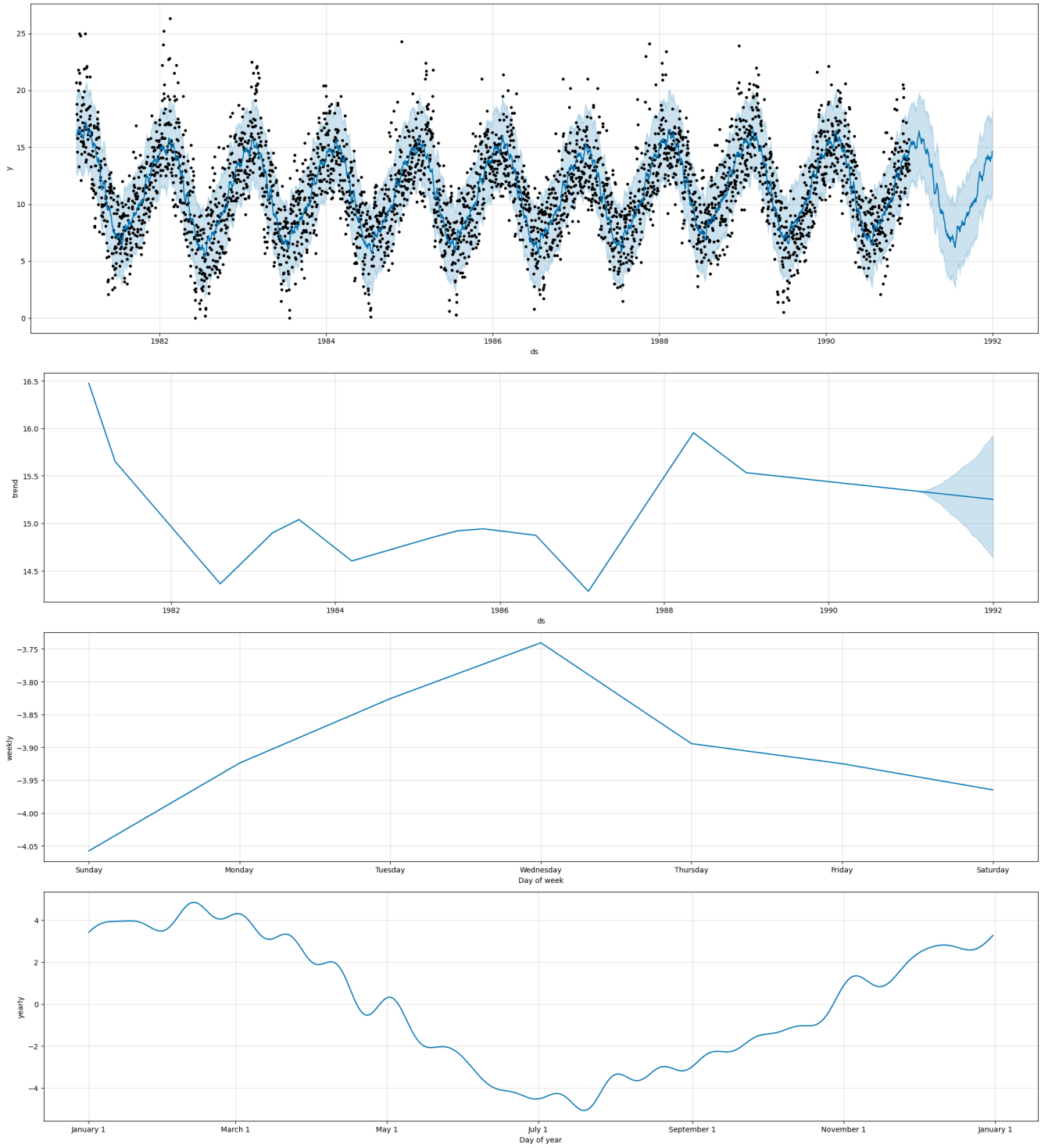

## #06. 모델의 계절성(Seasonality) 조절

| 파라미터 | 설명 |
|---|---|
| `yearly_seasonality` | 연 계절성(기본값=auto) |
| `weekly_seasonality` | 주 계절성(기본값=auto) |
| `daily_seasonality` | 일 계절성(기본값=auto) |
| `seasonality_prior_scale` | 계절성 반영 강도 |
| `seasonality_mode` | `additive`(기본값) or `multiplicative` |

