In [1]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in d:\anaconda3\lib\site-packag
es (4.9.0)
Requirement already satisfied: retrying>=1.3.3 in d:\anaconda3\lib\si
te-packages (from plotly) (1.3.3)
Requirement already satisfied: six in d:\anaconda3\lib\site-packages
(from plotly) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import numpy as np
from scipy import stats
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.api import qqplot
import statsmodels.api as sm
from pandas import Series, DataFrame, Panel
import plotnine
from pandas import Timestamp
from ggplot import*
from plotnine import *
from plotly.tools import mpl_to_plotly as ggplotly
```

```
<ipython-input-1-9ff5a2b86c0d>:9: FutureWarning: The Panel class is r
emoved from pandas. Accessing it from the top-level namespace will al
so be removed in the next version
  from pandas import Series, DataFrame, Panel
```

In [2]:
```
dta1= pd.read_csv("D:/Documents/Data/unemp.csv")
dta2= pd.read_csv("D:/Documents/Data/extpred.csv")
```

In [3]:
```
dta1.index = pd.Index(sm.tsa.datetools.dates_from_range('1997m6', '2005
m9'))
del dta1["date"]
dta2.index = pd.Index(sm.tsa.datetools.dates_from_range('1997m6', '2005
m9'))
del dta2["date"]
```

In [4]:
```
def floor_decade(date_value):
    "Takes a date. Returns the decade."
    return (date_value.year // 10) * 10

pd.to_datetime(2013-10-9)
```

Out[4]:
```
Timestamp('1970-01-01 00:00:00.000001994')
```

In [5]:
```
floor_decade(_)
```

Out[5]:
```
1970
```

In [6]:
```
import ggplot
from ggplot import *
```

In [7]:
```
dta2.head()
```

Out[7]:

	cci	11-Sep
1997-06-30	129.9	0
1997-07-31	126.3	0
1997-08-31	127.6	0
1997-09-30	130.2	0
1997-10-31	123.4	0

In [8]:
```
from pandasql import sqldf, load_meat
```

In [9]:
```
import pandasql
from pandasql import sqldf, load_meat, load_births
pysqldf = lambda q: sqldf(q, globals())
meat = load_meat()
births = load_births()
#print pysqldf("SELECT * FROM meat LIMIT 10;").head()
```

In [10]:
```
meat.head()
```

Out[10]:

	date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
0	1944-01-01	751.0	85.0	1280.0	89.0	NaN	NaN	NaN
1	1944-02-01	713.0	77.0	1169.0	72.0	NaN	NaN	NaN
2	1944-03-01	741.0	90.0	1128.0	75.0	NaN	NaN	NaN
3	1944-04-01	650.0	89.0	978.0	66.0	NaN	NaN	NaN
4	1944-05-01	681.0	106.0	1029.0	78.0	NaN	NaN	NaN

In [11]:
```
ts = meat.set_index(['date'])
```

In [12]:
```
ts = ts.rename(columns={'lamb_and_mutton':'lamb', 'other_chicken':'poul
try'})
```

In [13]:
```
ts = ts[['beef','pork','lamb','veal','poultry']]
```

In [14]:
```
ts = ts[60:]
```

In [ ]:
```

```

In [15]:
```
ts.head(40)
```

Out[15]:

date	beef	pork	lamb	veal	poultry
1949-01-01	779.0	915.0	60.0	91.0	NaN
1949-02-01	697.0	701.0	51.0	82.0	NaN
1949-03-01	794.0	737.0	47.0	98.0	NaN
1949-04-01	721.0	647.0	34.0	91.0	NaN
1949-05-01	747.0	630.0	38.0	92.0	NaN
1949-06-01	765.0	657.0	42.0	103.0	NaN
1949-07-01	743.0	584.0	45.0	103.0	NaN
1949-08-01	835.0	605.0	54.0	123.0	NaN
1949-09-01	821.0	637.0	55.0	121.0	NaN
1949-10-01	772.0	764.0	56.0	120.0	NaN
1949-11-01	751.0	957.0	52.0	119.0	NaN
1949-12-01	717.0	1041.0	53.0	97.0	NaN
1950-01-01	780.0	965.0	55.0	87.0	NaN
1950-02-01	676.0	699.0	46.0	80.0	NaN
1950-03-01	775.0	807.0	49.0	96.0	NaN
1950-04-01	694.0	702.0	44.0	87.0	NaN
1950-05-01	774.0	716.0	47.0	93.0	NaN
1950-06-01	753.0	718.0	48.0	97.0	NaN
1950-07-01	754.0	617.0	46.0	97.0	NaN
1950-08-01	829.0	633.0	52.0	108.0	NaN
1950-09-01	832.0	667.0	51.0	107.0	NaN
1950-10-01	814.0	806.0	51.0	105.0	NaN
1950-11-01	794.0	978.0	47.0	98.0	NaN
1950-12-01	773.0	1089.0	45.0	82.0	NaN
1951-01-01	842.0	1085.0	54.0	81.0	NaN
1951-02-01	650.0	720.0	38.0	66.0	NaN
1951-03-01	696.0	837.0	39.0	73.0	NaN
1951-04-01	658.0	813.0	36.0	67.0	NaN
1951-05-01	716.0	798.0	34.0	71.0	NaN
1951-06-01	591.0	798.0	39.0	78.0	NaN
1951-07-01	676.0	686.0	42.0	86.0	NaN
1951-08-01	767.0	753.0	44.0	98.0	NaN
1951-09-01	697.0	715.0	41.0	87.0	NaN
1951-10-01	789.0	885.0	53.0	106.0	NaN

date	beef	pork	lamb	veal	poultry
1951-11-01	768.0	1023.0	47.0	91.0	NaN
1951-12-01	699.0	1077.0	41.0	68.0	NaN
1952-01-01	810.0	1130.0	54.0	75.0	NaN
1952-02-01	721.0	945.0	52.0	66.0	NaN

In [16]:
```
beef=ts["beef"]
beef.head(10)
```

Out[16]:
```
date
1949-01-01    779.0
1949-02-01    697.0
1949-03-01    794.0
1949-04-01    721.0
1949-05-01    747.0
1949-06-01    765.0
1949-07-01    743.0
1949-08-01    835.0
1949-09-01    821.0
1949-10-01    772.0
Name: beef, dtype: float64
```

In [17]:
```
ts = meat.set_index(['date'])
beef=ts["beef"]
```

In [18]:
```
births.head(5)
```

Out[18]:

	date	births
0	1975-01-01	265775
1	1975-02-01	241045
2	1975-03-01	268849
3	1975-04-01	247455
4	1975-05-01	254545

In [19]:
```
ts2 = births.set_index(['date'])
ts2.head(5)
```

Out[19]:

date	births
1975-01-01	265775
1975-02-01	241045
1975-03-01	268849
1975-04-01	247455
1975-05-01	254545

In [20]:
```
ts.groupby(ts.index.year).sum()
```

Out[20]:

date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
1944	8801.0	1629.0	11502.0	1001.0	0.0	0.0	0.0
1945	9936.0	1552.0	8843.0	1030.0	0.0	0.0	0.0
1946	9010.0	1329.0	9220.0	946.0	0.0	0.0	0.0
1947	10096.0	1493.0	8811.0	779.0	0.0	0.0	0.0
1948	8766.0	1323.0	8486.0	728.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
2008	26561.2	143.1	23346.9	173.8	36906.3	559.3	6247.0
2009	25965.4	138.4	22999.0	170.7	35510.4	500.1	5663.4
2010	26304.3	134.2	22436.5	163.6	36909.8	503.9	5644.3
2011	26195.3	129.5	22758.2	148.7	37200.8	521.5	5790.4
2012	23891.9	108.2	21296.5	143.3	34179.8	480.0	5525.7

69 rows × 7 columns

In [21]:
```
ts
```

Out[21]:

date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
1944-01-01	751.0	85.0	1280.0	89.0	NaN	NaN	NaN
1944-02-01	713.0	77.0	1169.0	72.0	NaN	NaN	NaN
1944-03-01	741.0	90.0	1128.0	75.0	NaN	NaN	NaN
1944-04-01	650.0	89.0	978.0	66.0	NaN	NaN	NaN
1944-05-01	681.0	106.0	1029.0	78.0	NaN	NaN	NaN
...	...	...	...	...	...	...	...
2012-07-01	2200.8	9.5	1721.8	12.5	3127.0	43.4	497.2
2012-08-01	2367.5	10.1	1997.9	14.2	3317.4	51.0	530.1
2012-09-01	2016.0	8.8	1911.0	12.5	2927.1	43.7	453.1
2012-10-01	2343.7	10.3	2210.4	14.2	3335.0	43.8	579.9
2012-11-01	2206.6	10.1	2078.7	12.4	3006.7	37.5	515.3

827 rows × 7 columns

In [22]:
```
import numpy as np
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.api import qqplot
import statsmodels.api as sm
from pandas import Series, DataFrame, Panel
import matplotlib
from matplotlib import *
```

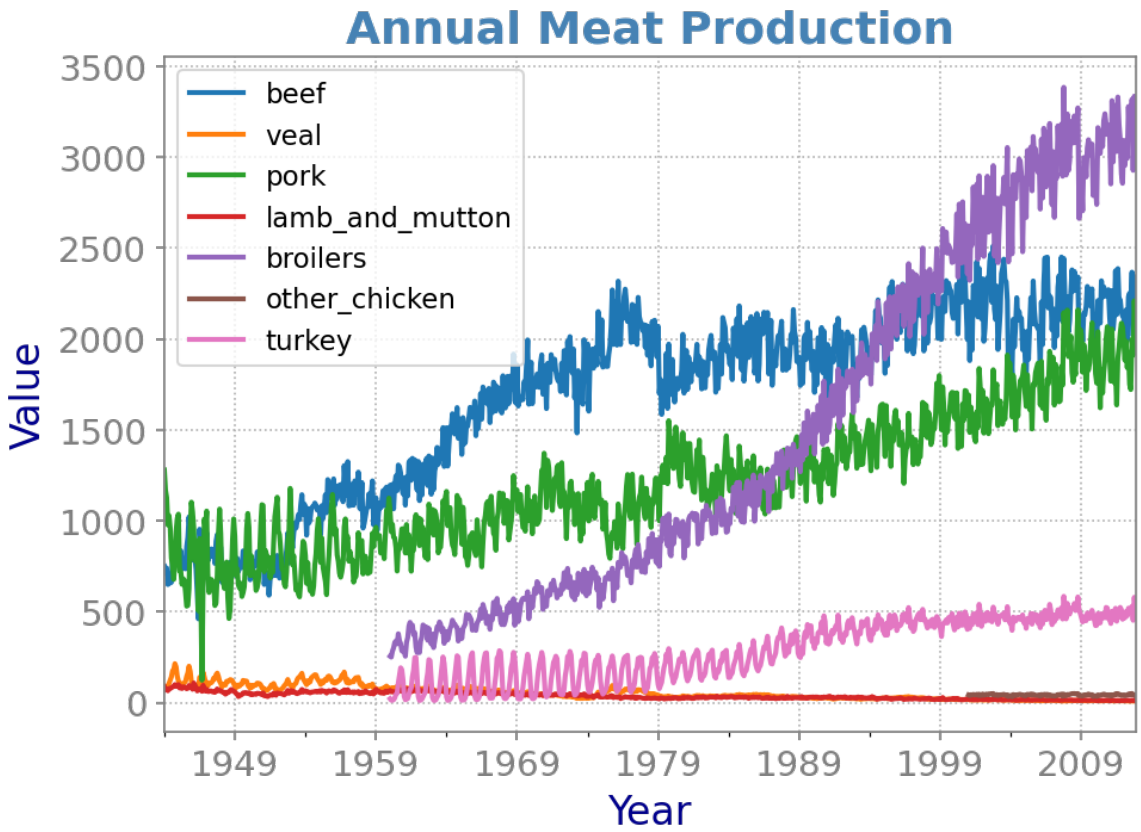
```
<ipython-input-22-16ca69bc74df>:8: FutureWarning:

The Panel class is removed from pandas. Accessing it from the top-lev
el namespace will also be removed in the next version
```

In [23]:
```
import matplotlib
from matplotlib import *
```

In [24]:
```
beef
```

Out[24]:
```
date
1944-01-01     751.0
1944-02-01     713.0
1944-03-01     741.0
1944-04-01     650.0
1944-05-01     681.0
               ...
2012-07-01    2200.8
2012-08-01    2367.5
2012-09-01    2016.0
2012-10-01    2343.7
2012-11-01    2206.6
Name: beef, Length: 827, dtype: float64
```

In [25]:
```
fig, ax = plt.subplots(figsize=(7, 5), dpi = 150)
#beef = df["cattle"]
ax = ts.plot(figsize=(7,5), linewidth=2, fontsize=14, ax=ax)
plt.xlabel('Year', fontsize=16, color = 'darkblue')
plt.ylabel('Value', fontsize=16, color = 'darkblue')
plt.title('Annual Meat Production', fontsize=18, color = 'steelblue', f
ontdict = dict(weight = 'bold'))
plt.legend(fontsize=11)
plt.grid(True, linestyle = ':')
ax.spines['bottom'].set_color('gray')
ax.spines['left'].set_color('gray')
ax.spines['right'].set_color('gray')
ax.spines['top'].set_color('gray')
ax.tick_params(axis='x', colors='gray')
ax.tick_params(axis='y', colors='gray')
plt.savefig('03_meat_ts.png', figsize = (7,5), dpi = 150, bbox_inches =
'tight')
plt.show()
#pyplot.show()
```



In [520]:
```
df = pd.read_csv("D:\\Documents\\DATA\\meat2.csv")
df['date'] = pd.to_datetime(df.date)
df.head(5)
```

Out[520]:

	date	cattle	hogs	sheep	poultry
0	1948-01-01	51.3	144.6	1464.0	10.4
1	1948-02-01	37.7	665.6	1306.7	87.6
2	1948-03-01	35.7	1242.0	1297.2	79.6
3	1948-04-01	45.1	377.5	1178.1	14.7
4	1948-05-01	52.9	833.5	1095.3	67.2

In [521]:
```
df.set_index('date', inplace=True)
df.index
```

Out[521]:
```
DatetimeIndex(['1948-01-01', '1948-02-01', '1948-03-01', '1948-04-01
',
               '1948-05-01', '1948-06-01', '1948-07-01', '1948-08-01
',
               '1948-09-01', '1948-10-01',
               ...
               '2019-10-01', '2019-11-01', '2019-12-01', '2020-01-01
',
               '2020-02-01', '2020-03-01', '2020-04-01', '2020-05-01
',
               '2020-06-01', '2020-07-01'],
              dtype='datetime64[ns]', name='date', length=871, freq=N
one)
```

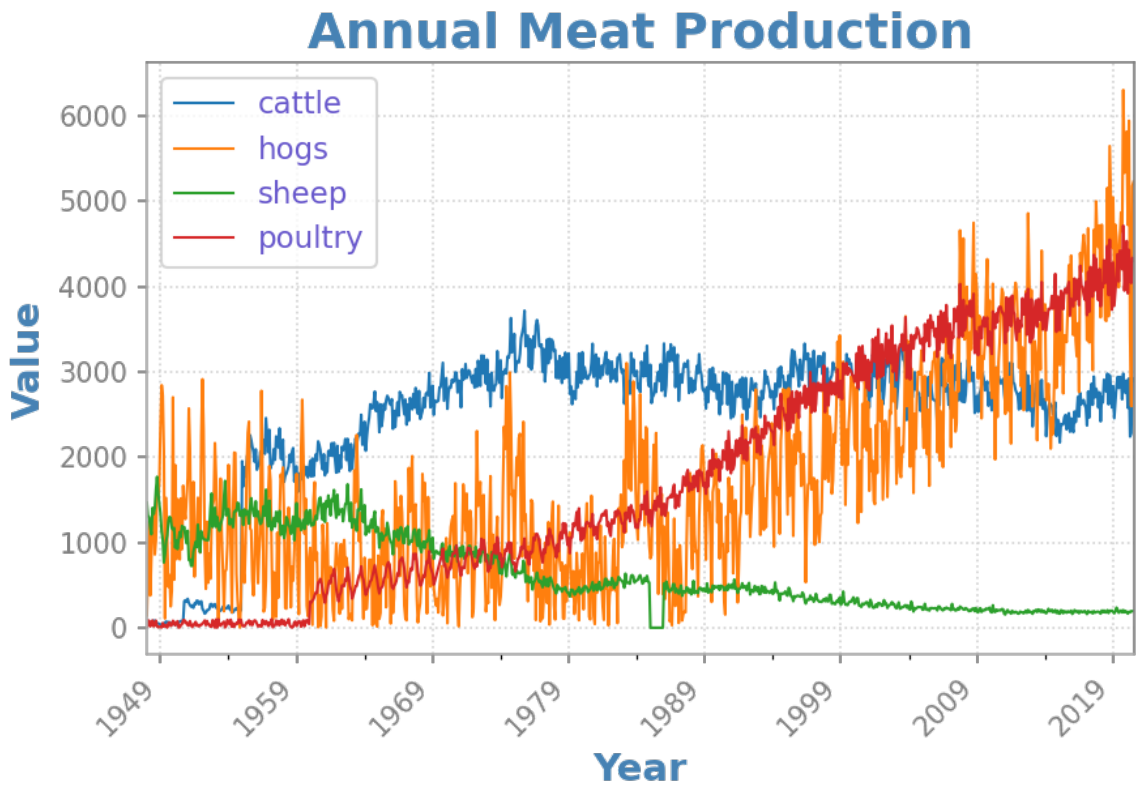
In [522]:
```
df.groupby(df.index.year).sum()
```

Out[522]:

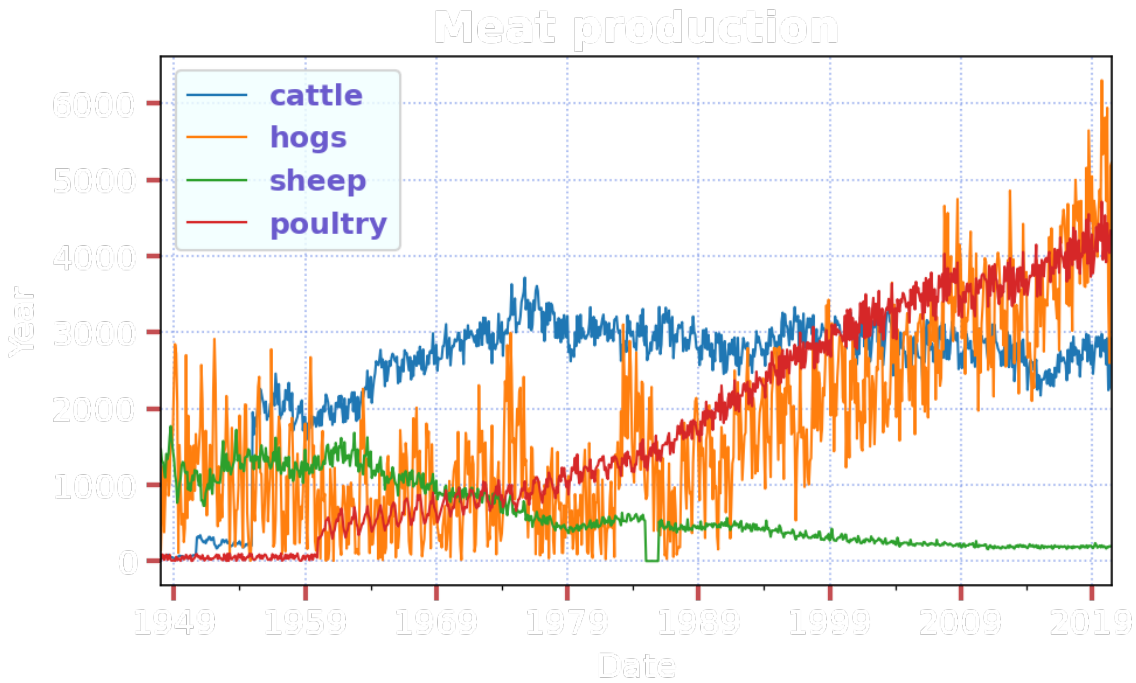
date	cattle	hogs	sheep	poultry
1948	681.1	12805.0	16896.9	505.5
1949	630.7	17095.0	13376.6	388.5
1950	1465.0	15056.0	12852.3	596.3
1951	3392.1	16528.5	11074.8	425.0
1952	2765.7	16198.9	13962.4	559.6
...	...	...	...	...
2016	30578.2	46219.9	2237.9	47364.4
2017	32189.4	49316.5	2178.2	48315.2
2018	33004.7	52432.5	2264.9	49161.7
2019	33555.3	57913.0	2321.7	50395.1
2020	18710.4	32887.9	1298.6	29652.8

73 rows × 4 columns

In [523]:
```
import matplotlib.dates as mdates
from matplotlib.dates import YearLocator
from matplotlib.dates import AutoDateFormatter, AutoDateLocator, DateL
ocator
fig, ax = plt.subplots(figsize=(6.5, 4), dpi = 150)
plt.xticks(np.arange(1, 767, 12*6))

#ax.plot(df[['cattle','hogs','sheep','poultry']])
ax = df.plot(figsize=(6.5,4), linewidth=1, fontsize=10, ax=ax)

plt.xticks(rotation = 45, ha = 'right')
plt.xlabel('Year', fontsize=14, color = 'steelblue', fontdict = dict(w
eight = 'bold'))
plt.ylabel('Value', fontsize=14, color = 'steelblue', fontdict = dict
(weight = 'bold'))
plt.title('Annual Meat Production', fontsize=18, color = 'steelblue',
fontdict = dict(weight = 'bold'))
plt.legend(fontsize=11)
plt.grid(True, linestyle = ':')
ax.spines['bottom'].set_color('darkgray')
ax.spines['left'].set_color('darkgray')
ax.spines['right'].set_color('gray')
ax.spines['top'].set_color('gray')
ax.tick_params(axis='x', colors='gray')
ax.tick_params(axis='y', colors='gray')
plt.savefig('03_meat_ts.png', figsize = (6.5,4), dpi = 150, bbox_inche
s = 'tight')
plt.show()
```



In [524]:
```
fig, ax1 = plt.subplots(figsize=(7, 4), dpi = 150)
plt.xticks(np.arange(1, 767, 12*6))
ax = df.plot(figsize=(7,4), linewidth=1, fontsize=14, ax=ax1)
plt.rcParams.update({"figure.figsize" : (7, 4),
                     "figure.facecolor": "navy",
                     "axes.facecolor" : "azure",
                     "axes.edgecolor": "white",
                     "text.color": "slateblue",
                     "font.weight": "bold",
                     "axes.labelweight": "bold"}),
plt.tick_params(axis = 'both', direction ='out', length = 6,
                width = 2, labelcolor = 'w', colors = 'r',
                grid_color = 'white', grid_alpha = 0.5)
ax.set_xlabel('Date', color = "white", fontsize=14)
ax.set_ylabel('Year', color = "white", fontsize=14)
plt.title('Meat production', fontsize=18, color = 'white', fontdict =
dict(weight = 'bold'))
plt.legend(fontsize=12)
plt.grid(True, color = 'royalblue', linestyle = ':')
plt.savefig('03_meat_ts2.png', figsize = (7,4), dpi = 150, bbox_inches
= 'tight')
pyplot.show()
```



In [525]:
```
def floor_decade(date_value):
    "Takes a date. Returns the decade."
    return (date_value.year // 10) * 10
pd.to_datetime(2019-12-31)
```

Out[525]:
```
Timestamp('1970-01-01 00:00:00.000001976')
```

In [532]:
```
#ts.groupby(floor_decade(_)).sum()
```

In [531]:
```
dd = df.groupby(floor_decade).sum()
dd
```

Out[531]:

	cattle	hogs	sheep	poultry
1940	1311.8	29900.0	30273.5	894.0
1950	131274.6	136436.7	146286.3	5833.0
1960	285752.8	88021.7	141454.2	68782.4
1970	366708.6	117982.6	81315.7	109069.6
1980	356206.1	134759.4	53007.9	173902.4
1990	346481.7	220211.0	47148.0	298300.0
2000	343625.3	328155.2	28947.5	403541.6
2010	321995.8	442543.2	22659.5	461291.1
2020	18710.4	32887.9	1298.6	29652.8

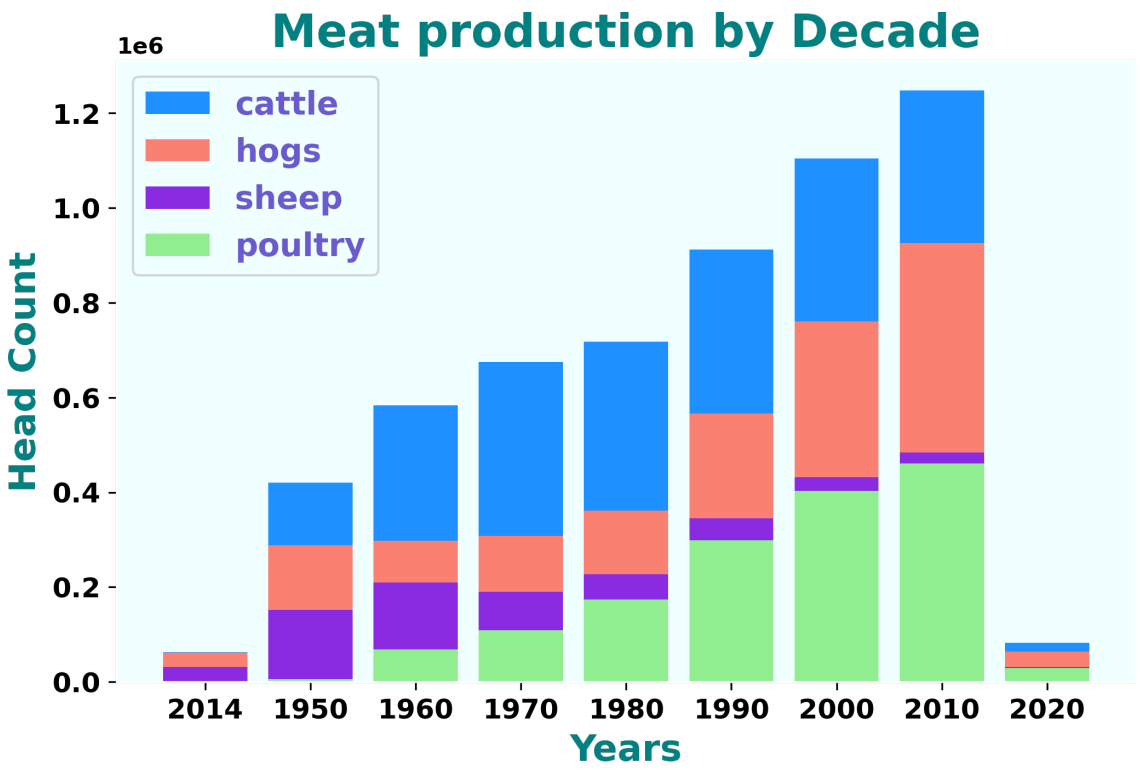
In [533]:
```
%matplotlib inline
by_decade = df.groupby(floor_decade).sum()
by_decade.index.name = 'year'
by_decade = by_decade.reset_index()
by_decade.head(5)
```

Out[533]:

	year	cattle	hogs	sheep	poultry
0	1940	1311.8	29900.0	30273.5	894.0
1	1950	131274.6	136436.7	146286.3	5833.0
2	1960	285752.8	88021.7	141454.2	68782.4
3	1970	366708.6	117982.6	81315.7	109069.6
4	1980	356206.1	134759.4	53007.9	173902.4

In [534]:
```
from ggplot import*
import numpy as np
import pandas as pd
from plotnine import *
from plotly.tools import mpl_to_plotly as ggplotly

%matplotlib inline
```

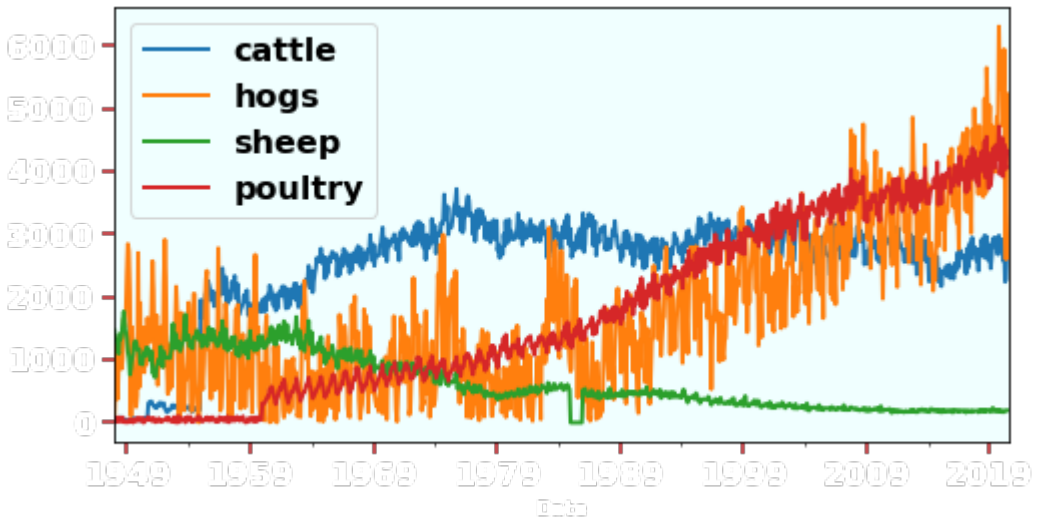
In [535]:
```
# Bar plot data wrangling
years = ['2014','1950','1960','1970','1980','1990','2000','2010','2020
']
cattle = dd['cattle']
hogs   = dd['hogs']
sheep  = dd['sheep']
poultry = dd['poultry']
ind = [x for x, _ in enumerate(years)]

# Set up figure of bars
fig, ax = plt.subplots(figsize=(8, 5), dpi = 300)
plt.bar(ind, cattle, width=0.8, label='cattle', color='dodgerblue', bo
ttom=hogs+sheep+poultry) +\
plt.bar(ind, hogs, width=0.8, label='hogs', color='salmon', bottom=she
ep+poultry) +\
plt.bar(ind, sheep, width=0.8, label='sheep', color='blueviolet', bott
om=poultry) +\
plt.bar(ind, poultry, width=0.8, label='poultry', color='lightgreen')

# Set up axes
plt.xticks(ind, years)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)

# Set up labels & legend
plt.ylabel("Head Count", fontsize=16, color = 'teal', fontdict = dict
(weight = 'bold'))
plt.xlabel("Years", fontsize=16, color = 'teal', fontdict = dict(weigh
t = 'bold'))
plt.title('Meat production by Decade', fontsize=20, color = 'teal', fo
ntdict = dict(weight = 'bold'))
plt.legend(loc="upper left", fontsize=14)

plt.savefig('03_meat_bar01.png', figsize = (8,5), dpi = 300, bbox_inch
es = 'tight')
#ax.plt.show()
```



In [536]:
```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
inline_rc = dict(mpl.rcParams)
```

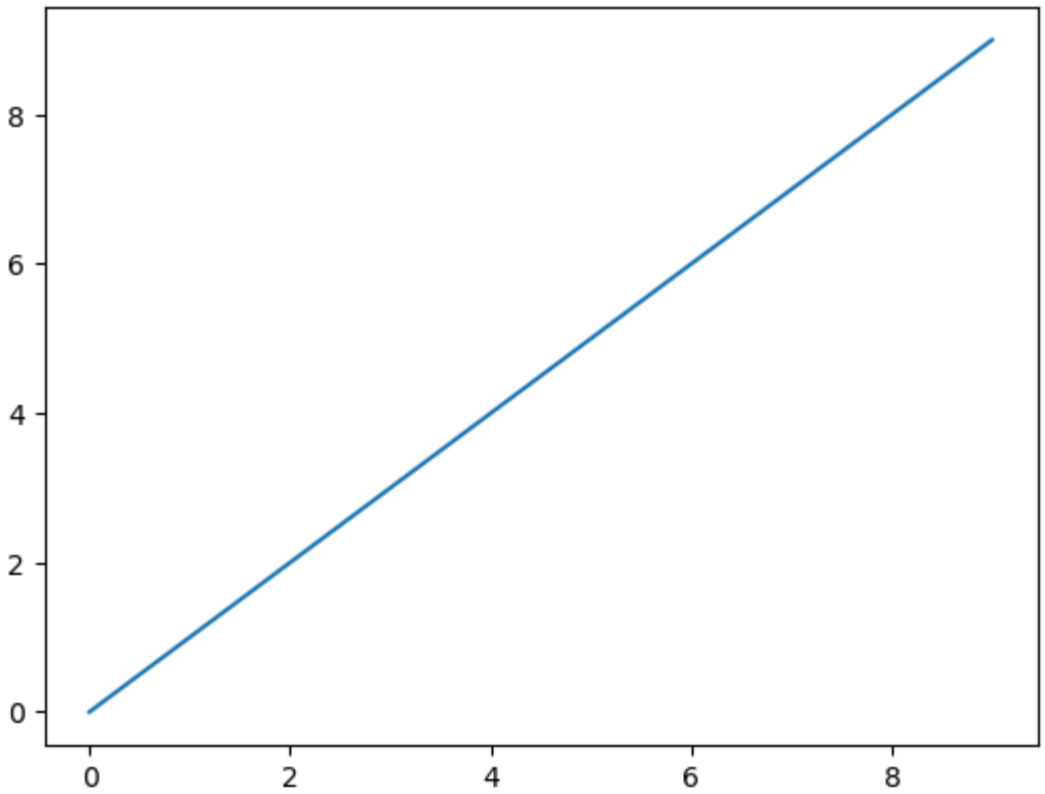
In [537]:
```
plt.rcParams.update({"axes.edgecolor": "black",
                     "text.color": "black"})
```

In [539]:
```
ax = df.plot(figsize = (8,4), linewidth = 2, fontsize = 16)
plt.rcParams.update({"axes.edgecolor": "black"}),
plt.tick_params(axis = 'both', direction ='out', length = 6,
                width = 2, labelcolor = 'w', colors = 'r',
                grid_color = 'white', grid_alpha = 0.5)
ax.set_xlabel('Date', color = "white");
ax.legend(fontsize = 16);
plt.savefig('03_meat_ts2.png', figsize = (7,4), dpi = 150, bbox_inches
= 'tight')
```



In [540]:
```
mpl.rcParams.update(mpl.rcParamsDefault)
plt.plot(range(10))
```

Out[540]:
```
[<matplotlib.lines.Line2D at 0x141113363a0>]
```



In [541]:
```
index = pd.date_range('10/1/1999', periods=1100)
ts = df
ts = ts.rolling(window=100, min_periods=100).mean().dropna()
ts.head()
```

Out[541]:

date	cattle	hogs	sheep	poultry
1956-04-01	457.409	1195.997	1216.872	44.458
1956-05-01	477.497	1205.538	1214.378	44.662
1956-06-01	497.705	1226.648	1213.594	43.913
1956-07-01	518.651	1231.867	1213.705	44.023
1956-08-01	540.172	1229.975	1216.164	44.472

In [543]:
```
import numpy as np
import pandas as pd
import dfply
from dfply.group import group_by as groupby
import matplotlib.pyplot as plt
from ggplot import *

meat = meat.dropna(thresh=800, axis=1) # drop columns that have fewer
than 800 observations
ts = meat.set_index(['date'])

ts.groupby(ts.index.year).sum().head(10)

#the1940s = groupby(ts.index.year).sum().ix['1940-01-01':'1949-12-31']
#the1940s
```

Out[543]:

date	beef	veal	pork	lamb_and_mutton
1944	8801.0	1629.0	11502.0	1001.0
1945	9936.0	1552.0	8843.0	1030.0
1946	9010.0	1329.0	9220.0	946.0
1947	10096.0	1493.0	8811.0	779.0
1948	8766.0	1323.0	8486.0	728.0
1949	9142.0	1240.0	8875.0	587.0
1950	9248.0	1137.0	9397.0	581.0
1951	8549.0	972.0	10190.0	508.0
1952	9337.0	1080.0	10321.0	635.0
1953	12055.0	1451.0	8971.0	715.0

In [544]:
```
df.groupby(df.index.year).sum()
```

Out[544]:

date	cattle	hogs	sheep	poultry
1948	681.1	12805.0	16896.9	505.5
1949	630.7	17095.0	13376.6	388.5
1950	1465.0	15056.0	12852.3	596.3
1951	3392.1	16528.5	11074.8	425.0
1952	2765.7	16198.9	13962.4	559.6
...	...	...	...	...
2016	30578.2	46219.9	2237.9	47364.4
2017	32189.4	49316.5	2178.2	48315.2
2018	33004.7	52432.5	2264.9	49161.7
2019	33555.3	57913.0	2321.7	50395.1
2020	18710.4	32887.9	1298.6	29652.8

73 rows × 4 columns

In [545]:
```
the1940s.sum().reset_index()
```

Out[545]:

	index	cattle	hogs	sheep	poultry
0	1948-01-01	51.3	144.6	1464.0	10.4
1	1948-02-01	37.7	665.6	1306.7	87.6
2	1948-03-01	35.7	1242.0	1297.2	79.6
3	1948-04-01	45.1	377.5	1178.1	14.7
4	1948-05-01	52.9	833.5	1095.3	67.2
...	...	...	...	...	...
866	2020-03-01	2921.7	5942.0	187.5	4427.8
867	2020-04-01	2239.1	3412.0	180.8	4098.9
868	2020-05-01	2277.5	2595.5	195.3	4039.1
869	2020-06-01	2876.3	5176.8	193.4	4334.7
870	2020-07-01	2918.4	5225.4	195.1	4303.0

871 rows × 5 columns

In [546]:
```
ts2 = births.set_index(['date'])
ts2.head(5)
ts2.groupby(ts2.index.year).sum().head(10)
```

Out[546]:

	births
date	
1975	3136965
1976	6304156
1979	3333279
1982	3612258
1983	7333238
1986	7308074
1987	3760561
1988	3756547
1990	7718904
1991	11714356

In [547]:
```
the1940s
```

Out[547]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001410AB
C6CD0>
```

In [548]:
```
by_decade = df.groupby(floor_decade).sum()

by_decade.index.name = 'year'

by_decade = by_decade.reset_index()
```

In [549]:
```
by_decade_long = pd.melt(by_decade, id_vars="year")
```

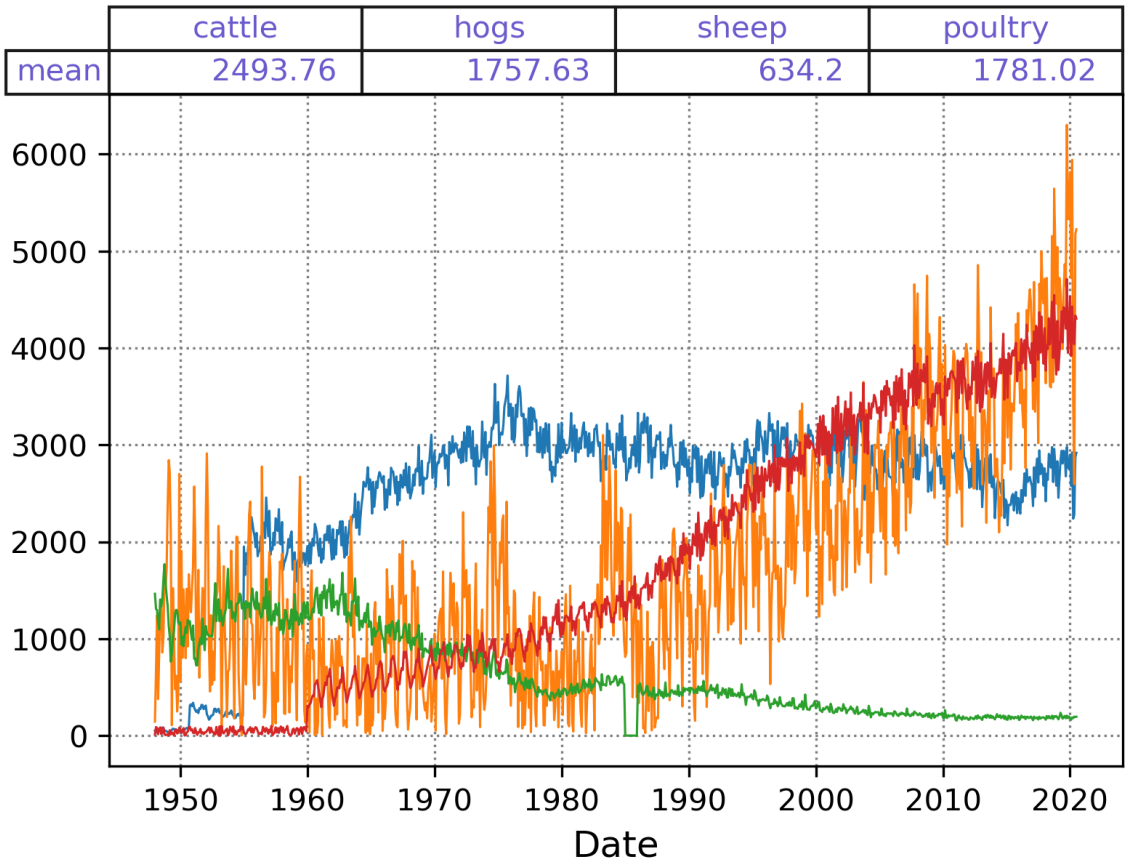
In [550]:
```
warnings.filterwarnings('ignore')
#ggplot() + geom_bar(aes(x='year', weight = 'value', color = 'variable
', fill='variable'), data=by_decade_long)
```

In [551]:
```
meat_sum = df.sum().to_frame().T
meat_mean = df.mean().to_frame().T
meat_sum.rename({0:'sum'}, inplace = True)
meat_mean.rename({0:'mean'}, inplace = True)
meat_sum, meat_mean
```

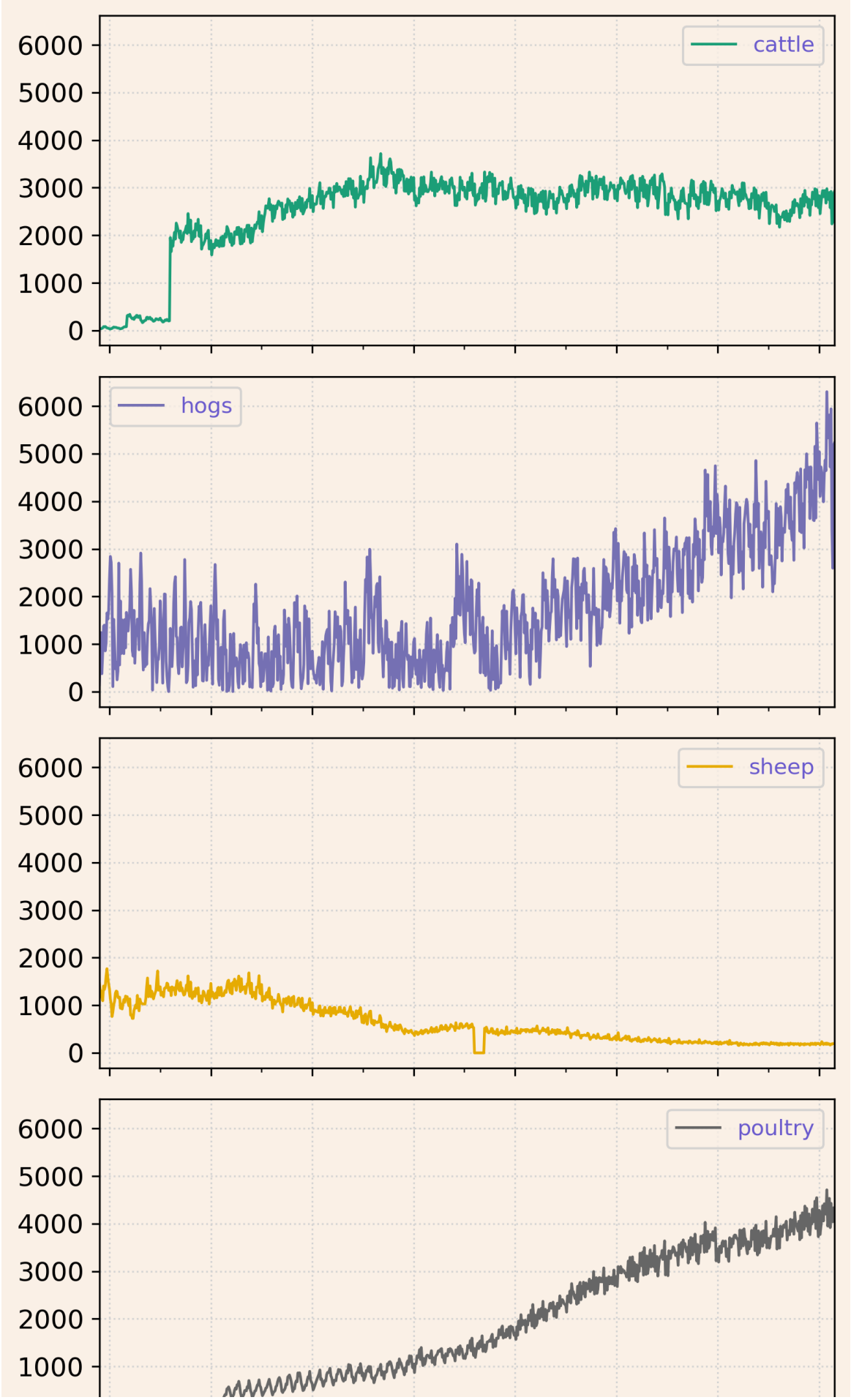
Out[551]:
```
(        cattle       hogs     sheep    poultry
 sum  2172067.1  1530897.7  552391.2  1551266.9,
             cattle         hogs       sheep      poultry
 mean  2493.762457  1757.632262  634.203444  1781.018255)
```

In [552]:
```
from matplotlib.lines import Line2D
fig, ax = plt.subplots(figsize=(6, 4), dpi = 300)
ax.plot(df, linewidth = 0.75)

plt.rcParams.update({"figure.facecolor": "linen",
                     "axes.facecolor" : "linen",
                     "text.color": "slateblue",}),
# Add x-axis labels
ax.set_xlabel('Date', fontsize = 12)

# Add summary table information to the plot
sax=ax.table(cellText=meat_mean.values.round(2),
        colWidths = [0.25] * len(meat_mean.columns),
        rowLabels = meat_mean.index,
        colLabels = meat_mean.columns,
        loc = 'top')
sax.auto_set_font_size(False)
sax.set_fontsize(10)
sax.scale(1, 1.2)
ax.grid(b = None, which = 'major', axis = 'both', color = 'gray', line
style = ':')
# Specify the fontsize and location of your legend
#colors = ['steelblue', 'green', 'lightseagreen', 'salmon']
lines = [Line2D([0], [0], color=c, linewidth=2, linestyle='-') for c i
n colors]
labels = ['Cattle', 'Hogs', 'Sheep', 'Poultry']
plt.legend(loc = 'upper left', labels = labels);
plt.savefig('03_meat_freq.png', figsize = (6,4), dpi = 150, bbox_inche
s = 'tight')
```

```
---------------------------------------------------------------------
------
TypeError                                 Traceback (most recent call
last)
<ipython-input-552-a25582a1b047> in <module>
     21 # Specify the fontsize and location of your legend
     22 #colors = ['steelblue', 'green', 'lightseagreen', 'salmon']
---> 23 lines = [Line2D([0], [0], color=c, linewidth=2, linestyle=
'-') for c in colors]
     24 labels = ['Cattle', 'Hogs', 'Sheep', 'Poultry']
     25 plt.legend(loc = 'upper left', labels = labels);

TypeError: 'module' object is not iterable
```



In [553]:
```
fig, ax = plt.subplots(figsize=(5.5, 10), dpi = 300)
plt.rcParams.update({"grid.color": "lightgray",
                     "grid.linestyle" : ":"}),
#ax.grid(b = None, which = 'major', axis = 'both', color = 'lightgray
', linestyle = ':')
df.plot(subplots = True,
          layout = (4, 1),
          grid = True,
          sharex = True,
          sharey = True,
          colormap = 'Dark2',
          fontsize = 12,
          rot = 90,
          legend = True,
          linewidth = 1.25,
          ax = ax);
plt.savefig('0000_60.png', figsize = (5.5,10), dpi = 150, bbox_inches
= 'tight')
plt.tight_layout();
```


cattle
6000
5000
4000
3000
2000
1000
0
hogs
6000
5000
4000
3000
2000
1000
0
sheep
6000
5000
4000
3000
2000
1000
0
poultry
6000
5000
4000
3000
2000
1000

In [554]:
```
def floor_decade(date_value):
    "Takes a date. Returns the decade."
    return (date_value.year // 10) * 10
pd.to_datetime(2013-10-9)
```

Out[554]:
```
Timestamp('1970-01-01 00:00:00.000001994')
```

In [555]:
```
floor_decade(_)
```

Out[555]:
```
1970
```

In [556]:
```
df.groupby(floor_decade).sum()
```

Out[556]:

	cattle	hogs	sheep	poultry
1940	1311.8	29900.0	30273.5	894.0
1950	131274.6	136436.7	146286.3	5833.0
1960	285752.8	88021.7	141454.2	68782.4
1970	366708.6	117982.6	81315.7	109069.6
1980	356206.1	134759.4	53007.9	173902.4
1990	346481.7	220211.0	47148.0	298300.0
2000	343625.3	328155.2	28947.5	403541.6
2010	321995.8	442543.2	22659.5	461291.1
2020	18710.4	32887.9	1298.6	29652.8

In [557]:
```
dd = df.groupby(floor_decade).sum()
dd
```

Out[557]:

	cattle	hogs	sheep	poultry
1940	1311.8	29900.0	30273.5	894.0
1950	131274.6	136436.7	146286.3	5833.0
1960	285752.8	88021.7	141454.2	68782.4
1970	366708.6	117982.6	81315.7	109069.6
1980	356206.1	134759.4	53007.9	173902.4
1990	346481.7	220211.0	47148.0	298300.0
2000	343625.3	328155.2	28947.5	403541.6
2010	321995.8	442543.2	22659.5	461291.1
2020	18710.4	32887.9	1298.6	29652.8

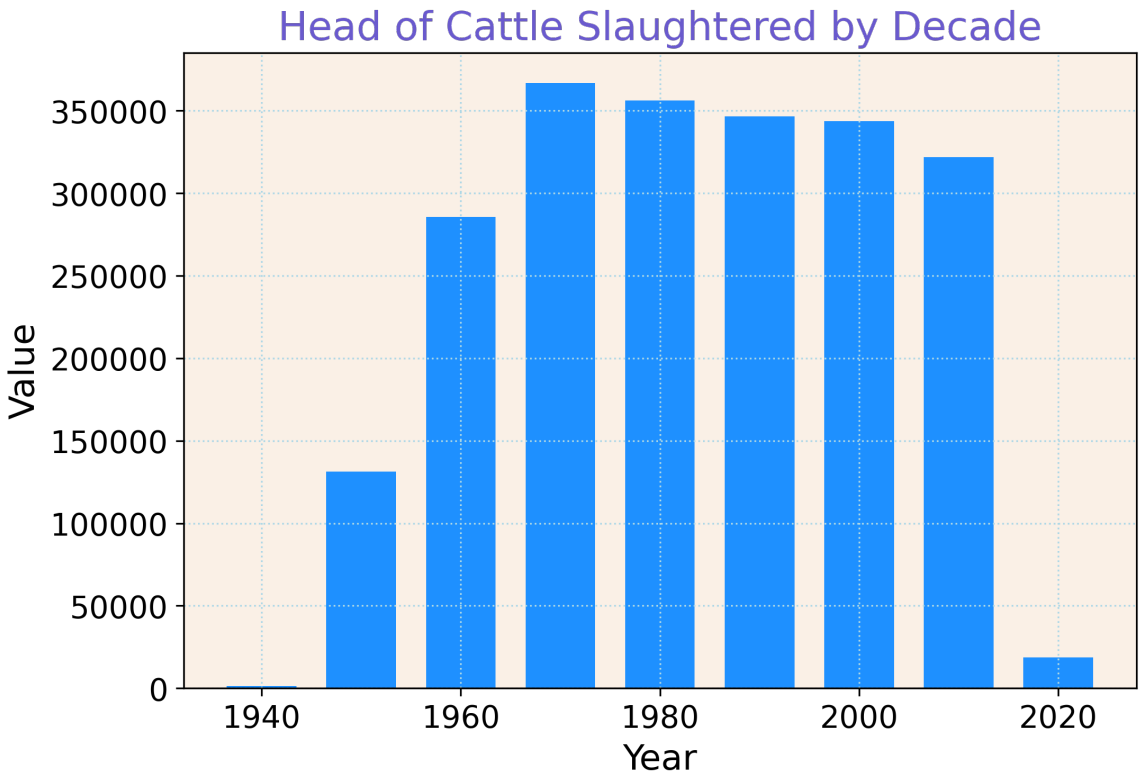
In [558]:
```
%matplotlib inline
by_decade = df.groupby(floor_decade).sum()
by_decade.index.name = 'year'
by_decade = by_decade.reset_index()
by_decade.head(5)
```

Out[558]:

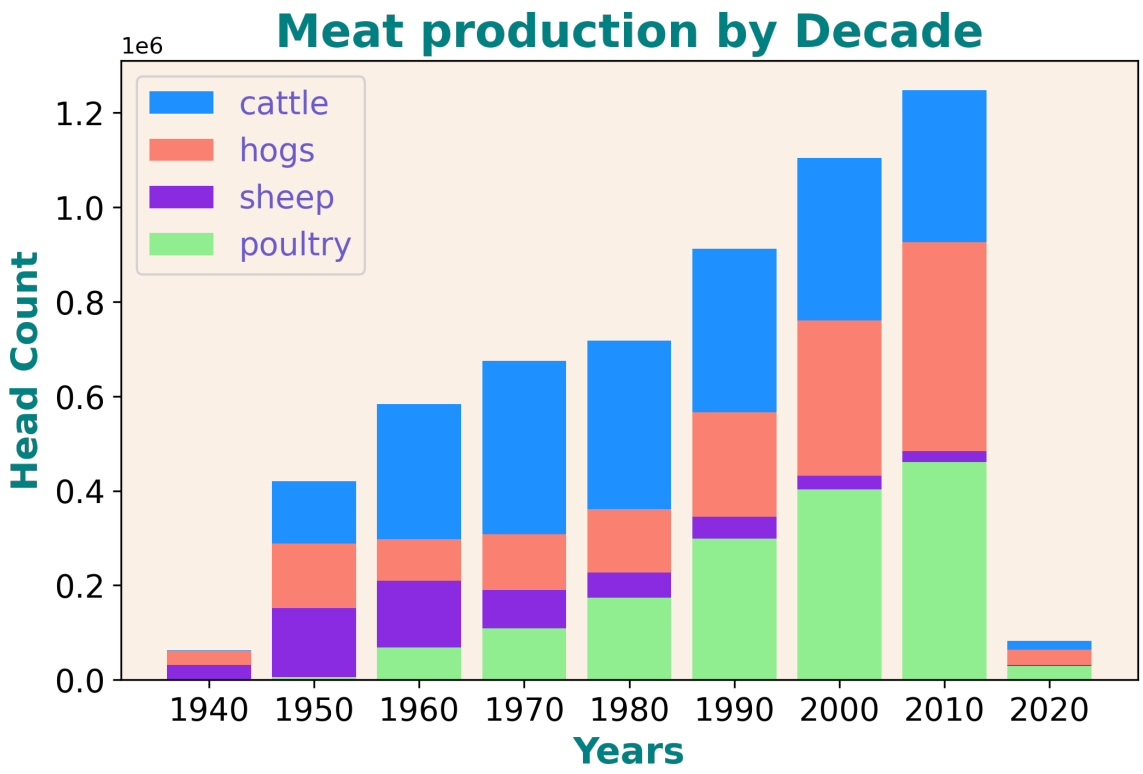
	year	cattle	hogs	sheep	poultry
0	1940	1311.8	29900.0	30273.5	894.0
1	1950	131274.6	136436.7	146286.3	5833.0
2	1960	285752.8	88021.7	141454.2	68782.4
3	1970	366708.6	117982.6	81315.7	109069.6
4	1980	356206.1	134759.4	53007.9	173902.4

In [559]:
```
from ggplot import*
import numpy as np
import pandas as pd
from plotnine import *
from plotly.tools import mpl_to_plotly as ggplotly

%matplotlib inline
```

In [560]:
```
matplotlib.rcParams['figure.dpi'] = 300
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(by_decade['year'], by_decade['cattle'], width = 7,
color = 'dodgerblue')
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
ax.set_xlabel('Year', size = 16)
ax.set_ylabel('Value', size = 16)
ax.set_title('Head of Cattle Slaughtered by Decade', size = 18)
ax.grid(color = 'lightblue')
plt.savefig('0000_91.png', figsize = (8,5), dpi = 300, bbox_inches = '
tight')
plt.show()
```



In [561]:
```
# Bar plot data wrangling
years = ['1940','1950','1960','1970','1980','1990','2000','2010','2020
']
cattle = dd['cattle']
hogs   = dd['hogs']
sheep  = dd['sheep']
poultry = dd['poultry']
ind = [x for x, _ in enumerate(years)]

# Set up figure of bars
fig, ax = plt.subplots(figsize=(8, 5), dpi = 300)
plt.bar(ind, cattle, width=0.8, label='cattle', color='dodgerblue', bo
ttom=hogs+sheep+poultry) +\
plt.bar(ind, hogs, width=0.8, label='hogs', color='salmon', bottom=she
ep+poultry) +\
plt.bar(ind, sheep, width=0.8, label='sheep', color='blueviolet', bott
om=poultry) +\
plt.bar(ind, poultry, width=0.8, label='poultry', color='lightgreen')

# Set up axes
plt.xticks(ind, years)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)

# Set up labels & legend
plt.ylabel("Head Count", fontsize=16, color = 'teal', fontdict = dict
(weight = 'bold'))
plt.xlabel("Years", fontsize=16, color = 'teal', fontdict = dict(weigh
t = 'bold'))
plt.title('Meat production by Decade', fontsize=20, color = 'teal', fo
ntdict = dict(weight = 'bold'))
plt.legend(loc="upper left", fontsize=14)

plt.savefig('0000_61.png', figsize = (8,5), dpi = 300, bbox_inches = '
tight')
#ax.plt.show()
```

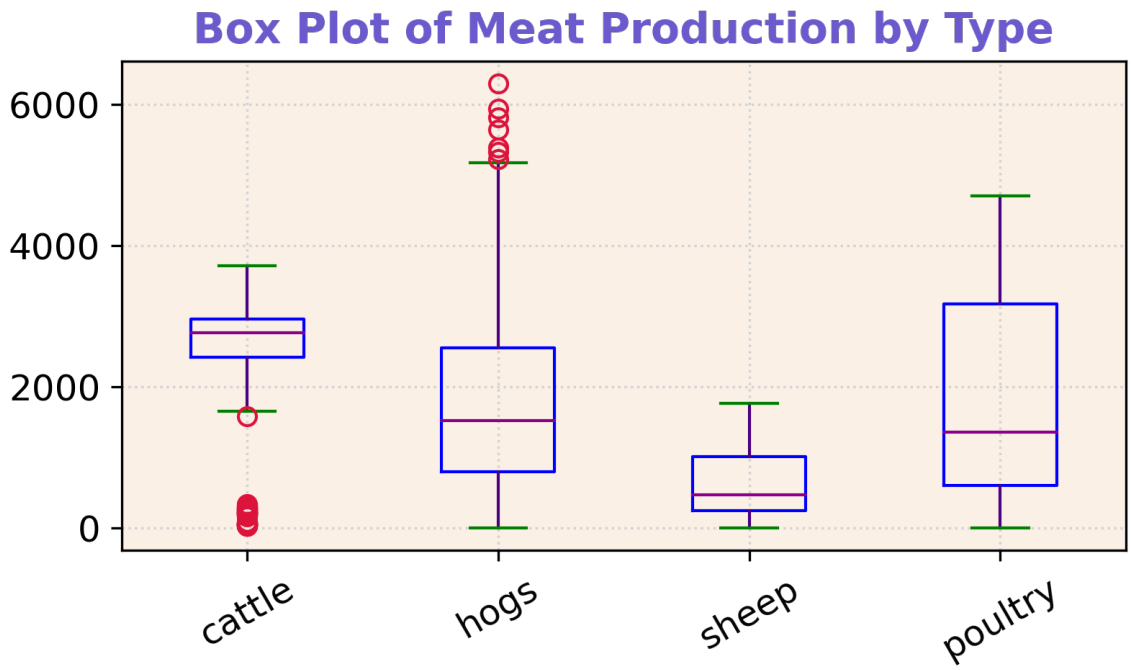


In [562]:
```
dd.head(5)
```

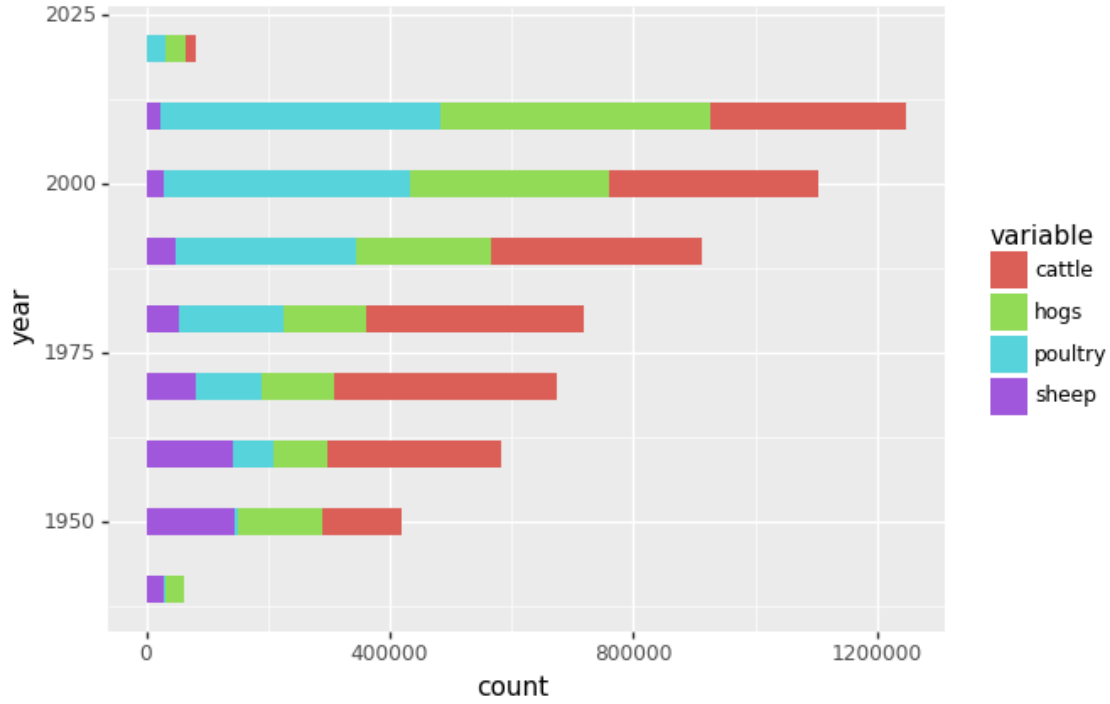
Out[562]:

	cattle	hogs	sheep	poultry
1940	1311.8	29900.0	30273.5	894.0
1950	131274.6	136436.7	146286.3	5833.0
1960	285752.8	88021.7	141454.2	68782.4
1970	366708.6	117982.6	81315.7	109069.6
1980	356206.1	134759.4	53007.9	173902.4

In [563]:
```
fig, ax = plt.subplots(linewidth = 3, figsize = (6, 3), dpi = 300)
plt.rcParams.update({"figure.facecolor": "white",
                     "lines.linewidth": 2,
                     "axes.facecolor" : "white"}),
ax = df.boxplot(figsize=(6,3), rot=30, fontsize=12,
            boxprops=dict(color="blue"),
            capprops=dict(color="green"),
            whiskerprops=dict(color="indigo"),
            medianprops=dict(color="darkmagenta"),
            flierprops=dict(color="yellow", markeredgecolor="crimso
n"),
            )
plt.title('Box Plot of Meat Production by Type', fontsize = 14, fontdi
ct = dict(weight = 'bold'))
plt.savefig('0000_62.png', figsize = (6,3), dpi = 300, bbox_inches = '
tight')
pyplot.show()
```

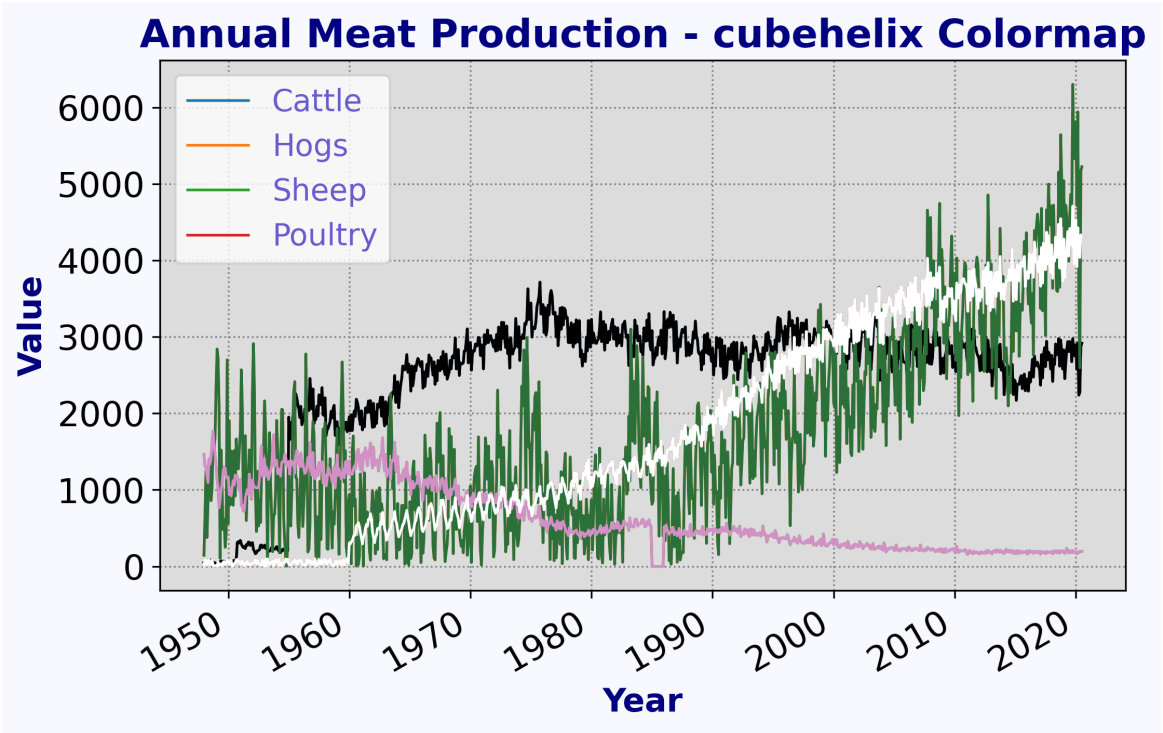


In [564]:
```
ggplot(by_decade_long, aes(x='year', weight = 'value', fill = 'variable')) + geom_bar(width = 4) +  coord_flip()
```

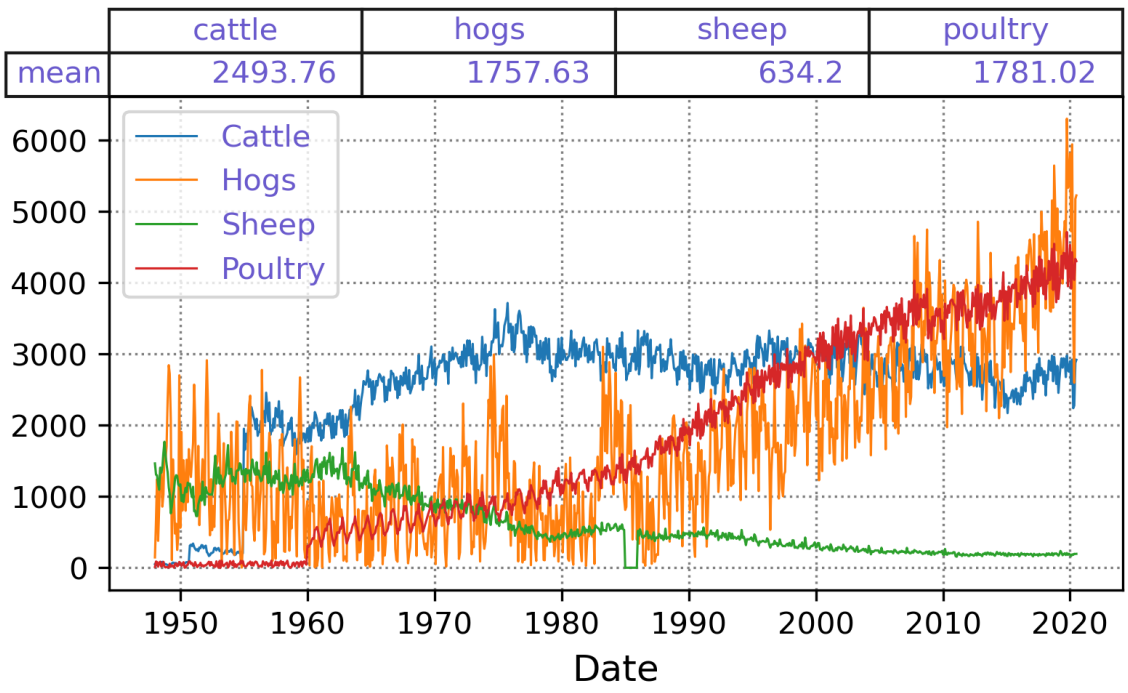


Out[564]: <ggplot: (86179373559)>

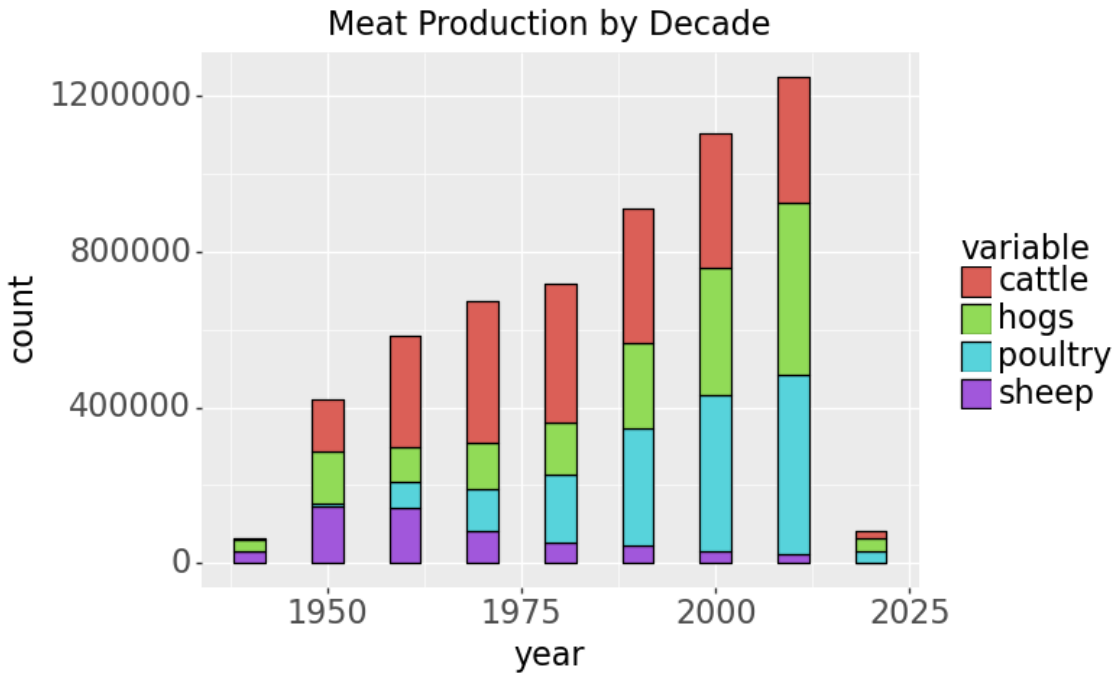
In [565]:
```
from matplotlib import *
fig, ax = plt.subplots(figsize = (8, 5), dpi = 300, facecolor = 'ghost
white')
ax.plot(df, linewidth = 1.25)
df.plot(figsize = (8,5), linewidth = 1.25, fontsize = 16, colormap = '
cubehelix', ax=ax)
fig.patch.set_facecolor('ghostwhite')
plt.xlabel('Year', fontsize=15, color = 'navy', fontdict = dict(weight
= 'bold'))
plt.ylabel('Value', fontsize=15, color = 'navy', fontdict = dict(weigh
t = 'bold'))
plt.title('Annual Meat Production - cubehelix Colormap', fontsize=18,
color = 'navy', fontdict = dict(weight = 'bold'))
plt.grid(True, color = 'gray', linestyle = ':')
ax.set_facecolor("gainsboro")
plt.legend(['Cattle', 'Hogs', 'Sheep', 'Poultry'], loc = 'upper left',
fontsize=14);
plt.savefig('0000_58.png', figsize = (8,5), dpi = 300, bbox_inches = '
tight',
            facecolor = fig.get_facecolor(), edgecolor='navy');

plt.show()
```



In [566]:
```
from matplotlib.lines import Line2D
fig, ax = plt.subplots(figsize = (6, 3), dpi = 300)
ax.plot(df, linewidth = 0.75)

plt.rcParams.update({'figure.facecolor': 'white',
                     'axes.facecolor' :  'white',
                     'text.color': 'slateblue',}),
# Add x-axis labels
ax.set_xlabel('Date', fontsize = 12)

# Add summary table information to the plot
sax = ax.table(cellText = meat_mean.values.round(2),
colWidths = [0.25] * len(meat_mean.columns),
rowLabels = meat_mean.index,
colLabels = meat_mean.columns, loc = 'top')
sax.auto_set_font_size(False)
sax.set_fontsize(10)
sax.scale(1, 1.2)
ax.grid(b = None, which = 'major', axis = 'both',
color = 'gray', linestyle = ':')
# Specify the fontsize and location of your legend
#Lines = [Line2D([0], [0], color = c, linewidth = 2, linestyle = '-')
for c in colors]
labels = ['Cattle', 'Hogs', 'Sheep', 'Poultry']
plt.legend(loc = 'upper left', labels = labels);
plt.savefig('0000_59.png', figsize = (6,3), dpi = 300, bbox_inches = '
tight',
            facecolor = fig.get_facecolor(), edgecolor='navy');

plt.show()
```



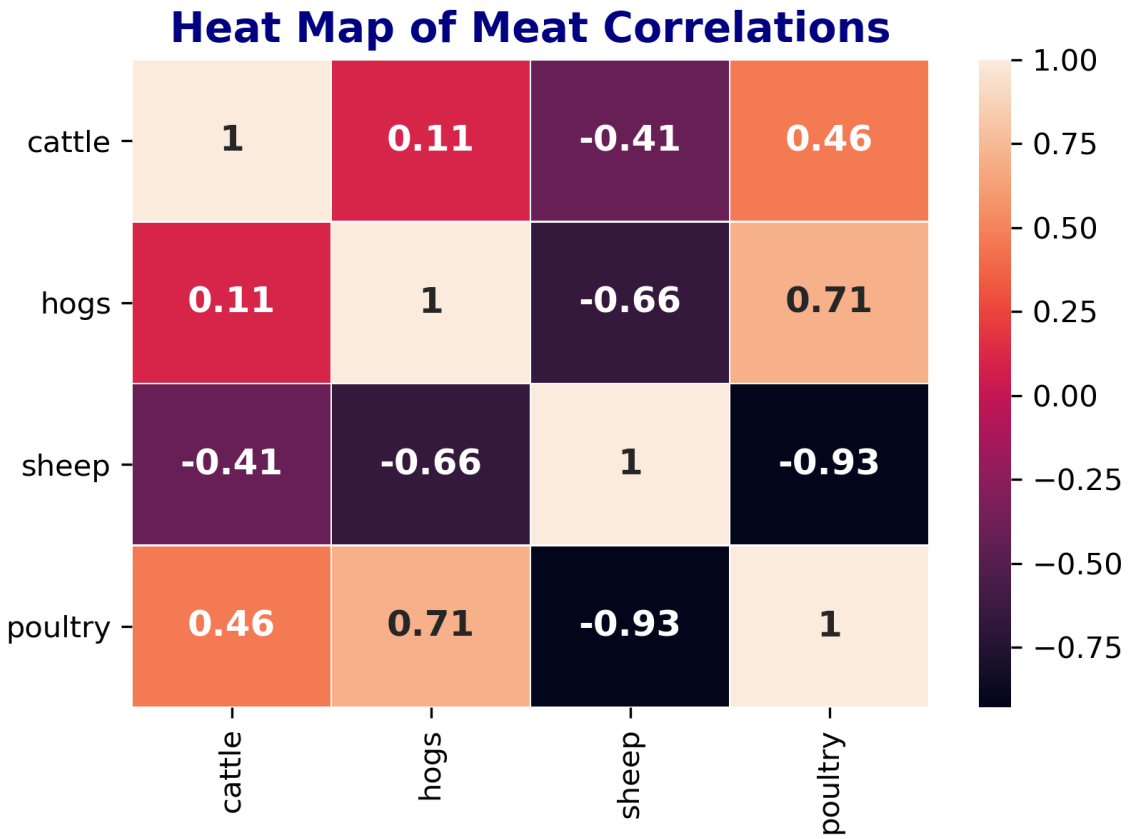
In [567]:
```
from ggplot import*
from plotnine import *
from plotly.tools import mpl_to_plotly as ggplotly
by_decade_long = pd.melt(by_decade, id_vars="year")
p2 = ggplot(aes(x='year', weight='value', fill='variable'), data=by_de
cade_long) +\
geom_bar(colour='black', width=4) + theme(text = element_text(size=1
6)) + ggtitle("Meat Production by Decade")
ggplotly(p2.draw())
plt.show(p2.draw())
```



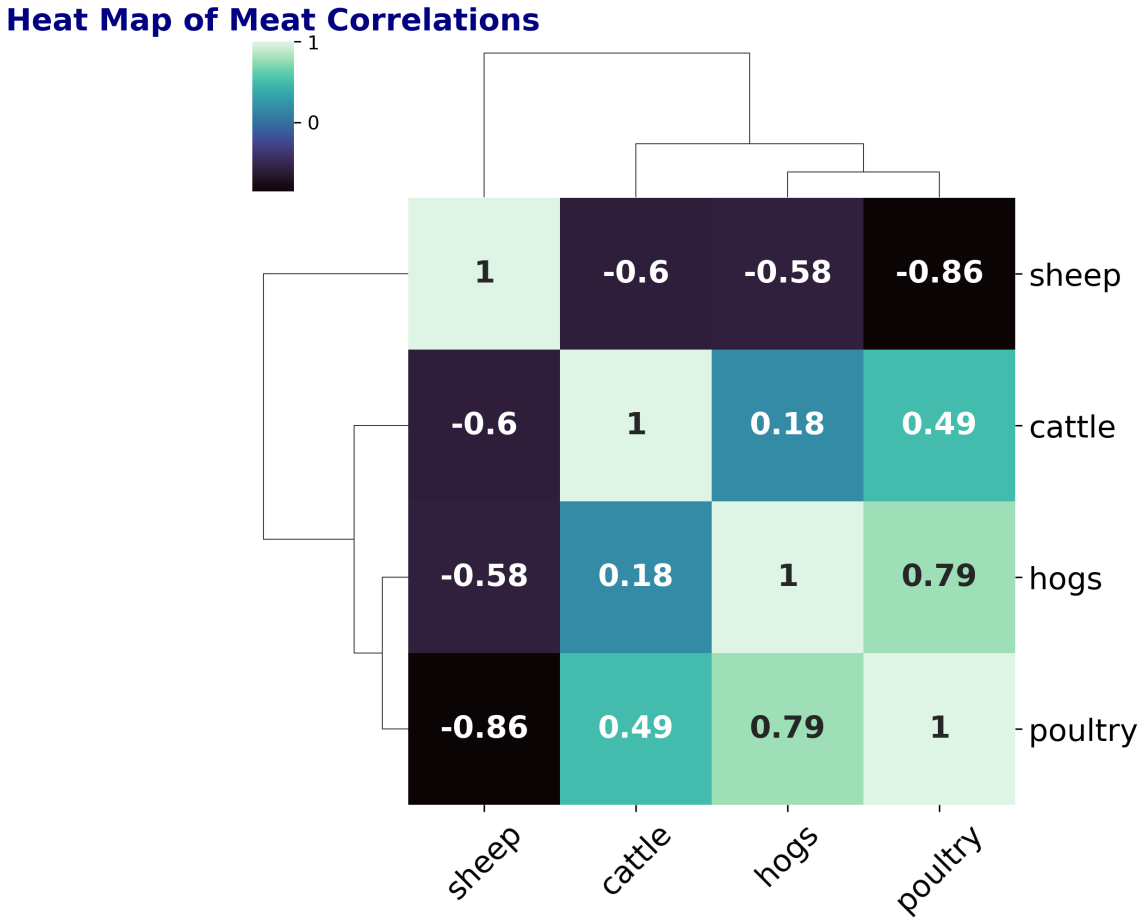
In [568]:
```
print(df[['cattle','hogs','sheep','poultry']].corr(method='pearson'))
```

```
            cattle      hogs     sheep   poultry
cattle    1.000000  0.175676 -0.600732  0.494452
hogs      0.175676  1.000000 -0.579480  0.785756
sheep    -0.600732 -0.579480  1.000000 -0.858030
poultry   0.494452  0.785756 -0.858030  1.000000
```

In [569]:
```
import seaborn as sns
corr_meat = df.corr(method='spearman')
fig, ax = plt.subplots(linewidth=2, figsize=(6, 4), dpi = 300)
# Customize the heatmap of the corr_meat correlation matrix
sns.heatmap(corr_meat,
        annot=True,
        linewidths=0.4,
        annot_kws={'size': 12, 'weight' : 'bold'});
plt.title('Heat Map of Meat Correlations', fontsize = 14, color = 'nav
y', fontdict = dict(weight = 'bold'))
plt.savefig('03_meat_heat.png', figsize = (6,4), dpi = 300, bbox_inche
s = 'tight');
plt.xticks(rotation=90);
plt.yticks(rotation=0);
```



In [570]:
```
corr_meat = df.corr(method='pearson')

# Customize the heatmap of the corr_meat correlation matrix
fig = sns.clustermap(corr_meat,
            row_cluster=True,
            col_cluster=True,
            figsize=(6, 6),
            cmap = 'mako',
            annot = True,
            annot_kws={'size': 16, 'weight' : 'bold'});
plt.setp(fig.ax_heatmap.xaxis.get_majorticklabels(), rotation=45, font
size = 16);
plt.setp(fig.ax_heatmap.yaxis.get_majorticklabels(), rotation=0, fonts
ize = 16);
plt.title('Heat Map of Meat Correlations', fontsize = 16, color = 'nav
y', fontdict = dict(weight = 'bold'))
plt.savefig('0000_63.png', figsize = (6,6), dpi = 300, bbox_inches = '
tight');
```

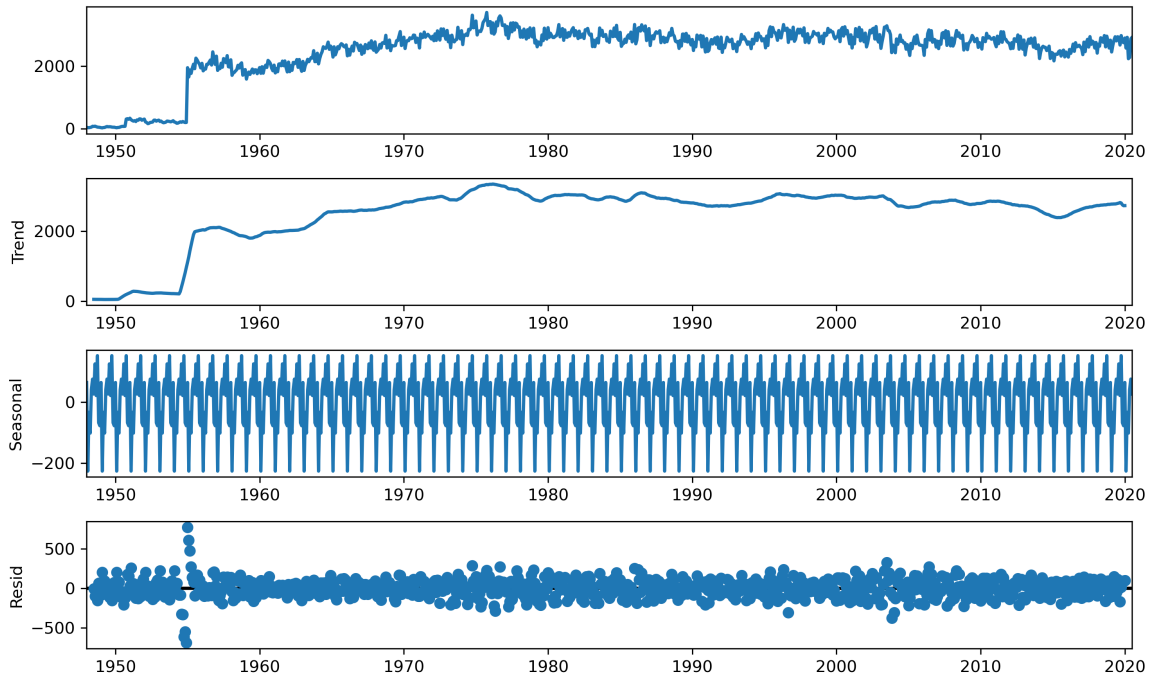


In [571]:
```
meat_yr = df.resample('1Y').sum()
meat_yr.head(10)
```

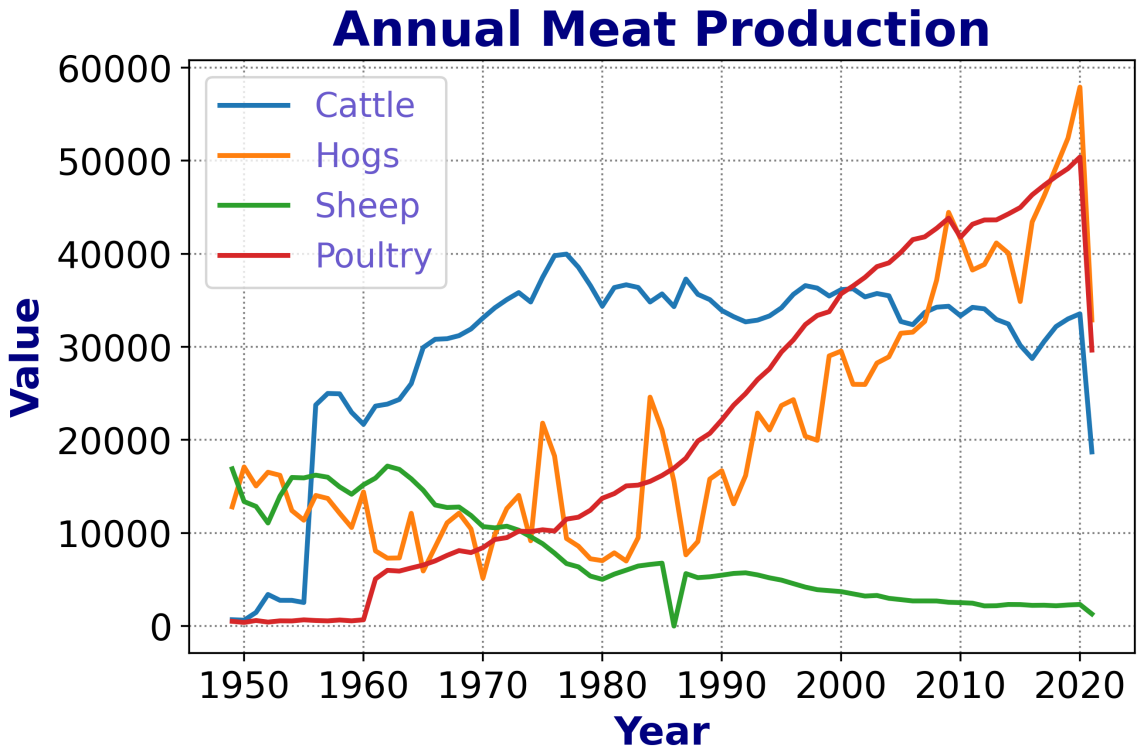
Out[571]:

date	cattle	hogs	sheep	poultry
1948-12-31	681.1	12805.0	16896.9	505.5
1949-12-31	630.7	17095.0	13376.6	388.5
1950-12-31	1465.0	15056.0	12852.3	596.3
1951-12-31	3392.1	16528.5	11074.8	425.0
1952-12-31	2765.7	16198.9	13962.4	559.6
1953-12-31	2761.1	12407.1	15967.4	547.5
1954-12-31	2530.5	11374.8	15919.6	666.8
1955-12-31	23762.3	14033.7	16215.1	596.0
1956-12-31	24996.1	13713.3	15993.4	555.7
1957-12-31	24956.8	12122.2	14957.4	651.0

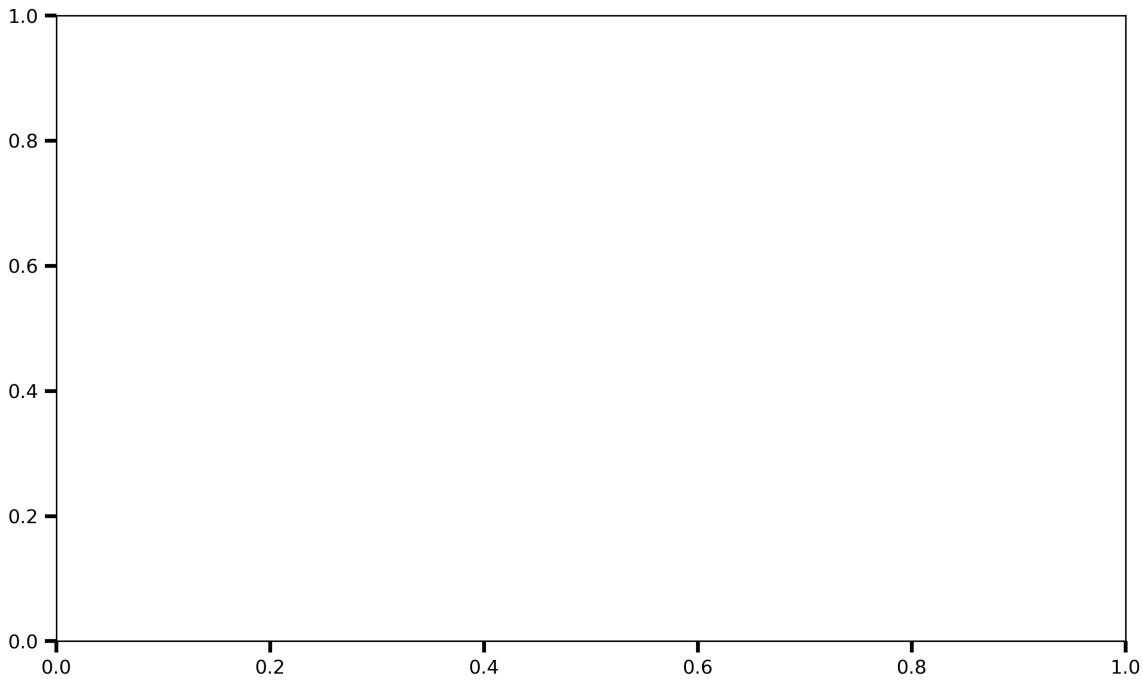
In [575]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df[['cattle']], freq = 12)
plt.rcParams.update({'figure.figsize': (10,6)})
decomposition.plot()
plt.show()
```



In [576]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
fig, ax = plt.subplots(linewidth = 1, figsize = (7,4.5), dpi = 400)
fig.patch.set_facecolor('xkcd:white')
ax.set_facecolor('xkcd:white')
ax.tick_params(axis = 'x', labelsize = 15)
ax.tick_params(axis = 'y', labelsize = 15)
plt.plot(meat_yr)
plt.xlabel('Year', fontsize=16, color = 'navy', fontdict = dict(weight
= 'bold'))
plt.ylabel('Value', fontsize=16, color = 'navy', fontdict = dict(weigh
t = 'bold'))
plt.title('Annual Meat Production', fontsize=20, color = 'navy', fontd
ict = dict(weight = 'bold'))
#lines = [Line2D([0], [0], color=c, linewidth=2, linestyle='-') for c
in colors]
plt.grid(True, color = 'gray', linestyle = ':')
#labels = ['Cattle', 'Hogs', 'Sheep', 'Poultry']
plt.legend(['Cattle', 'Hogs', 'Sheep', 'Poultry'], loc = 'upper left',
fontsize=14);
plt.savefig('0000_56.png', figsize = (6.5,4), dpi = 400, bbox_inches =
'tight',
            facecolor=fig.get_facecolor(), edgecolor='navy');

plt.show()
```



In [577]:
```
plt.rcParams.update({"axes.edgecolor": "black",
                     "text.color": "black",
                     "axes.labelweight": "bold"}),
plt.tick_params(axis = 'both', direction ='out', length = 6,
                width = 2, labelcolor = 'black', colors = 'black')
```

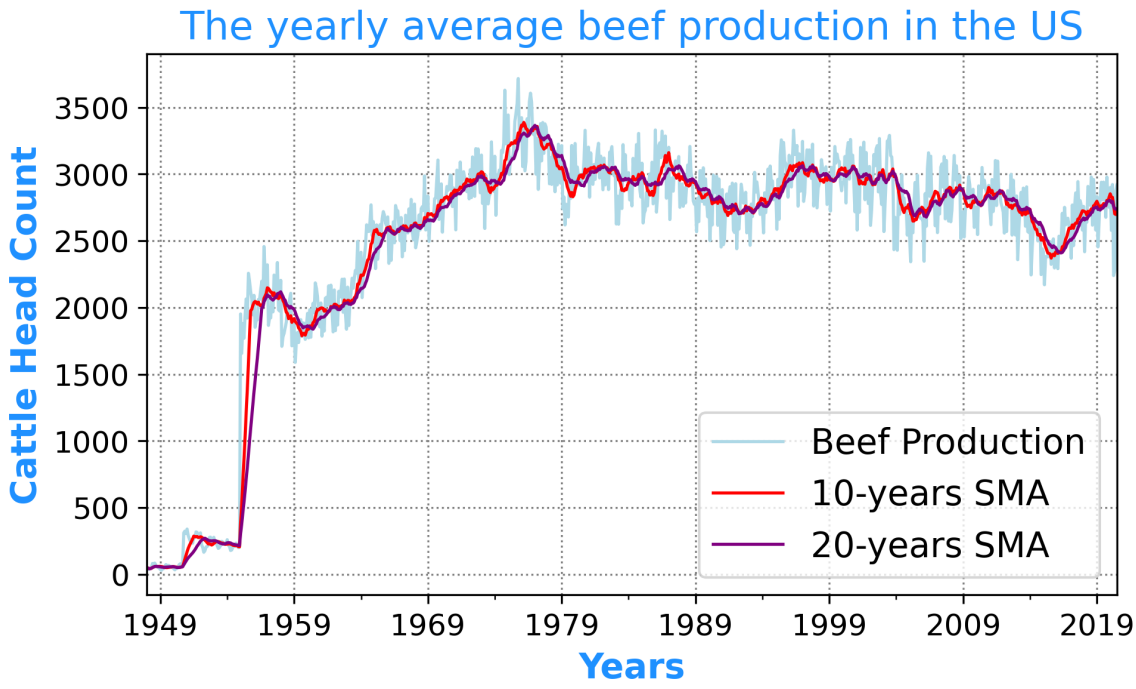


In [578]:
```
by_decade_long['category'] = pd.Categorical(by_decade_long['variable
'])
#by_decade_long
```

In [579]:
```
df_beef = df[['cattle']]
df_beef.index.name = 'date'
df_beef['avg_beef'] = df_beef.mean(axis = 1)
df_beef = df_beef[['avg_beef']]
# the simple moving average over a period of 10 years
ma1 = df_beef['SMA_10'] = df_beef.avg_beef.rolling(10,
min_periods = 1).mean()
# the simple moving average over a period of 20 year
ma2 = df_beef['SMA_20'] = df_beef.avg_beef.rolling(20,
min_periods = 1).mean()
```

In [ ]:

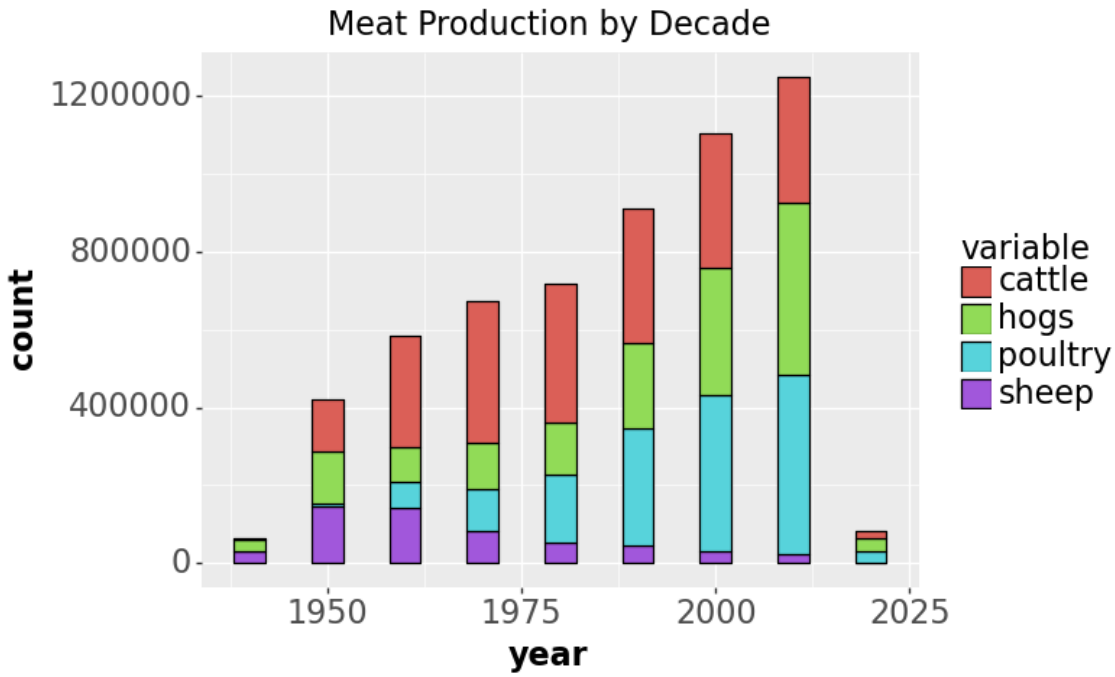
In [580]:
```
fig, ax = plt.subplots(linewidth=2, figsize = (7,4), dpi = 300)

beef_yr = df[['cattle']]

# line plot - the yearly average air temperature in Barcelona
beef_yr.plot(color = 'lightblue', linewidth = 1.25, ax = ax)
ma1.plot(color = 'red', linewidth = 1.25, ax = ax)
ma2.plot(color = 'purple', linewidth = 1.25, ax = ax)
# modify ticks size
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.legend(labels = ['Beef Production', '10-years SMA',
'20-years SMA'], fontsize = 14)

# title and labels
plt.title('The yearly average beef production in the US',
fontsize = 16, color = 'dodgerblue')
plt.xlabel('Years', fontsize = 14, color = 'dodgerblue')
plt.ylabel('Cattle Head Count', fontsize = 14,
color = 'dodgerblue')

plt.grid(True, color = 'gray', linestyle = ':')
plt.savefig('0000_65', figsize = (7.4), dpi = 300,
bbox_inches = 'tight');
plt.show()
```



In [581]:
```
X=by_decade_long[['variable']]
```

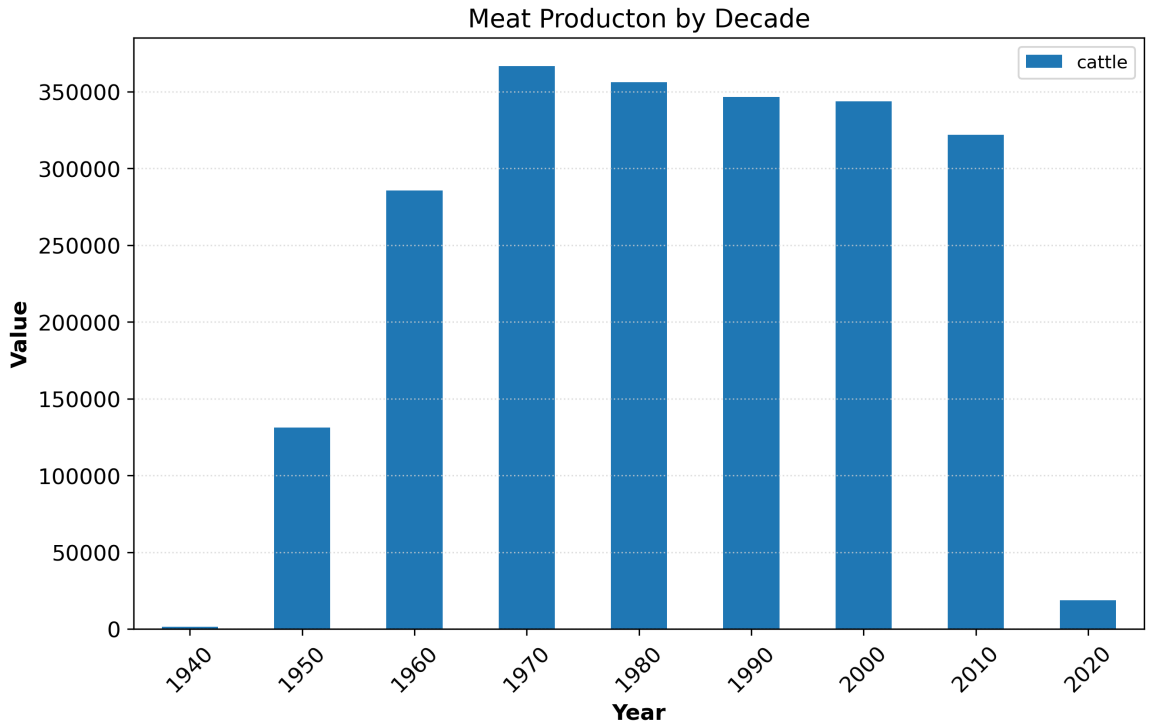
In [582]:
```
by_decade_long = pd.melt(by_decade, id_vars="year")
#by_decade_long
```

In [583]:
```
from ggplot import*
from plotnine import *
from plotly.tools import mpl_to_plotly as ggplotly
by_decade_long = pd.melt(by_decade, id_vars="year")
p2 = ggplot(aes(x='year', weight='value', fill='variable'), data=by_de
cade_long) +\
geom_bar(colour='black', width=4) + theme(text = element_text(size=1
6)) + ggtitle("Meat Production by Decade")
ggplotly(p2.draw())
plt.show(p2.draw())
```

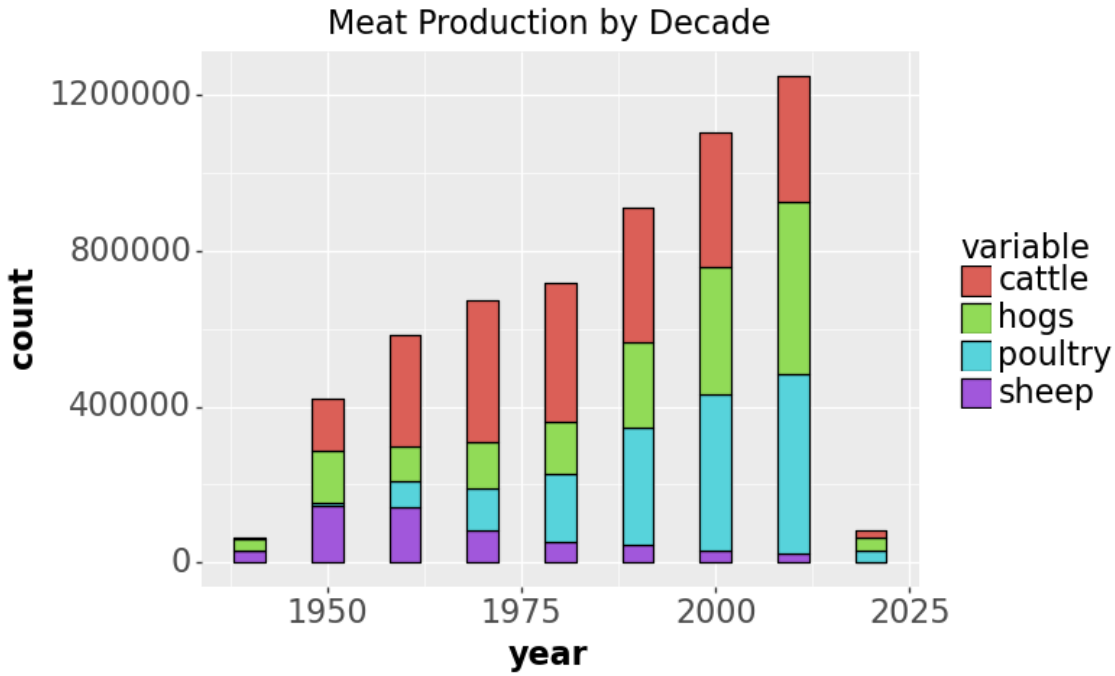


In [584]:
```
plt.figure(figsize=[10,10])
by_decade[['cattle']].plot.bar(ylim=0)
plt.grid(axis='y', alpha=.75)
n=9
ind = np.arange(n)
plt.xticks(ind,('1940','1950','1960','1970','1980','1990','2000','2010
','2020'), fontsize=12, rotation=45)
plt.yticks(fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.xlabel('Year', fontsize=12)
plt.title('Meat Producton by Decade', fontsize=14)
plt.savefig('03_livestock_bar.png', figsize = (7,4), dpi = 300, bbox_i
nches = 'tight');
leg = ax.legend()
leg.remove()
plt.show()
```

```
<Figure size 3000x3000 with 0 Axes>
```



In [585]:
```
by_decade_long = pd.melt(by_decade, id_vars="year")
p2 = ggplot(aes(x='year', weight='value', fill='variable'), data=by_de
cade_long) +\
geom_bar(colour='black', width=4) + theme(text = element_text(size=1
6)) + ggtitle("Meat Production by Decade")
ggplotly(p2.draw())
plt.show(p2.draw())
```

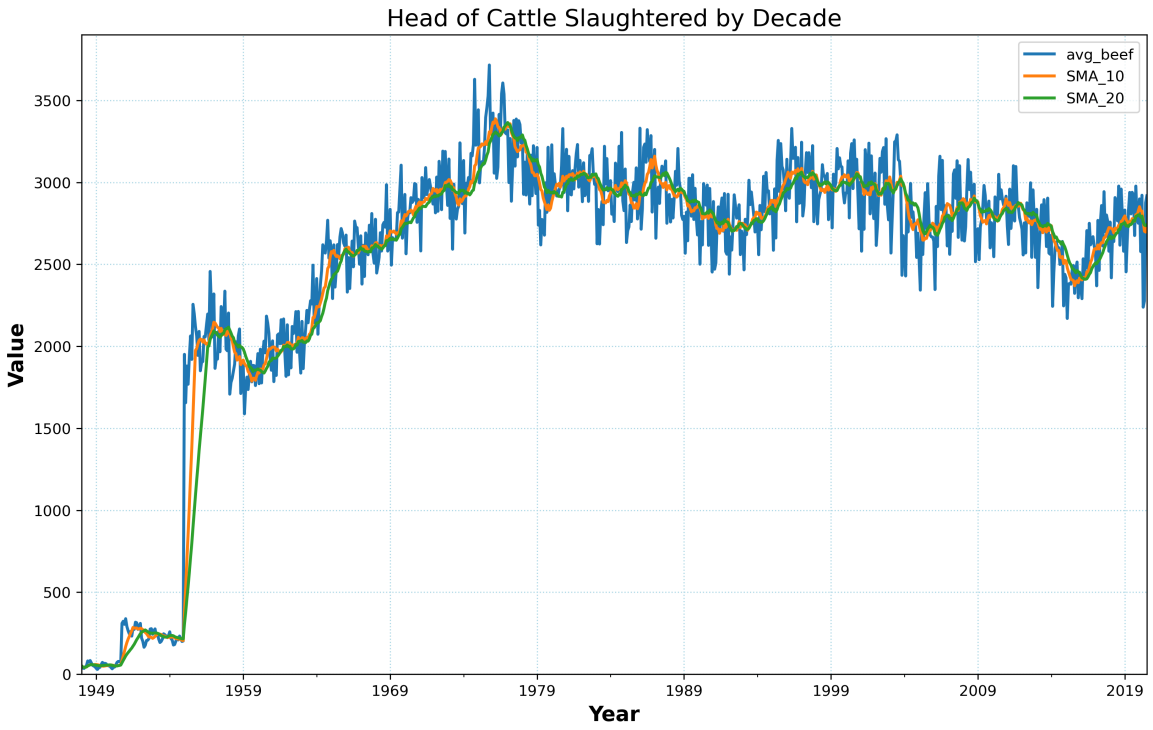


In [586]:
```
beef_df= meat[["date","beef"]]
beef_df.head(5)
beef_ts = beef_df.set_index(['date'])
beef_ts.head(10)
ts2 = meat.set_index(['date'])
ts2.head(5)
ts3=ts2.groupby(ts2.index.year).sum().head(10)
ts3
```

Out[586]:

date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
1944	8801.0	1629.0	11502.0	1001.0	0.0	0.0	0.0
1945	9936.0	1552.0	8843.0	1030.0	0.0	0.0	0.0
1946	9010.0	1329.0	9220.0	946.0	0.0	0.0	0.0
1947	10096.0	1493.0	8811.0	779.0	0.0	0.0	0.0
1948	8766.0	1323.0	8486.0	728.0	0.0	0.0	0.0
1949	9142.0	1240.0	8875.0	587.0	0.0	0.0	0.0
1950	9248.0	1137.0	9397.0	581.0	0.0	0.0	0.0
1951	8549.0	972.0	10190.0	508.0	0.0	0.0	0.0
1952	9337.0	1080.0	10321.0	635.0	0.0	0.0	0.0
1953	12055.0	1451.0	8971.0	715.0	0.0	0.0	0.0

In [593]:
```
matplotlib.rcParams['figure.dpi'] = 300
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
df_beef.plot(ylim=0, ax=ax)
ax.set_xlabel('Year', size = 14)
ax.set_ylabel('Value', size = 14)
ax.set_title('Head of Cattle Slaughtered by Decade', size = 16)
ax.grid(color = 'lightblue')
plt.show()
```



In [594]:
```
ts.groupby(ts.index.year).sum().head(10)
```

Out[594]:

date	beef	veal	pork	lamb_and_mutton
1944	8801.0	1629.0	11502.0	1001.0
1945	9936.0	1552.0	8843.0	1030.0
1946	9010.0	1329.0	9220.0	946.0
1947	10096.0	1493.0	8811.0	779.0
1948	8766.0	1323.0	8486.0	728.0
1949	9142.0	1240.0	8875.0	587.0
1950	9248.0	1137.0	9397.0	581.0
1951	8549.0	972.0	10190.0	508.0
1952	9337.0	1080.0	10321.0	635.0
1953	12055.0	1451.0	8971.0	715.0

In [595]:
```
y = df['cattle']
y.head(5)
```

Out[595]:
```
date
1948-01-01    51.3
1948-02-01    37.7
1948-03-01    35.7
1948-04-01    45.1
1948-05-01    52.9
Name: cattle, dtype: float64
```

In [596]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1534      0.045     -3.436      0.001      -0.241      -0.066
ma.L1         -0.3914      0.040     -9.800      0.000      -0.470      -0.313
ar.S.L12      -0.2223      0.012    -18.771      0.000      -0.245      -0.199
sigma2      2.518e+04    401.997     62.634      0.000    2.44e+04    2.6e+04
==============================================================================
```

In [597]:
```
import itertools
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in
                list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [598]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
seasonal_order=param_seasonal,
enforce_stationarity=False,
enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasona
l, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:16175.378659699785
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:14885.70286673959
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:11991.213306306003
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:11835.712905497232
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:12005.577898098692
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:11993.22274699329
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:11848.767695643653
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:11837.415189833366
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:15532.893182604368
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:13846.575806172135
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:11535.778734459482
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:11387.0464195048
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:11563.143321222704
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:11539.165451643086
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:11412.306551725806
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:11328.614639183697
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:11787.520863154414
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:11281.092026971632
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:11333.646759052614
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:10973.308588584325
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:11192.962289891519
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:11175.863185298655
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:11191.21618571537
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:10900.519051544876
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:11484.680986629673
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:11064.748227137454
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:11139.279531004217
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:10746.63314247163
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:10988.684583531734
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:10895.363711124948
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:10985.706378194305
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:10699.140943049317
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:11801.208967216911
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:11293.789837708706
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:11240.295902183712
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:10970.573963394607
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:11188.887158723504
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:11184.838082169075
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:11090.502152997524
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:10893.303801306207
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:11500.612297076554
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:11078.993458236424
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:11121.504899861826
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:10755.1013312102
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:10990.006951153966
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:10904.40245146786
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:10963.563730997312
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:10705.301597042984
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:11490.891791092205
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:11077.238900499107
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:11187.869972559194
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:10804.122875818866
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:11005.917119655962
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:11006.339537803156
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:11020.91109000023
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:10750.076281127022
```

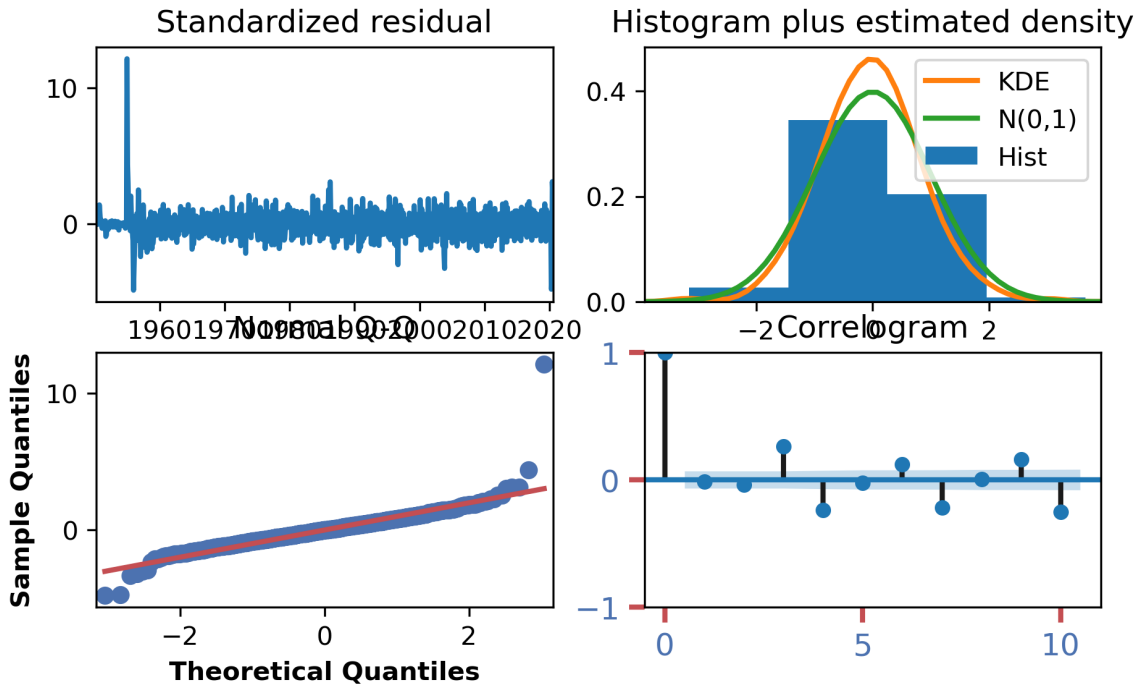
```
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:11449.23577949869
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:11046.130492006661
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:11136.98264430419
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:10742.044861724195
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:10969.554791004102
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:10890.369233026986
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:10969.058880358542
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:10696.678505990962
```

In [599]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1371      0.046     -3.002      0.003      -0.227      -0.048
ma.L1         -0.3748      0.042     -9.022      0.000      -0.456      -0.293
ar.S.L12       0.2627      0.016     16.586      0.000       0.232       0.294
ma.S.L12      -1.0626      0.020    -54.304      0.000      -1.101      -1.024
sigma2      1.597e+04    427.560     37.350      0.000    1.51e+04    1.68e+04
==============================================================================
```

In [600]:
```
#plt.subplots(figsize=(7, 4), dpi = 150)
from matplotlib import *
plt.subplots_adjust(wspace=0.5, hspace=0.5)
results.plot_diagnostics(figsize = (7,4))
# modify ticks size
plt.rcParams.update({"figure.figsize" : (7, 4),
                     "axes.edgecolor": "black",
                     "text.color": "slateblue",
                     "figure.figsize": [7, 4],
                     "figure.dpi"    : 300,
                     "savefig.dpi"   : 100,
                     "font.size"     : 14.0}),
plt.tick_params(axis = 'both', direction ='out', length = 6,
                width = 2, labelcolor = 'b', colors = 'r',
                grid_color = 'gray', grid_alpha = 0.5)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
ax.grid(True, color = 'gray', linestyle = ':')
plt.savefig('03_beef_line.png', figsize = (12,5), dpi = 300, bbox_inch
es = 'tight');
plt.show()
```

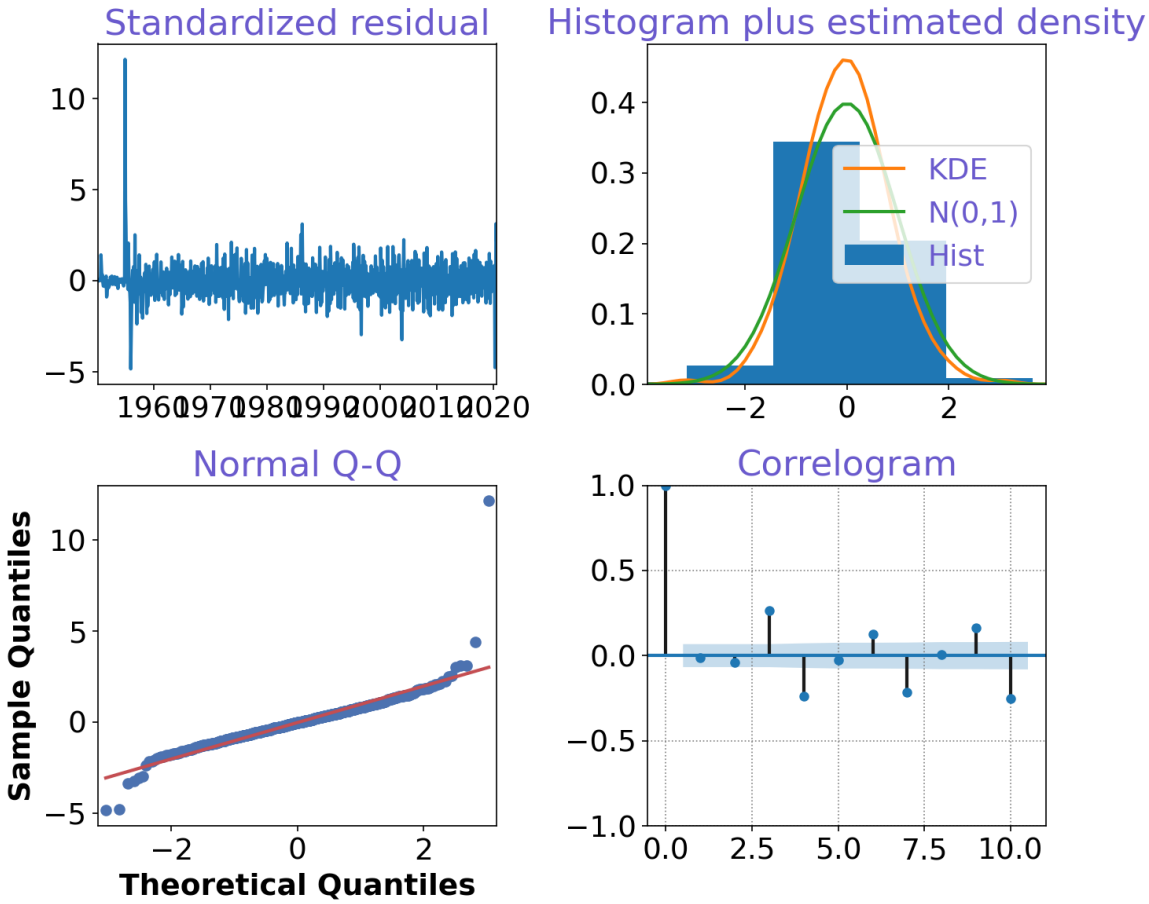
```
<Figure size 3000x1800 with 0 Axes>
```



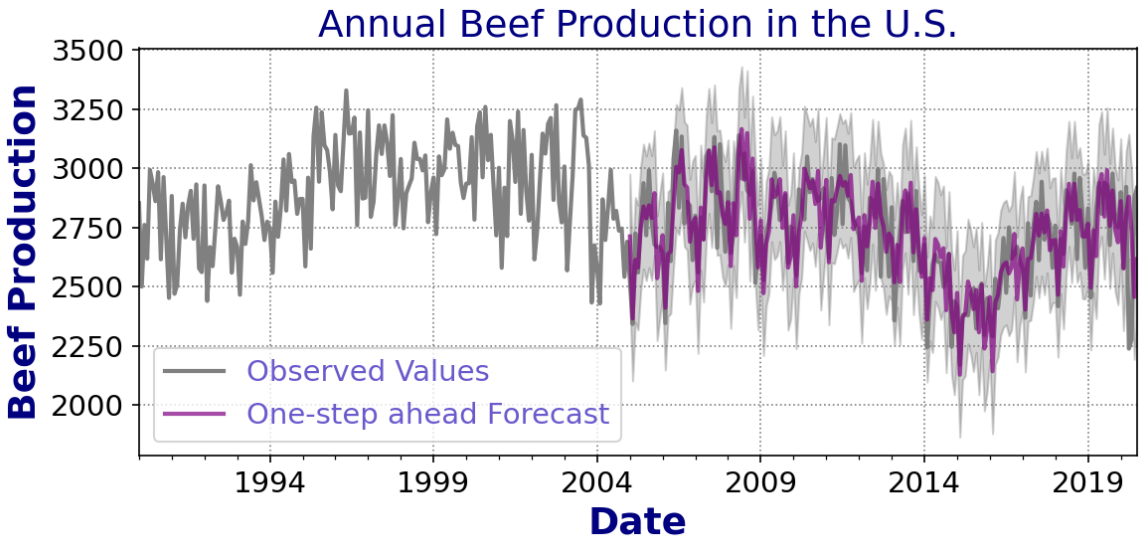
In [601]:
```
plt.rcParams.update({"figure.figsize" : (8, 5),
                     "axes.edgecolor": "black",
                     "text.color": "slateblue",
                     "figure.figsize": [8, 5],
                     "figure.dpi"     : 150,
                     "savefig.dpi"    : 150,
                     "font.size"      : 18.0}),
#plt.tick_params(axis = 'both', direction ='out', length = 6,
                # width = 2, labelcolor = 'b', colors = 'r',
                # grid_color = 'gray', grid_alpha = 0.5)
```

Out[601]: (None,)

In [602]:
```
results.plot_diagnostics(figsize=(10, 8))
mpl.rcParams['font.size'] = 14
plt.grid(True, color = 'gray', linestyle = ':')
plt.tight_layout()
plt.savefig('0000_66', figsize = (10,8), dpi = 200);
plt.show()
```



In [603]:
```
pred = results.get_prediction(start=pd.to_datetime('2005-01-01'), dyna
mic=False)
pred_ci = pred.conf_int()
ax = y['1990':].plot(label='Observed Values', color='gray')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha
=.7, figsize=(8, 4), color='purple')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date', fontsize = 18, color = 'navy')
ax.set_ylabel('Beef Production', fontsize = 18, color = 'navy')
ax.set_title('Annual Beef Production in the U.S.', fontsize = 18, colo
r = 'navy')
plt.grid(True, color = 'gray', linestyle = ':')
plt.tight_layout()
ax.legend()
plt.savefig('0000_67.png', figsize = (8,4), dpi = 300);
plt.show()
```



In [604]:
```
meat = meat.dropna(thresh=800, axis=1) # drop columns that have fewer
than 800 observations
ts = meat.set_index(['date'])
```

In [605]:
```
ts.groupby(ts.index.year).sum().head(10)
```

Out[605]:

date	beef	veal	pork	lamb_and_mutton
1944	8801.0	1629.0	11502.0	1001.0
1945	9936.0	1552.0	8843.0	1030.0
1946	9010.0	1329.0	9220.0	946.0
1947	10096.0	1493.0	8811.0	779.0
1948	8766.0	1323.0	8486.0	728.0
1949	9142.0	1240.0	8875.0	587.0
1950	9248.0	1137.0	9397.0	581.0
1951	8549.0	972.0	10190.0	508.0
1952	9337.0	1080.0	10321.0	635.0
1953	12055.0	1451.0	8971.0	715.0

In [606]:
```
the1940s.sum().reset_index()
```

Out[606]:

	index	cattle	hogs	sheep	poultry
0	1948-01-01	51.3	144.6	1464.0	10.4
1	1948-02-01	37.7	665.6	1306.7	87.6
2	1948-03-01	35.7	1242.0	1297.2	79.6
3	1948-04-01	45.1	377.5	1178.1	14.7
4	1948-05-01	52.9	833.5	1095.3	67.2
...	...	...	...	...	...
866	2020-03-01	2921.7	5942.0	187.5	4427.8
867	2020-04-01	2239.1	3412.0	180.8	4098.9
868	2020-05-01	2277.5	2595.5	195.3	4039.1
869	2020-06-01	2876.3	5176.8	193.4	4334.7
870	2020-07-01	2918.4	5225.4	195.1	4303.0

871 rows × 5 columns

In [607]:
```
def floor_decade(date_value):
    "Takes a date. Returns the decade."
    return (date_value.year // 10) * 10
```

In [608]:
```
ts.groupby(floor_decade).sum()

the1940s.sum().reset_index()

by_decade = ts.groupby(floor_decade).sum()

by_decade.index.name = 'year'

by_decade = by_decade.reset_index()
```

In [609]:
```
by_decade
```

Out[609]:

	year	beef	veal	pork	lamb_and_mutton
0	1940	55751.0	8566.0	55737.0	5071.0
1	1950	119161.0	12693.0	98450.0	6724.0
2	1960	177754.0	8577.0	116587.0	6873.0
3	1970	228947.0	5713.0	132539.0	4256.0
4	1980	230100.0	4278.0	150528.0	3394.0
5	1990	243579.0	2938.0	173519.0	2986.0
6	2000	260540.7	1685.3	208211.3	1964.7
7	2010	76391.5	371.9	66491.2	455.6

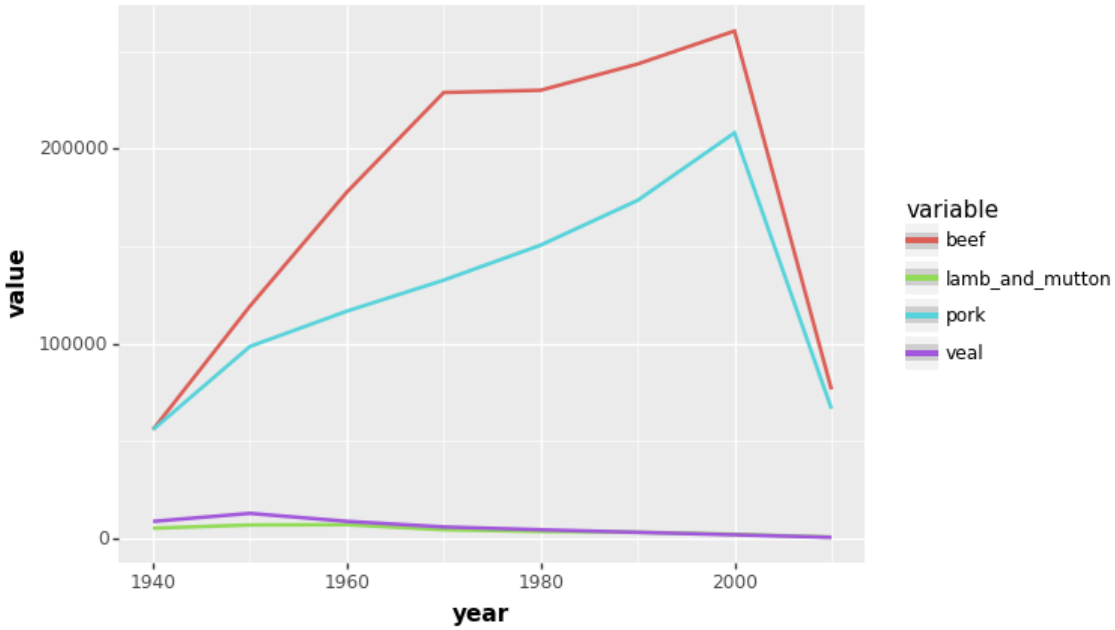
In [610]:
```
by_decade_long = pd.melt(by_decade, id_vars="year")
by_decade_long
```

Out[610]:

	year	variable	value
0	1940	beef	55751.0
1	1950	beef	119161.0
2	1960	beef	177754.0
3	1970	beef	228947.0
4	1980	beef	230100.0
5	1990	beef	243579.0
6	2000	beef	260540.7
7	2010	beef	76391.5
8	1940	veal	8566.0
9	1950	veal	12693.0
10	1960	veal	8577.0
11	1970	veal	5713.0
12	1980	veal	4278.0
13	1990	veal	2938.0
14	2000	veal	1685.3
15	2010	veal	371.9
16	1940	pork	55737.0
17	1950	pork	98450.0
18	1960	pork	116587.0
19	1970	pork	132539.0
20	1980	pork	150528.0
21	1990	pork	173519.0
22	2000	pork	208211.3
23	2010	pork	66491.2
24	1940	lamb_and_mutton	5071.0
25	1950	lamb_and_mutton	6724.0
26	1960	lamb_and_mutton	6873.0
27	1970	lamb_and_mutton	4256.0
28	1980	lamb_and_mutton	3394.0
29	1990	lamb_and_mutton	2986.0
30	2000	lamb_and_mutton	1964.7
31	2010	lamb_and_mutton	455.6

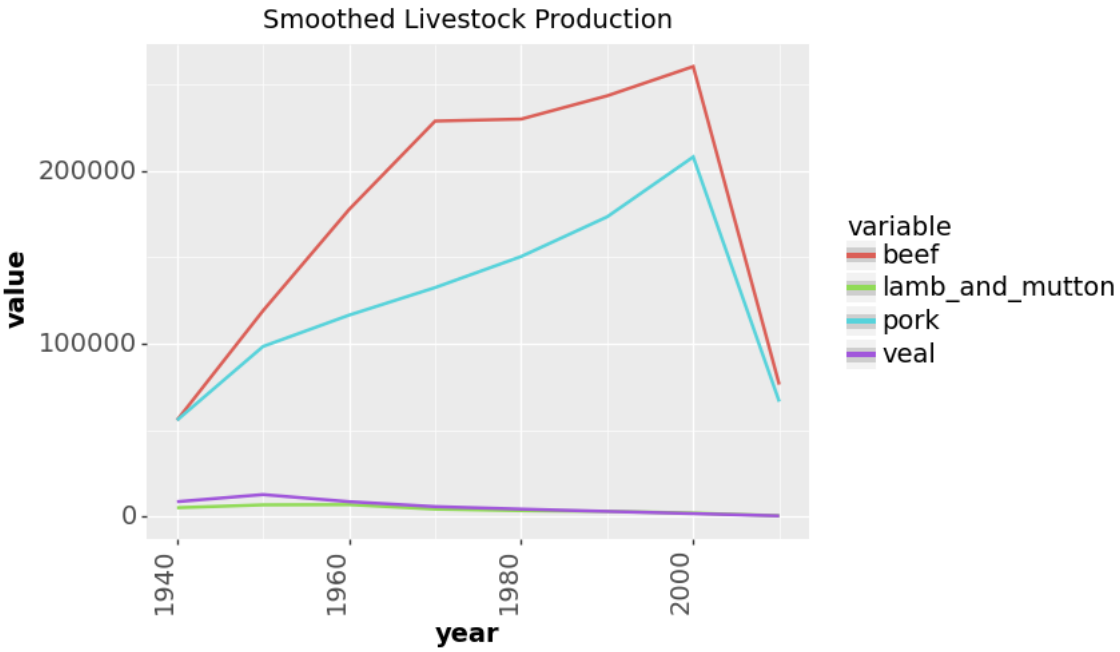
In [627]:
```
import seaborn as sns
sns.set()
#by_decade_long.set_index('variable').T.plot(kind='bar', stacked=True)
```

In [622]:
```
ggplot(aes(x='year', y='value', colour='variable'), data=by_decade_lon
g) + stat_smooth(span=0.10)
```



Out[622]: <ggplot: (86192714484)>

In [630]:
```
ggplot(aes(x='year', y='value', colour='variable'), data=by_decade_lon
g) + stat_smooth(span=0.10) +\
     theme(text = element_text(size=14), axis_text_x  = element_text(a
ngle = 90, hjust = 1)) +\
     ggtitle("Smoothed Livestock Production")
```



Out[630]: <ggplot: (86193397429)>

In [631]:
```
meat = load_meat()
meat = meat[['date','beef','veal','lamb_and_mutton','broilers','turkey
']]
meat.head(10)
```

Out[631]:

	date	beef	veal	lamb_and_mutton	broilers	turkey
0	1944-01-01	751.0	85.0	89.0	NaN	NaN
1	1944-02-01	713.0	77.0	72.0	NaN	NaN
2	1944-03-01	741.0	90.0	75.0	NaN	NaN
3	1944-04-01	650.0	89.0	66.0	NaN	NaN
4	1944-05-01	681.0	106.0	78.0	NaN	NaN
5	1944-06-01	658.0	125.0	79.0	NaN	NaN
6	1944-07-01	662.0	142.0	82.0	NaN	NaN
7	1944-08-01	787.0	175.0	87.0	NaN	NaN
8	1944-09-01	774.0	182.0	91.0	NaN	NaN
9	1944-10-01	834.0	215.0	100.0	NaN	NaN

In [632]:
```
meat_ts = meat.set_index('date')['1960':'2009']
meat_ts
```

Out[632]:

date	beef	veal	lamb_and_mutton	broilers	turkey
1960-01-01	1196.0	79.0	68.0	255.9	22.1
1960-02-01	1089.0	73.0	60.0	250.9	14.0
1960-03-01	1201.0	83.0	61.0	283.6	13.4
1960-04-01	1066.0	75.0	59.0	301.6	16.2
1960-05-01	1202.0	77.0	61.0	331.7	27.1
...	...	...	...	...	...
2009-08-01	2184.2	10.6	13.3	3012.1	463.4
2009-09-01	2234.1	11.7	14.8	3064.3	467.9
2009-10-01	2275.7	12.2	14.3	3071.9	508.3
2009-11-01	2016.1	11.7	14.3	2788.8	477.5
2009-12-01	2134.4	13.1	15.9	2966.2	456.3

600 rows × 5 columns

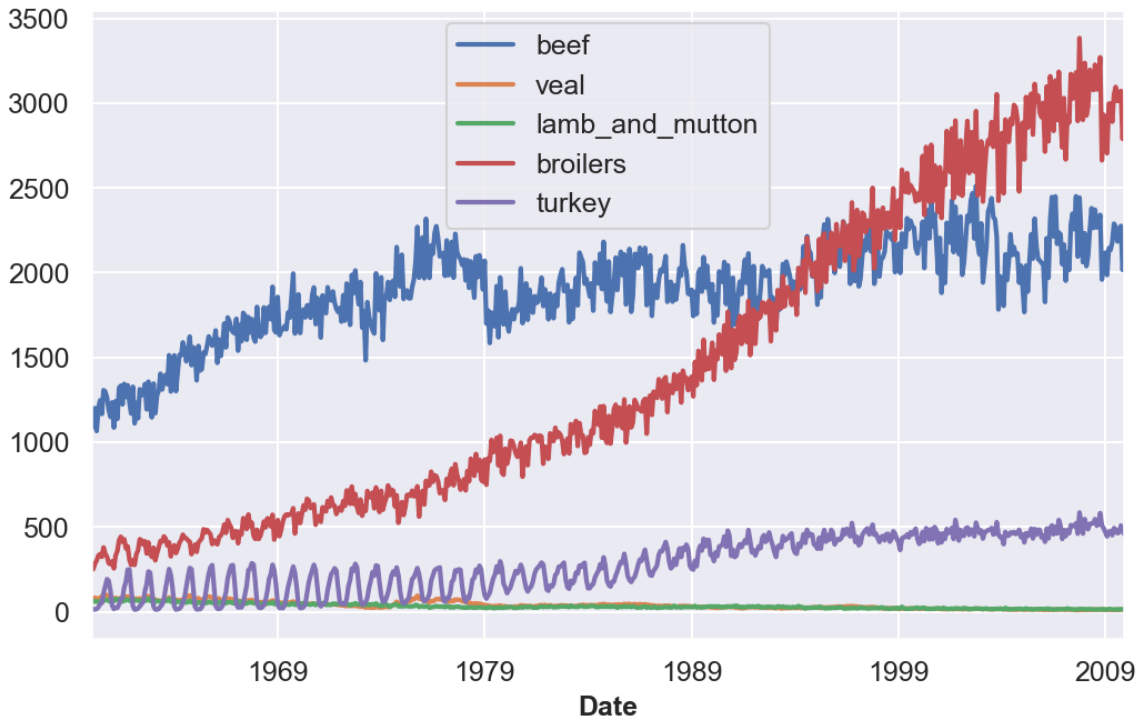
In [633]:
```
meat_ts['beef']
```

Out[633]:
```
date
1960-01-01    1196.0
1960-02-01    1089.0
1960-03-01    1201.0
1960-04-01    1066.0
1960-05-01    1202.0
               ...
2009-08-01    2184.2
2009-09-01    2234.1
2009-10-01    2275.7
2009-11-01    2016.1
2009-12-01    2134.4
Name: beef, Length: 600, dtype: float64
```

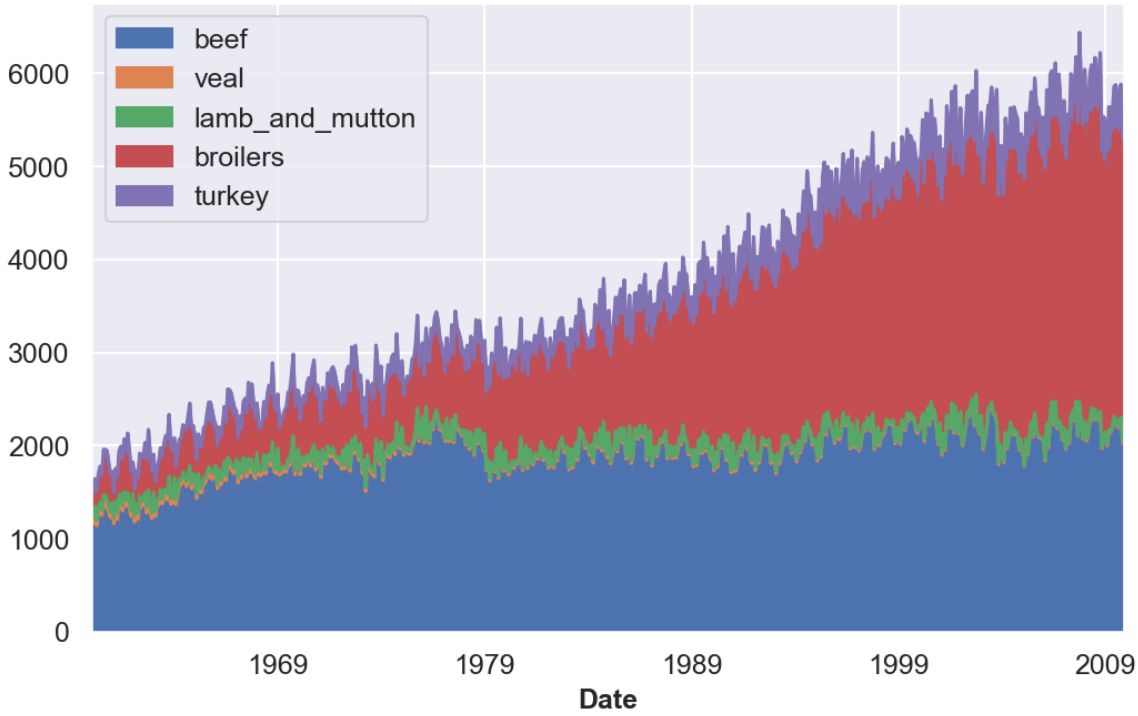
In [634]:
```
# Print the summary statistics of the DataFrame
print(meat_ts.describe())
```

```
                beef        veal  lamb_and_mutton     broilers       tur
key
count    600.000000  600.000000       600.000000   600.000000  600.000
000
mean    1901.534500   38.652167        32.456167  1424.565833  281.628
167
std      285.976654   21.960290        15.131929   909.040251  160.007
076
min     1066.000000    9.400000        12.700000   250.900000   12.400
000
25%     1759.000000   22.000000        21.000000   620.675000  147.850
000
50%     1936.000000   33.000000        28.000000  1099.500000  271.150
000
75%     2102.250000   53.000000        42.250000  2273.750000  439.325
000
max     2512.000000  101.000000        77.000000  3383.800000  585.100
000
```

In [635]:
```
ax = meat_ts.plot(linewidth=2, fontsize=12);

# Additional customizations
ax.set_xlabel('Date');
ax.legend(fontsize=12);
```

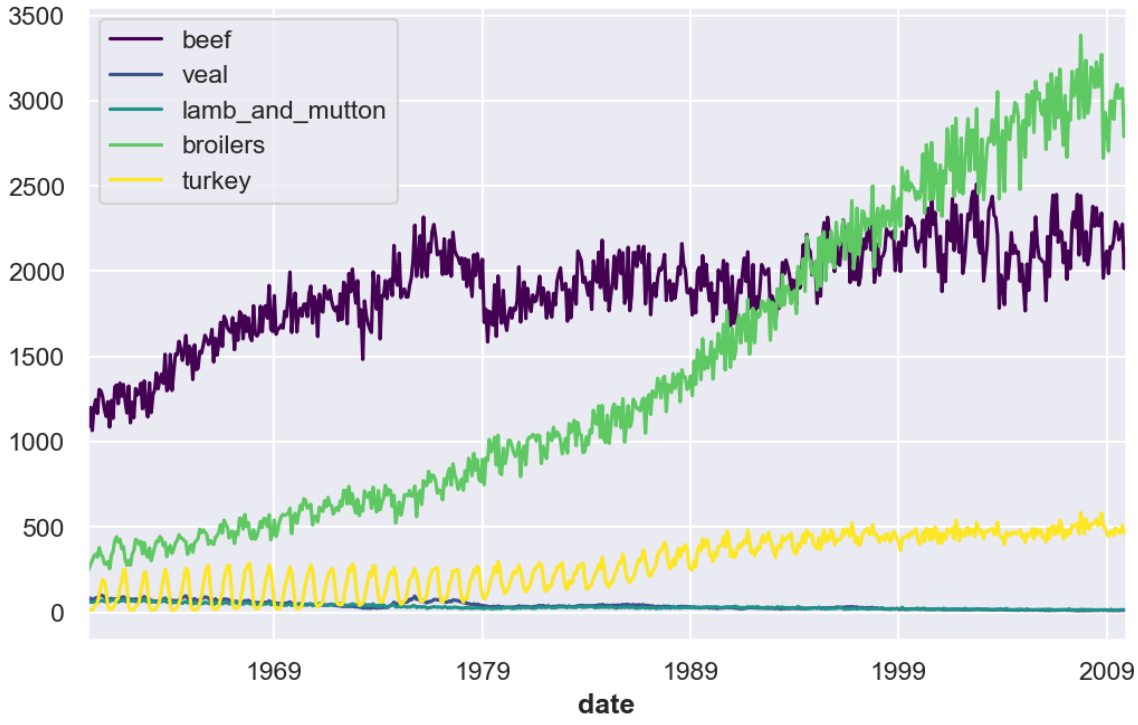


In [636]:
```
ax = meat_ts.plot.area(fontsize=12);

# Additional customizations
ax.set_xlabel('Date');
ax.legend(fontsize=12);
```

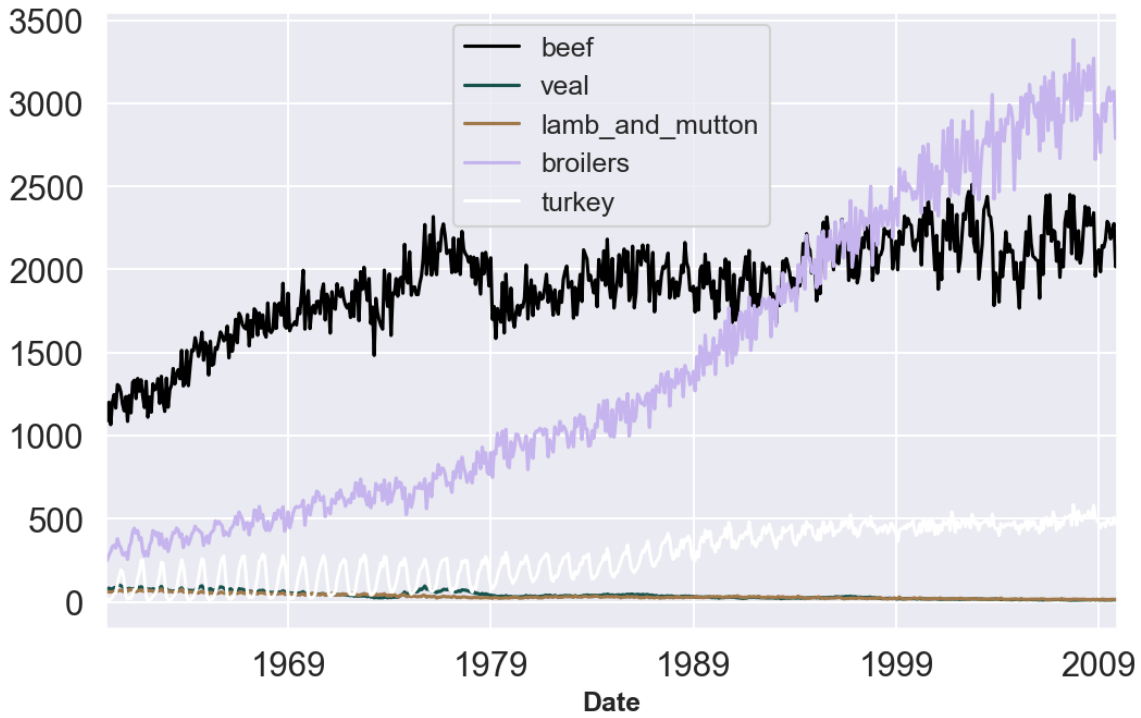


In [637]:
```
meat_ts.plot(colormap='viridis')
```

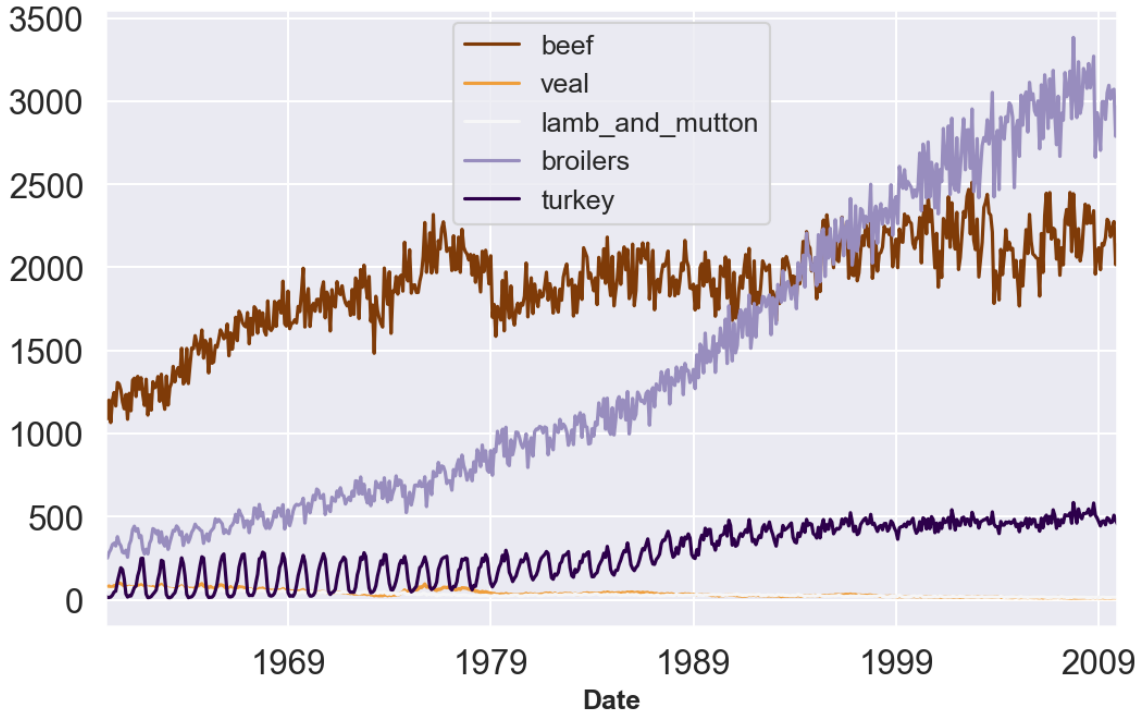
Out[637]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1411aefc460>
```



In [638]:
```
ax = meat_ts.plot(colormap='cubehelix', fontsize=15);
# Additional customizations
ax.set_xlabel('Date');
ax.legend(fontsize=12);
```



In [639]:
```
ax = meat_ts.plot(colormap='PuOr', fontsize=15);

# Additional customizations
ax.set_xlabel('Date');
ax.legend(fontsize=12);
```

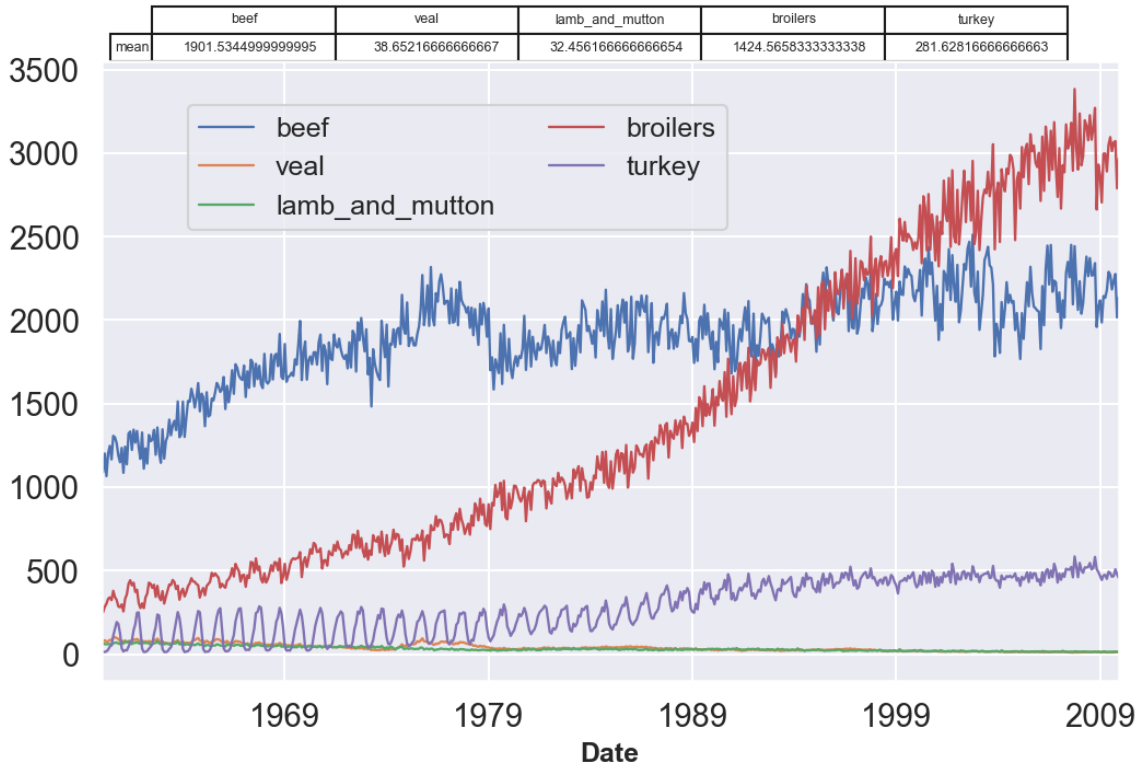


In [640]:
```
meat_mean = meat_ts.mean().to_frame().T
meat_mean.rename({0:'mean'}, inplace=True)
meat_mean
```

Out[640]:

	beef	veal	lamb_and_mutton	broilers	turkey
mean	1901.5345	38.652167	32.456167	1424.565833	281.628167

In [641]:
```
ax = meat_ts.plot(fontsize=14, linewidth=1.1);

# Add x-axis labels
ax.set_xlabel('Date', fontsize=12);

# Add summary table information to the plot
ax.table(cellText=meat_mean.values,
        colWidths=[0.18] * len(meat_mean.columns),
        rowLabels=meat_mean.index,
        colLabels=meat_mean.columns,
        loc='top');

# Specify the fontsize and location of your legend
ax.legend(loc='upper center', bbox_to_anchor=(0.35, 0.95), ncol=2, fon
tsize=12);
```



In [642]:
```
import pandasql
from pandasql import sqldf, load_meat, load_births
pysqldf = lambda q: sqldf(q, globals())
meat = load_meat()
births = load_births()
meat.head(5)
```

Out[642]:

	date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
0	1944-01-01	751.0	85.0	1280.0	89.0	NaN	NaN	NaN
1	1944-02-01	713.0	77.0	1169.0	72.0	NaN	NaN	NaN
2	1944-03-01	741.0	90.0	1128.0	75.0	NaN	NaN	NaN
3	1944-04-01	650.0	89.0	978.0	66.0	NaN	NaN	NaN
4	1944-05-01	681.0	106.0	1029.0	78.0	NaN	NaN	NaN

In [643]:
```
beef_df= meat[["date","beef"]]
beef_df.head(5)
```

Out[643]:

	date	beef
0	1944-01-01	751.0
1	1944-02-01	713.0
2	1944-03-01	741.0
3	1944-04-01	650.0
4	1944-05-01	681.0

In [644]:
```
beef_df= meat[["date","beef"]]
beef_df.head(5)
beef_ts = beef_df.set_index(['date'])
beef_ts.head(10)
```

Out[644]:

date	beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0
1944-06-01	658.0
1944-07-01	662.0
1944-08-01	787.0
1944-09-01	774.0
1944-10-01	834.0

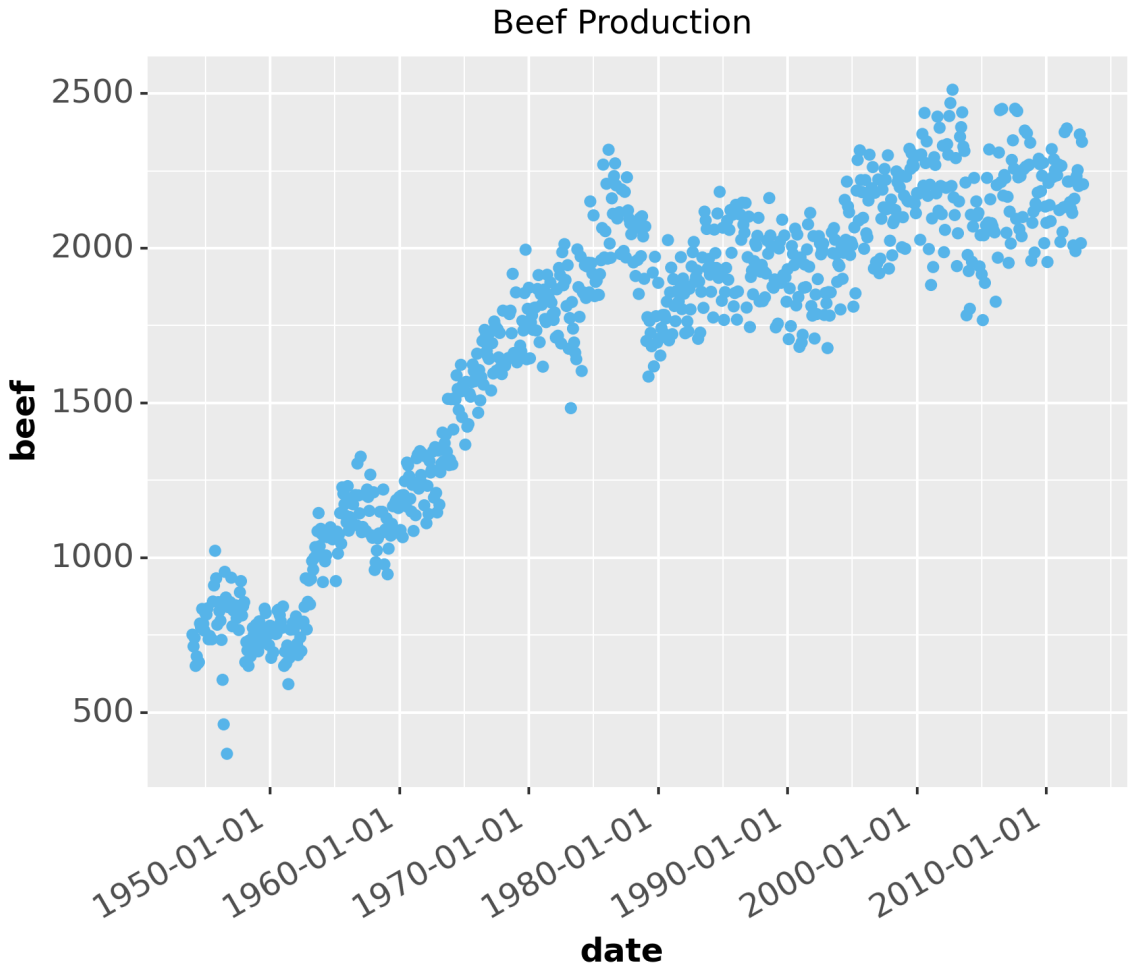
In [645]:
```
beef_df= meat[["date","beef"]]
beef_df.head(5)
beef_ts = beef_df.set_index(['date'])
beef_ts.head(10)
ts2 = meat.set_index(['date'])
ts2.head(5)
ts3=ts2.groupby(ts2.index.year).sum().head(10)
ts3
```

Out[645]:

date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
1944	8801.0	1629.0	11502.0	1001.0	0.0	0.0	0.0
1945	9936.0	1552.0	8843.0	1030.0	0.0	0.0	0.0
1946	9010.0	1329.0	9220.0	946.0	0.0	0.0	0.0
1947	10096.0	1493.0	8811.0	779.0	0.0	0.0	0.0
1948	8766.0	1323.0	8486.0	728.0	0.0	0.0	0.0
1949	9142.0	1240.0	8875.0	587.0	0.0	0.0	0.0
1950	9248.0	1137.0	9397.0	581.0	0.0	0.0	0.0
1951	8549.0	972.0	10190.0	508.0	0.0	0.0	0.0
1952	9337.0	1080.0	10321.0	635.0	0.0	0.0	0.0
1953	12055.0	1451.0	8971.0	715.0	0.0	0.0	0.0

In [646]:
```
# The palette with grey:
cbp1=("#999999", "#E69F00", "#56B4E9", "#009E73",
          "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
p = ggplot(aes(x='date', y='beef'), data=meat)
p1 = p + geom_point(color=cbp1[2]) +\
     theme(text = element_text(size=12), dpi = 300, axis_text_x  = ele
ment_text(angle = 30, hjust = 1)) +\
     ggtitle("Beef Production")

ggsave(plot = p1, filename = '0000_70.png', format = 'png', width = 7,
       height = 4, units = 'in', dpi = 300)
```

In [647]:
```
p1
```



Out[647]: <ggplot: (86193276796)>

In [648]:
```
beef_df.head(5)
```

Out[648]:

	date	beef
0	1944-01-01	751.0
1	1944-02-01	713.0
2	1944-03-01	741.0
3	1944-04-01	650.0
4	1944-05-01	681.0

In [649]:
```
beef_ts
```

Out[649]:

	beef
date	
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0
...	...
2012-07-01	2200.8
2012-08-01	2367.5
2012-09-01	2016.0
2012-10-01	2343.7
2012-11-01	2206.6

827 rows × 1 columns

In [650]:
```
#beef_ts.index.date
```

In [651]:
```
beef_ts.groupby(beef_ts.index.date).sum().head(10)
```

Out[651]:

	beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0
1944-06-01	658.0
1944-07-01	662.0
1944-08-01	787.0
1944-09-01	774.0
1944-10-01	834.0

In [652]:
```
# translate index name into English
beef_ts.index.name = 'date'
beef_ts.head(5)
```

Out[652]:

	beef
date	
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0

In [653]:
```
beef_df
```

Out[653]:

	date	beef
0	1944-01-01	751.0
1	1944-02-01	713.0
2	1944-03-01	741.0
3	1944-04-01	650.0
4	1944-05-01	681.0
...	...	...
822	2012-07-01	2200.8
823	2012-08-01	2367.5
824	2012-09-01	2016.0
825	2012-10-01	2343.7
826	2012-11-01	2206.6

827 rows × 2 columns

In [654]:
```
# calculate the yearly average air temperature
beef_ts['avg_beef'] = beef_ts.mean(axis=1)

# drop columns containing monthly values
beef_df2 = beef_ts['avg_beef']

# visualize the first 5 columns
beef_df2.head()
```

Out[654]:
```
date
1944-01-01    751.0
1944-02-01    713.0
1944-03-01    741.0
1944-04-01    650.0
1944-05-01    681.0
Name: avg_beef, dtype: float64
```

In [655]:
```
# The palette with grey:
cbp1=("#999999", "#E69F00", "#56B4E9", "#009E73",
          "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

In [656]:
```
meat.head(5)
```

Out[656]:

	date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
0	1944-01-01	751.0	85.0	1280.0	89.0	NaN	NaN	NaN
1	1944-02-01	713.0	77.0	1169.0	72.0	NaN	NaN	NaN
2	1944-03-01	741.0	90.0	1128.0	75.0	NaN	NaN	NaN
3	1944-04-01	650.0	89.0	978.0	66.0	NaN	NaN	NaN
4	1944-05-01	681.0	106.0	1029.0	78.0	NaN	NaN	NaN

In [657]:
```
pd.melt(meat, id_vars=['date'])
```

Out[657]:

	date	variable	value
0	1944-01-01	beef	751.0
1	1944-02-01	beef	713.0
2	1944-03-01	beef	741.0
3	1944-04-01	beef	650.0
4	1944-05-01	beef	681.0
...	...	...	...
5784	2012-07-01	turkey	497.2
5785	2012-08-01	turkey	530.1
5786	2012-09-01	turkey	453.1
5787	2012-10-01	turkey	579.9
5788	2012-11-01	turkey	515.3

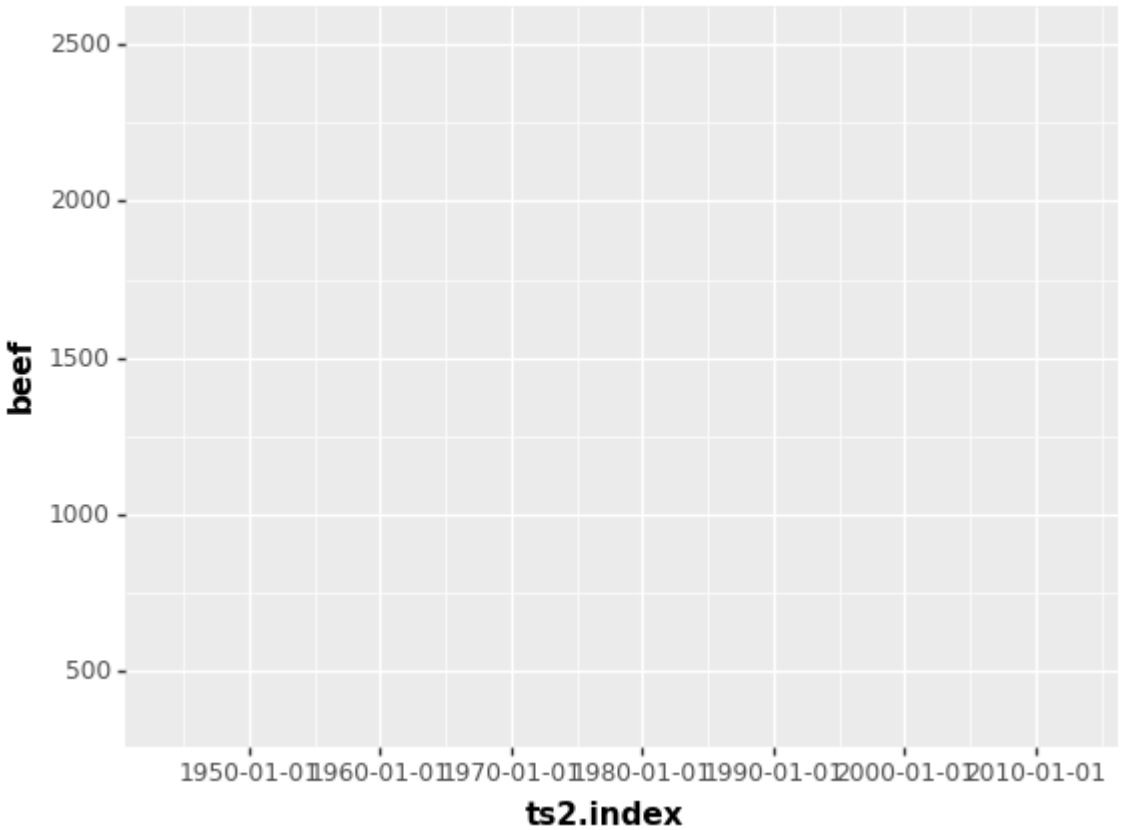
5789 rows × 3 columns

In [658]:
```
ts2 = meat.set_index(['date'])
ts2.head(5)
ts2.groupby(ts2.index.year).sum().head(10)
```

Out[658]:

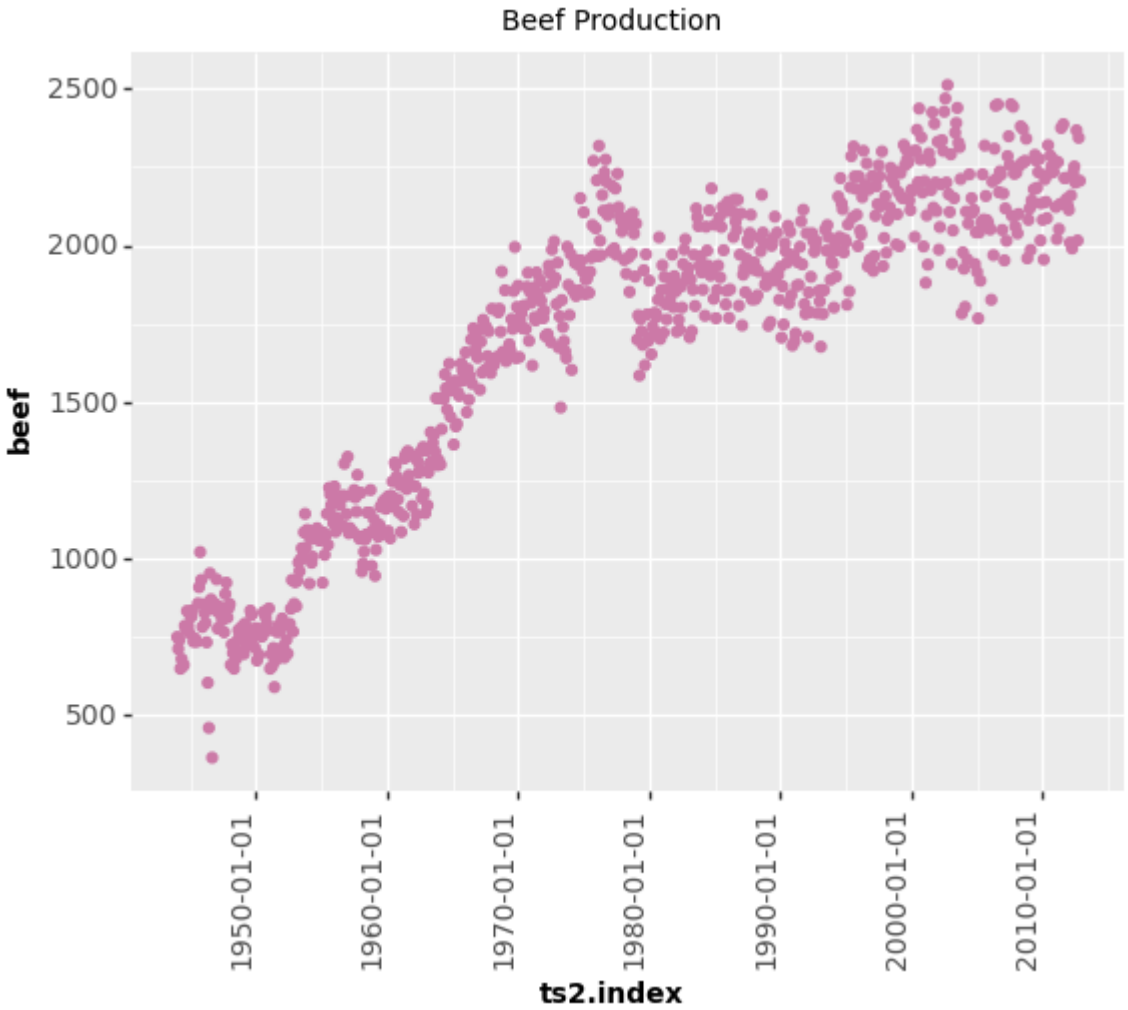
date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
1944	8801.0	1629.0	11502.0	1001.0	0.0	0.0	0.0
1945	9936.0	1552.0	8843.0	1030.0	0.0	0.0	0.0
1946	9010.0	1329.0	9220.0	946.0	0.0	0.0	0.0
1947	10096.0	1493.0	8811.0	779.0	0.0	0.0	0.0
1948	8766.0	1323.0	8486.0	728.0	0.0	0.0	0.0
1949	9142.0	1240.0	8875.0	587.0	0.0	0.0	0.0
1950	9248.0	1137.0	9397.0	581.0	0.0	0.0	0.0
1951	8549.0	972.0	10190.0	508.0	0.0	0.0	0.0
1952	9337.0	1080.0	10321.0	635.0	0.0	0.0	0.0
1953	12055.0	1451.0	8971.0	715.0	0.0	0.0	0.0

In [659]:
```
p = ggplot(aes(x='ts2.index', y='beef'), data=ts2)
p
```



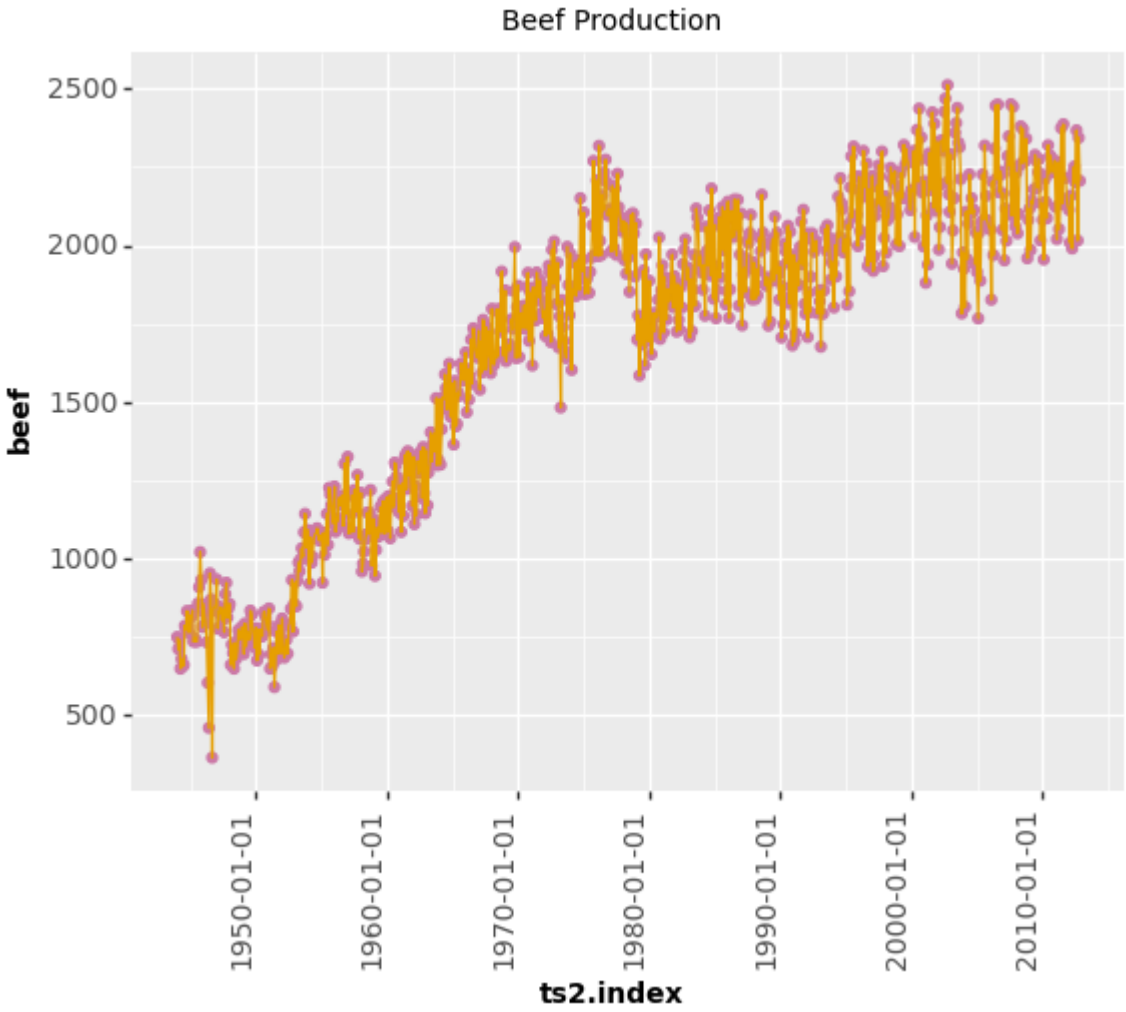
Out[659]: <ggplot: (86192698700)>

In [660]:
```
p + geom_point(color=cbp1[7]) +\
     theme(text = element_text(size=10), axis_text_x  = element_text(a
ngle = 90, hjust = 1)) +\
     ggtitle("Beef Production")
```



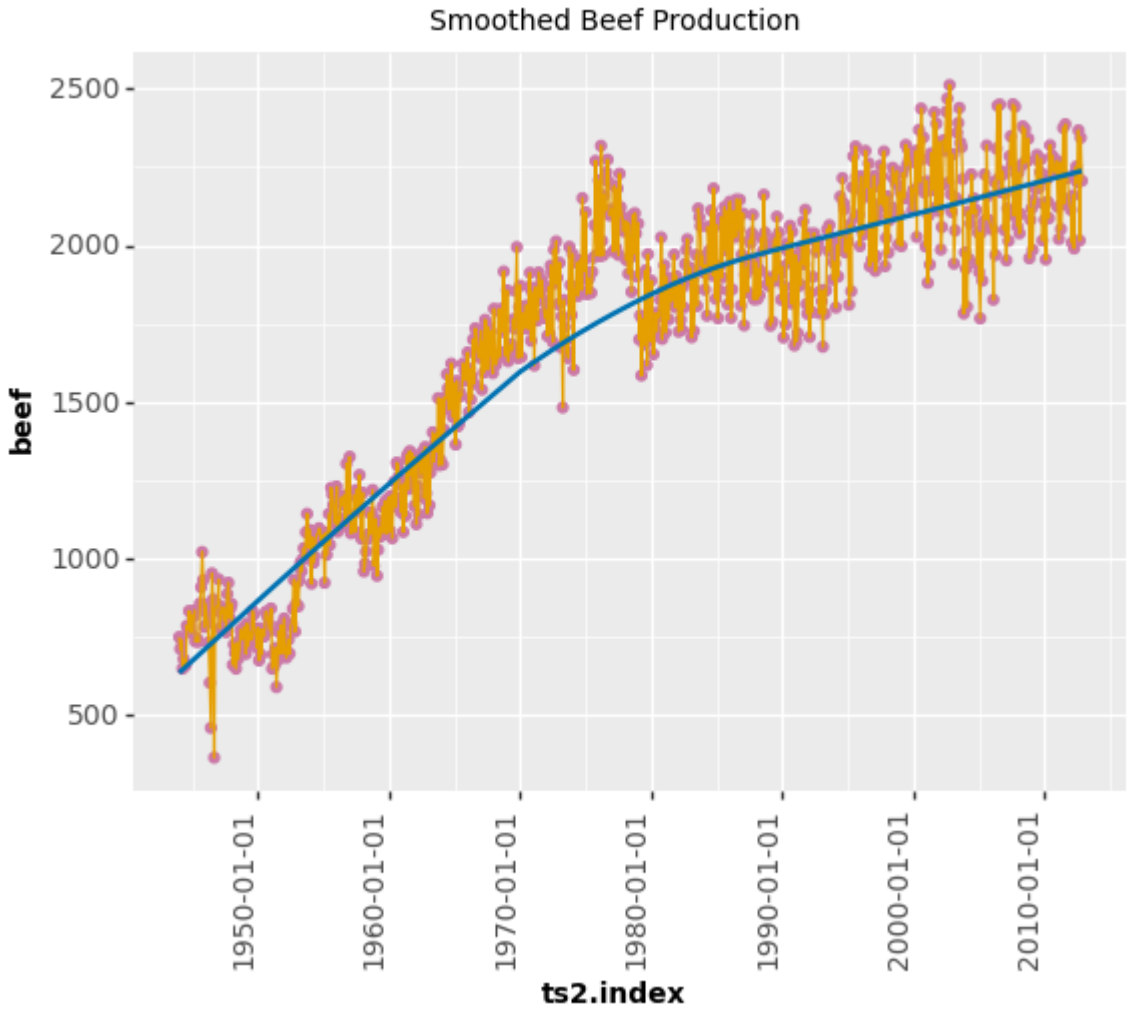
Out[660]: <ggplot: (86194464082)>

In [661]:
```
p + geom_point(color=cbp1[7]) + geom_line(color=cbp1[1]) +\
     theme(text = element_text(size=10), axis_text_x  = element_text(a
ngle = 90, hjust = 1)) +\
     ggtitle("Beef Production")
```



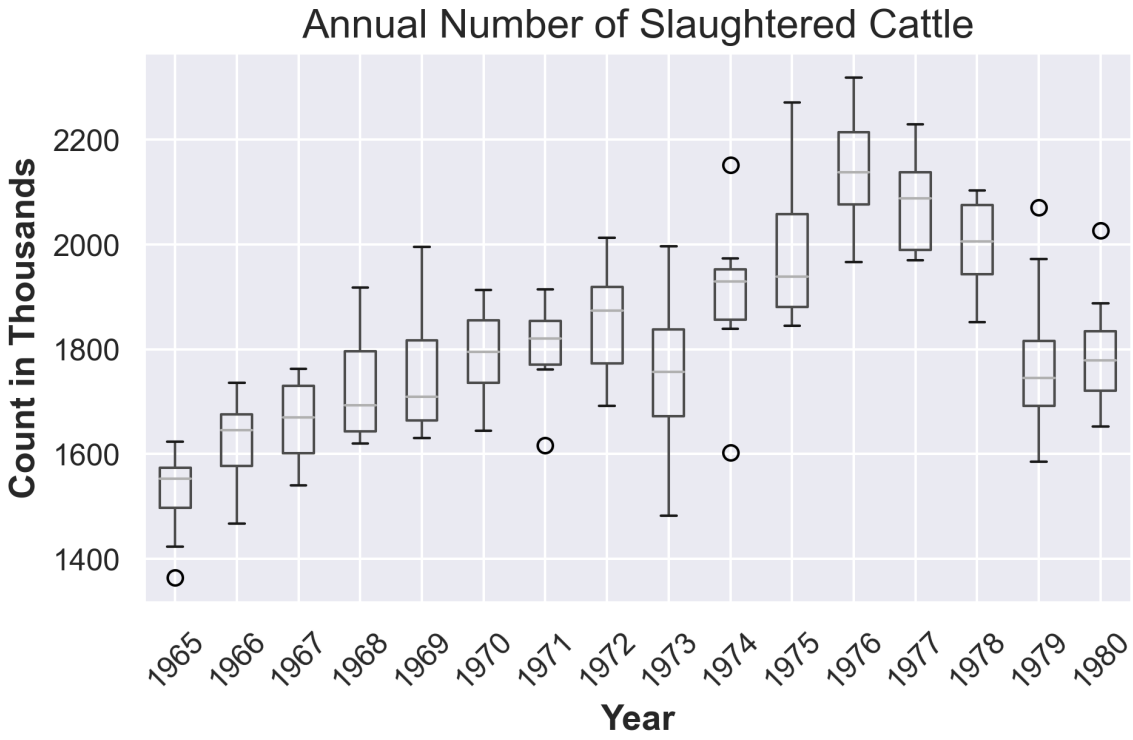
Out[661]: <ggplot: (86190157921)>

In [662]:
```
p + geom_point(color=cbp1[7]) + geom_line(color=cbp1[1]) + stat_smooth
(color=cbp1[5]) +\
     theme(text = element_text(size=10), axis_text_x  = element_text(a
ngle = 90, hjust = 1)) +\
     ggtitle("Smoothed Beef Production")
```

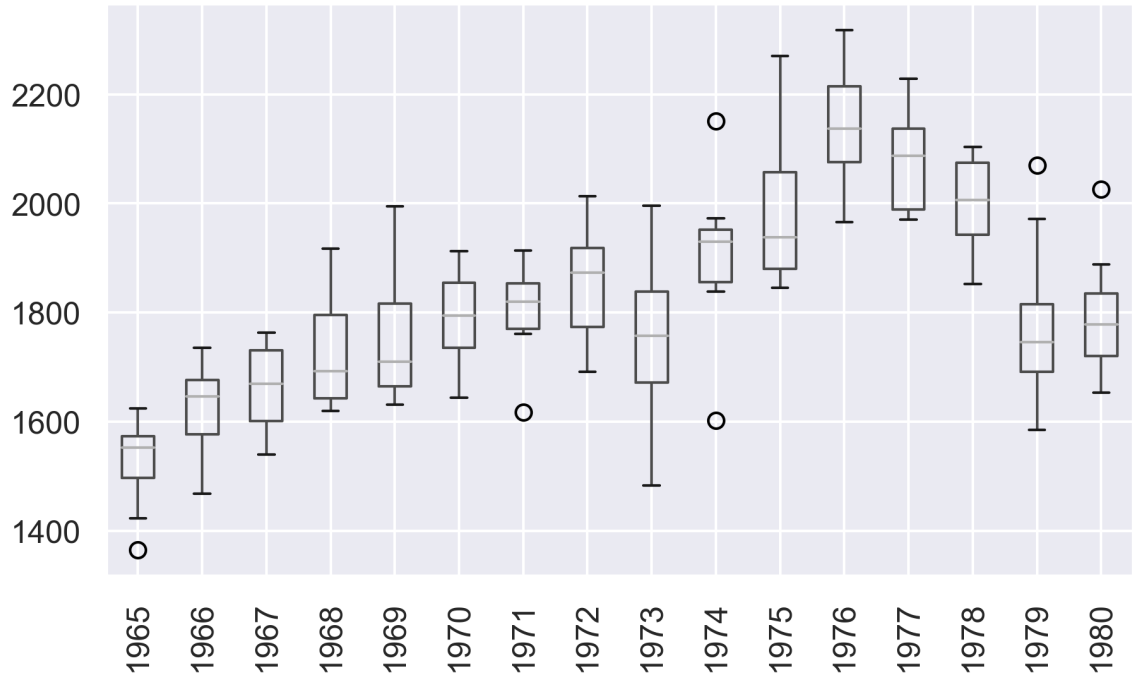


Out[662]: <ggplot: (86190196595)>

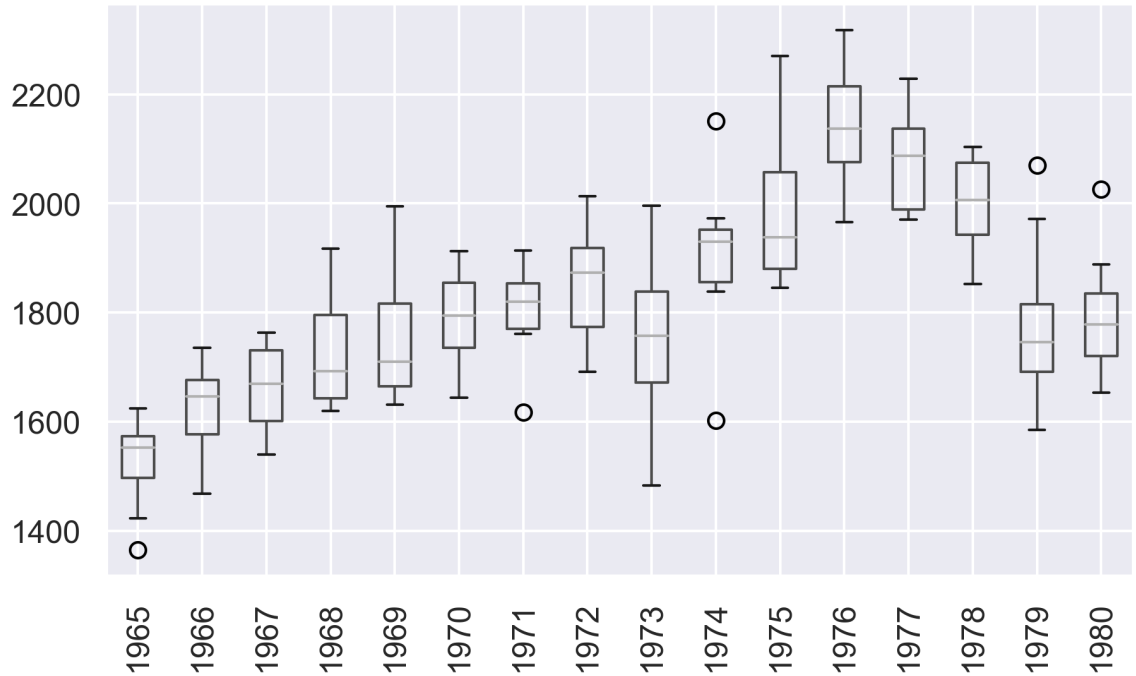
In [663]:
```
from pandas import read_csv
from pandas import DataFrame
from pandas import Grouper
from matplotlib import pyplot
matplotlib.rcParams['figure.dpi'] = 300
groups = beef['1965':'1980'].groupby(Grouper(freq = 'A'))
date = DataFrame()
for name, group in groups:
        date[name.year] = group.values
date.boxplot(figsize=(7,4), rot = 45, fontsize = 12)
plt.title('Annual Number of Slaughtered Cattle', size = 16)
plt.xlabel('Year', size = 14)
plt.ylabel('Count in Thousands', size = 14)
pyplot.show()
```



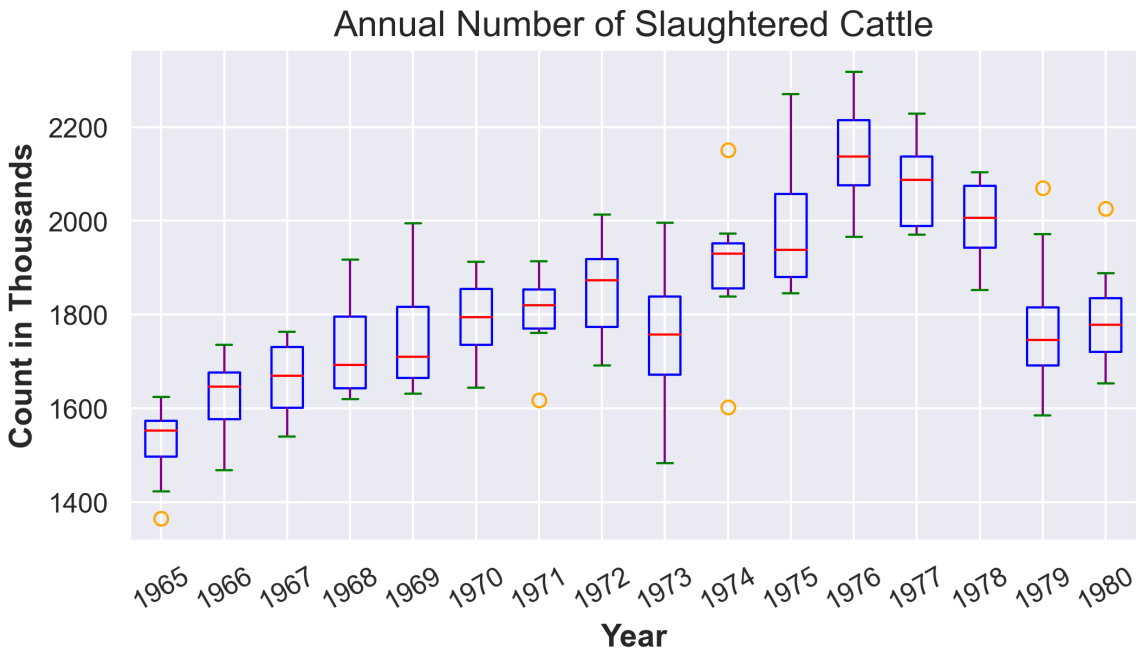
In [664]:
```
from pandas import read_csv
from pandas import DataFrame
from pandas import Grouper
from matplotlib import pyplot
groups = beef['1965':'1980'].groupby(Grouper(freq='A'))
date = DataFrame()
for name, group in groups:
        date[name.year] = group.values
date.boxplot(figsize=(7,4), rot=90, fontsize=12)
pyplot.show()
```



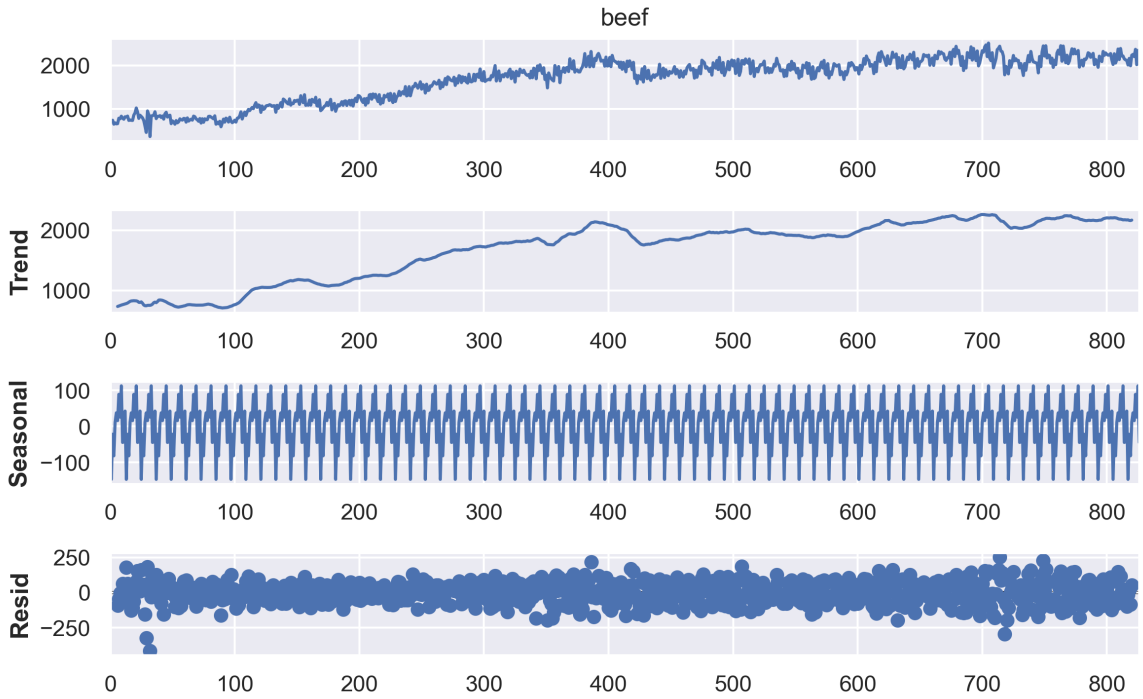
In [665]:
```
from pandas import read_csv
from pandas import DataFrame
from pandas import Grouper
from matplotlib import pyplot
groups = beef['1965':'1980'].groupby(Grouper(freq='A'))
date = DataFrame()
for name, group in groups:
        date[name.year] = group.values
date.boxplot(figsize=(7,4), rot=90, fontsize=12)
pyplot.show()
```



In [666]:
```
# option 1, specify props dictionaries
bp1 = date.boxplot(figsize=(8,4), rot=30, fontsize=12,
            boxprops=dict(color="blue"),
            capprops=dict(color="green"),
            whiskerprops=dict(color="purple"),
            medianprops=dict(color="red"),
            flierprops=dict(color="yellow", markeredgecolor="orange"),
            )
plt.title('Annual Number of Slaughtered Cattle', size = 16)
plt.xlabel('Year', size = 14)
plt.ylabel('Count in Thousands', size = 14)
pyplot.show()
```

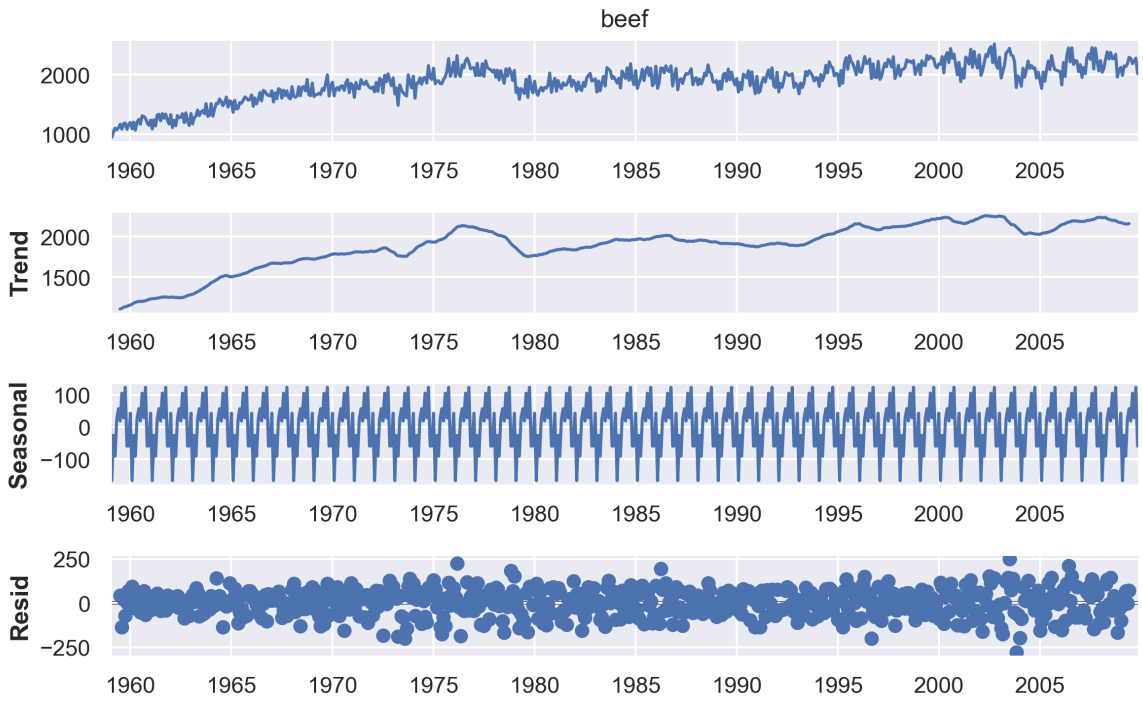


In [667]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
#fig, ax = plt.subplots(linewidth=2, figsize = (6,4), dpi = 300)
decomposition = seasonal_decompose(meat['beef'], freq=12)
decomposition.plot()
plt.savefig('0000_64.png', figsize = (6,4), dpi = 300, bbox_inches = '
tight');
plt.show()
```



In [668]:
```
meat1 = meat.set_index('date')['1959':'2009']
beef1 = meat1['beef']
```

In [669]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(beef1, freq=12)
decomposition.plot()
plt.show()
```



In [670]:
```
df_beef = meat[["date","beef"]]
df_beef.head(5)
```

Out[670]:

	date	beef
0	1944-01-01	751.0
1	1944-02-01	713.0
2	1944-03-01	741.0
3	1944-04-01	650.0
4	1944-05-01	681.0

In [671]:
```
df_beef.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 827 entries, 0 to 826
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    827 non-null    datetime64[ns]
 1   beef    827 non-null    float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 13.0 KB
```

In [672]:
```
# set year column as index
df_beef.set_index('date', inplace=True)
```

In [673]:
```
# translate index name into English
df_beef.index.name = 'month_year'
```

In [674]:
```
df_beef.mean(axis=1)
```

Out[674]:
```
month_year
1944-01-01     751.0
1944-02-01     713.0
1944-03-01     741.0
1944-04-01     650.0
1944-05-01     681.0
               ...
2012-07-01    2200.8
2012-08-01    2367.5
2012-09-01    2016.0
2012-10-01    2343.7
2012-11-01    2206.6
Length: 827, dtype: float64
```

In [675]:
```
# calculate the yearly average air temperature
df_beef['avg_beef'] = df_beef.mean(axis=1)
```

In [676]:
```
# drop columns containing monthly values
df_beef = df_beef[['avg_beef']]
```

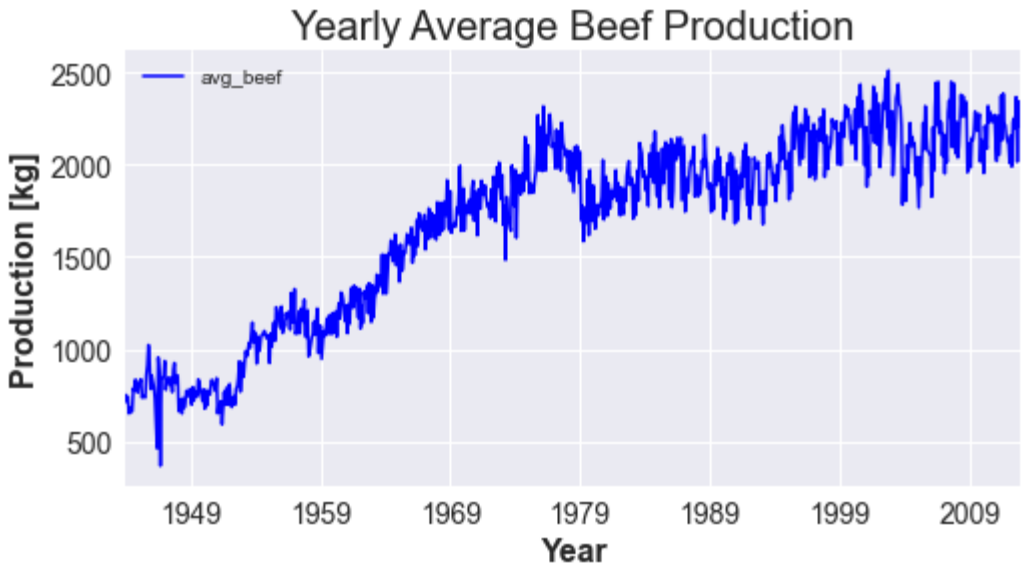
In [677]:
```
# visualize the first 5 columns
df_beef.head()
```

Out[677]:

month_year	avg_beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0

In [678]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn')
# line plot - the yearly average air temperature in Barcelona
df_beef.plot(color='blue', linewidth=1.5, figsize=(8,4))
# modify ticks size
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.legend('')
# title and labels
plt.title('Yearly Average Beef Production', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Production [kg]', fontsize=16)
```

Out[678]:
```
Text(0, 0.5, 'Production [kg]')
```



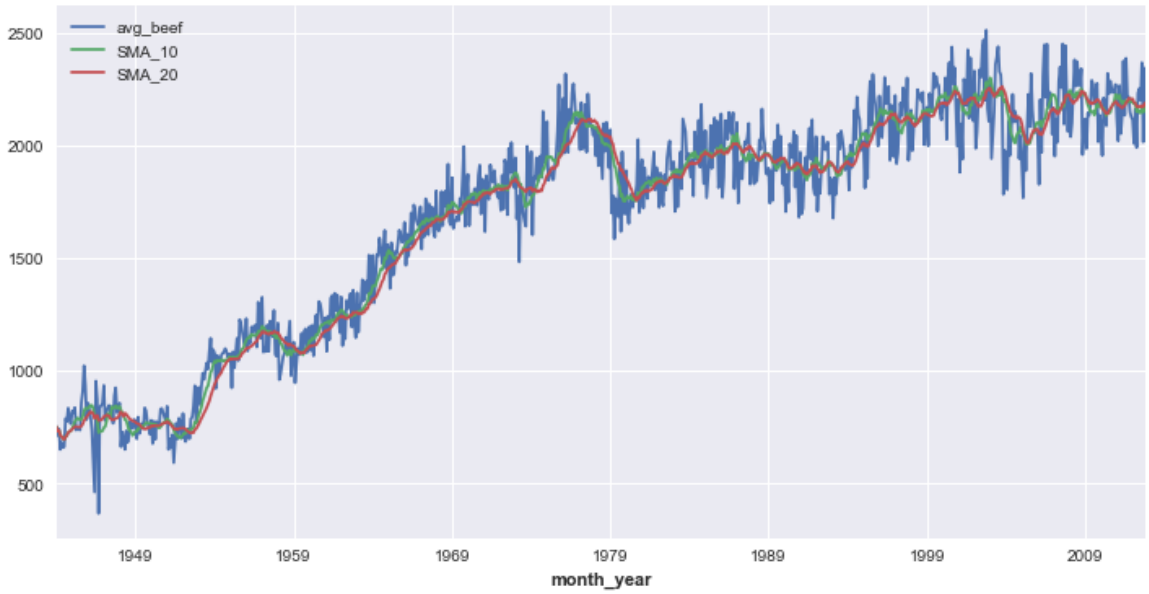
In [679]:
```
# ambient air temperature
# the simple moving average over a period of 10 years
df_beef['SMA_10'] = df_beef.avg_beef.rolling(10, min_periods=1).mean()

# the simple moving average over a period of 20 year
df_beef['SMA_20'] = df_beef.avg_beef.rolling(20, min_periods=1).mean()
```

In [680]:
```
df_beef['SMA_20']
```

Out[680]:
```
month_year
1944-01-01     751.000
1944-02-01     732.000
1944-03-01     735.000
1944-04-01     713.750
1944-05-01     707.200
                 ...
2012-07-01    2171.215
2012-08-01    2176.045
2012-09-01    2170.700
2012-10-01    2186.865
2012-11-01    2183.885
Name: SMA_20, Length: 827, dtype: float64
```

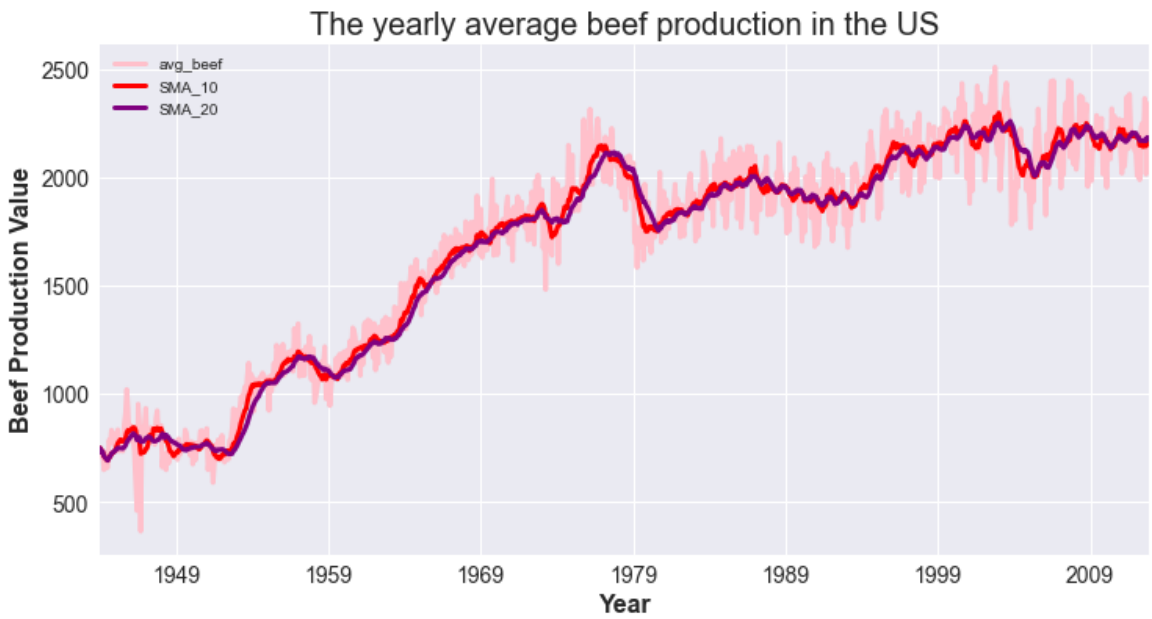
In [681]:
```
df_beef.plot(figsize=(12,6))
```

Out[681]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1410aad12e0>
```

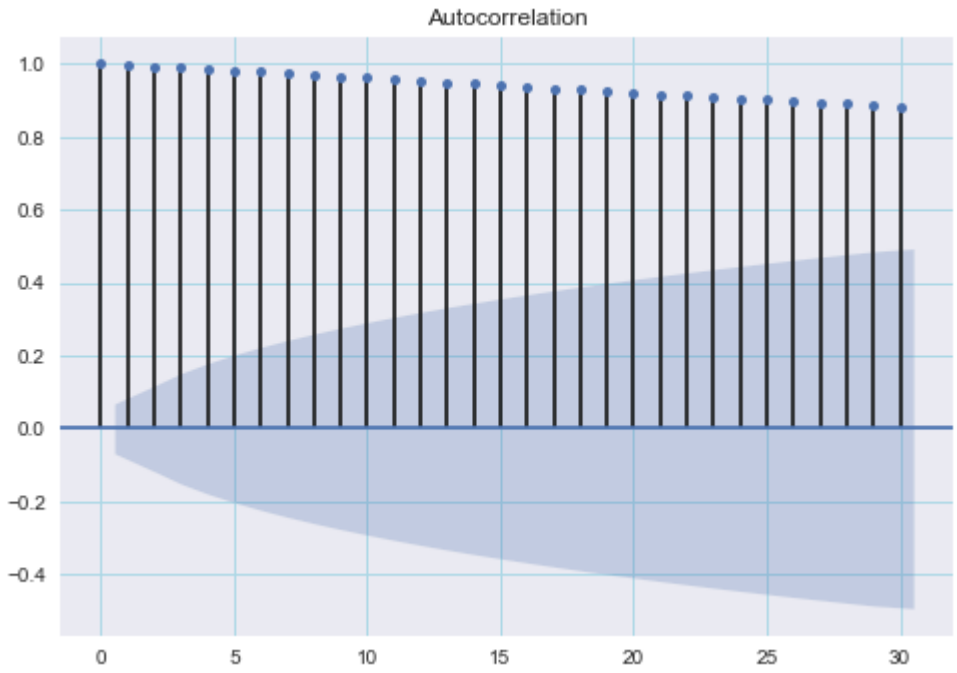


In [682]:
```
# colors for the line plot
colors = ['pink', 'red', 'purple']

# line plot - the yearly average air temperature in Barcelona
df_beef.plot(color=colors, linewidth=3, figsize=(12,6))

# modify ticks size
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.legend(labels =['Beef Production', '10-years SMA', '20-years SMA
'], fontsize=14)

# title and labels
plt.title('The yearly average beef production in the US', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Beef Production Value', fontsize=16)
```

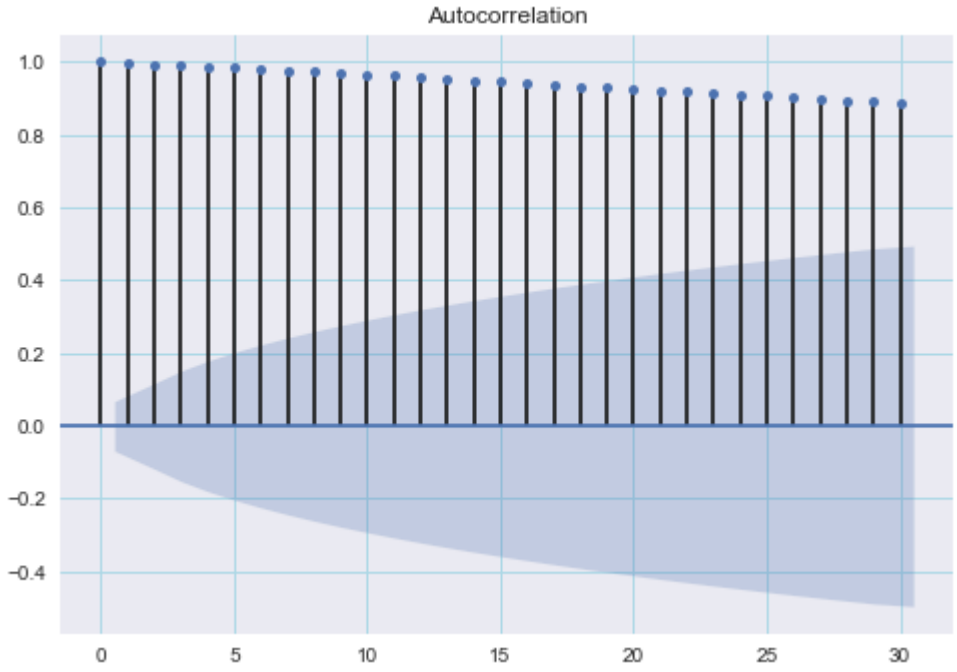
Out[682]:
```
Text(0, 0.5, 'Beef Production Value')
```



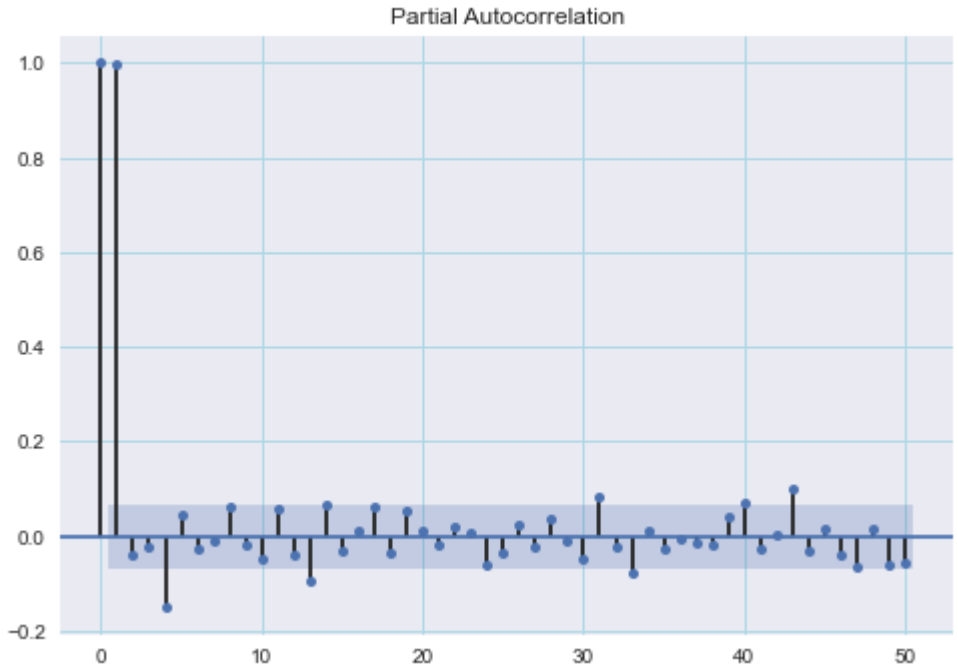
In [683]:
```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df_beef['SMA_10']);plt.grid(color = 'lightblue')
pyplot.show()
```



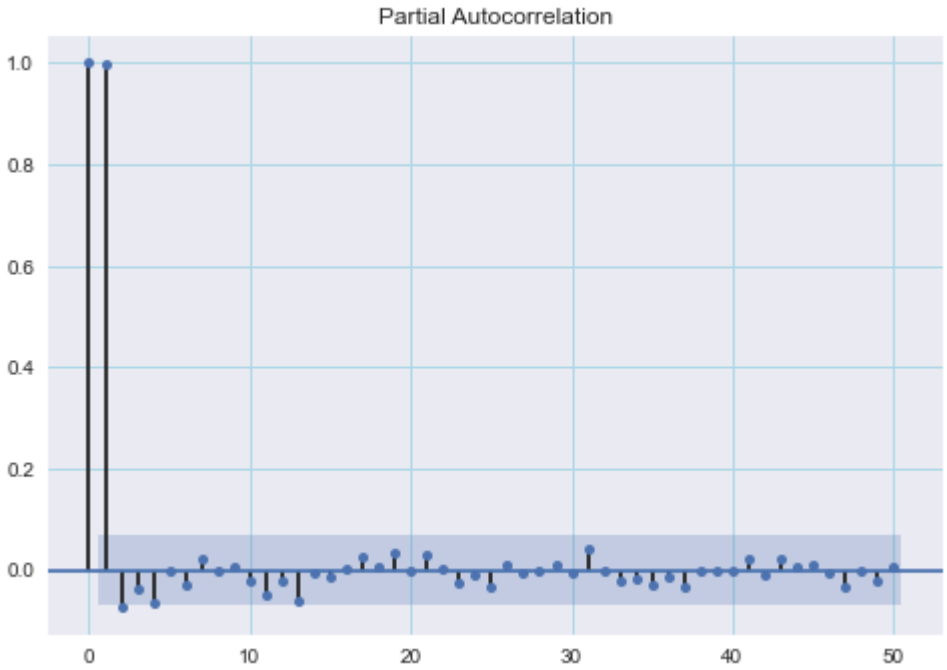
In [684]:
```
plot_acf(df_beef['SMA_20']);plt.grid(color = 'lightblue')
pyplot.show()
```



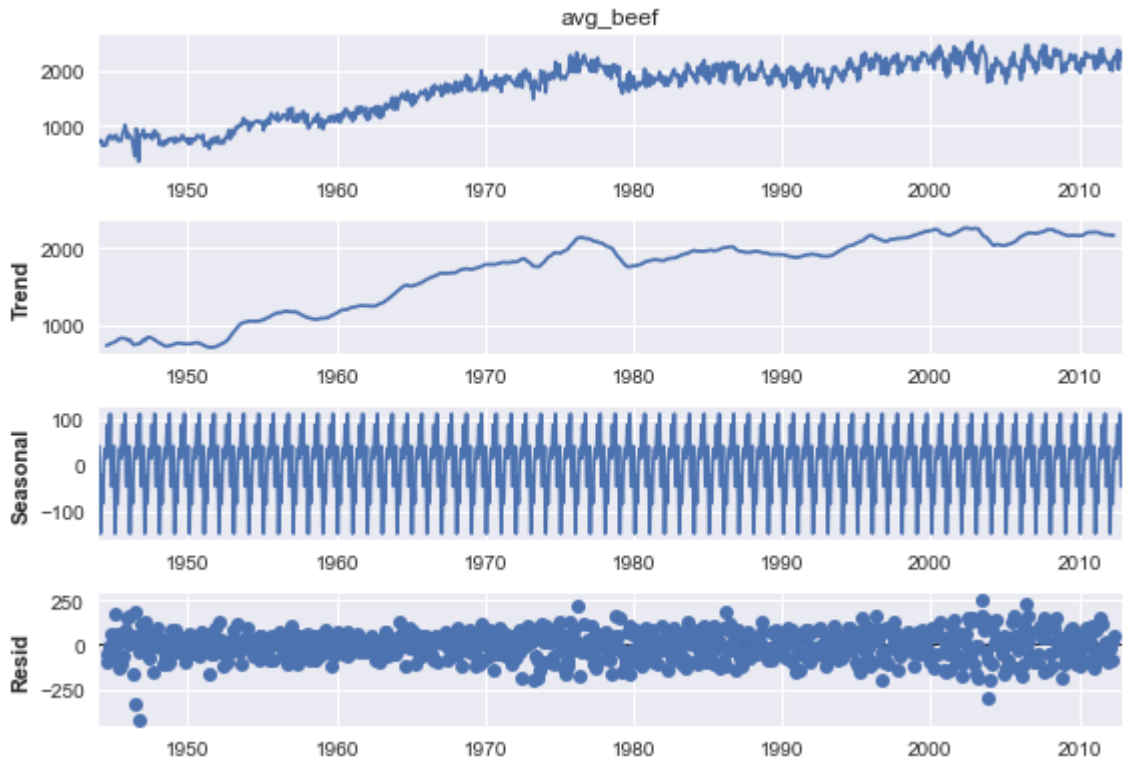
In [685]:
```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(df_beef['SMA_10'], lags=50);plt.grid(color = 'lightblue')
pyplot.show()
```



In [686]:
```
plot_pacf(df_beef['SMA_20'], lags=50);plt.grid(color = 'lightblue')
pyplot.show()
```



In [687]:
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df_beef['avg_beef'], freq=12)
decomposition.plot()
plt.show()
```



We can apply statistical tests and Augmented Dickey-Fuller test is the widely used one. The null hypothesis of the test is time series has a unit root, meaning that it is non-stationary. We interpret the test result using the p-value of the test. If the p-value is lower than the threshold value (5% or 1%), we reject the null hypothesis and time series is stationary. If the p-value is higher than the threshold, we fail to reject the null hypothesis and time series is non-stationary.

In [688]:
```
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(df_beef['SMA_10'])
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#
Lags Used','Number of Observations Used'])
for key, value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Test Statistic                  -1.470261
p-value                          0.548168
#Lags Used                      21.000000
Number of Observations Used    805.000000
Critical Value (1%)             -3.438499
Critical Value (5%)             -2.865137
Critical Value (10%)            -2.568685
dtype: float64
```

In [689]:
```
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(df_beef['SMA_20'])
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#
Lags Used','Number of Observations Used'])
for key, value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Test Statistic                  -1.748185
p-value                          0.406473
#Lags Used                      21.000000
Number of Observations Used    805.000000
Critical Value (1%)             -3.438499
Critical Value (5%)             -2.865137
Critical Value (10%)            -2.568685
dtype: float64
```

In [690]:
```
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

In [691]:
```
import numpy as np
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.api import qqplot
import statsmodels.api as sm
from pandas import Series, DataFrame, Panel
```

In [692]:
```
import pandasql
from pandasql import sqldf, load_meat, load_births
pysqldf = lambda q: sqldf(q, globals())
meat = load_meat()
meat.head(10)
```

Out[692]:

	date	beef	veal	pork	lamb_and_mutton	broilers	other_chicken	turkey
0	1944-01-01	751.0	85.0	1280.0	89.0	NaN	NaN	NaN
1	1944-02-01	713.0	77.0	1169.0	72.0	NaN	NaN	NaN
2	1944-03-01	741.0	90.0	1128.0	75.0	NaN	NaN	NaN
3	1944-04-01	650.0	89.0	978.0	66.0	NaN	NaN	NaN
4	1944-05-01	681.0	106.0	1029.0	78.0	NaN	NaN	NaN
5	1944-06-01	658.0	125.0	962.0	79.0	NaN	NaN	NaN
6	1944-07-01	662.0	142.0	796.0	82.0	NaN	NaN	NaN
7	1944-08-01	787.0	175.0	748.0	87.0	NaN	NaN	NaN
8	1944-09-01	774.0	182.0	678.0	91.0	NaN	NaN	NaN
9	1944-10-01	834.0	215.0	777.0	100.0	NaN	NaN	NaN

In [772]:
```
df_beef = meat[["date","beef"]]
df_beef.head(10)
```

Out[772]:

	date	beef
0	1944-01-01	751.0
1	1944-02-01	713.0
2	1944-03-01	741.0
3	1944-04-01	650.0
4	1944-05-01	681.0
5	1944-06-01	658.0
6	1944-07-01	662.0
7	1944-08-01	787.0
8	1944-09-01	774.0
9	1944-10-01	834.0

In [773]:
```
births.head(5)
```

Out[773]:

	date	births
0	1975-01-01	265775
1	1975-02-01	241045
2	1975-03-01	268849
3	1975-04-01	247455
4	1975-05-01	254545

In [774]:
```
beef_ts = df_beef.set_index(['date'])
beef_ts.index
```

Out[774]:
```
DatetimeIndex(['1944-01-01', '1944-02-01', '1944-03-01', '1944-04-01
',
               '1944-05-01', '1944-06-01', '1944-07-01', '1944-08-01
',
               '1944-09-01', '1944-10-01',
               ...
               '2012-02-01', '2012-03-01', '2012-04-01', '2012-05-01
',
               '2012-06-01', '2012-07-01', '2012-08-01', '2012-09-01
',
               '2012-10-01', '2012-11-01'],
              dtype='datetime64[ns]', name='date', length=827, freq=N
one)
```

In [775]:
```
births_ts = births.set_index(['date'])
births_ts.index
```

Out[775]:
```
DatetimeIndex(['1975-01-01', '1975-02-01', '1975-03-01', '1975-04-01
',
               '1975-05-01', '1975-06-01', '1975-07-01', '1975-08-01
',
               '1975-09-01', '1975-10-01',
               ...
               '2012-03-01', '2012-04-01', '2012-05-01', '2012-06-01
',
               '2012-07-01', '2012-08-01', '2012-09-01', '2012-10-01
',
               '2012-11-01', '2012-12-01'],
              dtype='datetime64[ns]', name='date', length=408, freq=N
one)
```

In [776]:
```
births_ts.head(5)
```

Out[776]:

date	births
1975-01-01	265775
1975-02-01	241045
1975-03-01	268849
1975-04-01	247455
1975-05-01	254545

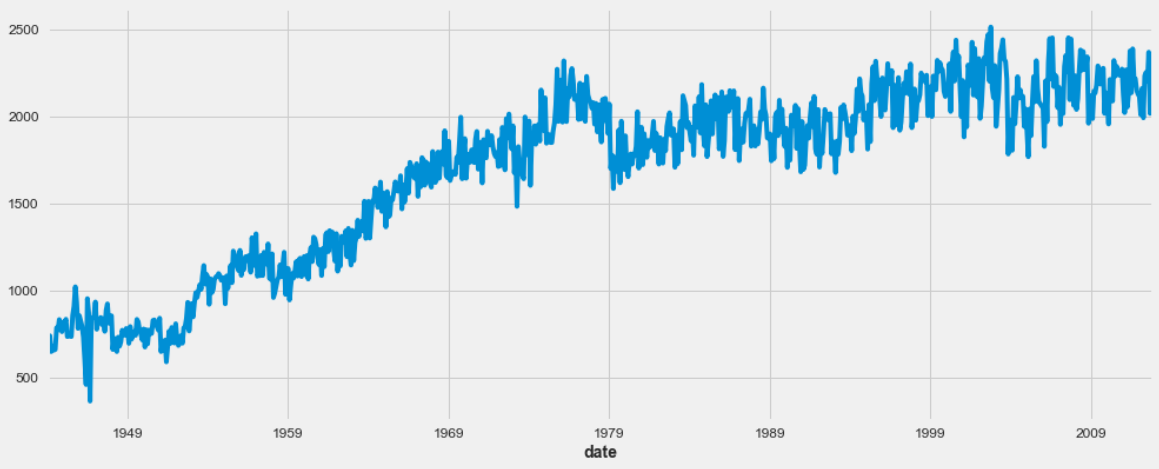
In [777]:
```
y = beef_ts['beef'].resample('MS').mean()
y['2000':]
```

Out[777]:
```
date
2000-01-01    2178.0
2000-02-01    2175.0
2000-03-01    2300.0
2000-04-01    2027.0
2000-05-01    2303.0
                ...
2012-07-01    2200.8
2012-08-01    2367.5
2012-09-01    2016.0
2012-10-01    2343.7
2012-11-01    2206.6
Freq: MS, Name: beef, Length: 155, dtype: float64
```

In [778]:
```
w = births_ts['births'].resample('MS').sum()
w['2000':]
```

Out[778]:
```
date
2000-01-01    319340
2000-02-01    298711
2000-03-01    329436
2000-04-01    319758
2000-05-01    330519
                ...
2012-08-01    359554
2012-09-01    361922
2012-10-01    347625
2012-11-01    320195
2012-12-01    340995
Freq: MS, Name: births, Length: 156, dtype: int64
```

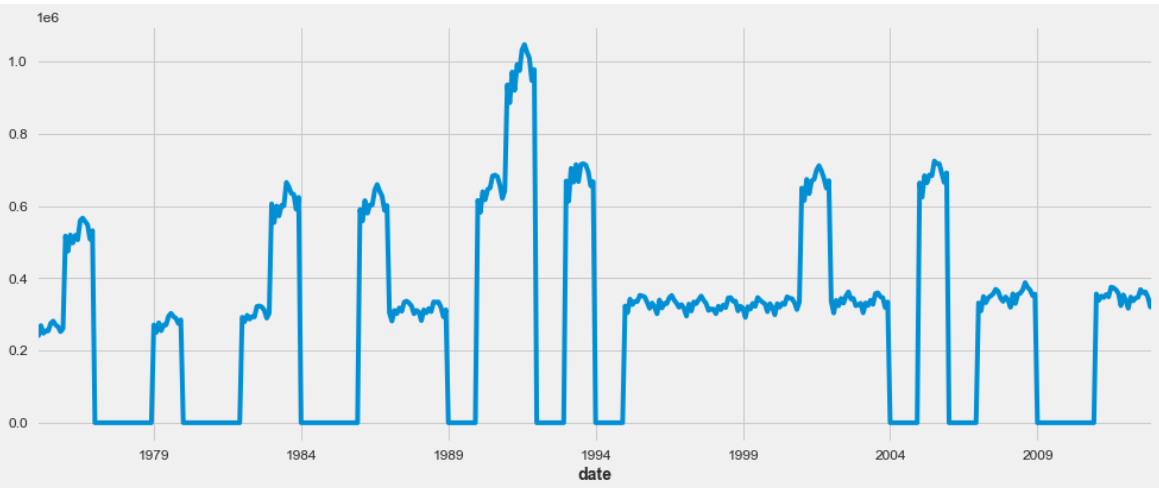
In [779]:
```
y.plot(figsize=(15, 6))
plt.show()
```



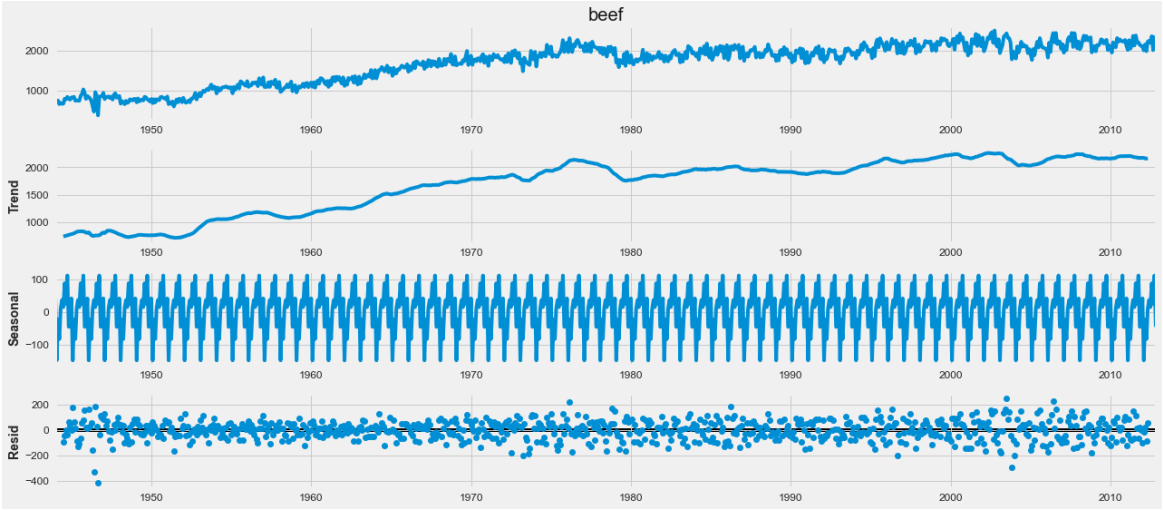
In [780]:
```
joined['births']
```

Out[780]:
```
0      265775
1      241045
2      268849
3      247455
4      254545
        ...
95     303957
96     317969
97     325516
98     323960
99     335222
Name: births, Length: 100, dtype: int64
```

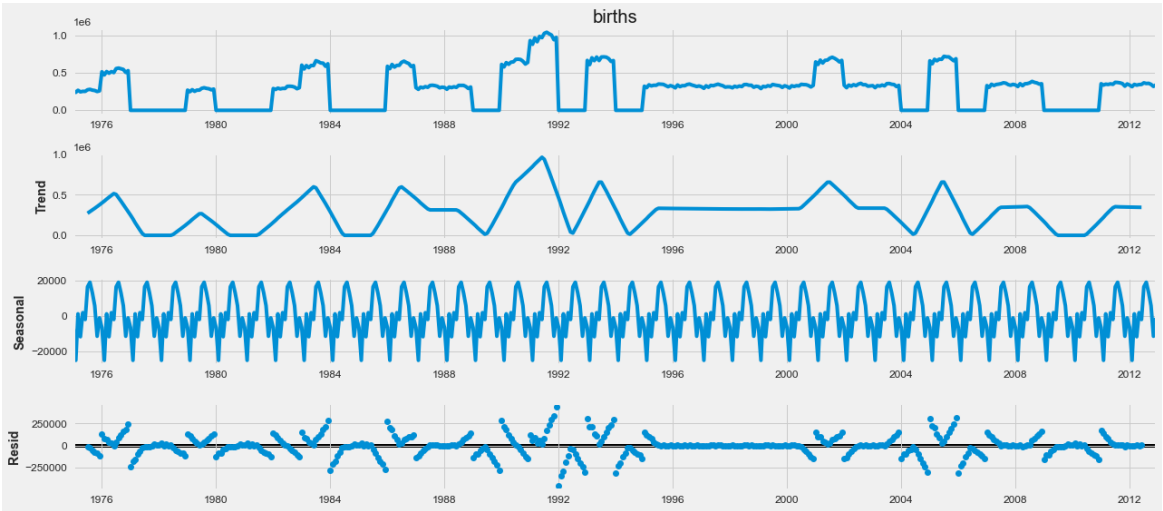
In [702]:
```
w.plot(figsize=(15, 6))
plt.show()
```



In [703]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



In [704]:
```
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(w, model='additive')
fig = decomposition.plot()
plt.show()
```



In [705]:
```
beef_ts['avg_beef'] = beef_ts.mean(axis=1)
beef_ts = beef_ts[['avg_beef']]
beef_ts.head(5)
```

Out[705]:

date	avg_beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0

In [706]:
```
beef_ts.index.name = 'month_year'
beef_ts.head(10)
```

Out[706]:

month_year	avg_beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0
1944-06-01	658.0
1944-07-01	662.0
1944-08-01	787.0
1944-09-01	774.0
1944-10-01	834.0

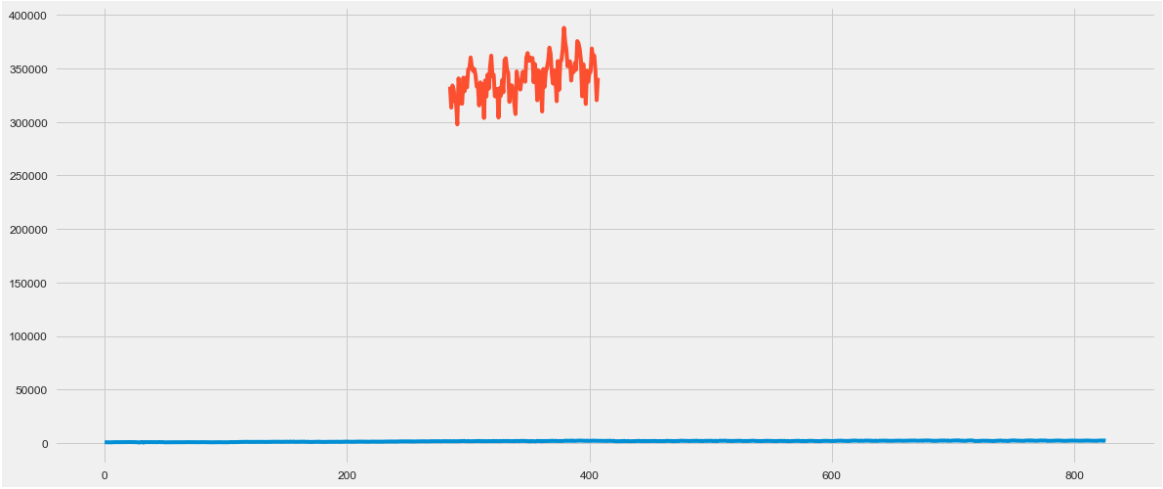
In [707]:
```
X = beef_ts.values
train_size = int(len(X) * 0.7)
train, test = X[0:train_size], X[train_size:len(X)]
print('Observations: %d' % (len(X)))
print('Training Observations: %d' % (len(train)))
print('Testing Observations: %d' % (len(test)))
```

```
Observations: 827
Training Observations: 578
Testing Observations: 249
```

In [708]:
```
Z = births_ts.values
train_size = int(len(Z) * 0.7)
train, test = Z[0:train_size], Z[train_size:len(X)]
print('Observations: %d' % (len(Z)))
print('Training Observations: %d' % (len(train)))
print('Testing Observations: %d' % (len(test)))
```

```
Observations: 408
Training Observations: 285
Testing Observations: 123
```

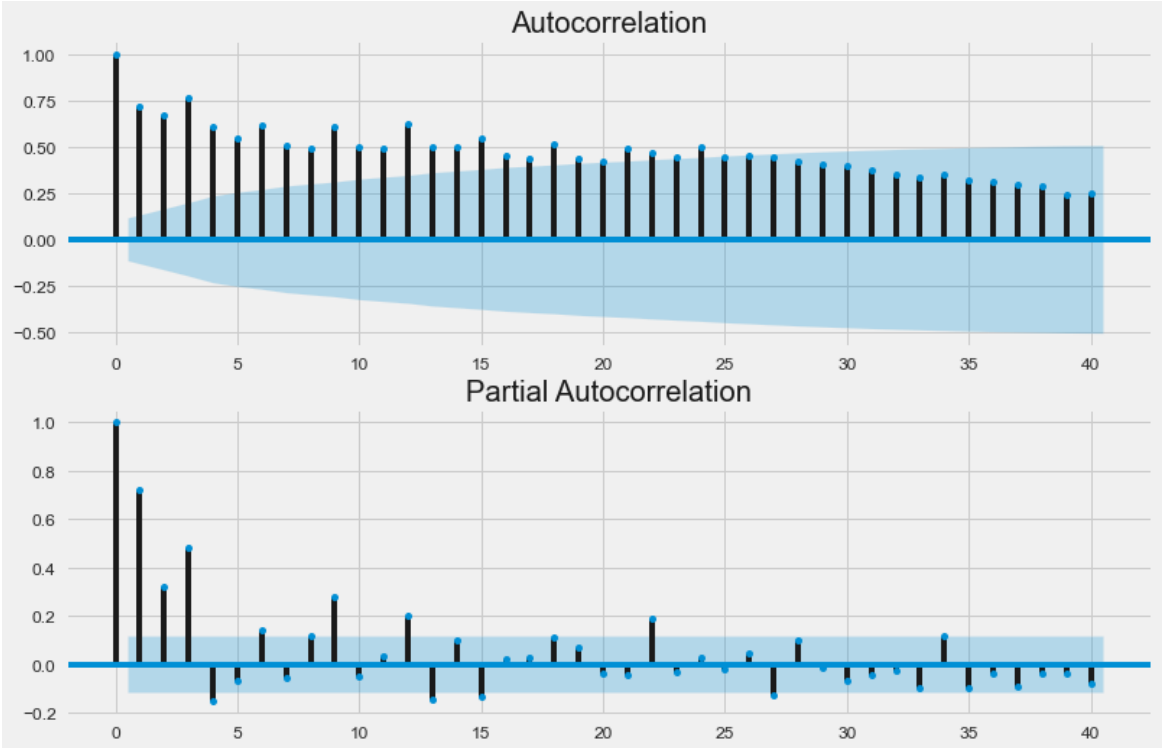
In [709]:
```
from matplotlib import pyplot
pyplot.plot(X)
pyplot.plot([None for i in train] + [x for x in test])
pyplot.show()
```



In [710]:
```
sm.stats.durbin_watson(train)
```

Out[710]:
```
array([0.00470735])
```

In [711]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(train, lags=40, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(train, lags=40, ax=ax2)
```

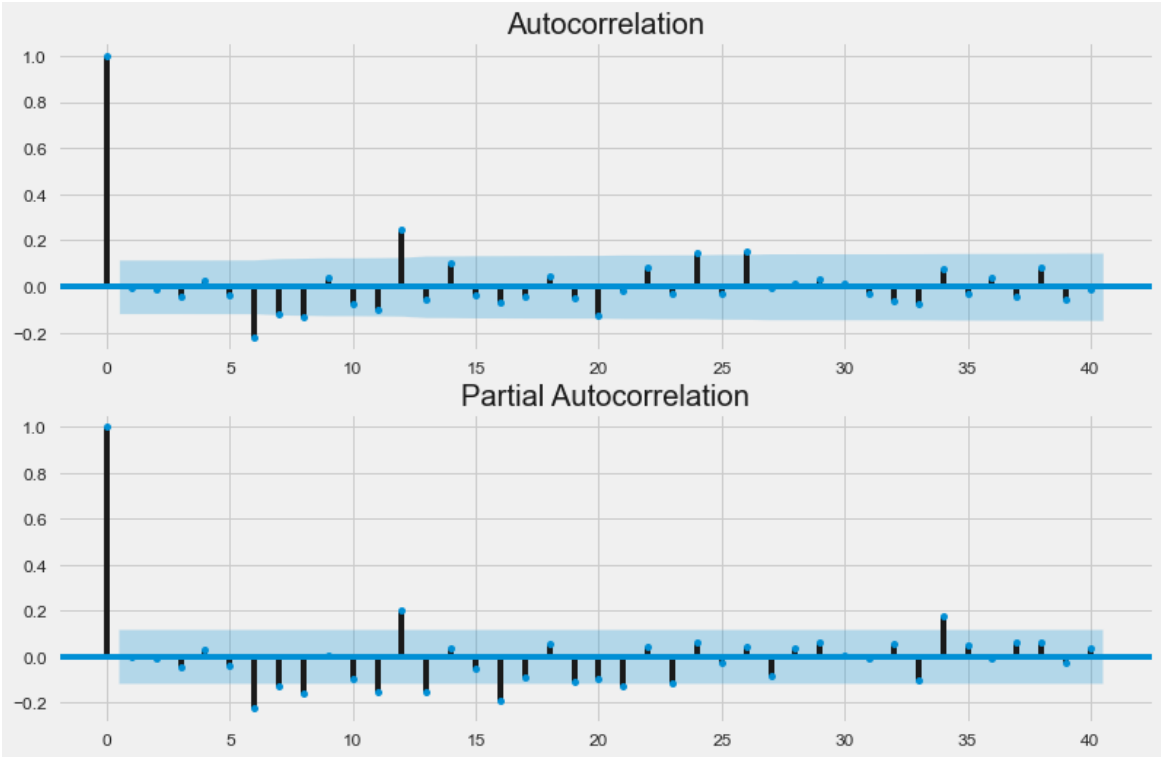


In [712]:
```
arma_mod_1 = sm.tsa.ARIMA(train, order=(0,1,0))
arma_res_1 = arma_mod_1.fit(trend='c', disp=1)
print(arma_res_1.summary())
```

```
                             ARIMA Model Results
==============================================================================
=========
Dep. Variable:                      D.y   No. Observations:
284
Model:                 ARIMA(0, 1, 0)   Log Likelihood
-3236.057
Method:                            css   S.D. of innovations
21496.112
Date:                Mon, 30 Nov 2020   AIC
6476.113
Time:                        06:02:15   BIC
6483.411
Sample:                             1   HQIC
6479.039

==============================================================================
=========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const        273.2711   1275.560      0.214      0.830   -2226.780
2773.322
==============================================================================
=========
```

In [718]:
```
#resid_opt_1 = arma_res_1.resid
#fig = plt.figure(figsize=(12,8))
#ax1 = fig.add_subplot(211)
#fig = sm.graphics.tsa.plot_acf(resid_opt_1, lags=40, ax=ax1)
#ax2 = fig.add_subplot(212)
#fig = sm.graphics.tsa.plot_pacf(resid_opt_1, lags=40, ax=ax2)
```

In [714]:
```
arma_mod_2 = sm.tsa.ARIMA(train, order=(6,1,0))
arma_res_2 = arma_mod_2.fit(trend='nc', disp=1)
resid_opt_2 = arma_res_2.resid
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(resid_opt_2, lags=40, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(resid_opt_2, lags=40, ax=ax2)
```



In [715]:
```
print(arma_res_1.summary())
```

```
                             ARIMA Model Results
==============================================================================
=========
Dep. Variable:                      D.y   No. Observations:
284
Model:                 ARIMA(0, 1, 0)   Log Likelihood
-3236.057
Method:                            css   S.D. of innovations
21496.112
Date:                Mon, 30 Nov 2020   AIC
6476.113
Time:                        06:02:46   BIC
6483.411
Sample:                             1   HQIC
6479.039

==============================================================================
=========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const        273.2711   1275.560      0.214      0.830   -2226.780
2773.322
==============================================================================
=========
```

In [716]:
```
print(arma_res_2.summary())
```

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	284
Model:	ARIMA(6, 1, 0)	Log Likelihood	-3153.950
Method:	css-mle	S.D. of innovations	16069.507
Date:	Mon, 30 Nov 2020	AIC	6321.900
Time:	06:02:49	BIC	6347.443
Sample:	1	HQIC	6332.140

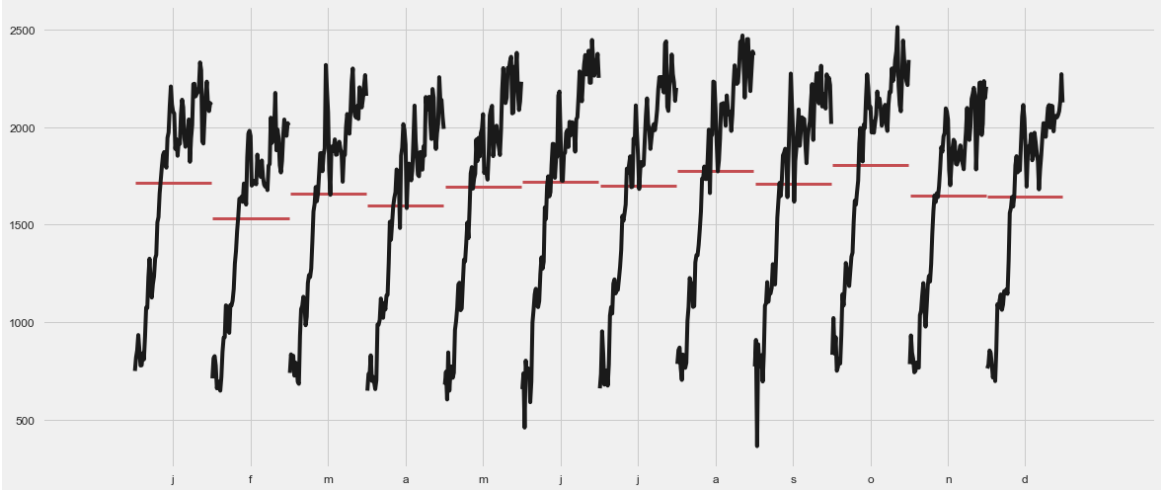
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.y	-0.5971	0.059	-10.060	0.000	-0.713	-0.481
ar.L2.D.y	-0.4200	0.069	-6.130	0.000	-0.554	-0.286
ar.L3.D.y	0.0515	0.073	0.703	0.482	-0.092	0.195
ar.L4.D.y	-0.0430	0.073	-0.588	0.557	-0.186	0.100
ar.L5.D.y	-0.1480	0.069	-2.160	0.031	-0.282	-0.014
ar.L6.D.y	0.0304	0.060	0.510	0.610	-0.086	0.147

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.5440	-1.0153j	1.1519	-0.3283
AR.2	-0.5440	+1.0153j	1.1519	0.3283
AR.3	-1.4916	-0.0000j	1.4916	-0.5000
AR.4	1.1187	-1.3922j	1.7860	-0.1423
AR.5	1.1187	+1.3922j	1.7860	0.1423
AR.6	5.2082	-0.0000j	5.2082	-0.0000

In [717]:
```
fig = plt.figure(figsize=(12,8))
#ax = fig.add_subplot(111)
fig = sm.graphics.tsa.month_plot(beef_ts)
```

```
<Figure size 864x576 with 0 Axes>
```



In [719]:
```
# births per year
q = """
    SELECT
        strftime("%Y", date)
        , SUM(births)
    FROM births
    GROUP BY 1
    ORDER BY 1;
            """

sqldf(q, locals())
```

Out[719]:

	strftime("%Y", date)	SUM(births)
0	1975	3136965
1	1976	6304156
2	1979	3333279
3	1982	3612258
4	1983	7333238
5	1986	7308074
6	1987	3760561
7	1988	3756547
8	1990	7718904
9	1991	11714356
10	1993	8194907
11	1995	4000240
12	1996	3952767
13	1997	3899589
14	1998	3891494
15	1999	3880894
16	2000	3941553
17	2001	8018231
18	2002	4025933
19	2003	4021726
20	2005	8210950
21	2007	4138349
22	2008	4265555
23	2011	4247694
24	2012	4130665

In [720]:
```
def pysqldf(q):
    return sqldf(q, globals())

q = """
        SELECT
            *
        FROM
            births
        LIMIT 10;"""

pysqldf(q)
```

Out[720]:

	date	births
0	1975-01-01 00:00:00.000000	265775
1	1975-02-01 00:00:00.000000	241045
2	1975-03-01 00:00:00.000000	268849
3	1975-04-01 00:00:00.000000	247455
4	1975-05-01 00:00:00.000000	254545
5	1975-06-01 00:00:00.000000	254096
6	1975-07-01 00:00:00.000000	275163
7	1975-08-01 00:00:00.000000	281300
8	1975-09-01 00:00:00.000000	270738
9	1975-10-01 00:00:00.000000	265494

In [721]:
```
 # joining meats + births on date
q = """
        SELECT
            m.date
            , b.births
            , m.beef
        FROM
            meat m
        INNER JOIN
            births b
                on m.date = b.date
        ORDER BY
            m.date
        LIMIT 100;
        """

joined = pysqldf(q)
joined.head()
```

Out[721]:

	date	births	beef
0	1975-01-01 00:00:00.000000	265775	2106.0
1	1975-02-01 00:00:00.000000	241045	1845.0
2	1975-03-01 00:00:00.000000	268849	1891.0
3	1975-04-01 00:00:00.000000	247455	1895.0
4	1975-05-01 00:00:00.000000	254545	1849.0

In [722]:
```
births = load_births()
births.head(5)
```

Out[722]:

	date	births
0	1975-01-01	265775
1	1975-02-01	241045
2	1975-03-01	268849
3	1975-04-01	247455
4	1975-05-01	254545

In [725]:
```
births_ts['births'] = births_ts.mean(axis=1)
births_ts_ts = births_ts[['births']]
births_ts_ts.head(5)
```

Out[725]:

date	births
1975-01-01	265775.0
1975-02-01	241045.0
1975-03-01	268849.0
1975-04-01	247455.0
1975-05-01	254545.0

In [726]:
```
beef_ts.head(10)
```

Out[726]:

month_year	avg_beef
1944-01-01	751.0
1944-02-01	713.0
1944-03-01	741.0
1944-04-01	650.0
1944-05-01	681.0
1944-06-01	658.0
1944-07-01	662.0
1944-08-01	787.0
1944-09-01	774.0
1944-10-01	834.0

In [727]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1754      0.043     -4.091      0.000      -0.259      -0.091
ma.L1         -0.6502      0.033    -19.777      0.000      -0.715      -0.586
ar.S.L12      -0.2392      0.029     -8.239      0.000      -0.296      -0.182
sigma2      9325.1986    359.347     25.950      0.000    8620.891    1e+04
==============================================================================
```

In [728]:
```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in
                list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [729]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
seasonal_order=param_seasonal,
enforce_stationarity=False,
enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasona
l, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:14689.166432367274
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:13465.176599348248
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:10051.608773940949
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:9906.38668856096
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:10053.833767516373
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:10039.669508190076
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:9918.1844737085
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:9908.25538359395
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:13610.489472477842
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:12454.94298483342
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:9915.589019250343
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:9774.613907352787
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:9936.417049515607
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:9910.143858178644
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:9798.448947100711
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:9749.542006734038
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:10559.150662430065
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:10051.860117739889
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:10147.65175091831
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:9804.029840722837
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:10006.51057716779
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:9981.505831994633
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:10016.77781447033
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:9739.649056896113
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:10178.590335733701
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:9731.8103858793
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:9774.113647109307
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:9402.371558747496
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:9641.994182967152
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:9543.329177687663
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:9625.529947785973
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:9363.395799083004
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:10572.212694394122
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:10063.888511754578
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:9887.724062670171
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:9744.54432867298
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:9888.06560685212
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:9880.96056105255
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:9748.032381153243
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:9678.112372996025
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:10190.803076768945
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:9745.244995784808
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:9754.138739363718
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:9413.434587340738
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:9644.006818516486
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:9554.586195600772
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:9606.29322846887
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:9374.902205761002
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:10241.239691320767
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:9813.589416938194
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:9945.317024843087
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:9583.386846076639
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:9765.759341807641
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:9762.665163880922
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:9792.177377399905
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:9532.056646787012
```

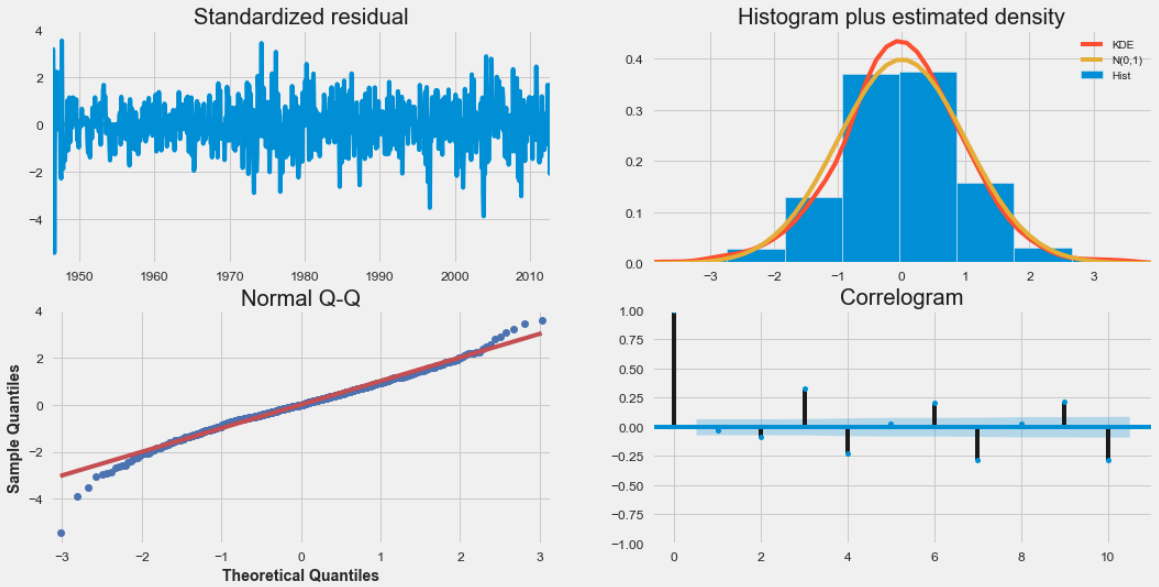
```
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:10166.449943373576
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:9727.347376650938
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:9770.118057797648
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:9393.904613113136
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:9626.71424740387
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:9534.893296215272
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:9603.123626401208
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:9359.009975398578
```

In [730]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1754      0.043     -4.091      0.000      -0.259      -0.091
ma.L1         -0.6502      0.033    -19.777      0.000      -0.715      -0.586
ar.S.L12      -0.2392      0.029     -8.239      0.000      -0.296      -0.182
sigma2      9325.1986    359.347     25.950      0.000    8620.891    1e+04
==============================================================================
```

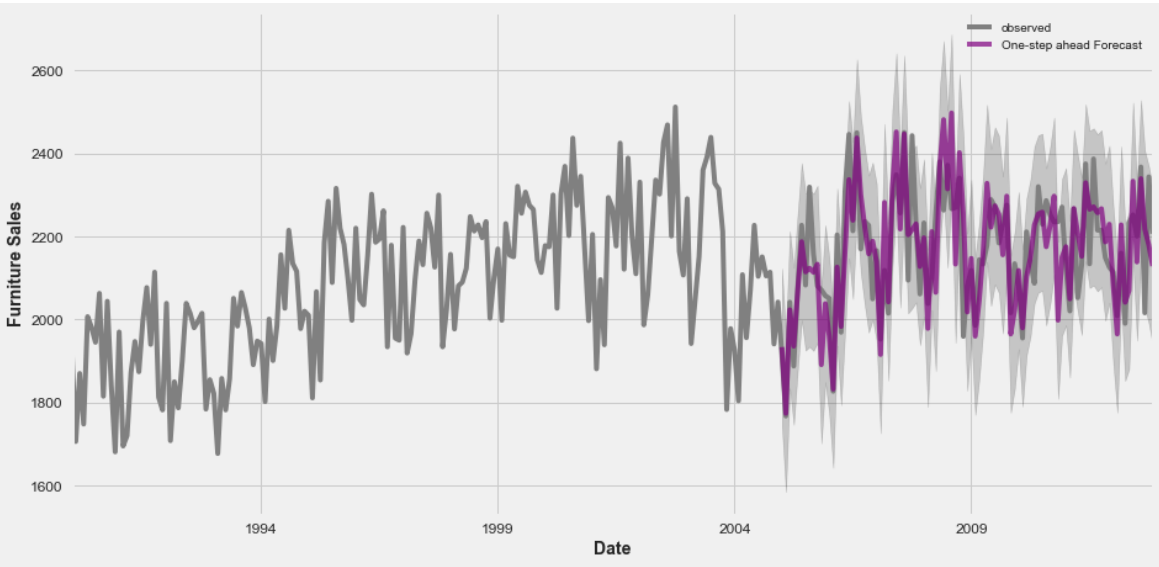
In [ ]:
```
```

In [731]:
```
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



In [732]:
```
pred = results.get_prediction(start=pd.to_datetime('1944-01-01'), dyna
mic=False)
```

In [733]:
```
pred = results.get_prediction(start=pd.to_datetime('2005-01-01'), dyna
mic=False)
pred_ci = pred.conf_int()
ax = y['1990':].plot(label='observed', color='gray')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha
=.7, figsize=(14, 7), color='purple')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Furniture Sales')
plt.legend()
plt.show()
```



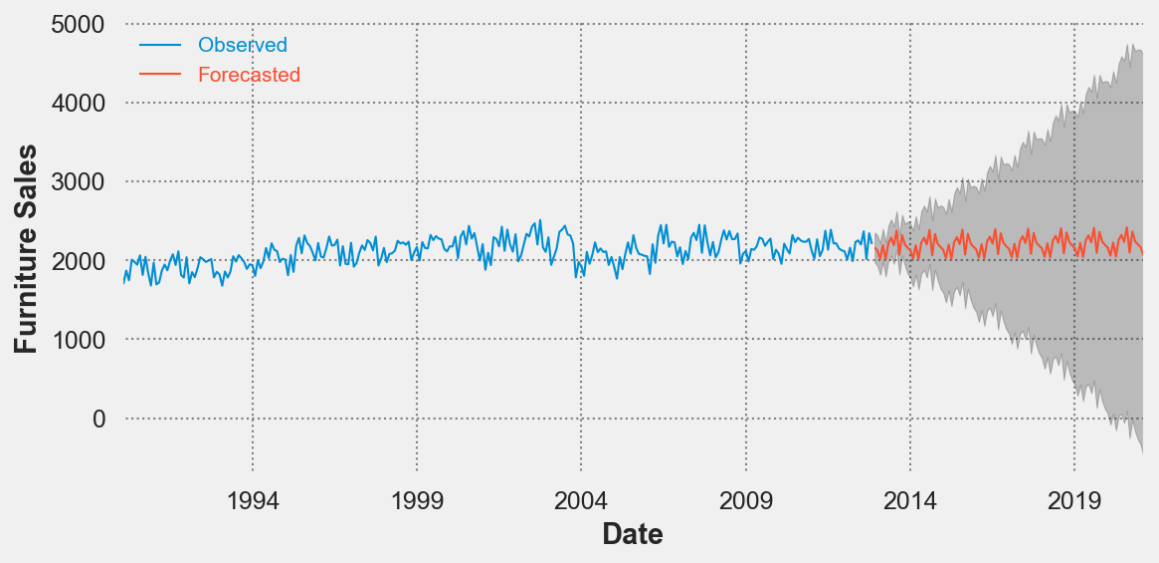
In [734]:
```
forecasted = pred.predicted_mean
truth = y['2005-01-01':]
mse = ((forecasted - truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(ms
e, 2)))
```

```
The Mean Squared Error of our forecasts is 9922.52
```

In [735]:
```
print('The Root Mean Squared Error of our forecasts is {}'.format(roun
d(np.sqrt(mse), 2)))
```

```
The Root Mean Squared Error of our forecasts is 99.61
```

In [744]:
```
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
fig, ax = plt.subplots(figsize=(7, 5), dpi = 150)
ax = y['1990':].plot(label='Observed', figsize=(8, 4), linewidth=1)
pred_uc.predicted_mean.plot(ax=ax, label='Forecasted', linewidth=1)
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('Furniture Sales')
plt.grid(True, color = 'gray', linestyle = ':')

leg = ax.legend(loc = 'upper left')
for line, text in zip(leg.get_lines(), leg.get_texts()):
    text.set_color(line.get_color())
plt.tight_layout()
plt.savefig('0000_68', figsize = (8,4), dpi = 300, bbox_inches = 'tigh
t')
plt.show()
```



In [745]:
```
print('The Root Mean Squared Error of our forecasts is {}'.format(roun
d(np.sqrt(mse), 2)))
```

```
The Root Mean Squared Error of our forecasts is 99.61
```

In [ ]:

In [ ]:

In [ ]:

In [96]:
```
import sklearn
from sklearn import *
```

In [755]:
```
milk= pd.read_csv("D:/Documents/Data/milk_monthly.txt")
```

In [756]:
```
milk.head(5)
```

Out[756]:

	month	milk_prod_per_cow_kg
0	1962-01-01	265.05
1	1962-02-01	252.45
2	1962-03-01	288.00
3	1962-04-01	295.20
4	1962-05-01	327.15

In [757]:
```
milk
```

Out[757]:

	month	milk_prod_per_cow_kg
0	1962-01-01	265.05
1	1962-02-01	252.45
2	1962-03-01	288.00
3	1962-04-01	295.20
4	1962-05-01	327.15
...	...	...
163	1975-08-01	386.10
164	1975-09-01	367.65
165	1975-10-01	372.15
166	1975-11-01	358.65
167	1975-12-01	379.35

168 rows × 2 columns

In [758]:
```
milk.index = pd.Index(sm.tsa.datetools.dates_from_range('1962m1', '1975m12'))
#del milk["month"]
```

In [759]:
```
milk.head(10)
```

Out[759]:

	month	milk_prod_per_cow_kg
1962-01-31	1962-01-01	265.05
1962-02-28	1962-02-01	252.45
1962-03-31	1962-03-01	288.00
1962-04-30	1962-04-01	295.20
1962-05-31	1962-05-01	327.15
1962-06-30	1962-06-01	313.65
1962-07-31	1962-07-01	288.00
1962-08-31	1962-08-01	269.55
1962-09-30	1962-09-01	255.60
1962-10-31	1962-10-01	259.65

In [767]:
```
milk_lng = pd.melt(milk, id_vars=['month'])
milk_lng
```

Out[767]:

	month	variable	value
0	1962-01-01	milk_prod_per_cow_kg	265.05
1	1962-02-01	milk_prod_per_cow_kg	252.45
2	1962-03-01	milk_prod_per_cow_kg	288.00
3	1962-04-01	milk_prod_per_cow_kg	295.20
4	1962-05-01	milk_prod_per_cow_kg	327.15
...	...	...	...
163	1975-08-01	milk_prod_per_cow_kg	386.10
164	1975-09-01	milk_prod_per_cow_kg	367.65
165	1975-10-01	milk_prod_per_cow_kg	372.15
166	1975-11-01	milk_prod_per_cow_kg	358.65
167	1975-12-01	milk_prod_per_cow_kg	379.35

168 rows × 3 columns

In [771]:
```
ts.groupby(floor_decade).sum()

by_decade = ts.groupby(floor_decade).sum()

by_decade.index.name = 'year'

by_decade = by_decade.reset_index()
by_decade
```

Out[771]:

	year	beef	veal	pork	lamb_and_mutton
0	1940	55751.0	8566.0	55737.0	5071.0
1	1950	119161.0	12693.0	98450.0	6724.0
2	1960	177754.0	8577.0	116587.0	6873.0
3	1970	228947.0	5713.0	132539.0	4256.0
4	1980	230100.0	4278.0	150528.0	3394.0
5	1990	243579.0	2938.0	173519.0	2986.0
6	2000	260540.7	1685.3	208211.3	1964.7
7	2010	76391.5	371.9	66491.2	455.6

In [781]:
```
ggplot(aes(x='year', y='value'), data=by_decade) + geom_line() + \
ggtitle("Meat Production by Decade")
```

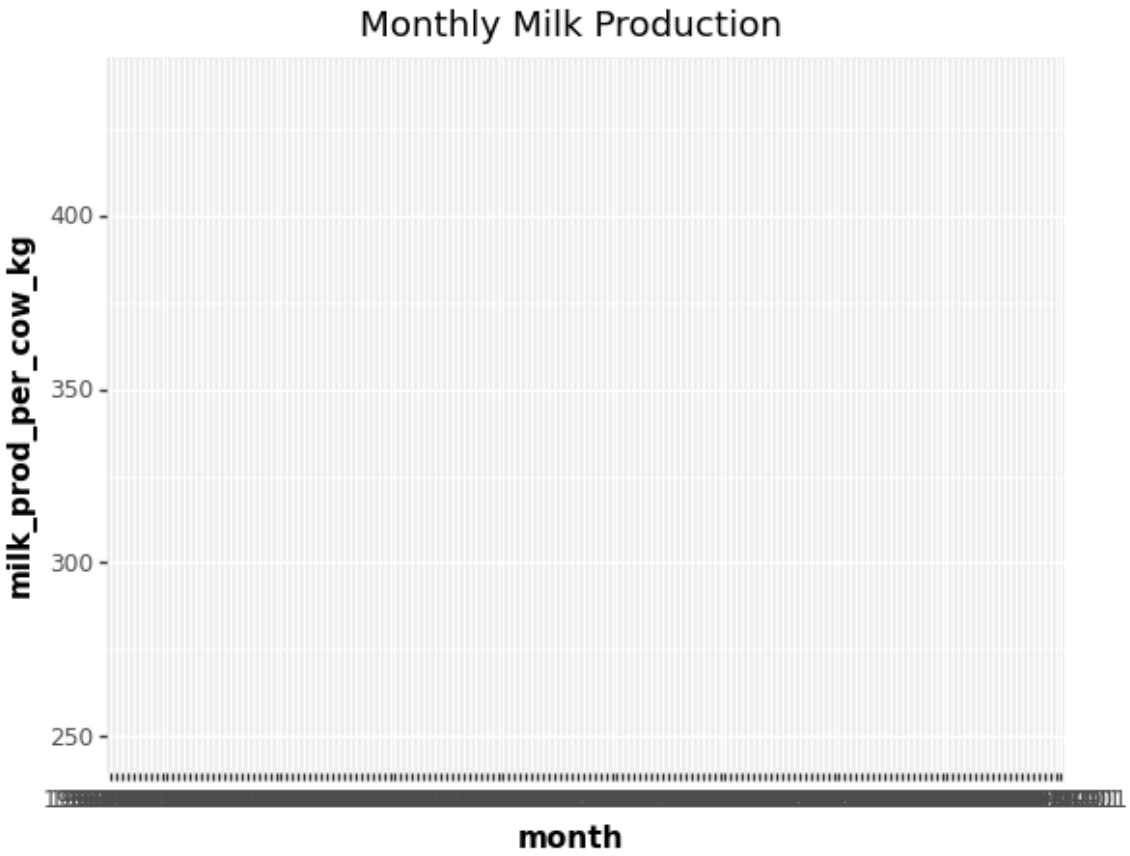
Out[781]: <ggplot: (86185821832)>

In [782]:
```
def floor_year(date_value):
    "Takes a date. Returns the year."
    return (date_value.month // 12) * 12

pd.to_datetime(2013-10-9)
```

Out[782]: Timestamp('1970-01-01 00:00:00.000001994')

In [783]:
```
%matplotlib inline
by_year = milk.groupby(floor_year).sum()
by_year.index.name = 'month'
by_year = by_year.reset_index()
```

In [784]:
```
meat_lng = pd.melt(milk, id_vars=['month'])
ggplot(aes(x='month', y='milk_prod_per_cow_kg'), data=milk) + geom_lin
e() + \
ggtitle("Monthly Milk Production")
```



Out[784]: <ggplot: (86186110392)>

In [785]:
```
ggplot(milk_mon, aes(x='month', y='milk_prod_per_cow_kg')) + geom_poin
t()
```

```
---------------------------------------------------------------------
------
NameError                                 Traceback (most recent call
last)
<ipython-input-785-2baee82daefd> in <module>
----> 1 ggplot(milk_mon, aes(x='month', y='milk_prod_per_cow_kg')) +
geom_point()

NameError: name 'milk_mon' is not defined
```

In [786]:
```
ggplot(milk_mon, aes(x='month', y='milk_prod_per_cow_kg'), color = 'red') + geom_point()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-786-b0bec418ebc9> in <module>
----> 1 ggplot(milk_mon, aes(x='month', y='milk_prod_per_cow_kg'), color = 'red') + geom_point()

NameError: name 'milk_mon' is not defined
```

In [787]:
```
import plotnine as p9
```

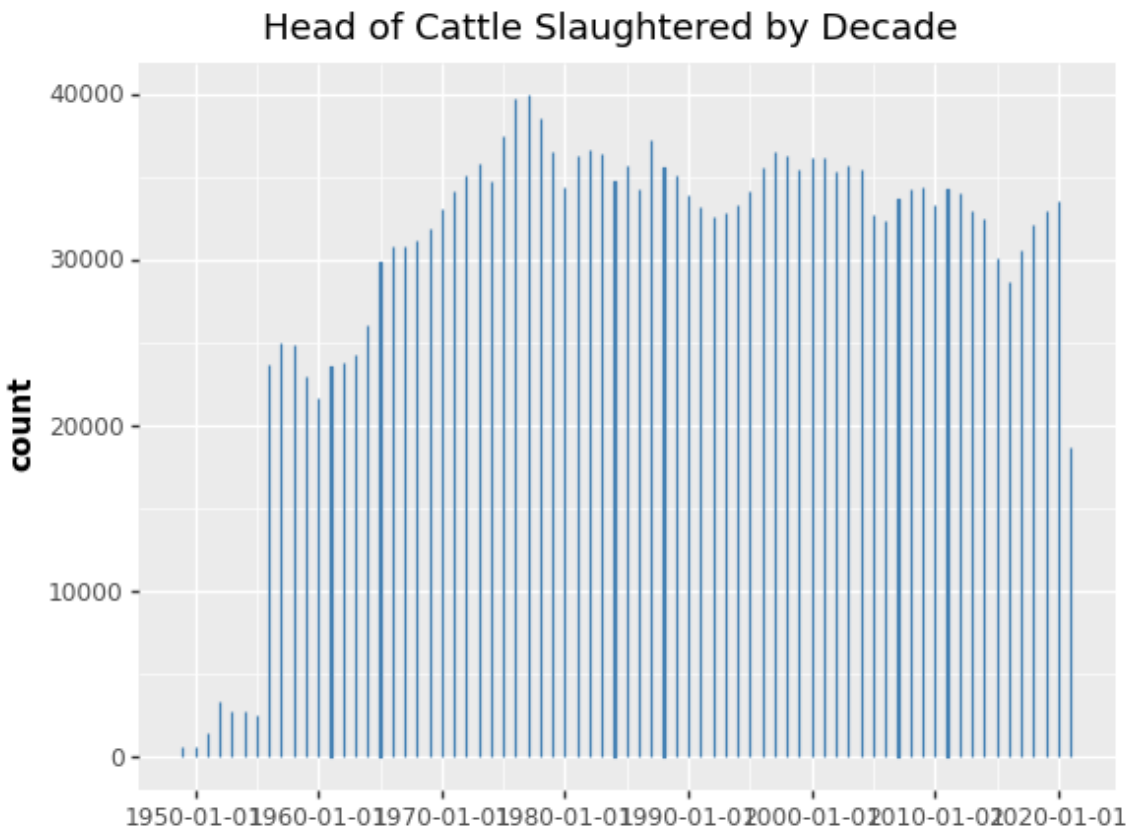
In [788]:
```
(p9.ggplot(data=milk_mon,
           mapping=p9.aes(x='month',
                          y='milk_prod_per_cow_kg',
                          color='milk_prod_per_cow_kg'))
    + p9.geom_line()
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-788-718e311bbd4a> in <module>
----> 1 (p9.ggplot(data=milk_mon,
      2            mapping=p9.aes(x='month',
      3                           y='milk_prod_per_cow_kg',
      4                           color='milk_prod_per_cow_kg'))
      5     + p9.geom_line()

NameError: name 'milk_mon' is not defined
```

In [789]:
```
ggplot(aes(x='month', y='milk_prod_per_cow_kg'), data=milk_mon) + geom_line()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-789-9a2f8f70595f> in <module>
----> 1 ggplot(aes(x='month', y='milk_prod_per_cow_kg'), data=milk_mon) + geom_line()

NameError: name 'milk_mon' is not defined
```

In [790]:
```
ggplot(meat_yr, aes(x = meat_yr.index, weight = 'cattle')) + \
    geom_bar(fill = 'blue', color = 'steelblue', width = 5) + \
    ggtitle('Head of Cattle Slaughtered by Decade')
```



Out[790]: <ggplot: (86196020484)>

In [791]:
```
meat_yr
```

Out[791]:

date	cattle	hogs	sheep	poultry
1948-12-31	681.1	12805.0	16896.9	505.5
1949-12-31	630.7	17095.0	13376.6	388.5
1950-12-31	1465.0	15056.0	12852.3	596.3
1951-12-31	3392.1	16528.5	11074.8	425.0
1952-12-31	2765.7	16198.9	13962.4	559.6
...	...	...	...	...
2016-12-31	30578.2	46219.9	2237.9	47364.4
2017-12-31	32189.4	49316.5	2178.2	48315.2
2018-12-31	33004.7	52432.5	2264.9	49161.7
2019-12-31	33555.3	57913.0	2321.7	50395.1
2020-12-31	18710.4	32887.9	1298.6	29652.8

73 rows × 4 columns

In [ ]:

In [ ]: