## 7.2.2. Обработка данных: стандартные алгоритмы машинного обучения в Julia

### 7.2.2.1. Кластеризация данных. Метод k-средних

In [1]:

```
# Загрузка пакетов:
#import Pkg
#Pkg.add("DataFrames")
#Pkg.add("Statistics")
using DataFrames
using CSV
#import Pkg
#Pkg.add("Plots")
```

In [2]:

```
# Загрузка данных:
houses = CSV.File("houses.csv") |> DataFrame
```

Out[2]: 985 rows × 12 columns (omitted printing of 5 columns)

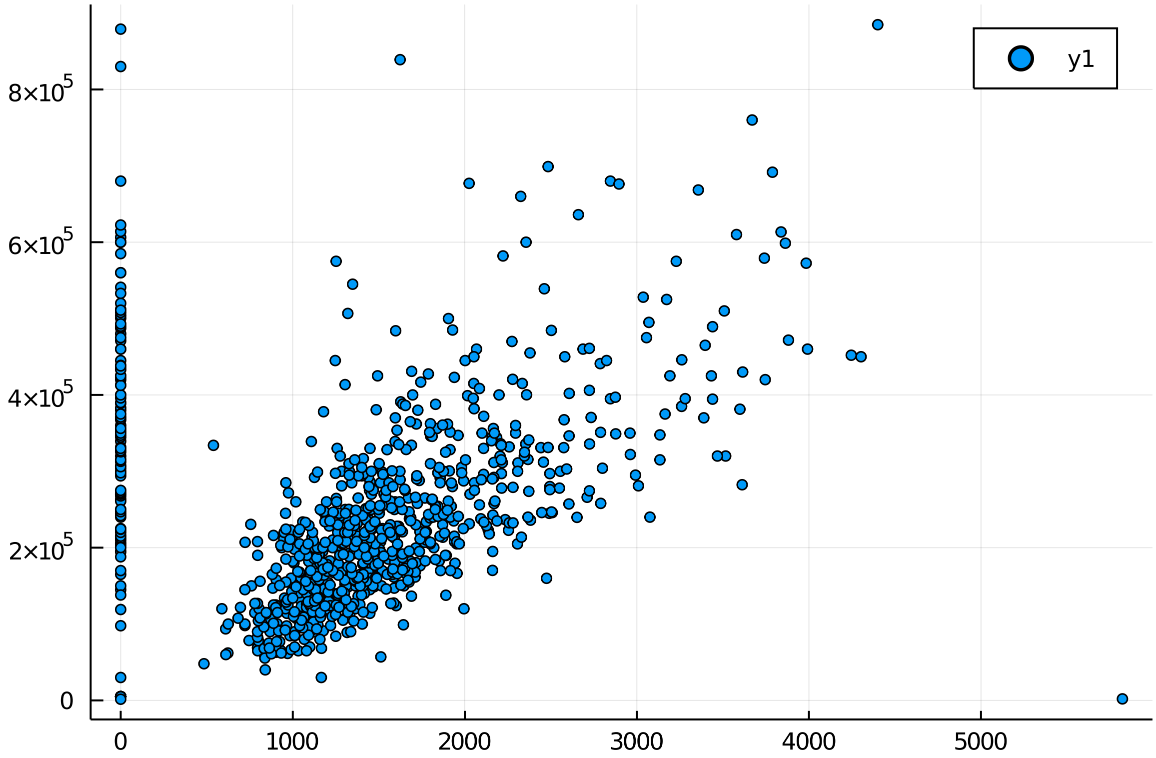
| | street | city | zip | state | beds | baths | sq__ft |
|---|---|---|---|---|---|---|---|
| | String | String | Int64 | String | Int64 | Int64 | Int64 |
| 1 | 3526 HIGH ST | SACRAMENTO | 95838 | CA | 2 | 1 | 836 |
| 2 | 51 OMAHA CT | SACRAMENTO | 95823 | CA | 3 | 1 | 1167 |
| 3 | 2796 BRANCH ST | SACRAMENTO | 95815 | CA | 2 | 1 | 796 |
| 4 | 2805 JANETTE WAY | SACRAMENTO | 95815 | CA | 2 | 1 | 852 |
| 5 | 6001 MCMAHON DR | SACRAMENTO | 95824 | CA | 2 | 1 | 797 |
| 6 | 5828 PEPPERMILL CT | SACRAMENTO | 95841 | CA | 3 | 1 | 1122 |
| 7 | 6048 OGDEN NASH WAY | SACRAMENTO | 95842 | CA | 3 | 2 | 1104 |
| 8 | 2561 19TH AVE | SACRAMENTO | 95820 | CA | 3 | 1 | 1177 |
| 9 | 11150 TRINITY RIVER DR Unit 114 | RANCHO CORDOVA | 95670 | CA | 2 | 2 | 941 |
| 10 | 7325 10TH ST | RIO LINDA | 95673 | CA | 3 | 2 | 1146 |
| 11 | 645 MORRISON AVE | SACRAMENTO | 95838 | CA | 3 | 2 | 909 |
| 12 | 4085 FAWN CIR | SACRAMENTO | 95823 | CA | 3 | 2 | 1289 |
| 13 | 2930 LA ROSA RD | SACRAMENTO | 95815 | CA | 1 | 1 | 871 |
| 14 | 2113 KIRK WAY | SACRAMENTO | 95822 | CA | 3 | 1 | 1020 |
| 15 | 4533 LOCH HAVEN WAY | SACRAMENTO | 95842 | CA | 2 | 2 | 1022 |
| 16 | 7340 HAMDEN PL | SACRAMENTO | 95842 | CA | 2 | 2 | 1134 |
| 17 | 6715 6TH ST | RIO LINDA | 95673 | CA | 2 | 1 | 844 |
| 18 | 6236 LONGFORD DR Unit 1 | CITRUS HEIGHTS | 95621 | CA | 2 | 1 | 795 |
| 19 | 250 PERALTA AVE | SACRAMENTO | 95833 | CA | 2 | 1 | 588 |
| 20 | 113 LEEWILL AVE | RIO LINDA | 95673 | CA | 3 | 2 | 1356 |
| 21 | 6118 STONEHAND AVE | CITRUS HEIGHTS | 95621 | CA | 3 | 2 | 1118 |
| 22 | 4882 BANDALIN WAY | SACRAMENTO | 95823 | CA | 4 | 2 | 1329 |
| 23 | 7511 OAKVALE CT | NORTH HIGHLANDS | 95660 | CA | 4 | 2 | 1240 |
| 24 | 9 PASTURE CT | SACRAMENTO | 95834 | CA | 3 | 2 | 1601 |
| 25 | 3729 BAINBRIDGE DR | NORTH HIGHLANDS | 95660 | CA | 3 | 2 | 901 |
| 26 | 3828 BLACKFOOT WAY | ANTELOPE | 95843 | CA | 3 | 2 | 1088 |
| 27 | 4108 NORTON WAY | SACRAMENTO | 95820 | CA | 3 | 1 | 963 |
| 28 | 1469 JANRICK AVE | SACRAMENTO | 95832 | CA | 3 | 2 | 1119 |
| 29 | 9861 CULP WAY | SACRAMENTO | 95827 | CA | 4 | 2 | 1380 |
| 30 | 7825 CREEK VALLEY CIR | SACRAMENTO | 95828 | CA | 3 | 2 | 1248 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [3]:

```
# Построение графика:
using Plots
plot(size=(500,500),leg=false)
```

```
x = houses[:sq__ft]
y = houses[:price]
scatter(x,y,markersize=3)
```

