In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
data = pd.read_csv('forestfires.csv')
```

In [3]:

```
X = data.drop(['month', 'day', 'area'], axis=1).as_matrix()
y = data.area.values

def ind(x):
    if x in ['jun','jul','aug']:
        return 1
    else:
        return 0
X = np.concatenate([X, np.array(map(ind, data.month.values)).reshape(-1, 1),
np.ones(len(X)).reshape(-1, 1)], axis=1)
print X.shape
```

```
(517, 12)
```

In [16]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE

# this function calculates the estimation for y and returns the root MSE loss
def quolity(X=X, y=y, transform=False, c=1):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    if transform:
        y_train = np.log(c + y_train)
        y_test = np.log(c + y_test)
    W = np.dot(np.linalg.inv(np.dot(X_train.transpose(), X_train)), np.dot(X_tra
in.transpose(), y_train))
    y_est = np.dot(X_test, W)
    if transform:
        return np.sqrt(MSE(np.exp(y_est), np.exp(y_test)))
    else:
        return np.sqrt(MSE(y_est, y_test))
```

In [7]:

```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (15, 10)

# this function collects the results of N experiments (with different samples sp
lits) and prints the results
def exp_range(N=200, transform=False, c=1, show=True):
    results = [quolity(transform=transform, c=c) for i in range(N)]
    if show:
        print "average = ", np.mean(results)
        plt.plot(range(N), results)
        plt.xlabel('number of experimet', fontsize=20)
        plt.ylabel('RMSE', fontsize=20)
        plt.show()
    return np.mean(results)
```

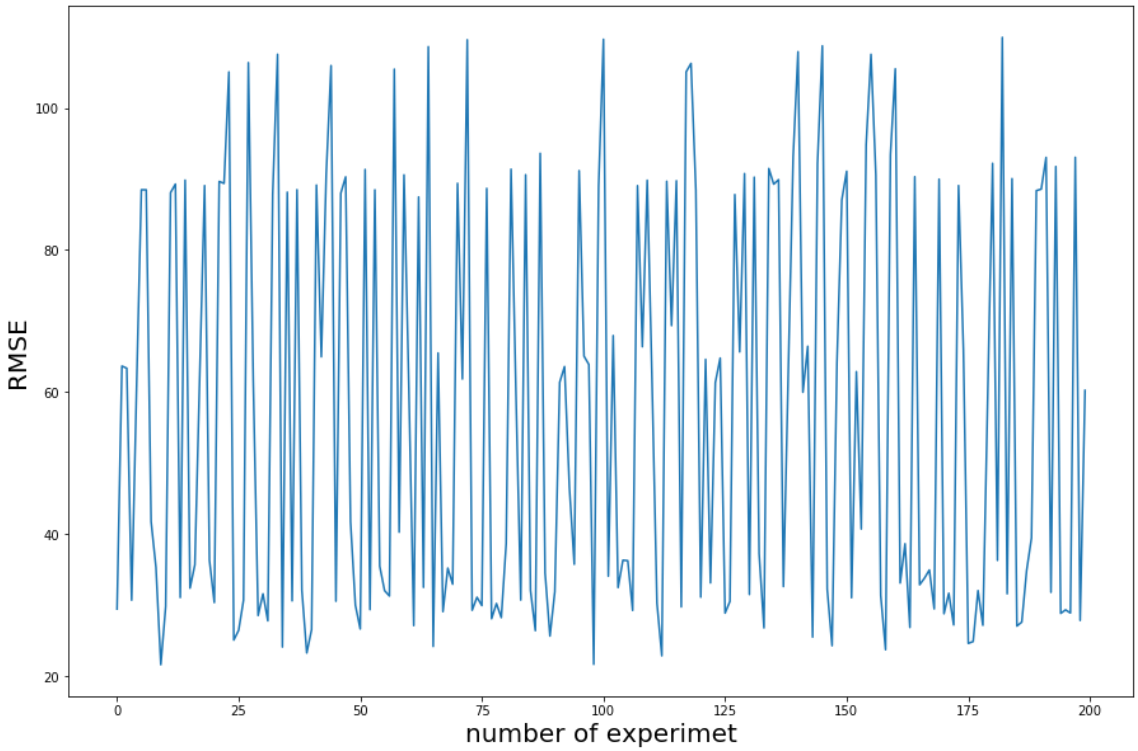
Считаем оценки:

Без преобразования area:

In [6]:

```
exp_range()
```

```
average =  57.4711360369
```



Видим, что результат сильно зависит от разбиения данных, следовательно, для получения достоверной оценки качества нужно брать много примеров.

In [38]:

```
exp_range(N=1000, show=False)
```

Out[38]:

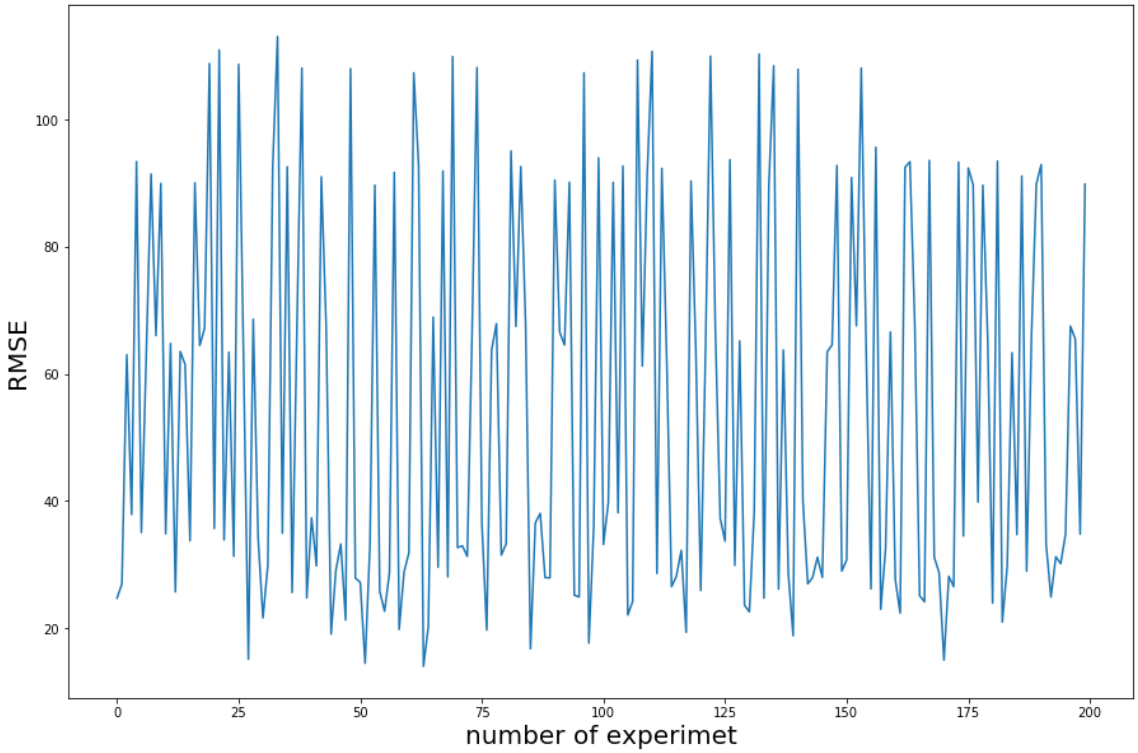
56.748701446137176

С преобразованием area:

In [17]:

```
exp_range(transform=True)
```

```
average =  54.5923327944
```



Out[17]:

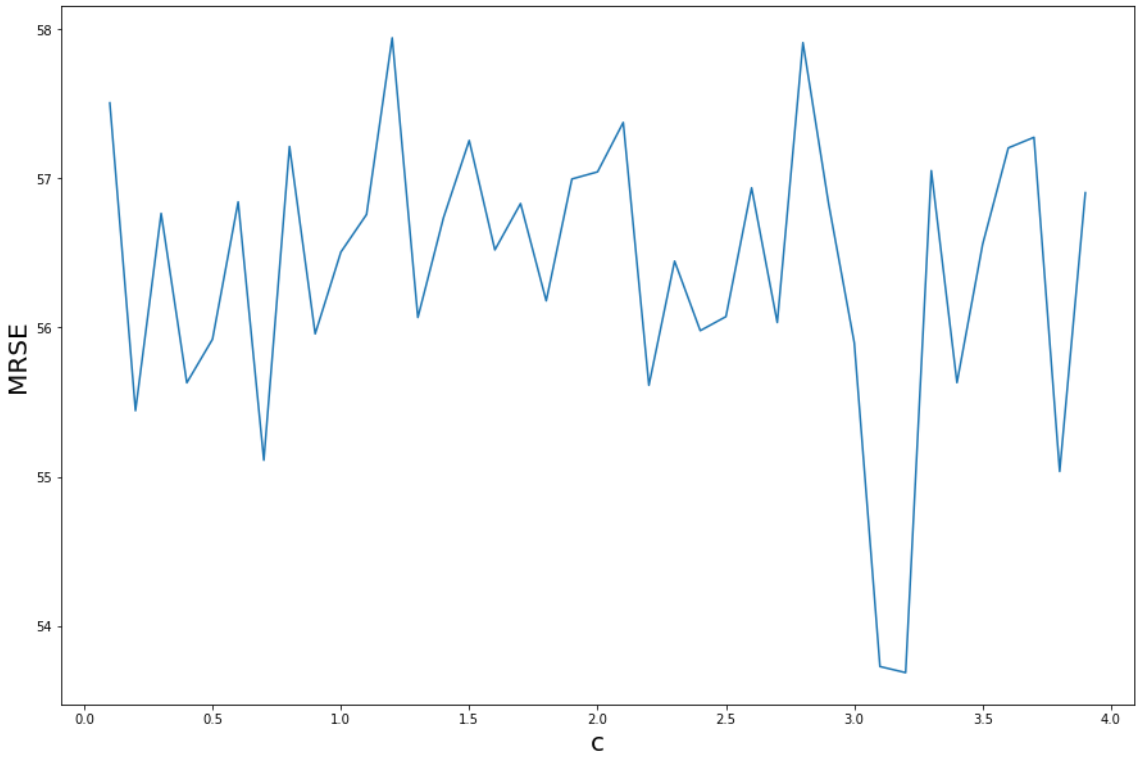
54.592332794366754

Результат так же сильно зависит от разбиения, оценка чуть лучше. Займемся подбором константы c.

In [35]:

```
c_range = np.arange(0.1, 4, 0.1)
scores = []
for c in c_range:
    scores.append(exp_range(N=1000, transform=True, c=c, show=False))
print 'c: ', c_range[np.argmin(scores)], 'MRSE: ', np.min(scores)
plt.plot(c_range, scores)
plt.xlabel('c', fontsize=20)
plt.ylabel('MRSE', fontsize=20)
plt.show()
```

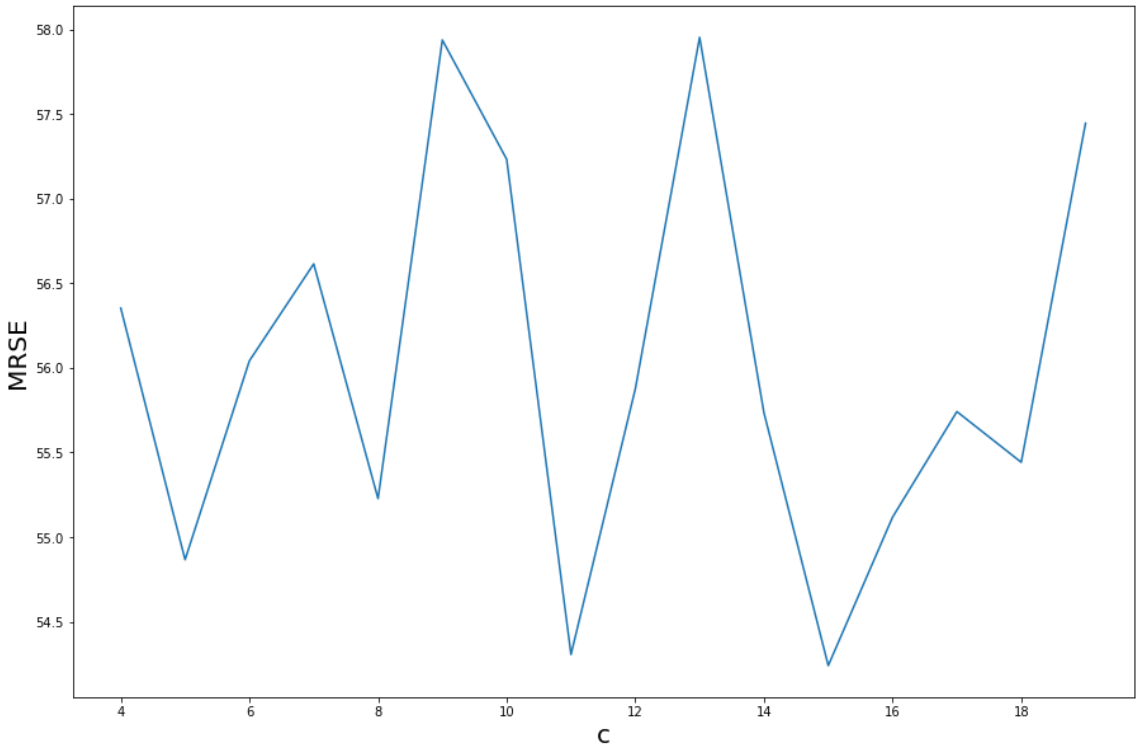
```
c:  3.2 MRSE:  53.6875779046
```



In [36]:

```
c_range = np.arange(4, 20, 1)
scores = []
for c in c_range:
    scores.append(exp_range(N=1000, transform=True, c=c, show=False))
print 'c: ', c_range[np.argmin(scores)], 'MRSE: ', np.min(scores)
plt.plot(c_range, scores)
plt.xlabel('c', fontsize=20)
plt.ylabel('MRSE', fontsize=20)
plt.show()
```

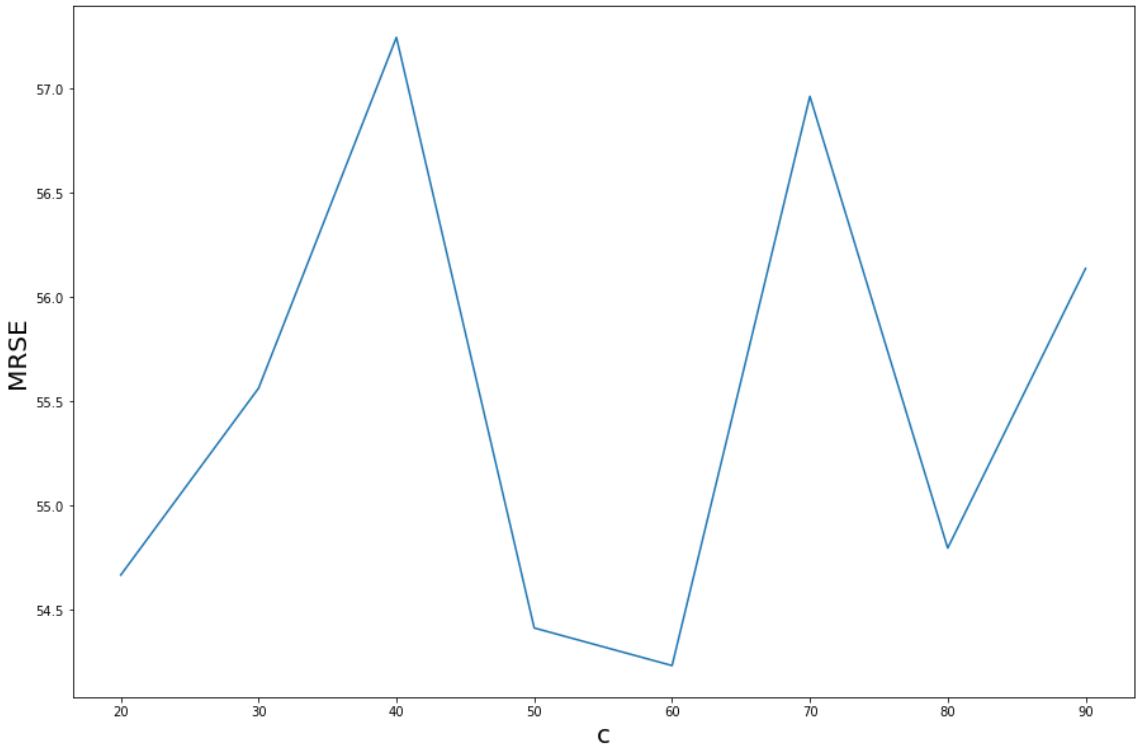
```
c:  15 MRSE:  54.2406726069
```



In [37]:

```
c_range = np.arange(20, 100, 10)
scores = []
for c in c_range:
    scores.append(exp_range(N=1000, transform=True, c=c, show=False))
print 'c: ', c_range[np.argmin(scores)], 'MRSE: ', np.min(scores)
plt.plot(c_range, scores)
plt.xlabel('c', fontsize=20)
plt.ylabel('MRSE', fontsize=20)
plt.show()
```

```
c:  60 MRSE:  54.2338538843
```



Оптимально брать c=3.2, при этом MRSE = 53.7, то есть качество восстановления выше, чем без преобразования y.

In [ ]:

```

```