> 푸리에급수 계절성 패턴을 추정.

```
m = Prophet(
    # Trend
    changepoint_prior_scale=0.3,
    # Seasonality
    weekly_seasonality=10,
    yearly_seasonality=20,
    daily_seasonality=False
)

m.fit(df)

# 예측
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)

# 시각화
fig1 = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)

fig2 = m.plot_components(forecast, figsize=(20, 15))
```

```
11:03:26 - cmdstanpy - INFO - Chain [1] start processing
11:03:27 - cmdstanpy - INFO - Chain [1] done processing
```

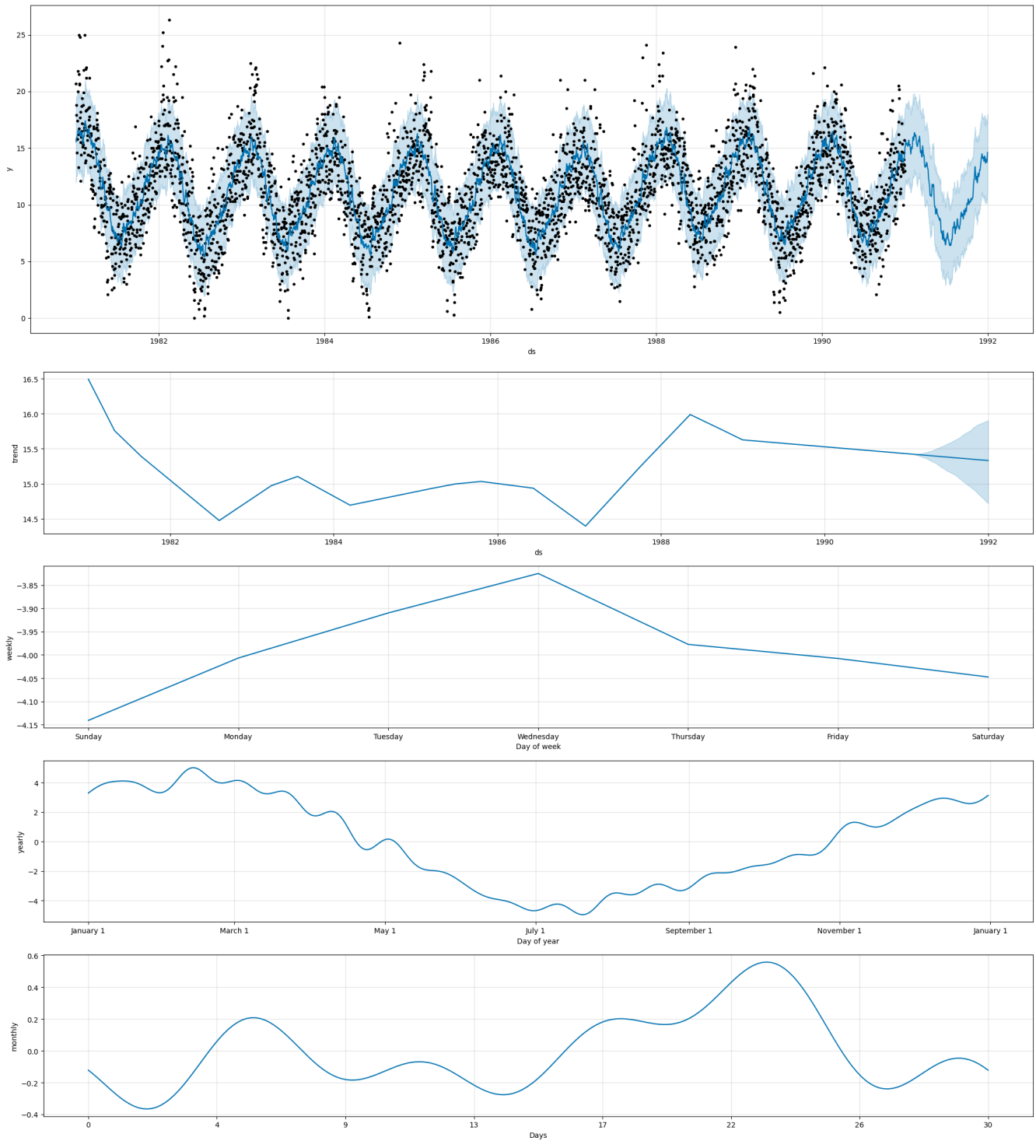



## #07. 계절성 주기 직접 생성

```
m = Prophet(
    # Trend
    changepoint_prior_scale=0.3,
    # Seasonality
    weekly_seasonality=10,
    yearly_seasonality=20,
    daily_seasonality=False
)

# 월단위 계절성 추가
m.add_seasonality(name='monthly', period=30.5, fourier_order=5)
```

```
m.fit(df)

# 예측
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)

# 시각화
fig1 = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)

fig2 = m.plot_components(forecast, figsize=(20, 15))
```

```
11:11:36 - cmdstanpy - INFO - Chain [1] start processing
11:11:38 - cmdstanpy - INFO - Chain [1] done processing
```



## #08. 공휴일 지정

### 직접 공휴일 데이터프레임 생성

`holiday`, `ds` 두 개의 필드를 갖는 데이터 프레임을 생성해야 한다.

`holiday` 필드에는 모두 `holiday`라는 문자열 값을 설정

`ds`필드에는 날짜값을 설정

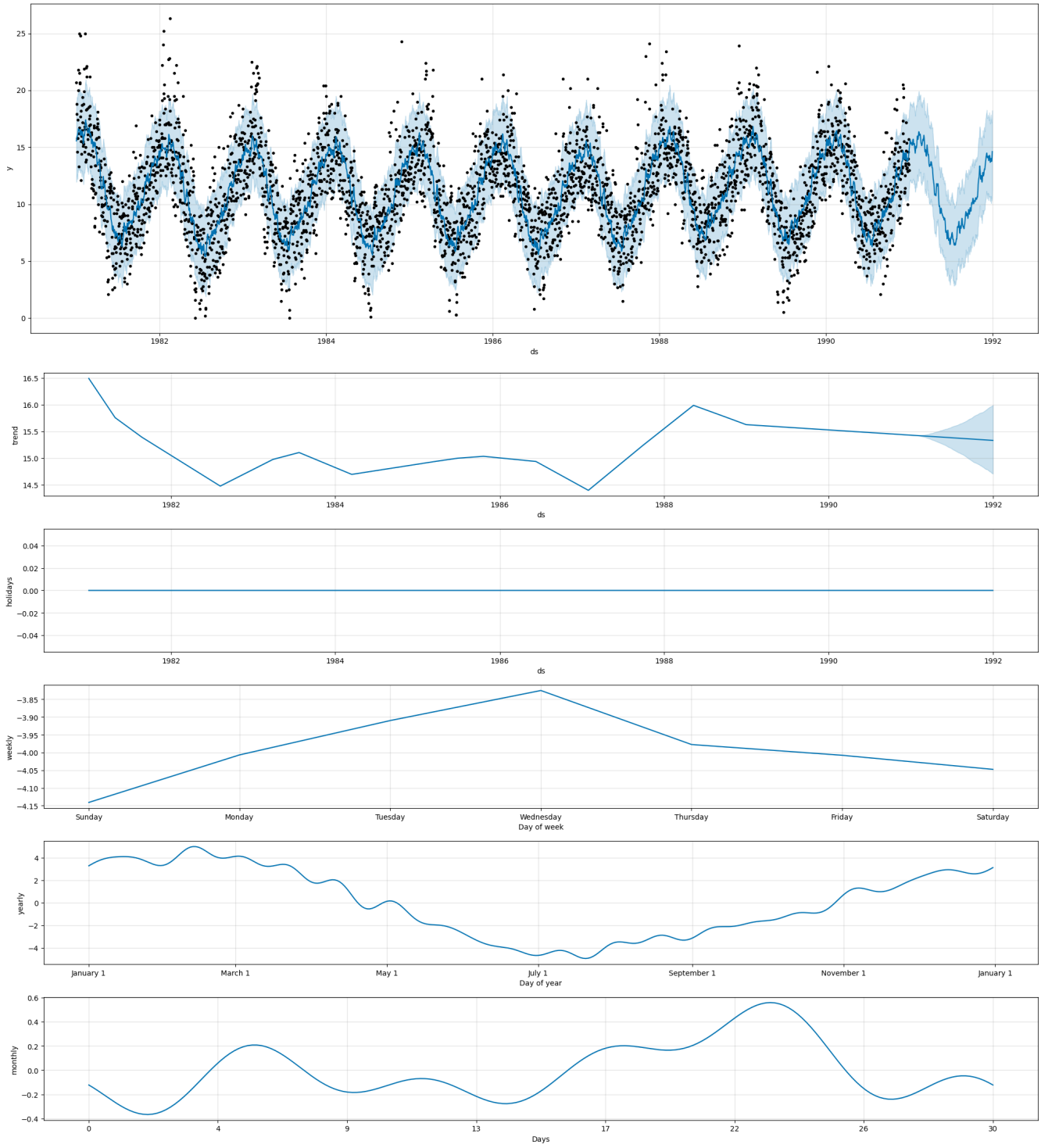
```
holidays = DataFrame({
    'holiday': 'holiday',
    'ds': concat([
        Series(date_range('2017-05-05', '2017-06-03', freq='D')),
        Series(date_range('2018-05-05', '2018-06-03', freq='D')),
        Series(date_range('2019-05-05', '2019-06-03', freq='D')),
        Series(date_range('2020-05-05', '2020-06-03', freq='D'))
    ])
})

holidays.head()
```

| | holiday | ds |
|---|---|---|
| 0 | holiday | 2017-05-05 |
| 1 | holiday | 2017-05-06 |
| 2 | holiday | 2017-05-07 |
| 3 | holiday | 2017-05-08 |
| 4 | holiday | 2017-05-09 |

```
m = Prophet(
    # Trend
    changepoint_prior_scale=0.3,
    # Seasonality
    weekly_seasonality=10,
    yearly_seasonality=20,
    daily_seasonality=False,
    # Holidays
    holidays=holidays,
    holidays_prior_scale = 15
)

# 월단위 계절성 추가
m.add_seasonality(name='monthly', period=30.5, fourier_order=5)

m.fit(df)

# 예측
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)

# 시각화
```

```
fig1 = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)

fig2 = m.plot_components(forecast, figsize=(20, 15))
```

```
11:36:48 - cmdstanpy - INFO - Chain [1] start processing
11:36:50 - cmdstanpy - INFO - Chain [1] done processing
```



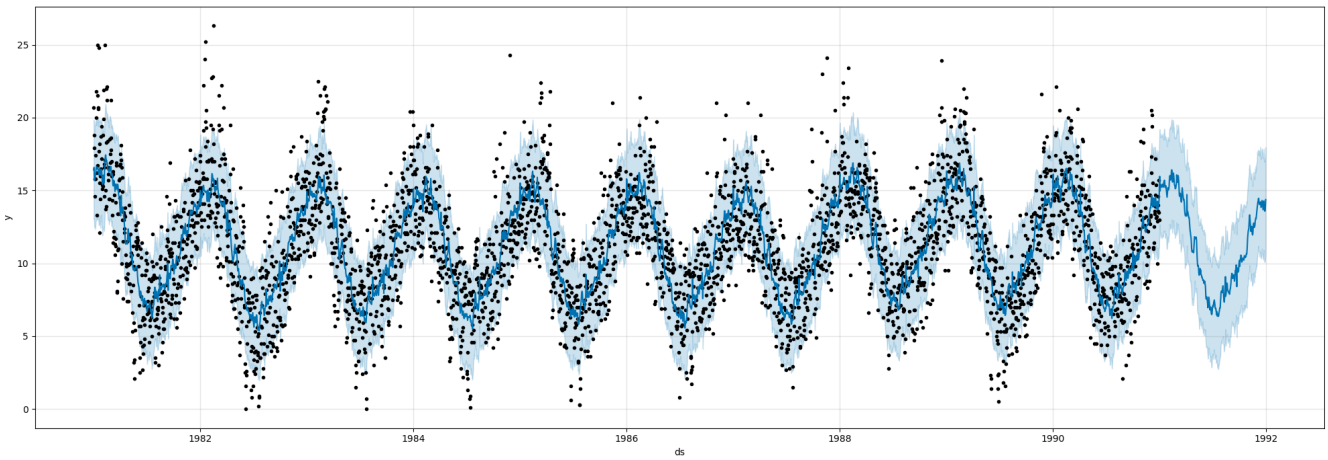

## API 기능을 활용한 공휴일 지정

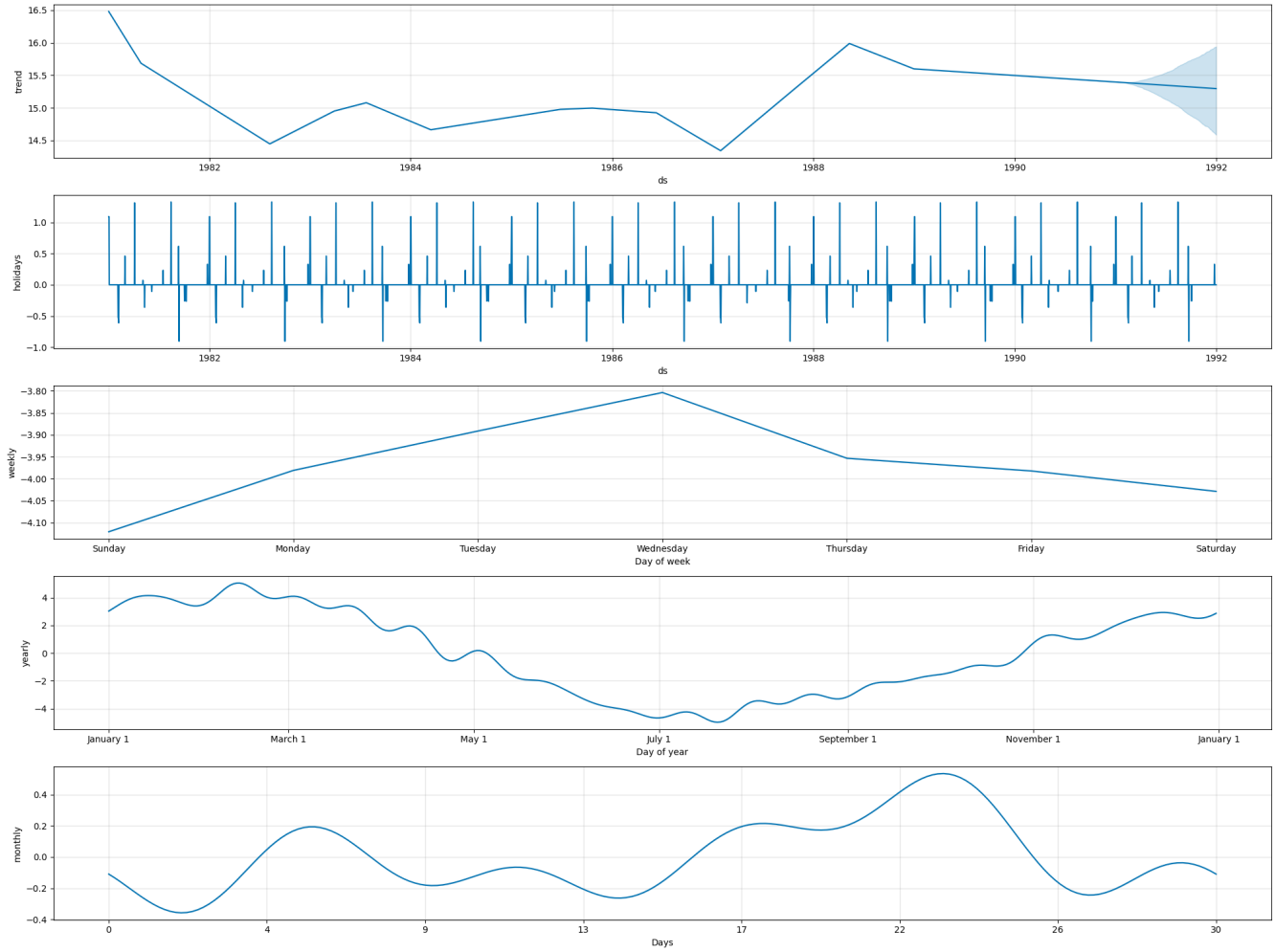
### 현재 사용가능한 국가 코드 목록

Brazil (BR), Indonesia (ID), India (IN), Malaysia (MY), Vietnam (VN), Thailand (TH), Philippines (PH), Pakistan (PK), Bangladesh (BD), Egypt (EG), China (CN), and Russian (RU), Korea (KR), Belarus (BY), and United Arab Emirates (AE)

```
m = Prophet(
    # Trend
    changepoint_prior_scale=0.3,
    # Seasonality
    weekly_seasonality=10,
    yearly_seasonality=20,
    daily_seasonality=False
)

# 공휴일 데이터 추가
m.add_country_holidays(country_name='KR')

# 월단위 계절성 추가
m.add_seasonality(name='monthly', period=30.5, fourier_order=5)

m.fit(df)

# 예측
future = m.make_future_dataframe(periods=365, freq='D')
forecast = m.predict(future)

# 시각화
fig1 = m.plot(forecast, figsize=(20, 7))
add_changepoints_to_plot(fig.gca(), m, forecast)

fig2 = m.plot_components(forecast, figsize=(20, 15))
```

```
11:40:28 - cmdstanpy - INFO - Chain [1] start processing
11:40:30 - cmdstanpy - INFO - Chain [1] done processing
```






### 사용된 공휴일 확인

```
m.train_holiday_names
```

```
0                       New Year's Day
1                       Lunar New Year
2     The day preceding Lunar New Year
3     The second day of Lunar New Year
4            Independence Movement Day
5                    Tree Planting Day
6                    Buddha's Birthday
7                       Children's Day
8                         Memorial Day
9                    Constitution Day
10                      Liberation Day
11             National Foundation Day
12                           Hangul Day
13                              Chuseok
14           The day preceding Chuseok
15           The second day of Chuseok
16                        Christmas Day
dtype: object
```

```
```