Out[3]:




In [4]:

```
# Фильтрация данных по заданному условию:
filter_houses = houses[houses[!,:sq__ft].>0,:]
```

Out[4]: 814 rows × 12 columns (omitted printing of 5 columns)

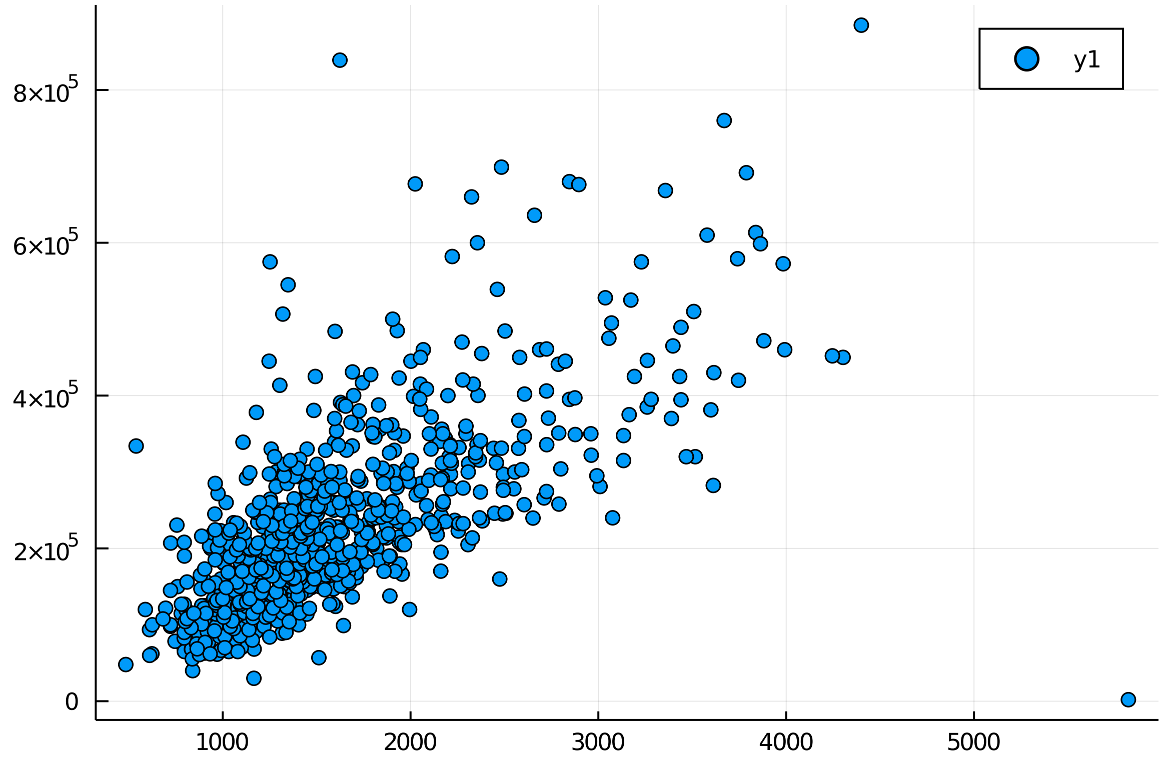
| | street | city | zip | state | beds | baths | sq__ft |
|---|---|---|---|---|---|---|---|
| | String | String | Int64 | String | Int64 | Int64 | Int64 |
| 1 | 3526 HIGH ST | SACRAMENTO | 95838 | CA | 2 | 1 | 836 |
| 2 | 51 OMAHA CT | SACRAMENTO | 95823 | CA | 3 | 1 | 1167 |
| 3 | 2796 BRANCH ST | SACRAMENTO | 95815 | CA | 2 | 1 | 796 |
| 4 | 2805 JANETTE WAY | SACRAMENTO | 95815 | CA | 2 | 1 | 852 |
| 5 | 6001 MCMAHON DR | SACRAMENTO | 95824 | CA | 2 | 1 | 797 |
| 6 | 5828 PEPPERMILL CT | SACRAMENTO | 95841 | CA | 3 | 1 | 1122 |
| 7 | 6048 OGDEN NASH WAY | SACRAMENTO | 95842 | CA | 3 | 2 | 1104 |
| 8 | 2561 19TH AVE | SACRAMENTO | 95820 | CA | 3 | 1 | 1177 |
| 9 | 11150 TRINITY RIVER DR Unit 114 | RANCHO CORDOVA | 95670 | CA | 2 | 2 | 941 |
| 10 | 7325 10TH ST | RIO LINDA | 95673 | CA | 3 | 2 | 1146 |
| 11 | 645 MORRISON AVE | SACRAMENTO | 95838 | CA | 3 | 2 | 909 |
| 12 | 4085 FAWN CIR | SACRAMENTO | 95823 | CA | 3 | 2 | 1289 |
| 13 | 2930 LA ROSA RD | SACRAMENTO | 95815 | CA | 1 | 1 | 871 |
| 14 | 2113 KIRK WAY | SACRAMENTO | 95822 | CA | 3 | 1 | 1020 |
| 15 | 4533 LOCH HAVEN WAY | SACRAMENTO | 95842 | CA | 2 | 2 | 1022 |
| 16 | 7340 HAMDEN PL | SACRAMENTO | 95842 | CA | 2 | 2 | 1134 |
| 17 | 6715 6TH ST | RIO LINDA | 95673 | CA | 2 | 1 | 844 |
| 18 | 6236 LONGFORD DR Unit 1 | CITRUS HEIGHTS | 95621 | CA | 2 | 1 | 795 |
| 19 | 250 PERALTA AVE | SACRAMENTO | 95833 | CA | 2 | 1 | 588 |
| 20 | 113 LEEWILL AVE | RIO LINDA | 95673 | CA | 3 | 2 | 1356 |
| 21 | 6118 STONEHAND AVE | CITRUS HEIGHTS | 95621 | CA | 3 | 2 | 1118 |
| 22 | 4882 BANDALIN WAY | SACRAMENTO | 95823 | CA | 4 | 2 | 1329 |
| 23 | 7511 OAKVALE CT | NORTH HIGHLANDS | 95660 | CA | 4 | 2 | 1240 |
| 24 | 9 PASTURE CT | SACRAMENTO | 95834 | CA | 3 | 2 | 1601 |
| 25 | 3729 BAINBRIDGE DR | NORTH HIGHLANDS | 95660 | CA | 3 | 2 | 901 |
| 26 | 3828 BLACKFOOT WAY | ANTELOPE | 95843 | CA | 3 | 2 | 1088 |
| 27 | 4108 NORTON WAY | SACRAMENTO | 95820 | CA | 3 | 1 | 963 |
| 28 | 1469 JANRICK AVE | SACRAMENTO | 95832 | CA | 3 | 2 | 1119 |
| 29 | 9861 CULP WAY | SACRAMENTO | 95827 | CA | 4 | 2 | 1380 |
| 30 | 7825 CREEK VALLEY CIR | SACRAMENTO | 95828 | CA | 3 | 2 | 1248 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [5]:

```
# Построение графика:
x = filter_houses[:sq__ft]
```

```
y = filter_houses[:price]
scatter(x,y)
```

Out[5]:




In [6]:

```
# Подключение пакета Statistics:
using Statistics
```

In [7]:

```
# Определение средней цены для определённого типа домов:
by(filter_houses,:type,filter_houses->mean(filter_houses[:price]))
```

Out[7]: 3 rows × 2 columns

| | type | x1 |
|---|---|---|
| | String | Float64 |
| **1** | Residential | 2.34802e5 |
| **2** | Condo | 1.34213e5 |
| **3** | Multi-Family | 2.24535e5 |

In [8]:

```
# Подключение пакета Clustering:
#import Pkg
#Pkg.add("Clustering")
using Clustering
```

In [9]:

```
# Добавление данных :latitude и :longitude в новый фрейм:
X = filter_houses[[:latitude,:longitude]]
```

Out[9]: 814 rows × 2 columns

| | latitude | longitude |
|---|---|---|
| | Float64 | Float64 |
| 1 | 38.6319 | -121.435 |
| 2 | 38.4789 | -121.431 |
| 3 | 38.6183 | -121.444 |
| 4 | 38.6168 | -121.439 |
| 5 | 38.5195 | -121.436 |
| 6 | 38.6626 | -121.328 |
| 7 | 38.6817 | -121.352 |
| 8 | 38.5351 | -121.481 |
| 9 | 38.6212 | -121.271 |
| 10 | 38.7009 | -121.443 |
| 11 | 38.6377 | -121.452 |
| 12 | 38.4707 | -121.459 |
| 13 | 38.6187 | -121.436 |
| 14 | 38.4822 | -121.493 |
| 15 | 38.6729 | -121.359 |
| 16 | 38.7001 | -121.351 |
| 17 | 38.6896 | -121.452 |
| 18 | 38.6798 | -121.314 |
| 19 | 38.6121 | -121.469 |
| 20 | 38.69 | -121.463 |
| 21 | 38.7079 | -121.321 |
| 22 | 38.4682 | -121.444 |
| 23 | 38.7028 | -121.382 |
| 24 | 38.6286 | -121.488 |
| 25 | 38.7015 | -121.376 |
| 26 | 38.7097 | -121.374 |
| 27 | 38.5375 | -121.478 |
| 28 | 38.4765 | -121.502 |
| 29 | 38.5584 | -121.328 |
| 30 | 38.4721 | -121.404 |
| ⋮ | ⋮ | ⋮ |

In [10]:

```
# Конвертация данных в матричный вид:
X = convert(Matrix{Float64}, X)
```

Out[10]:
```
814×2 Array{Float64,2}:
 38.6319  -121.435
 38.4789  -121.431
 38.6183  -121.444
 38.6168  -121.439
 38.5195  -121.436
 38.6626  -121.328
 38.6817  -121.352
 38.5351  -121.481
 38.6212  -121.271
 38.7009  -121.443
 38.6377  -121.452
 38.4707  -121.459
 38.6187  -121.436
  ⋮
 38.7035  -121.375
 38.7031  -121.235
 38.3898  -121.446
 38.8978  -121.325
 38.4679  -121.445
 38.4453  -121.442
 38.4174  -121.484
 38.4577  -121.36
 38.4999  -121.459
 38.7088  -121.257
 38.417   -121.397
 38.6552  -121.076
```

In [11]:
```
# Транспонирование матрицы с данными:
X = X'
```

Out[11]:
```
2×814 LinearAlgebra.Adjoint{Float64,Array{Float64,2}}:
   38.6319    38.4789    38.6183  …    38.7088    38.417     38.6552
 -121.435   -121.431   -121.444      -121.257   -121.397   -121.076
```

In [12]:
```
# Задание количества кластеров:
k = length(unique(filter_houses[:zip]))
```

Out[12]:
```
66
```

In [13]:
```
# Определение k-среднего:
C = kmeans(X,k) # попробуйте поменять k
```

Out[13]:
```
KmeansResult{Array{Float64,2},Float64,Int64}([38.680849300000006 38.28410599999999
4 … 38.51819435714285 38.251295999999996; -121.36293816666664 -121.29519644444443
… -121.49720271428573 -121.31266899999999], [35, 47, 35, 35, 10, 41, 1, 46, 60, 45
…  27, 36, 47, 62, 19, 61, 47, 23, 7, 28], [6.273316466831602e-5, 0.00050545454723
76049, 7.037215982563794e-5, 6.84302649460733e-5, 0.00012161941776867025, 0.000197
0666489796713, 0.00012683964814641513, 0.00023977080854820088, 1.144897032645531e-
5, 0.00011686658399412408  …  3.845047467621043e-5, 5.2413251978578046e-5, 0.00021
6632652154658, 0.00023574619626742788, 0.00013415837747743353, 0.00012178530232631
601, 0.0004134149457968306, 0.0005266233965812717, 7.109141370165162e-5, 0.0014500
179277092684], [30, 9, 5, 25, 1, 1, 10, 13, 5, 22  …  15, 18, 2, 9, 34, 36, 1, 1,
14, 6], [30, 9, 5, 25, 1, 1, 10, 13, 5, 22  …  15, 18, 2, 9, 34, 36, 1, 1, 14, 6],
0.2148637686877919, 15, true)
```

In [14]:
```
# Формирование фрейма данных:
df = DataFrame(cluster = C.assignments,city = filter_houses[:city],
    latitude = filter_houses[:latitude],longitude = filter_houses[:longitude],zip
```

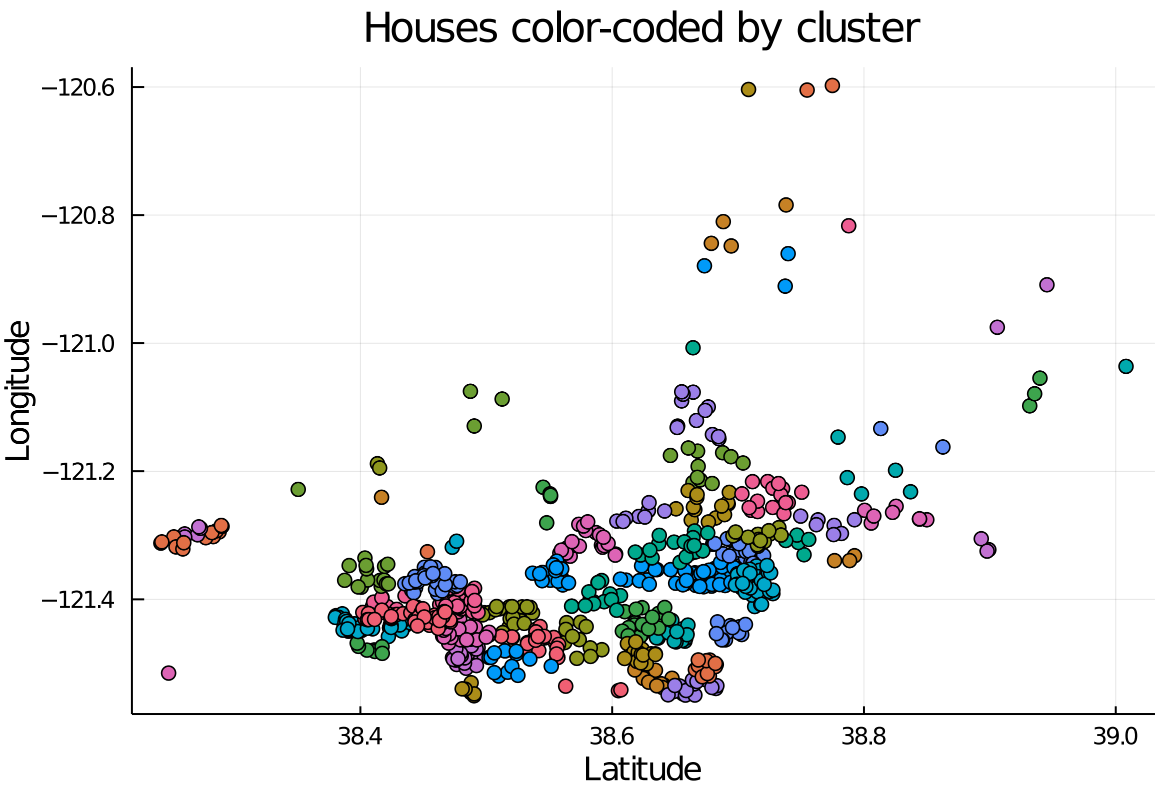
Out[14]: 814 rows × 5 columns

| | cluster | city | latitude | longitude | zip |
|---|---|---|---|---|---|
| | Int64 | String | Float64 | Float64 | Int64 |
| 1 | 35 | SACRAMENTO | 38.6319 | -121.435 | 95838 |
| 2 | 47 | SACRAMENTO | 38.4789 | -121.431 | 95823 |
| 3 | 35 | SACRAMENTO | 38.6183 | -121.444 | 95815 |
| 4 | 35 | SACRAMENTO | 38.6168 | -121.439 | 95815 |
| 5 | 10 | SACRAMENTO | 38.5195 | -121.436 | 95824 |
| 6 | 41 | SACRAMENTO | 38.6626 | -121.328 | 95841 |
| 7 | 1 | SACRAMENTO | 38.6817 | -121.352 | 95842 |
| 8 | 46 | SACRAMENTO | 38.5351 | -121.481 | 95820 |
| 9 | 60 | RANCHO CORDOVA | 38.6212 | -121.271 | 95670 |
| 10 | 45 | RIO LINDA | 38.7009 | -121.443 | 95673 |
| 11 | 22 | SACRAMENTO | 38.6377 | -121.452 | 95838 |
| 12 | 47 | SACRAMENTO | 38.4707 | -121.459 | 95823 |
| 13 | 35 | SACRAMENTO | 38.6187 | -121.436 | 95815 |
| 14 | 4 | SACRAMENTO | 38.4822 | -121.493 | 95822 |
| 15 | 1 | SACRAMENTO | 38.6729 | -121.359 | 95842 |
| 16 | 1 | SACRAMENTO | 38.7001 | -121.351 | 95842 |
| 17 | 45 | RIO LINDA | 38.6896 | -121.452 | 95673 |
| 18 | 13 | CITRUS HEIGHTS | 38.6798 | -121.314 | 95621 |
| 19 | 53 | SACRAMENTO | 38.6121 | -121.469 | 95833 |
| 20 | 45 | RIO LINDA | 38.69 | -121.463 | 95673 |
| 21 | 13 | CITRUS HEIGHTS | 38.7079 | -121.321 | 95621 |
| 22 | 47 | SACRAMENTO | 38.4682 | -121.444 | 95823 |
| 23 | 43 | NORTH HIGHLANDS | 38.7028 | -121.382 | 95660 |
| 24 | 53 | SACRAMENTO | 38.6286 | -121.488 | 95834 |
| 25 | 43 | NORTH HIGHLANDS | 38.7015 | -121.376 | 95660 |
| 26 | 43 | ANTELOPE | 38.7097 | -121.374 | 95843 |
| 27 | 46 | SACRAMENTO | 38.5375 | -121.478 | 95820 |
| 28 | 4 | SACRAMENTO | 38.4765 | -121.502 | 95832 |
| 29 | 15 | SACRAMENTO | 38.5584 | -121.328 | 95827 |
| 30 | 55 | SACRAMENTO | 38.4721 | -121.404 | 95828 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [15]:

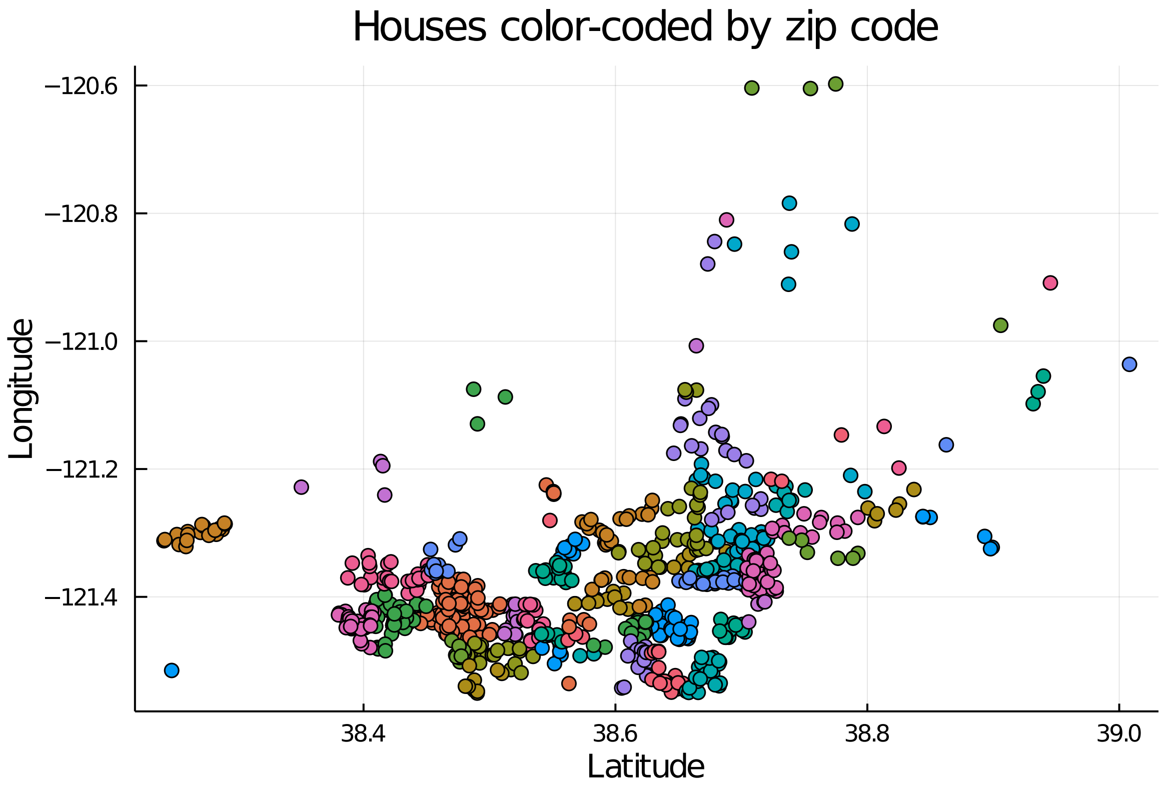
```
#Построим график, обозначив каждый кластер отдельным цветом:
clusters_figure = plot(legend = false)
for i = 1:k
```

```
    clustered_houses = df[df[:cluster].== i,:]
    xvals = clustered_houses[:latitude]
    yvals = clustered_houses[:longitude]
    scatter!(clusters_figure,xvals,yvals,markersize=4)
end
xlabel!("Latitude")
ylabel!("Longitude")
title!("Houses color-coded by cluster")
display(clusters_figure)
```




In [16]:

```
#Построим график, раскрасив кластеры по почтовому индексу:
unique_zips = unique(filter_houses[:zip])
zips_figure = plot(legend = false)
for uzip in unique_zips
    subs = filter_houses[filter_houses[:zip].==uzip,:]
    x = subs[:latitude]
    y = subs[:longitude]
    scatter!(zips_figure,x,y)
end
xlabel!("Latitude")
ylabel!("Longitude")
title!("Houses color-coded by zip code")
display(zips_figure)
```




7.2.2.2. Кластеризация данных. Метод k ближайших соседей

In [17]:
```
# Подключение пакета NearestNeighbors:
#import Pkg
#Pkg.add("NearestNeighbors")
using NearestNeighbors
```

In [18]:
```
#Найдём k-среднее одного из объектов недвижимости:
knearest = 10
id = 70
point = X[:,id]
```

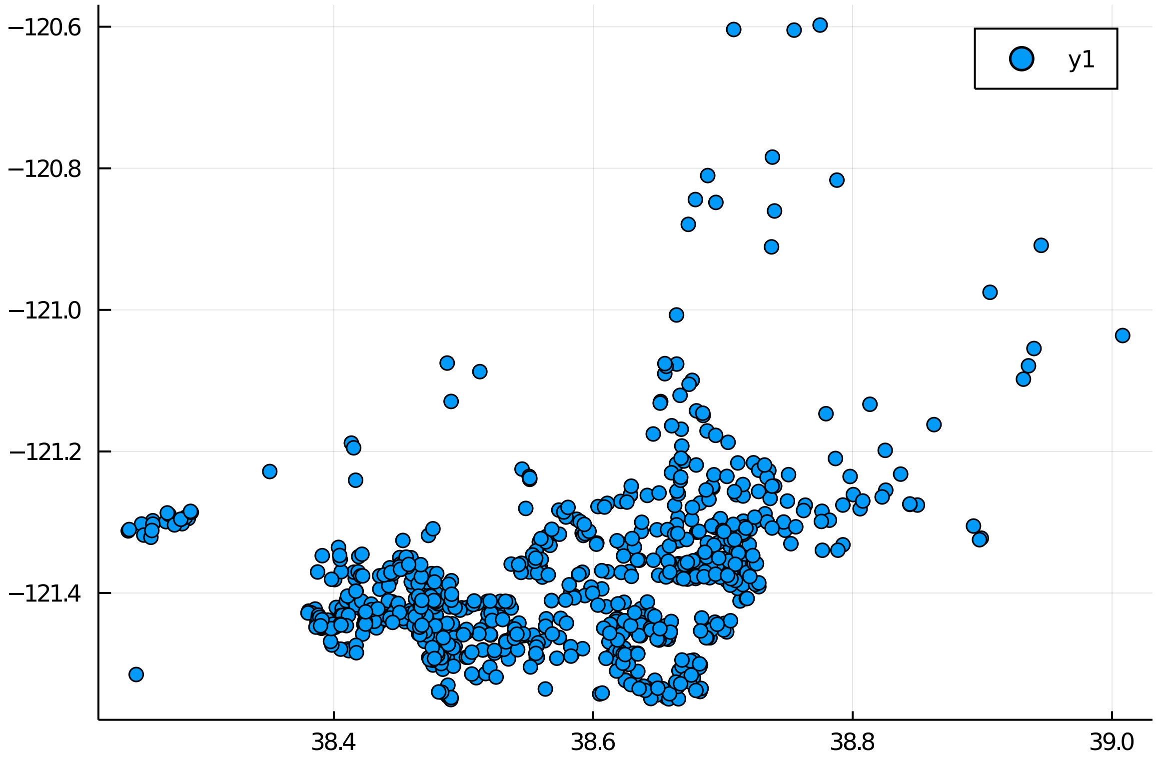
Out[18]:
```
2-element Array{Float64,1}:
   38.44004
 -121.421012
```

In [19]:
```
# Поиск ближайших соседей:
kdtree = KDTree(X)
idxs, dists = knn(kdtree, point, knearest, true)
```

Out[19]:
```
([70, 764, 196, 125, 557, 368, 415, 92, 112, 683], [0.0, 0.006264891539364138, 0.0
0825320259050462, 0.008473585132630057, 0.009164073548370188, 0.00940506512469770
6, 0.009921759722950759, 0.009941028618812013, 0.010332637707777167, 0.01116899391
1721985])
```

In [20]:
```
# Все объекты недвижимости:
x = filter_houses[!,:latitude];
y = filter_houses[!,:longitude];
scatter(x,y)
```

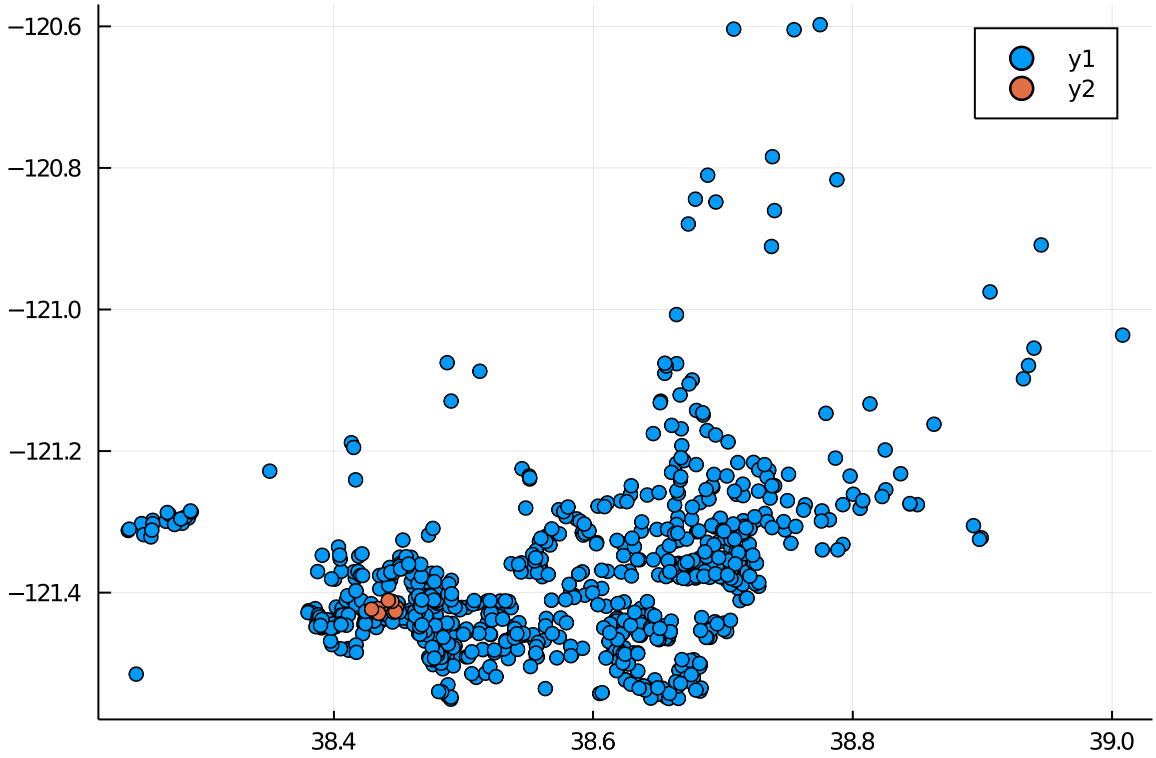
Out[20]:




In [21]:

```
# Соседи:
x = filter_houses[idxs,:latitude];
y = filter_houses[idxs,:longitude];
scatter!(x,y)
```

Out[21]:




In [22]:

```
# Фильтрация по районам соседних домов:
cities = filter_houses[idxs,:city]
```

Out[22]:

```
10-element Array{String,1}:
 "SACRAMENTO"
 "ELK GROVE"
 "SACRAMENTO"
 "SACRAMENTO"
 "SACRAMENTO"
 "SACRAMENTO"
 "ELK GROVE"
 "ELK GROVE"
 "ELK GROVE"
 "ELK GROVE"
```

7.2.2.3. Обработка данных. Метод главных компонент

In [23]:

```
# Фрейм с указанием площади и цены недвижимости:
F = filter_houses[[:sq__ft,:price]]
```

Out[23]: 814 rows × 2 columns

| | sq_ft | price |
|---|---|---|
| | Int64 | Int64 |
| 1 | 836 | 59222 |
| 2 | 1167 | 68212 |
| 3 | 796 | 68880 |
| 4 | 852 | 69307 |
| 5 | 797 | 81900 |
| 6 | 1122 | 89921 |
| 7 | 1104 | 90895 |
| 8 | 1177 | 91002 |
| 9 | 941 | 94905 |
| 10 | 1146 | 98937 |
| 11 | 909 | 100309 |
| 12 | 1289 | 106250 |
| 13 | 871 | 106852 |
| 14 | 1020 | 107502 |
| 15 | 1022 | 108750 |
| 16 | 1134 | 110700 |
| 17 | 844 | 113263 |
| 18 | 795 | 116250 |
| 19 | 588 | 120000 |
| 20 | 1356 | 121630 |
| 21 | 1118 | 122000 |
| 22 | 1329 | 122682 |
| 23 | 1240 | 123000 |
| 24 | 1601 | 124100 |
| 25 | 901 | 125000 |
| 26 | 1088 | 126640 |
| 27 | 963 | 127281 |
| 28 | 1119 | 129000 |
| 29 | 1380 | 131200 |
| 30 | 1248 | 132000 |
| ⋮ | ⋮ | ⋮ |

In [24]:

```
# Конвертация данных в массив:
F = convert(Array{Float64,2},F)'
```

Out[24]:
```
2×814 LinearAlgebra.Adjoint{Float64,Array{Float64,2}}:
   836.0    1167.0     796.0     852.0     797.0  …    1216.0     1685.0     1362.0
 59222.0   68212.0   68880.0   69307.0   81900.0     235000.0   235301.0   235738.0
```

In [25]:
```
# Подключение пакета MultivariateStats:
#import Pkg
#Pkg.add("MultivariateStats")
using MultivariateStats
```

In [26]:
```
# Приведение типов данных к распределению для PCA:
M = fit(PCA, F)
```

Out[26]:
```
PCA(indim = 2, outdim = 1, principalratio = 0.9999840784692097)
```

In [27]:
```
y = MultivariateStats.transform(M, F)
```

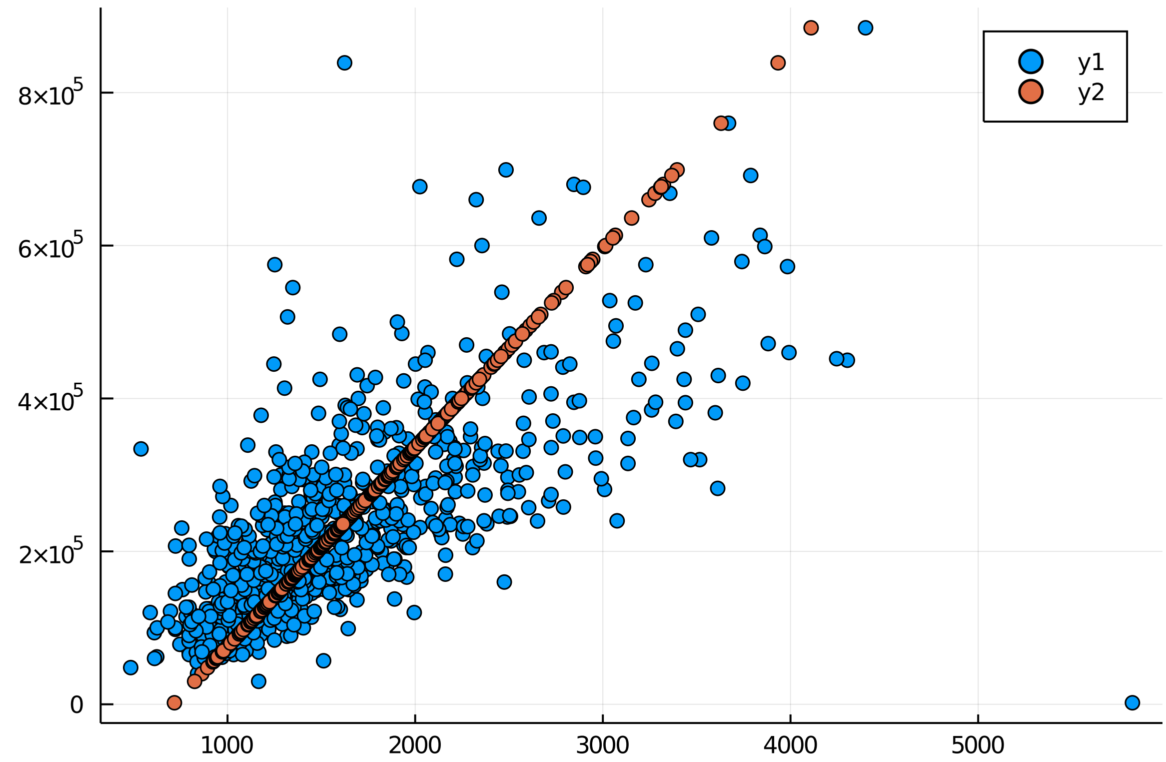
Out[27]:
```
1×814 Array{Float64,2}:
 -170228.0  -1.61237e5  -1.6057e5  …  4551.16  5550.15  5852.95  6288.7
```

In [28]:
```
# Выделение значений главных компонент в отдельную переменную:
Xr = reconstruct(M, y)
```

Out[28]:
```
2×814 Array{Float64,2}:
   936.922    971.477    974.039    975.681  …  1613.64          1615.32
 59221.6    68212.8    68879.3    69306.5          2.35301e5  235737.0
```

In [29]:
```
# Построение графика с выделением главных компонент:
scatter(F[1,:],F[2,:])
scatter!(Xr[1,:],Xr[2,:])
```

Out[29]:

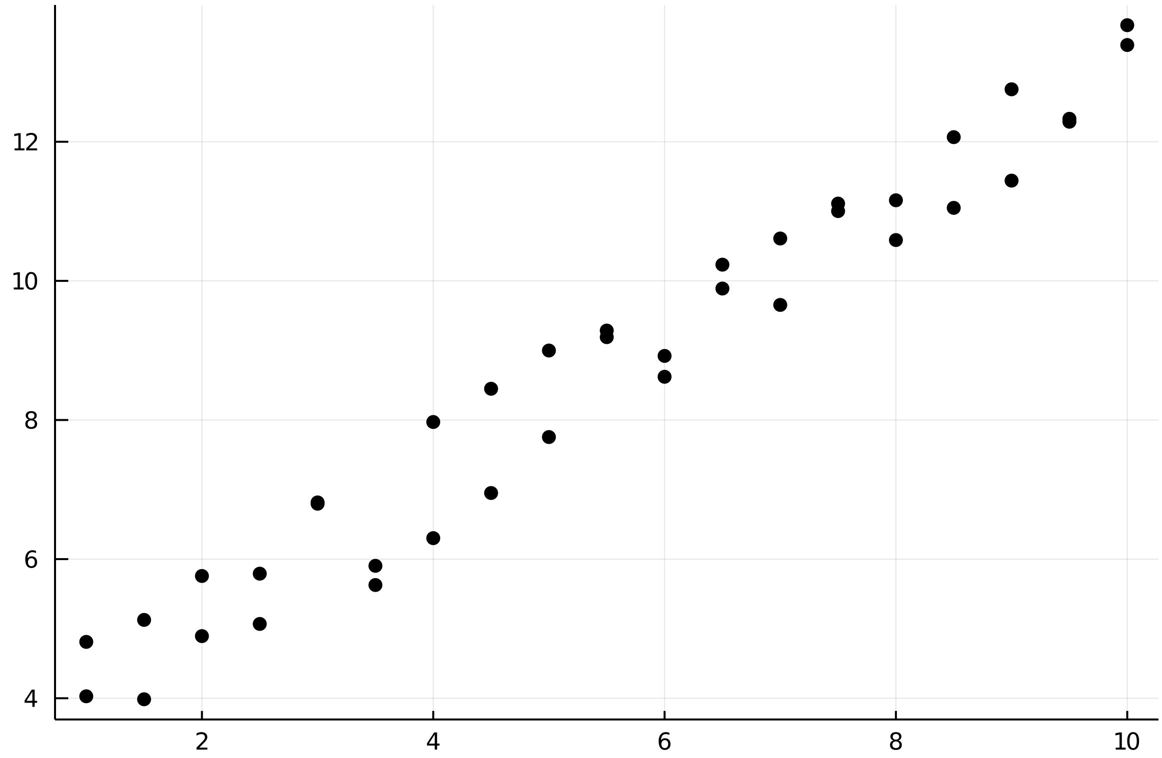



7.2.2.4. Обработка данных. Линейная регрессия

In [30]:
```
#Зададим случайный набор данных. Попробуем найти для данных лучшее соответствие:
```

```
xvals = repeat(1:0.5:10,inner=2)
yvals = 3 .+ xvals + 2*rand(length(xvals)) .- 1
scatter(xvals,yvals,color=:black,leg=false)
```

Out[30]:




In [31]:

```
#Определим функцию линейной регрессии:
function find_best_fit(xvals,yvals)
    meanx = mean(xvals)
    meany = mean(yvals)
    stdx = std(xvals)
    stdy = std(yvals)
    r = cor(xvals,yvals)
    a = r*stdy/stdx
    b = meany - a*meanx
    return a,b
end
```

Out[31]: find_best_fit (generic function with 1 method)

In [32]:

```
#Применим функцию линейной регрессии для построения соответствующего графика значе
a,b = find_best_fit(xvals,yvals)
ynew = a * xvals .+ b
```

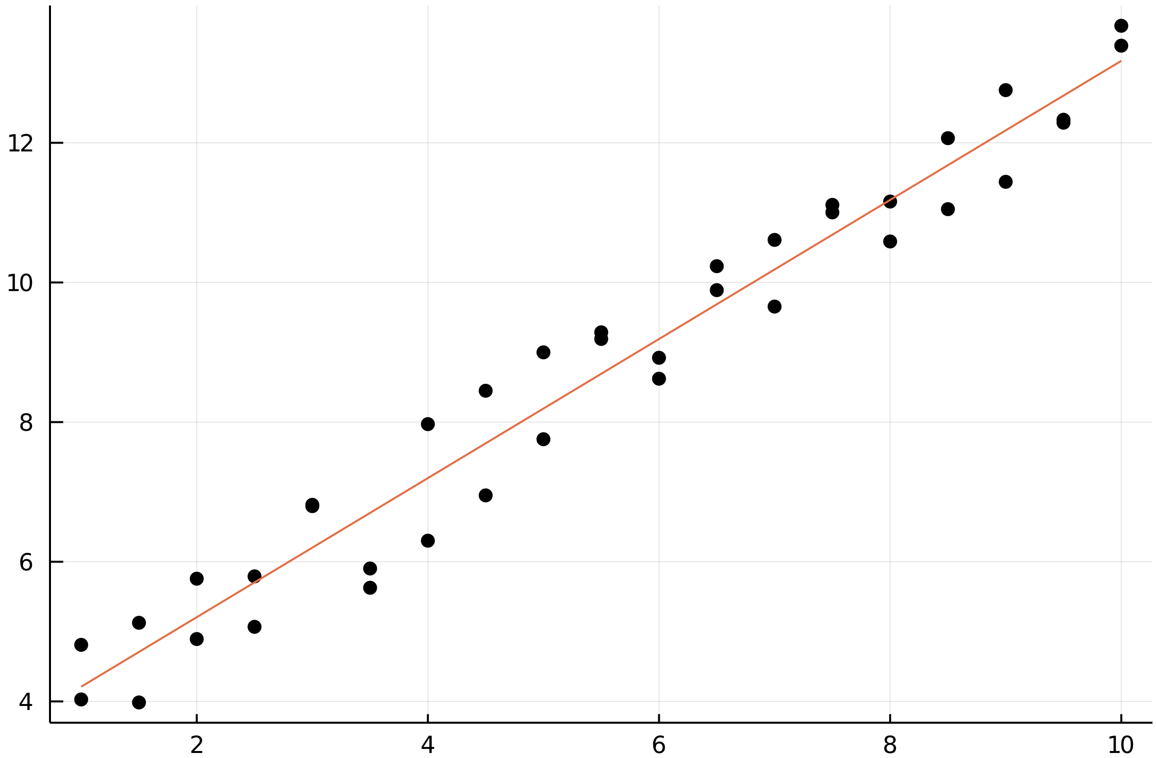
Out[32]:

```
38-element Array{Float64,1}:
  4.210197911424078
  4.210197911424078
  4.708165370535329
  4.708165370535329
  5.206132829646579
  5.206132829646579
  5.7041002887578305
  5.7041002887578305
  6.20206774786908
  6.20206774786908
  6.700035206980331
  6.700035206980331
  7.198002666091581
  ⋮
 10.683774879870334
 10.683774879870334
 11.181742338981586
 11.181742338981586
 11.679709798092835
 11.679709798092835
 12.177677257204087
 12.177677257204087
 12.675644716315336
 12.675644716315336
 13.173612175426587
 13.173612175426587
```

In [33]:

```
plot!(xvals,ynew)
```

Out[33]:




In [34]:

```
#Сгенерируем больший набор данных:
xvals = 1:100000;
xvals = repeat(xvals,inner=3);
yvals = 3 .+ xvals + 2*rand(length(xvals)) .- 1;
```

In [35]:

```
@show size(xvals)
```

```
@show size(yvals)
```

```
size(xvals) = (300000,)
size(yvals) = (300000,)
```

Out[35]:

```
(300000,)
```

In [36]:

```
#Определим, сколько времени потребуется, чтобы найти соответствие этим данным:
@time a,b = find_best_fit(xvals,yvals)
```

```
  0.132372 seconds (323.96 k allocations: 16.933 MiB)
```

Out[36]:

```
(1.0000000566451486, 2.996018700163404)
```

In [37]:

```
#Для сравнения реализуем подобный код на языке Python:
#import Pkg
#Pkg.add("PyCall")
#Pkg.add("Conda")
using PyCall
using Conda
```

In [38]:

```
py"""
import numpy
def find_best_fit_python(xvals,yvals):
    meanx = numpy.mean(xvals)
    meany = numpy.mean(yvals)
    stdx = numpy.std(xvals)
    stdy = numpy.std(yvals)
    r = numpy.corrcoef(xvals,yvals)[0][1]
    a = r*stdy/stdx
    b = meany - a*meanx
    return a,b
"""
```

In [39]:

```
find_best_fit_python = py"find_best_fit_python"
```

Out[39]:

```
PyObject <function find_best_fit_python at 0x0000000061344A60>
```

In [40]:

```
xpy = PyObject(xvals)
ypy = PyObject(yvals)
@time a,b = find_best_fit_python(xpy,ypy)
```

```
  0.096835 seconds (195.44 k allocations: 10.155 MiB)
```

Out[40]:

```
(1.0000000566451517, 2.996018700010609)
```

In [41]:

```
#Используем пакет для анализа производительности, чтобы провести сравнение:
#import Pkg
#Pkg.add("BenchmarkTools")
using BenchmarkTools
```

In [42]:

```
@btime a,b = find_best_fit_python(xvals,yvals)
```

```
  8.559 ms (27 allocations: 1.02 KiB)
```

Out[42]:

```
(1.0000000566451517, 2.996018700010609)
```

In [43]:
```
@btime a,b = find_best_fit(xvals,yvals)
```

```
  1.175 ms (1 allocation: 32 bytes)
```

Out[43]:
```
(1.0000000566451486, 2.996018700163404)
```

In [ ]: