In [1]:

```
%matplotlib inline

import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import sklearn
import statsmodels.api as sm

import seaborn as sns
sns.set_style("whitegrid")
sns.set_context("poster")

# special matplotlib argument for improved plots
from matplotlib import rcParams
```

In [2]:

```
ins = pd.read_csv("//home//yeshua//Documents//study//excel//Advertising.csv");
```

In [3]:

```
print(ins.keys())
```

```
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

In [4]:

```
ins.head()
```

Out[4]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [5]:

```
print(ins.describe())
```

```
       Unnamed: 0          TV       radio   newspaper       sales
count  200.000000  200.000000  200.000000  200.000000  200.000000
mean   100.500000  147.042500   23.264000   30.554000   14.022500
std     57.879185   85.854236   14.846809   21.778621    5.217457
min      1.000000    0.700000    0.000000    0.300000    1.600000
25%     50.750000   74.375000    9.975000   12.750000   10.375000
50%    100.500000  149.750000   22.900000   25.750000   12.900000
75%    150.250000  218.825000   36.525000   45.100000   17.400000
max    200.000000  296.400000   49.600000  114.000000   27.000000
```

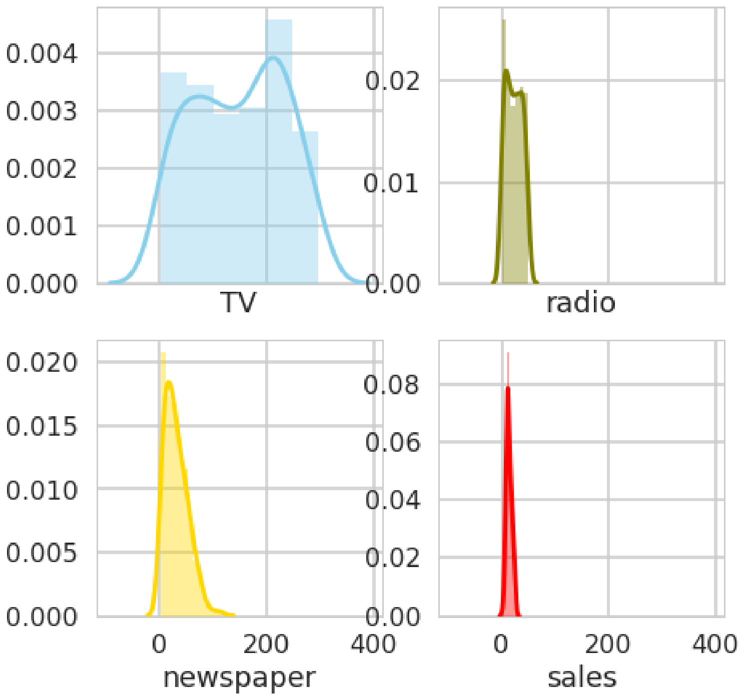
In [6]:

```
f, axes = plt.subplots(2, 2, figsize=(7, 7), sharex=True)
sns.distplot( ins["TV"] , color="skyblue", ax=axes[0, 0])
sns.distplot( ins["radio"] , color="olive", ax=axes[0, 1])
sns.distplot( ins["newspaper"] , color="gold", ax=axes[1, 0])
sns.distplot( ins["sales"] , color="red", ax=axes[1, 1])
```

```
/home/yeshua/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/home/yeshua/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/home/yeshua/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/home/yeshua/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: The
'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[6]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f9f985c4c18>

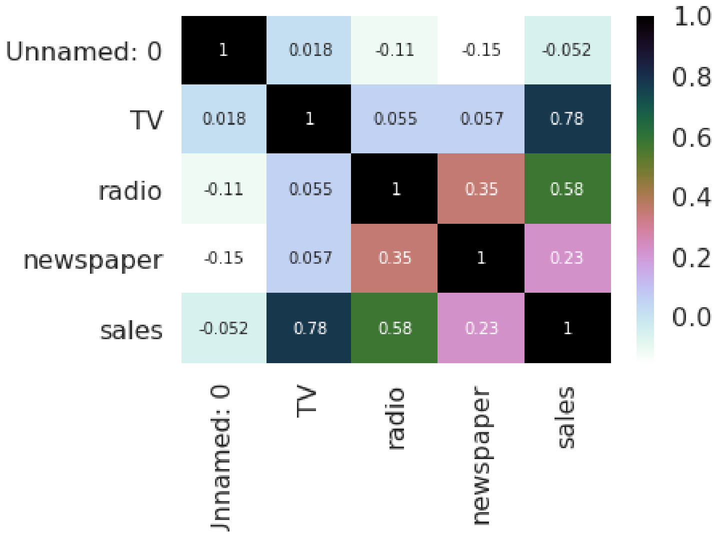


In [7]:

```
corr= ins.corr()
corr
sns.heatmap(corr,xticklabels=corr.columns.values,yticklabels=corr.columns.values,annot=True,cmap='c
ubehelix_r')
```

Out[7]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f9f983c1710>

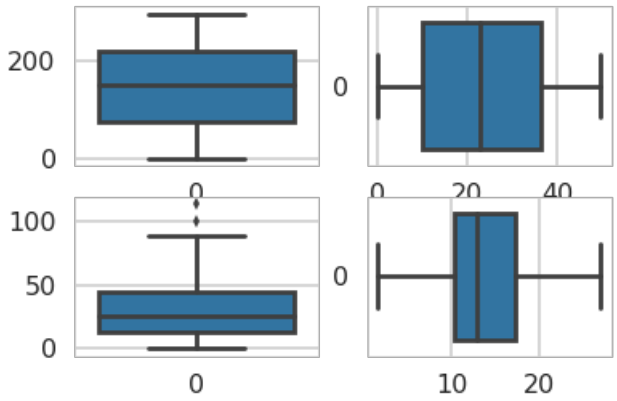


In [8]:

```
fig, axs = plt.subplots(ncols=2,nrows=2)
sns.boxplot(data= ins['TV'], ax=axs[0,0])
sns.boxplot(data= ins['radio'], ax=axs[0,1],orient='h')
sns.boxplot(data= ins['newspaper'], ax=axs[1,0])
sns.boxplot(data= ins['sales'], ax=axs[1,1],orient='h')
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f970683c8>
```

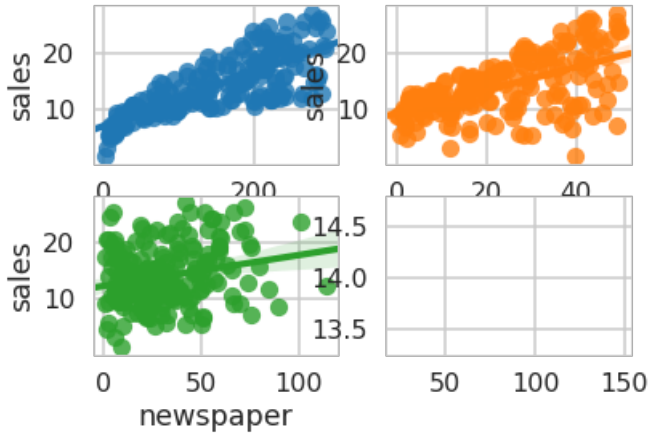


In [9]:

```
fig, axs = plt.subplots(ncols=2,nrows=2)
sns.regplot(x='TV',y='sales',data=ins,ax=axs[0,0])
sns.regplot(x='radio',y='sales',data=ins,ax=axs[0,1])
sns.regplot(x='newspaper',y='sales',data=ins,ax=axs[1,0])
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f96f67b00>
```



In [10]:

```
X = ins.drop('sales', axis = 1)
Y = ins['sales']
```

In [11]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, random_state = 5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(134, 4)
(66, 4)
```

(134,)
(66,)

In [12]:

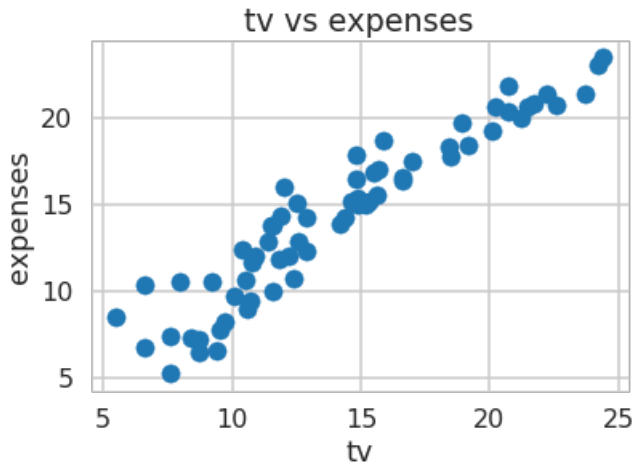
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, Y_train)

Y_pred = lm.predict(X_test)

plt.scatter(Y_test, Y_pred)
plt.xlabel("tv")
plt.ylabel("expenses")
plt.title("tv vs expenses")
```

Out[12]:

```
Text(0.5,1,'tv vs expenses')
```



In [13]:

```
y_predict=lm.predict(X_test)
```

In [14]:

```
mse = sklearn.metrics.mean_squared_error(Y_test, Y_pred)
print(mse)
```

```
2.4539450141659724
```

In [15]:

```
cofficient=lm.coef_
cofficient
```

Out[15]:

```
array([-0.00165547,  0.04685823,  0.1850065 , -0.00125919])
```

In [16]:

```
intercept=lm.intercept_
intercept
```

Out[16]:

```
3.0919757561429666
```

In [17]:

```
from sklearn.metrics import mean_squared_error
```

```
from sklearn.metrics import mean_squared_error
y_pred_general_train = lm.predict(X_train)
y_pred_general_test = lm.predict(X_test)
```

In [18]:

```
mse_general_train = mean_squared_error(y_pred_general_train, Y_train)
mse_general_test = mean_squared_error(y_pred_general_test, Y_test)
```

In [19]:

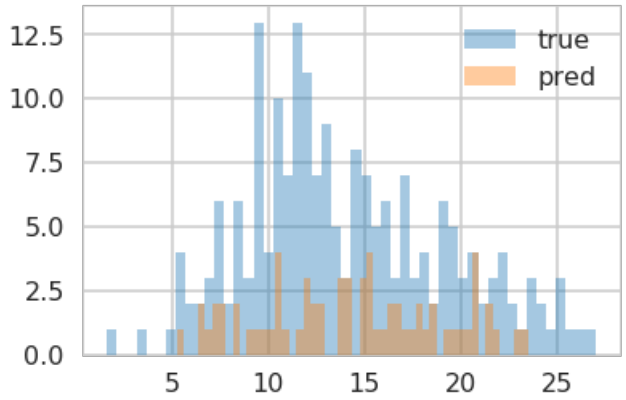
```
mse_general_train - mse_general_test, mse_general_train, mse_general_test
```

Out[19]:

```
(0.5165631748483746, 2.970508189014347, 2.4539450141659724)
```

In [20]:

```
plt.hist(Y, bins=50, alpha=0.4, label="true")
plt.hist(y_predict, bins=50, alpha=0.4, label="pred")
plt.legend()
plt.show()
```



In [21]:

```
lm.score(X,Y)
```

Out[21]:

```
0.8966230217948369
```

In [22]:

```
# Import the random forest model.
from sklearn.ensemble import RandomForestRegressor

# Initialize the model with some parameters.
model = RandomForestRegressor(n_estimators=100, min_samples_leaf=10, random_state=1)
# Fit the model to the data.
model.fit(X_train, Y_train)
# Make predictions.
predictions = model.predict(X_test)
# Compute the error.
mse = sklearn.metrics.mean_squared_error(predictions, Y_test)
print(mse)
```

```
1.3255947781523982
```