1. Carga del conjunto de datos:

A lo largo de esta práctica utilizaremos el conjunto de datos Forest Fires Data Set, el cual contiene información relevante acerca de distintos incendios acaecidos en el noreste de Portugal. Cada muestra del conjunto de datos estará formada por el área de bosque quemada y el valor de múltiples factores que podrían ser los detonantes o catalizadores del fuego. El conjunto de datos lo encontraréis en el siguiente enlace: https://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv.

Nota: para los ejercicios de esta PEC, utilizaremos como variable objetivo el "area": número de hectáreas afectadas por el incendio. El resto de variables del conjunto de datos conformarán los atributos descriptivos.

In [1]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn import preprocessing
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, cross_val_score

import matplotlib
import matplotlib.pyplot as plt

pd.set_option('display.max_columns', None)
seed = 100

%matplotlib inline
```

In [2]:
```
# Cargamos el dataset.
'''
La 'r' previa a la string con la ruta del archivo
indica 'raw' para que el intérprete de Python entienda los backlashes (\)
de forma literal y no dé error de unicode. También podríamos haber duplicado
el backlash en cada caso quedando como 'C:\\users\\fjrc9...'.
'''
ForestFires_df = pd.read_csv(r'C:\users\fjrc9\desktop\forestfires.csv')

# Número de filas y muestras del dataset.
print('El conjunto de datos (dataframe) tiene \033[1m{} filas\033[0m,'
      ' quitando la primera (el encabezado) tendríamos un total de'
      ' \033[1m{} muestras\033[0m.\n'.format(ForestFires_df.shape[0], ForestFires_df.shape[0]-1))

# Columnas y atributos del dataset.
print('Tenemos \033[1m{} columnas\033[0m y representan'
      ' los siguientes atributos:\033[1m\n{}\n\033[0m'.format(ForestFires_df.shape[1], [i for i in ForestFires_

# Valores perdidos.
missing_values_NA = ForestFires_df.isna()
print('Valores \033[1mNA\033[0m en cada columna: \n{}\n'.format(missing_values_NA.sum()))
missing_values_NULL = ForestFires_df.isnull()
print('Valores \033[1mNULL\033[0m en cada columna: \n{}\n'.format(missing_values_NULL.sum()))
print('\033[1mNo tenemos valores perdidos\033[0m en nuestro dataset.')
```

El conjunto de datos (dataframe) tiene **517 filas**, quitando la primera (el encabezado) tendríamos un total de **516 muestras**.

Tenemos **13 columnas** y representan los siguientes atributos:
['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH', 'wind', 'rain', 'area']

Valores **NA** en cada columna:
```
X        0
Y        0
month    0
day      0
FFMC     0
DMC      0
DC       0
ISI      0
temp     0
RH       0
wind     0
rain     0
area     0
dtype: int64
```

Valores **NULL** en cada columna:
```
X        0
Y        0
month    0
day      0
FFMC     0
DMC      0
DC       0
ISI      0
temp     0
RH       0
wind     0
rain     0
area     0
dtype: int64
```

No tenemos valores perdidos en nuestro dataset.

2. Análisis de los datos:

2.1 Análisis estadístico básico:

In [3]:
```
# Primero vemos que pinta tienen los valores en cada variable.
print(ForestFires_df.head())
```

```
   X  Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area
0  7  5   mar  fri  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0
1  7  4   oct  tue  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0
2  7  4   oct  sat  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0
3  8  6   mar  fri  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0
4  8  6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0
```

Vemos que 'month' y 'day' son variables categóricas, 'X' e 'Y' son variables geoespaciales y el resto son variables numéricas

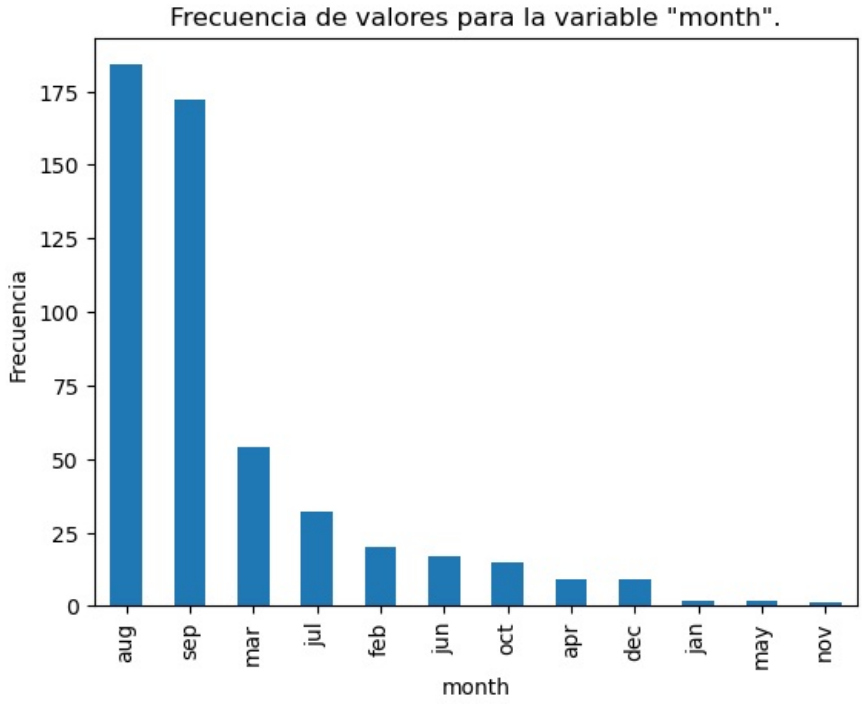
Variables categóricas:

In [4]:
```
# Frecuencia de valores de "month".
freq_month = ForestFires_df['month'].value_counts()
print('La \033[1mfrecuencia para'
      ' "month"\033[0m es:\n{}\n'.format(freq_month))
# Gráfico de barras para frecuencia de "month".
freq_month.plot(kind='bar')
plt.title('Frecuencia de valores para la variable "month".')
plt.ylabel('Frecuencia')
plt.xlabel('month')
plt.show()

# Frecuencia de valores para la variable "day".
freq_day = ForestFires_df['day'].value_counts()
print('La \033[1mfrecuencia para'
      ' "day"\033[0m es:\n{}\n'.format(freq_day))
# Gráfico de barras para frecuencia de "day".
freq_day.plot(kind='bar')
plt.title('Frecuencia de valores para la variable "day".')
plt.ylabel('Frecuencia')
plt.xlabel('day')
plt.show()
```

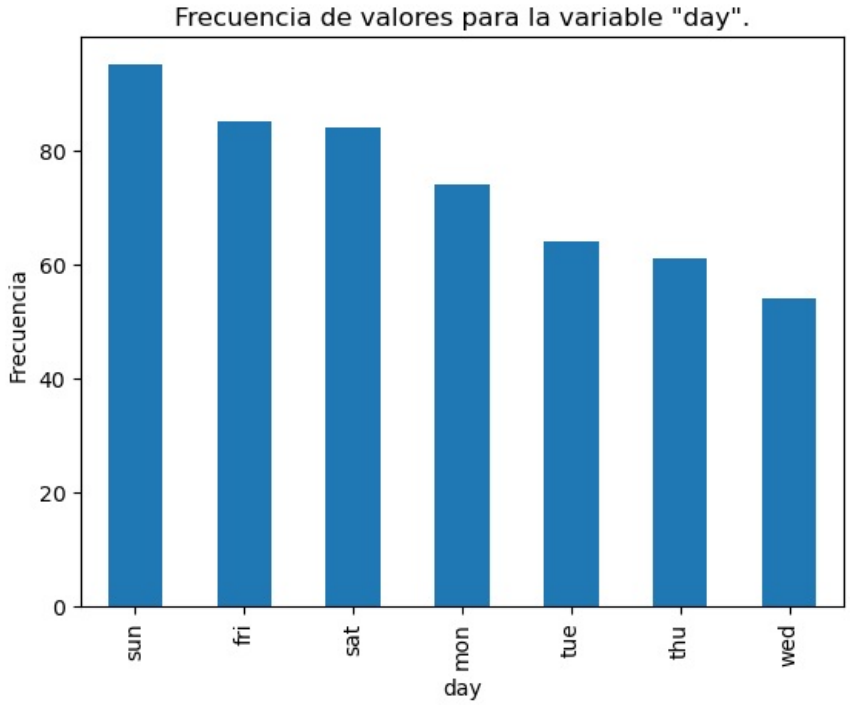
La **frecuencia para "month"** es:

```
aug    184
sep    172
mar     54
jul     32
feb     20
jun     17
oct     15
apr      9
dec      9
jan      2
may      2
nov      1
Name: month, dtype: int64
```



La **frecuencia para "day"** es:

```
sun    95
fri    85
sat    84
mon    74
tue    64
thu    61
wed    54
Name: day, dtype: int64
```



Variables numéricas:

In [5]:
```
# Media.
print('La \033[1mmedia\033[0m para cada atributo numérico es:')
media_por_columna = ForestFires_df.iloc[:, 3:].mean(numeric_only=True)
print(media_por_columna, '\n')

# Mediana.
print('La \033[1mmediana\033[0m para cada atributo numérico es:')
mediana_por_columna = ForestFires_df.iloc[:, 3:].median(numeric_only=True)
print(mediana_por_columna, '\n')

# Desviación estándar
print('La \033[1mdesviación estándar\033[0m para cada atributo numérico es:')
desviacion_por_columna = ForestFires_df.iloc[:, 3:].std(numeric_only=True)
print(desviacion_por_columna, '\n')
```

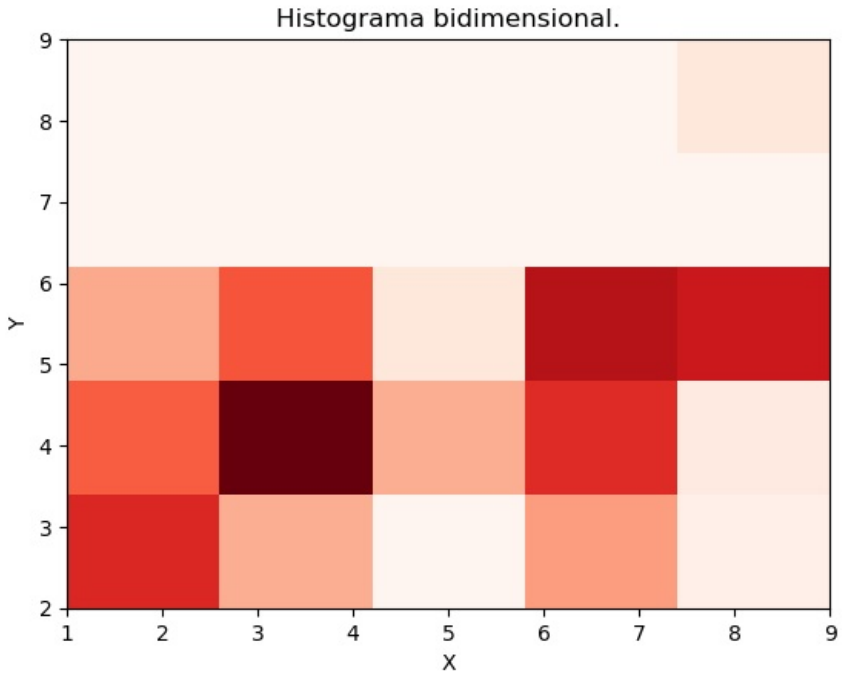
```
La media para cada atributo numérico es:
FFMC     90.644681
DMC     110.872340
DC      547.940039
ISI       9.021663
temp     18.889168
RH       44.288201
wind      4.017602
rain      0.021663
area     12.847292
dtype: float64

La mediana para cada atributo numérico es:
FFMC     91.60
DMC     108.30
DC      664.20
ISI       8.40
temp     19.30
RH       42.00
wind      4.00
rain      0.00
area      0.52
dtype: float64

La desviación estándar para cada atributo numérico es:
FFMC      5.520111
DMC      64.046482
DC      248.066192
ISI       4.559477
temp      5.806625
RH       16.317469
wind      1.791653
rain      0.295959
area     63.655818
dtype: float64
```

Variables geoespaciales:

In [6]:
```
# Hacemos el histograma en 2 dimensiones con la función 'hist2d' de matplotlib.
plt.hist2d(ForestFires_df['X'], ForestFires_df['Y'], bins=5, cmap='Reds')
plt.title('Histograma bidimensional.')
plt.ylabel('Y')
plt.xlabel('X')
plt.show()
```



Conclusiones del análisis estadístico básico:

Variables categóricas:

Atendiendo a las variables categóricas, parece ser que los días menos probables de que se produjera un incendio eran los miércoles ya que registran la menor de las frecuencias. En cuanto a los meses, vemos que el mes de noviembre es el más seguro en cuanto a incendios, seguid por mayo y enero donde sólo se registraron 1, 2 y 2 incendios respectivamente.

Variables numéricas:

Atendiendo a nuestra variable objetivo (el área de superficie quemada etiquetada como 'area' en nuestro dataset), vemos que el valor medio de superficie quemada es de unas 12.84 hectareas. La mediana es de sólo unos 0.52 hectareas, lo cual indica que la superficie quemada en el dataset está bastante localizada en una o unas regiiones en concreto, ya que si la distribución de superficie quemada fuese más homogénea, la mediana no sería un valor tan dispar con respecto de la media. Esta última idea la confirmamos al ver que la desviación estándar de la variable "area" es de 63.65 un valor demasiado alto que nos indica que los datos tienen una gran dispersión (una superficie quemada más heterogénea).

Variables geoespaciales:

Como comentamos en el apartado anterior, en base a la disparidad entre el valor medio y la mediana de "area" (además del valor tan elevado para la desviación estándar), al hacer un histograma bidimensional para las coordenadas X e Y de los sectores quemados, obtenemosun "mapa de calor" que nos confirma que hay unas 5 regiones quemadas que destacan sobre las demás, en especial el sector (X=3, Y=4) mientras que la gran mayoría de sectores no han sido quemados o apenas han sufrido los efectos de un incendio. Por lo que esto respalda nuestra idea de que los incendios han sido muy focalizados y por ello la mediana difiere tanto de la superficie quemada en promedio ya que los incendios más graves se han concentrado en unas regiones muy concretas.

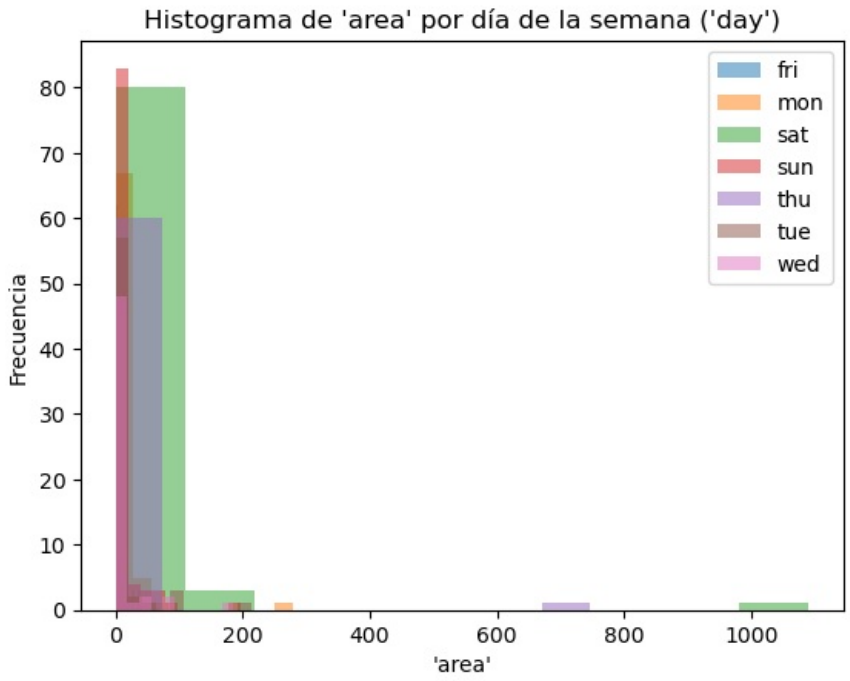
2.2 Análisis exploratorio de los datos:

'area' vs 'day'

In [7]:
```
'''
Agrupamos las variables por la variable 'day' de modo que obtenemos
un nuevo dataframe agrupado por dicha variable.
'''
grouped = ForestFires_df.groupby('day')

# Creamos un histograma para cada día de la semana.
'''
Cada iteración sobre nuestro dataframe agrupado por la variable 'day'
nos devolvera el valor del área (group['area']) y 'name' que será
la etiqueta correspondiente a la variable categórica 'day'.
'''
for name, group in grouped:
    plt.hist(group['area'], alpha=0.5, label=name)
```

```
# Añadimos título, etiquetas y leyenda antes de mostrar el resultado.
plt.xlabel('\'area\'')
plt.ylabel('Frecuencia')
plt.title('Histograma de \'area\' por día de la semana (\'day\')')
plt.legend()
plt.show()
```

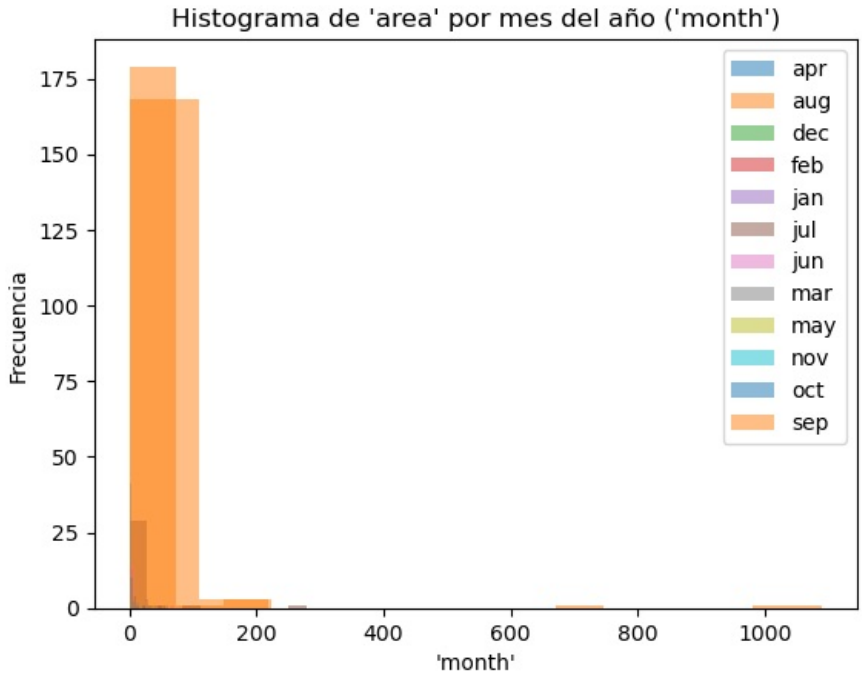


'area' vs 'month'

In [8]:
```
'''
Agrupamos las variables por la variable 'month' de modo que obtenemos
un nuevo dataframe agrupado por dicha variable.
'''
grouped = ForestFires_df.groupby('month')

# Creamos un histograma para cada día de la semana.
'''
Cada iteración sobre nuestro dataframe agrupado por la variable 'month'
nos devolvera el valor del área (group['area']) y 'name' que será
la etiqueta correspondiente a la variable categórica 'month'.
'''
for name, group in grouped:
    plt.hist(group['area'], alpha=0.5, label=name)

# Añadimos título, etiquetas y leyenda antes de mostrar el resultado.
plt.xlabel('\'month\'')
plt.ylabel('Frecuencia')
plt.title('Histograma de \'area\' por mes del año (\'month\')')
plt.legend()
plt.show()
```



Si atendemos al gráfico con las categorías de la variable 'day' vemos como predomina el verde correspondiente a 'sat' (saturday), es decir, al sábado. La diferencia respecto a los otros días de la semana no es muy exagerada pero si lo suficientemente notoria. Tiene lógica ya que podría deberse a que durante los fines de semana, la población suele tener los días libres y pueden realizar actividades al aire libre en la montaña o zonas boscosas. Es obvio que la actividad humana como barbacoas, campings, rutas de senderismo, tabaquismo de las personas que acudan a dichas actividades en dichas zonas (o su cercanía) y la basura que se puede generar debido a ello (colillas, ceniza, botellas de vidrio que hacen efecto lupa con el sol, envases de plástico y/o papel, etc) supone un riesgo potencial de incendio debido a la naturaleza del entorno, especialmente en días soleados y calurosos.

Siguiendo nuestro argumento, cuando vemos el gráfico para los meses del año predomina por mucho el color naranja, asociado a los meses de agosto y septiembre que son los meses finales de verano, lo cual concuerda ya que la población aprovecha el buen tiempo de la temporada estival y las vacaciones para realizar el tipo de actividades ya comentadas en el anterior párrafo. Además las condiciones de sol, calor y disminución de precipitaciones típicas del verano, también aumentan el riesgo de incendio, por lo que ambos riesgos alcanzan su máxima presencia en dicha época.

Podríamos concluir que, durante los fines de semana, en especial los sábados, y durante el verano, el riesgo de incendio y de que este afecte un área mayor, aumenta significativamente respecto al resto de días de la semana y épocas del año.

In [9]:
```
'''
Agrupamos la variable 'area' por etiquetas de 'day'.
Y hacemos el valor medio y desviación estándar del área
por cada categoría de 'day'.
'''
day_means = ForestFires_df.groupby('day')['area'].mean()
day_std = ForestFires_df.groupby('day')['area'].std()

# Repetimos agrupando por 'month'.
month_means = ForestFires_df.groupby('month')['area'].mean()
month_std = ForestFires_df.groupby('month')['area'].std()

# Mostramos los resultados
print("\033[1mMedia del área (\'area\') por día de la semana (\'day\')\033[0m:")
print(day_means, '\n')
print("\033[1mDesviación estándar del área (\'area\') por día de la semana (\'day\')\033[0m:")
print(day_std, '\n')

print("\033[1mMedia del área (\'area\') por mes del año (\'month\')\033[0m:")
print(month_means, '\n')
print("\033[1mDesviación estándar del área (\'area\') por dmes del año (\'month\')\033[0m:")
print(month_std, '\n')
```

Media del área ('area') por día de la semana ('day'):

```
day
fri     5.261647
mon     9.547703
sat    25.534048
sun    10.104526
thu    16.345902
tue    12.621719
wed    10.714815
Name: area, dtype: float64
```

Desviación estándar del área ('area') por día de la semana ('day'):

```
day
fri     10.012083
mon     33.703562
sat    122.698840
sun     26.076032
thu     95.351052
tue     33.568193
wed     30.285914
Name: area, dtype: float64
```

Media del área ('area') por mes del año ('month'):

```
month
apr     8.891111
aug    12.489076
dec    13.330000
feb     6.275000
jan     0.000000
jul    14.369687
jun     5.841176
mar     4.356667
may    19.240000
nov     0.000000
oct     6.638000
sep    17.942616
Name: area, dtype: float64
```

Desviación estándar del área ('area') por dmes del año ('month'):

```
month
apr    19.929092
aug    60.364174
dec     6.610747
feb    12.342510
jan     0.000000
jul    50.849299
jun    16.884945
mar     9.140107
may    27.209469
nov          NaN
oct    13.699522
sep    87.648175
Name: area, dtype: float64
```

- **Aclaración 1:** Vemos que el valor medio de área por mes del año, también es significativo en julio y en mayo (aunque no en junio), puede que debido a la asignación de colores en el gráfico, el naranja se haya impuesto sobre el color de esos meses.
- **Aclaración 2:** La desviación estándar para el mes de noviembre es ***NaN*** ya que sólo tenemos una instancia en noviembre.

Calculamos y mostramos la correlación entre todos los atributos numéricos (incluyendo los geoespaciales) y la respuesta.

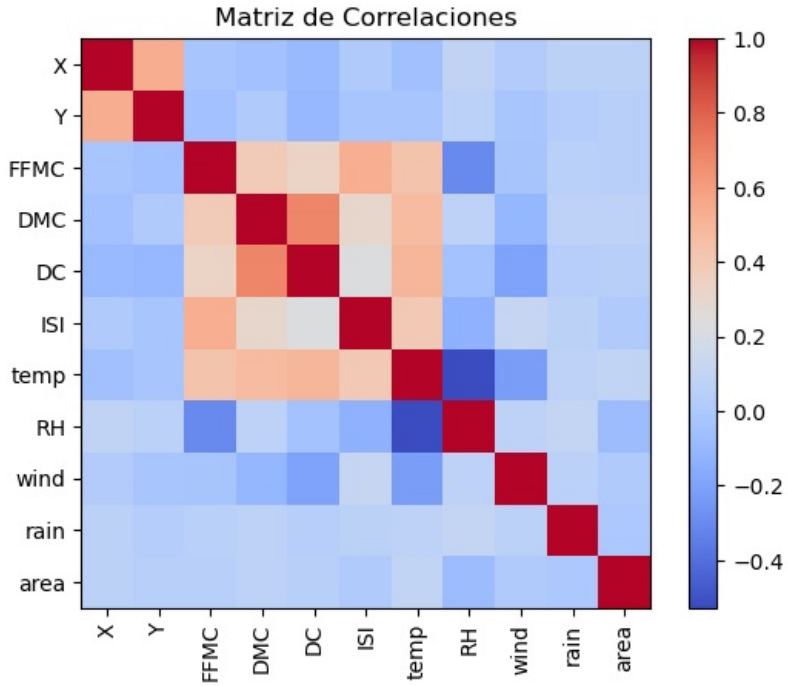
In [10]:
```python
# Creamos un dataframe en el que excluimos las variables categóricas 'day' y 'month'.
ForestFiresNumeric_df = ForestFires_df.drop(['day', 'month'], axis=1)

# Calculamos la matriz de correlaciones.
matriz_correlaciones = ForestFiresNumeric_df.corr()

# Mostramos el resultado de la correlación de todas las variables con la variable objetivo 'area'.
print(matriz_correlaciones['area'])
```

```
X       0.063385
Y       0.044873
FFMC    0.040122
DMC     0.072994
DC      0.049383
ISI     0.008258
temp    0.097844
RH     -0.075519
wind    0.012317
rain   -0.007366
area    1.000000
Name: area, dtype: float64
```

In [11]:
```
plt.imshow(matriz_correlaciones, cmap='coolwarm')
plt.colorbar()
plt.title('Matriz de Correlaciones')
plt.xticks(range(len(matriz_correlaciones.columns)), matriz_correlaciones.columns, rotation=90)
plt.yticks(range(len(matriz_correlaciones.columns)), matriz_correlaciones.columns)
plt.show()
```



Los dos atributos que tienen mayor correlación con 'area':

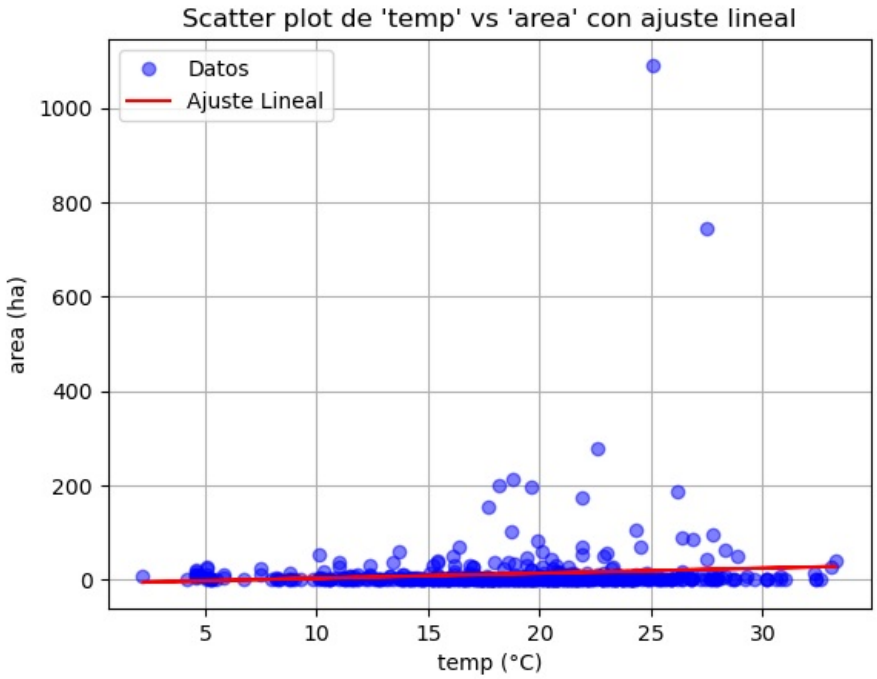
- **'temp':** con una correlación de 0.0978 con 'area'.
- **'RH':** con una correlación de -0.0755 con 'area'.

In [12]:
```
plt.plot(ForestFiresNumeric_df['temp'], ForestFiresNumeric_df['area'], 'ob', alpha=0.5, label='Datos')

# Hacemos el ajuste lineal.
pendiente, cte = np.polyfit(ForestFiresNumeric_df['temp'], ForestFiresNumeric_df['area'], 1)
plt.plot(ForestFiresNumeric_df['temp'], pendiente * ForestFiresNumeric_df['temp'] + cte, color='red', label='Aj

# Agregamos etiquetas, título, leyenda y cuadrícula.
plt.xlabel('temp (°C)')
plt.ylabel('area (ha)')
plt.title('Scatter plot de \'temp\' vs \'area\' con ajuste lineal')
plt.legend()
plt.grid()

# Mostramos el gráfico.
plt.show()
```

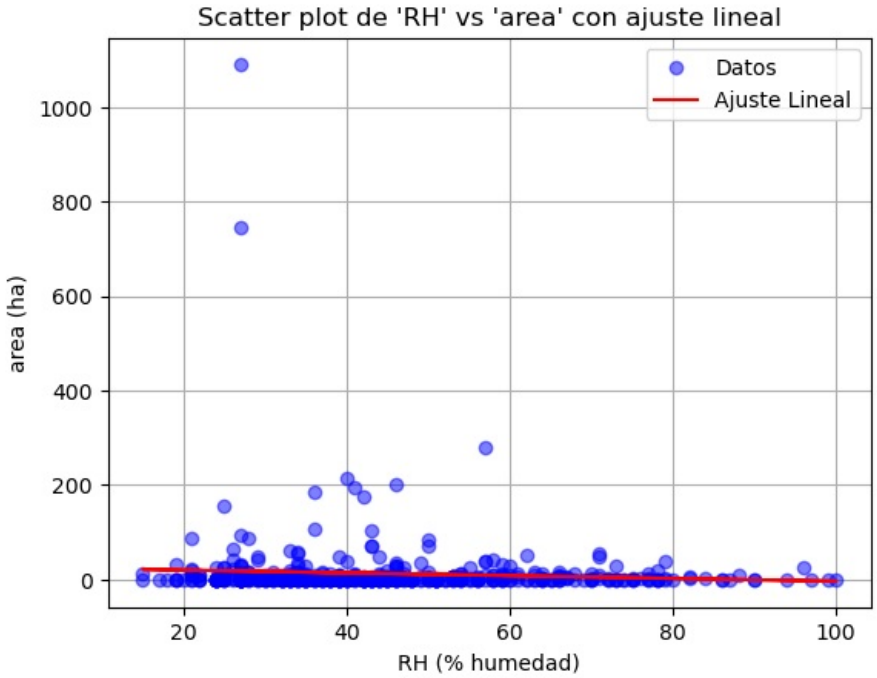


In [13]:
```
plt.plot(ForestFiresNumeric_df['RH'], ForestFiresNumeric_df['area'], 'ob', alpha=0.5, label='Datos')

# Hacemos el ajuste lineal.
pendiente, cte = np.polyfit(ForestFiresNumeric_df['RH'], ForestFiresNumeric_df['area'], 1)
plt.plot(ForestFiresNumeric_df['RH'], pendiente * ForestFiresNumeric_df['RH'] + cte, color='red', label='Ajuste

# Agregamos etiquetas, título, leyenda y cuadrícula.
plt.xlabel('RH (% humedad)')
plt.ylabel('area (ha)')
plt.title('Scatter plot de \'RH\' vs \'area\' con ajuste lineal')
plt.legend()
plt.grid()

# Mostramos el gráfico.
plt.show()
```



3. Preprocesado de los datos:

Una vez analizados los atributos descriptivos, es el momento de prepararlos para que nos sean útiles de cara a predecir valores. En este apartado:

- Transformaremos las variables categóricas en varias variables binarias, una para cada categoría.
- Estandarizaremos los valores de los atributos descriptivos para que sus escalas no sean muy diferentes.
- Separaremos el conjunto de datos original en dos subconjuntos: entrenamiento y test.

Eliminamos los atributos categóricos del conjunto de datos y en su lugar transformamos dichos atributos a tantas variables binarias

Eliminamos los atributos categóricos del conjunto de datos y en su lugar transformamos dichos atributos a tantas variables binarias como categorías tengan.

In [17]:
```
# Aplicamos get_dummies a la variables 'day' y 'month'.
ForestFiresBinary_df = pd.get_dummies(ForestFires_df, columns=['day', 'month'], prefix=('day', 'month'))

# Mostramos el resultado.
print(ForestFiresBinary_df.head())
```

```
   X  Y  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area  day_fri  day_mon  \
0  7  5  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0        1        0
1  7  4  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0        0        0
2  7  4  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0        0        0
3  8  6  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0        1        0
4  8  6  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0        0        0

   day_sat  day_sun  day_thu  day_tue  day_wed  month_apr  month_aug  \
0        0        0        0        0        0          0          0
1        0        0        0        1        0          0          0
2        1        0        0        0        0          0          0
3        0        0        0        0        0          0          0
4        0        1        0        0        0          0          0

   month_dec  month_feb  month_jan  month_jul  month_jun  month_mar  \
0          0          0          0          0          0          1
1          0          0          0          0          0          0
2          0          0          0          0          0          0
3          0          0          0          0          0          1
4          0          0          0          0          0          1

   month_may  month_nov  month_oct  month_sep
0          0          0          0          0
1          0          0          1          0
2          0          0          1          0
3          0          0          0          0
4          0          0          0          0
```

Estandarizamos todos los atributos descriptivos.

In [18]:
```
# Creamos un objeto StandardScaler.
scaler = preprocessing.StandardScaler()

# Estandarizamos las variables descriptivas (coinciden con las numéricas).
ForestFiresStandard_df = scaler.fit_transform(ForestFiresNumeric_df)

# Este método nos devuelve un array de numpy, lo transformamos en dataframe nuevamente.
ForestFiresStandard_df = pd.DataFrame(ForestFiresStandard_df, columns=ForestFiresNumeric_df.columns)

# Mostramos el resultado
print(ForestFiresStandard_df.head())
```

```
          X         Y      FFMC       DMC        DC       ISI      temp  \
0  1.008313  0.569860 -0.805959 -1.323326 -1.830477 -0.860946 -1.842640
1  1.008313 -0.244001 -0.008102 -1.179541  0.488891 -0.509688 -0.153278
2  1.008313 -0.244001 -0.008102 -1.049822  0.560715 -0.509688 -0.739383
3  1.440925  1.383722  0.191362 -1.212361 -1.898266 -0.004756 -1.825402
4  1.440925  1.383722 -0.243833 -0.931043 -1.798600  0.126966 -1.291012

         RH      wind      rain     area
0  0.411724  1.498614 -0.073268 -0.20202
1 -0.692456 -1.741756 -0.073268 -0.20202
2 -0.692456 -1.518282 -0.073268 -0.20202
3  3.233519 -0.009834  0.603155 -0.20202
4  3.356206 -1.238940 -0.073268 -0.20202
```

Nos extraña el hecho de tener estos valores tras la estandarización, se supone que la estandarización nos devuelve un conjunto de datos cuya media es 0 y su desviación estándar debería ser 1. Es cierto que en muchas instancias tenemos variables con valor 0 y la media de esas mismas varaibles es mayor que cero (en otras palabras, la distribución de los valores no es normal), por lo que al estandarizar mediante la fórmula:

$$z = \frac{x - \mu}{\sigma}$$

Hace que obtengamos un numerador negativo. No sé si deberíamos haber considerado las instancias con valor 0 como valores perdidos en algunos casos. Yo los interpreté como que el modelo no previó ningún incendio y por tanto la superficie quemada era 0.

Atendiendo a las explicaciones del link al dataset (https://archive.ics.uci.edu/dataset/162/forest+fires) se trata de un conjunto de datos que pretende entrenar un algoritmo de regresión para prever la superficie quemada en los incendios al noreste de portugal. Nos dice que antes de llevar a cabo la ejecución del modelo, los datos para el área fueron transformados mediante una función logarítmica y luego de ejecutar el modelo deshicieron dicha transformación. Puedo que este proceso de pasarlo primero a una escala logarítmica y luego deshacer esa manipulación en el *output* de su modelo, haga que la variable *'area'* sea muy sensible.

Creo que el proceso que hemos llevado hasta ahora es correcto, pero tengo dudas al ver el resultado que obtenemos con muchas muestras con valor negativo e incluso fuera del rango típico [-1, 1].

Vamos a realizar una comprobación de los valores medios y las desviaciones estándar de las variables tras el proceso de estandarización, como hemos visto en la teoría, la media debería ser cero y la desviación estándar debería ser 1.

In [19]:
```
print('\033[1mValores medios\033[0m de las variables estandarizadas:\n{}\n'.format(ForestFiresStandard_df.mean(
print('\033[1mDesviación estándar\033[0m de las variables estandarizadas:\n{}'.format(ForestFiresStandard_df.st
```

```
Valores medios de las variables estandarizadas:
X       2.113074e-16
Y       2.611279e-16
FFMC   -1.752306e-15
DMC    -2.748715e-17
DC      6.871787e-17
ISI     1.030768e-17
temp    2.542561e-16
RH      2.198972e-16
wind   -4.191790e-16
rain   -6.871787e-18
area    4.123072e-17
dtype: float64

Desviación estándar de las variables estandarizadas:
X       1.000969
Y       1.000969
FFMC    1.000969
DMC     1.000969
DC      1.000969
ISI     1.000969
temp    1.000969
RH      1.000969
wind    1.000969
rain    1.000969
area    1.000969
dtype: float64
```

Separamos los atributos descriptivos y la variable objetivo en los subconjuntos de entrenamiento y test.

In [20]:
```
# Separamos la variable objetivo del resto teniendo en cuenta el dataframe estandarizado.
X = ForestFiresStandard_df.drop('area', axis=1)
Y = ForestFiresStandard_df['area']

# Dividimos los datos en subconjuntos de entrenamiento y prueba.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Veamos cuantas muestras tenemos en cada subconjunto.
print('Nuestro subconjunto de entrenamiento tiene {} muestras.'.format(len(X_train)))
print('Nuestro subconjunto de prueba tiene {} muestras.'.format(len(X_test)))
```

```
Nuestro subconjunto de entrenamiento tiene 413 muestras.
Nuestro subconjunto de prueba tiene 104 muestras.
```

Si hubiéramos dividido primero los datos en los subconjuntos de entrenamiento y prueba, sería más problemático realizar la estandarización de las variables ya que la media de cada subconjunto sería distinta a la que tendría el conjunto original de datos, en cuanto a la desviación estándar tendríamos el mismo problema. Recordemos que la fórmula para estandarizar variables es:

$$z = \frac{x - \mu}{\sigma}$$

Donde z es el valor estandarizado, x el valor que se pretende estandarizar, μ el valor medio de la variable que se pretende estandarizar y σ la desviación estándar de la misma.

Otro punto a tener en cuenta es que al realizar primero la estandarización y transformación de los datos antes de dividirlos en subconjuntos, nuestro código es más simple y fácil de reutilizar, de algún modo conseguimos que sea modular ya que podemos manipular y transformar de distinta forma los datos antes de dividirlos nuevamente.

Estandarizar los valores de los atributos, entre otras cosas, nos aporta:

- **Entrenamiento más rápido de los modelos:** al tener escalas similares entre todas las variables, para el modelo de aprendizaje automático es más sencillo interpretar los coeficientes obtenidos en modelos lienales como, por ejemplo, una regresión, por lo tanto el desempeño del algoritmo será mejor ya que se entrena de forma más rápida y el número de instancias de entrenamiento se aprovecha mejor.
- **Reducción de riesgo de problemas numéricos:** La estandarización ayuda a mitigar o incluso eliminar la posible inestabilidad de los datos o la falta de convergencia en los mismos que pueden surgir cuando las variables tienen escalas muy dispares entre sí en un algoritmo.

Existen algunas situaciones en las que la estandarización es imprescindible, por ejemplo, para afrontar problemas de regresión lineal o

logística. O cuando nos vemos obligados a tener en cuenta *'sí o sí'* una variable con una escala muy dispar respecto a la de la variable objetivo en la ejecución de un modelo. En problemas de *clustering* ya que se basan en el concepto de distancia y los valores entre las distintas variables deben tener una escala comparable entre sí.

En general, estandarizar los atributos es una buena práctica.

4. Reducción de la dimensionalidad:

Con el propósito de comprobar visualmente la distribución de la variable objetivo teniendo en cuenta todos los atributos descriptivos a la vez, vamos a reducir la dimensionalidad del problema a solamente dos atributos que serán la proyección de los atributos descriptivos originales.

Generamos 2 PCA (componentes principales):

In [21]:
```
# Repetimos la definición de las variables para asegurarnos que partimos del dataframe ya estandarizado.
# previo a la división de subconjuntos de entrenamiento y prueba del apartado anterior.
X = ForestFiresStandard_df.drop('area', axis=1)
Y = ForestFiresStandard_df['area']

# Creamos una instancia de PCA con 2 componentes principales.
pca = PCA(n_components=2)

# Ajustamos el modelo PCA a los datos sin tener en cuenta la variable objetivo.
pca.fit(X)

# Transformamos las dimensiones originales en las nuevas dimensiones (reducidas a 2).
X_pca = pca.transform(X)

# Creamos un nuevo DataFrame con las dos dimensiones principales.
pca_df = pd.DataFrame(data=X_pca, columns=['PCA1', 'PCA2'])

# Agregamos la columna 'area' al nuevo dataframe.
pca_df['area'] = Y

# Vemos el nuevo dataframe con las dos componentes principales y la variable objetivo.
print(pca_df)
```

```
         PCA1      PCA2      area
0    3.312502  0.789929 -0.202020
1    0.245271  0.162852 -0.202020
2    0.470222  0.175328 -0.202020
3    3.081562  2.329574 -0.202020
4    2.705330  2.226644 -0.202020
..        ...       ...       ...
512  0.310514 -1.398239 -0.100753
513  1.562881 -1.045291  0.651674
514  1.840661  0.420487 -0.026532
515 -1.555718 -1.089617 -0.202020
516  3.389359 -1.017025 -0.202020

[517 rows x 3 columns]
```

Hacemos la representación gráfica del área de bosque quemada en función de las componentes principales que calculamos.

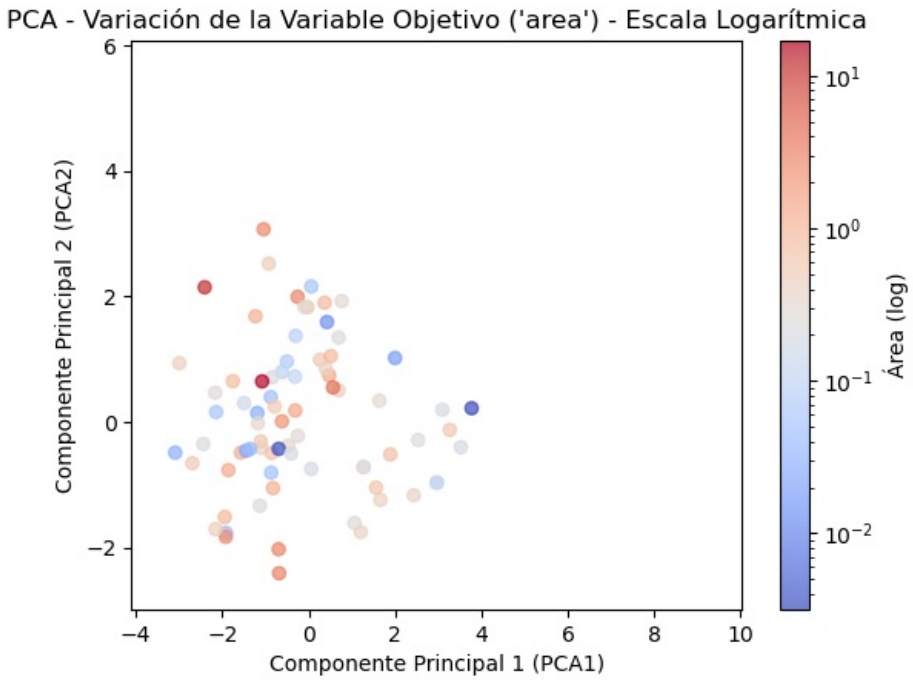
In [22]:
```
# Importamos la función LogNorm para poder usarla en nuestor gráfico.
from matplotlib.colors import LogNorm

# Creamos el gráfico usando un mapa de colores para la variable objetivo 'area' con escala logarítmica.
scatter = plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['area'], cmap='coolwarm', norm=LogNorm(), alpha=

# Agregamos barra de colores a modo de leyenda para la escala logarítmica de 'area'.
plt.colorbar(scatter, label='Área (log)', orientation='vertical')

# Etiquetas de los ejes y título del gráfico.
plt.xlabel('Componente Principal 1 (PCA1)')
plt.ylabel('Componente Principal 2 (PCA2)')
plt.title('PCA - Variación de la Variable Objetivo (\'area\') - Escala Logarítmica')

# Mostramos el gráfico.
plt.show()
```



Calculamos componentes TSNE.

In [23]:
```
# Creamos una instancia de TSNE con 2 componentes principales.
tsne = TSNE(n_components=2, learning_rate=10, perplexity=100)

# Aplicar TSNE al DataFrame (excluyendo la variable objetivo 'area')
X_tsne = tsne.fit_transform(X)

# Crear un nuevo DataFrame con las dos dimensiones principales
tsne_df = pd.DataFrame(data=X_tsne, columns=['t-SNE1', 't-SNE2'])

# Agregamos la columna 'area' al nuevo dataframe.
tsne_df['area'] = Y


# Vemos el nuevo dataframe con las dos componentes principales y la variable objetivo.
print(tsne_df)
```

```
       t-SNE1    t-SNE2      area
0    9.619971  2.359629 -0.202020
1    0.566107  0.038386 -0.202020
2    0.608145  0.117990 -0.202020
3    7.394624  0.243087 -0.202020
4    7.268708  0.162692 -0.202020
..        ...       ...       ...
512  3.396274  4.886887 -0.100753
513  2.756957  3.923809  0.651674
514  3.255217  2.577861 -0.026532
515 -5.901033 -0.125633 -0.202020
516  7.167114  4.553944 -0.202020

[517 rows x 3 columns]
```

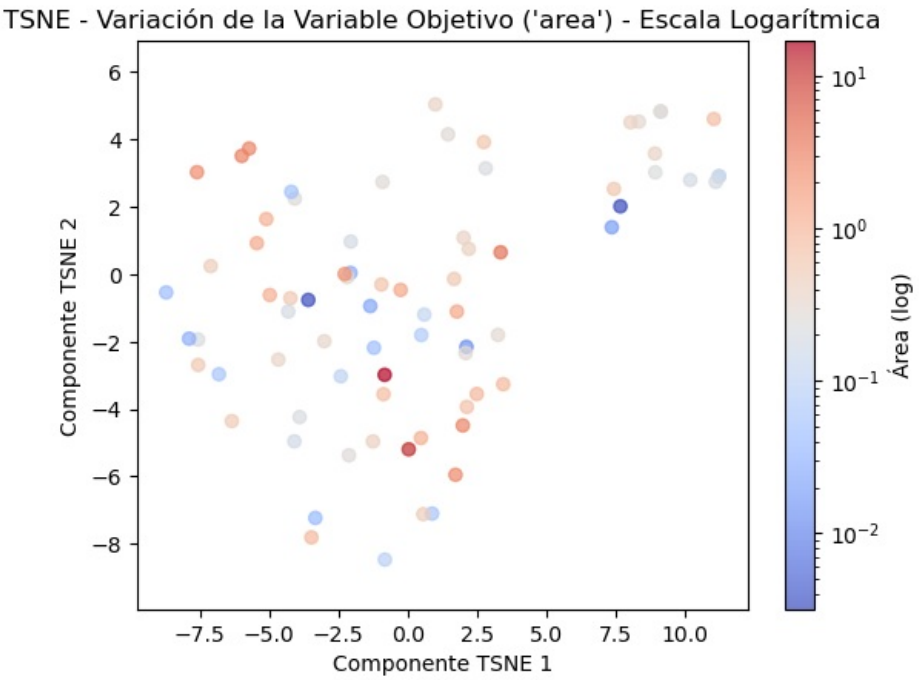
Gráfico para componentes TSNE.

In [24]:
```
# Creamos el gráfico usando un mapa de colores para la variable objetivo 'area' con escala logarítmica.
scatter = plt.scatter(tsne_df['t-SNE1'], tsne_df['t-SNE2'], c=tsne_df['area'], cmap='coolwarm', norm=LogNorm(),

# Agregamos barra de colores a modo de leyenda para la escala logarítmica de 'area'.
plt.colorbar(scatter, label='Área (log)', orientation='vertical')

# Etiquetas de los ejes y título del gráfico.
plt.xlabel('Componente TSNE 1')
plt.ylabel('Componente TSNE 2')
plt.title('TSNE - Variación de la Variable Objetivo (\'area\') - Escala Logarítmica')

# Mostramos el gráfico.
plt.show()
```



¿Por qué obtenemos resultados tan diferentes con los dos métodos?

La reducción de dimensionalidad ha funcionado, en ambos casos podemos diferenciar debidamente auqellos puntos que corresponden a las áreas de incendio grandes o pequeñas, gracias a la escala de color y que en ninguno de los casos los puntos han quedado demasiado agrupados en una misma zona al extremo de no poder diferenciarlos entre sí porque caigan todos en una misma región muy reducida.

Sin embargo, no apreciamos ningún patrón que diferencie dos regiones (o dos cúmulos/*'clusters'*) suficientemente marcadas de puntos con áreas de bosque quemadas grandes respecto de las pequeñas, es decir, en ambos casos tenemos ambos tipos de muestra bastante mezclados entre sí. Puede que hayamos cometido algún error del que no somos conscientes o estemos fallando a la interpretación de los gráficos, pero en nuestra opinión, en este caso, las técnicas *PCA* o *TSNE* no serían suficiente para predecir el valor de la variable objetivo.

Creemos que esto, en cierto modo es lógico, ya que las correlaciones con la variable objetivo tampoco eran notorias. Recordemos que en el mejor de los casos 'temp' y 'area' guardaban una correlación de 0.0978, o lo que es lo mismo, una correlación de sólo el 10% redondeando en el mejor de los casos. Esto sólo para una de las variables (*'temp'*), mientras que el resto tienen una correlación incluso menor con *'area'*.

De todos modos, consideramos que el método *TSNE*, al no ser un método lineal, es más adecuado para este caso ya que los puntos del gráfico son más fáciles de diferenciar y podemos interpretar mejor los valores de *'area'*, hemos probado con distintos valores para *perplexity* y *learning_rate* y creemos que puede haber alguna forma de encontrar los valores óptimos para estos parámetros de modo que podamos obtener (más o menos) 2 *clusters* bien diferenciados de puntos rojos (valores grandes de *'area'*) y azules (valores pequeños de *'area'*).

La diferencia entre ambos métodos es que, mientras *PCA* es una técnica lineal que busca las combinaciones lineales de las variables originales que maximizen la varianza del conjunto de datos, *TSNE* es un método no lineal que se centra en mantener la similitud local entre los puntos. De ahí que los gráficos sean distintos, quizá por ello el método *PCA* no sea el más adecuado para nuestro caso ya que estamos trabajando con la variable objetivo *'area'* en escala logarítmica mientras que el método (*PCA*) es lienal.