7.4. Задания для самостоятельного выполнения

7.4.1. Кластеризация

In [1]:
```
#загрузила необходимые пакеты
using DataFrames, RDatasets, Clustering, Plots, LinearAlgebra, Statistics
```

In [2]:
```
using RDatasets
iris = dataset("datasets", "iris")
```

Out[2]: 150 rows × 5 columns

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
	Float64	Float64	Float64	Float64	Cat...
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa
28	5.2	3.5	1.5	0.2	setosa
29	5.2	3.4	1.4	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
⋮	⋮	⋮	⋮	⋮	⋮

In [3]:
```
# Добавила данные в новый фрейм
X = iris[!, [:SepalLength,:PetalWidth]]
```

Out[3]: 150 rows × 2 columns

	SepalLength	PetalWidth
	Float64	Float64
1	5.1	0.2
2	4.9	0.2
3	4.7	0.2
4	4.6	0.2
5	5.0	0.2
6	5.4	0.4
7	4.6	0.3
8	5.0	0.2
9	4.4	0.2
10	4.9	0.1
11	5.4	0.2
12	4.8	0.2
13	4.8	0.1
14	4.3	0.1
15	5.8	0.2
16	5.7	0.4
17	5.4	0.4
18	5.1	0.3
19	5.7	0.3
20	5.1	0.3
21	5.4	0.2
22	5.1	0.4
23	4.6	0.2
24	5.1	0.5
25	4.8	0.2
26	5.0	0.2
27	5.0	0.4
28	5.2	0.2
29	5.2	0.2
30	4.7	0.2
⋮	⋮	⋮

In [4]:

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

Out[4]:
```
150×2 Array{Float64,2}:
 5.1  0.2
 4.9  0.2
 4.7  0.2
 4.6  0.2
 5.0  0.2
 5.4  0.4
 4.6  0.3
 5.0  0.2
 4.4  0.2
 4.9  0.1
 5.4  0.2
 4.8  0.2
 4.8  0.1
 ⋮
 6.0  1.8
 6.9  2.1
 6.7  2.4
 6.9  2.3
 5.8  1.9
 6.8  2.3
 6.7  2.5
 6.7  2.3
 6.3  1.9
 6.5  2.0
 6.2  2.3
 5.9  1.8
```

In [5]:
```
# Транспонировала матрицы с данными:
X = X'
```

Out[5]:
```
2×150 Adjoint{Float64,Array{Float64,2}}:
 5.1  4.9  4.7  4.6  5.0  5.4  4.6  5.0  …  6.8  6.7  6.7  6.3  6.5  6.2  5.9
 0.2  0.2  0.2  0.2  0.2  0.4  0.3  0.2     2.3  2.5  2.3  1.9  2.0  2.3  1.8
```

In [6]:
```
# Задала количества кластеров:
k = length(unique(iris[!,:Species]))
```

Out[6]:
```
3
```

In [7]:
```
# Определила k-среднего:
C = kmeans(X,k)
```

Out[7]:
```
KmeansResult{Array{Float64,2},Float64,Int64}([5.892592592592593 6.857142857142857
5.0055555555555555; 1.4629629629629628 2.0119047619047614 0.3037037037037036], [3,
3, 3, 3, 3, 3, 3, 3, 3, 3  …  2, 2, 1, 2, 2, 2, 2, 2, 2, 1], [0.01967421124828661,
0.02189643347050918, 0.10411865569273004, 0.17522976680384517, 0.0107853223593963
2, 0.1648593964334708, 0.16448902606310156, 0.010785322359396332, 0.3774519890260
35, 0.052637174211248805  …  0.17531179138320852, 0.08483560090702724, 0.199574759
94512208, 0.08626417233558925, 0.26293083900225156, 0.10769274376416149, 0.3229308
3900225383, 0.12769274376417172, 0.5148356009070199, 0.11364883401920167], [54, 4
2, 54], [54, 42, 54], 32.73746031746022, 3, true)
```

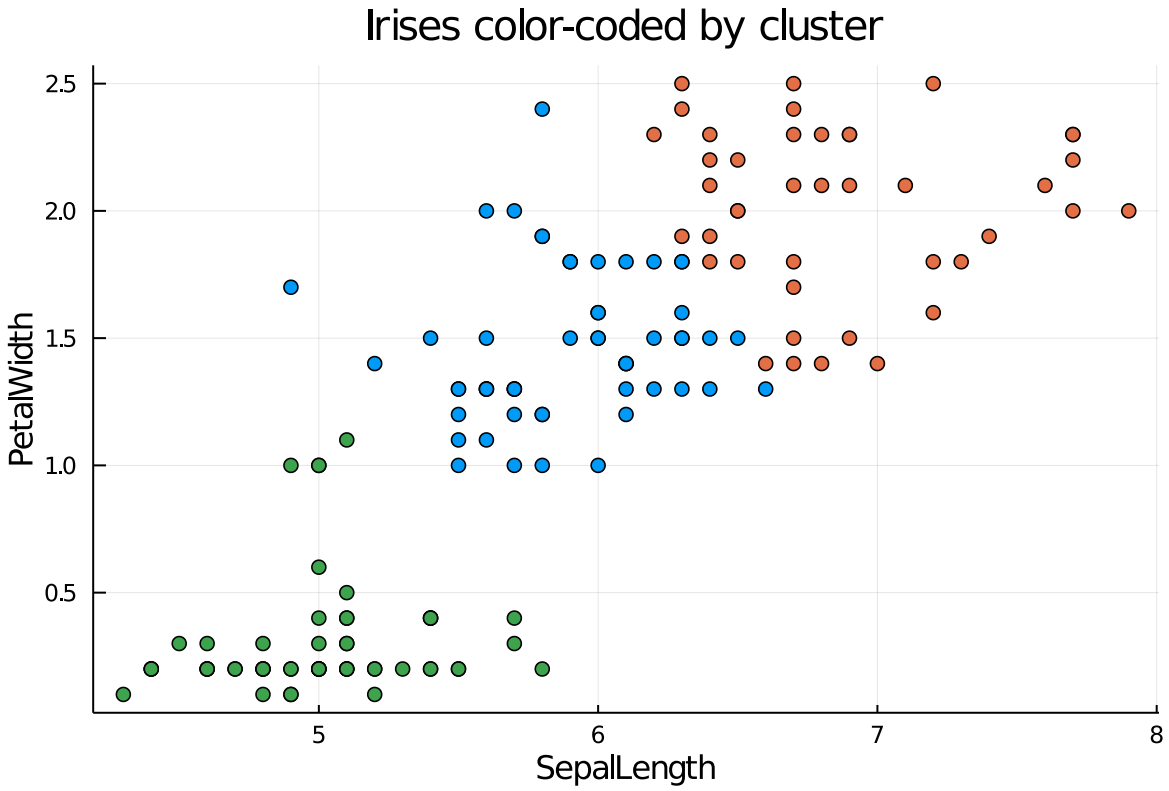
In [8]:
```
# Сформировала фрейм данных:
df = DataFrame(cluster = C.assignments,SepalLength = iris[!,:SepalLength],PetalWid
    Species = iris[!,:Species])
```

Out[8]: 150 rows × 4 columns

	cluster	SepalLength	PetalWidth	Species
	Int64	Float64	Float64	Cat...
1	3	5.1	0.2	setosa
2	3	4.9	0.2	setosa
3	3	4.7	0.2	setosa
4	3	4.6	0.2	setosa
5	3	5.0	0.2	setosa
6	3	5.4	0.4	setosa
7	3	4.6	0.3	setosa
8	3	5.0	0.2	setosa
9	3	4.4	0.2	setosa
10	3	4.9	0.1	setosa
11	3	5.4	0.2	setosa
12	3	4.8	0.2	setosa
13	3	4.8	0.1	setosa
14	3	4.3	0.1	setosa
15	3	5.8	0.2	setosa
16	3	5.7	0.4	setosa
17	3	5.4	0.4	setosa
18	3	5.1	0.3	setosa
19	3	5.7	0.3	setosa
20	3	5.1	0.3	setosa
21	3	5.4	0.2	setosa
22	3	5.1	0.4	setosa
23	3	4.6	0.2	setosa
24	3	5.1	0.5	setosa
25	3	4.8	0.2	setosa
26	3	5.0	0.2	setosa
27	3	5.0	0.4	setosa
28	3	5.2	0.2	setosa
29	3	5.2	0.2	setosa
30	3	4.7	0.2	setosa
⋮	⋮	⋮	⋮	⋮

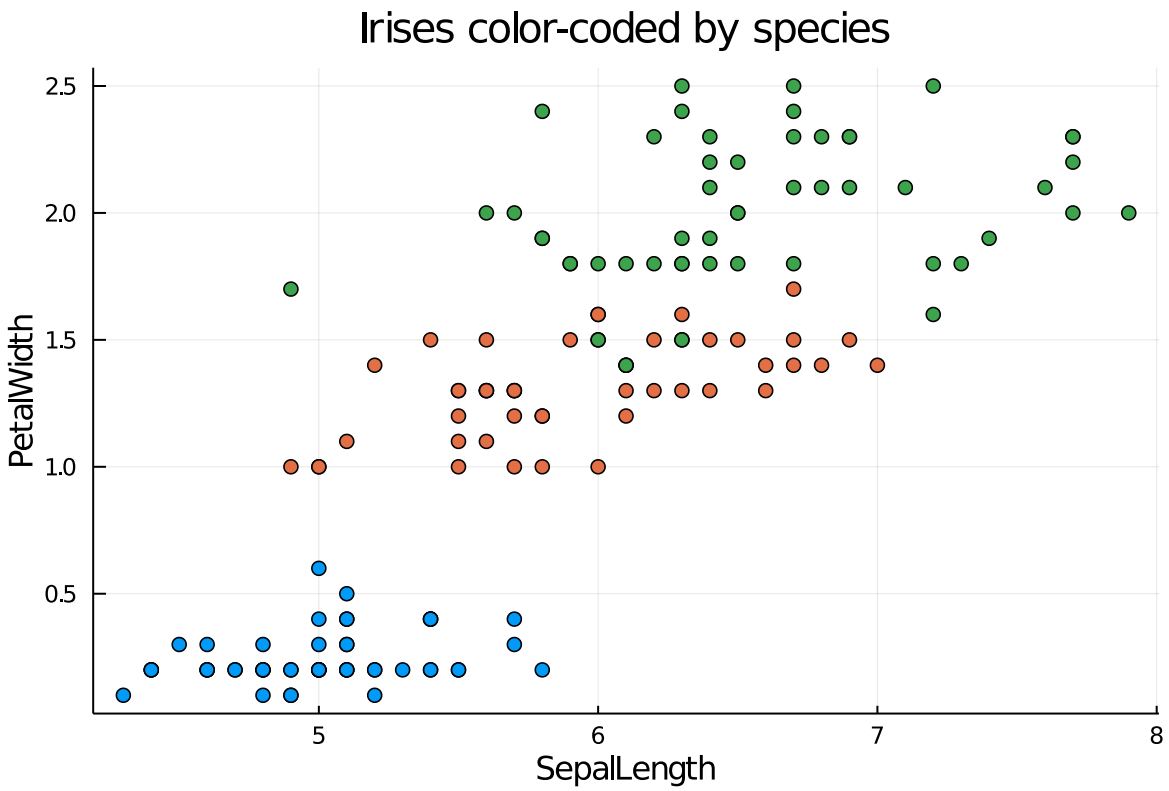
In [9]:
```
#Построла график, обозначив каждый кластер отдельным цветом:
clusters_figure = plot(legend = false)
for i = 1:k
```

```
    clustered_irises = df[df[!,:cluster].== i,:]
    xvals = clustered_irises[!,:SepalLength]
    yvals = clustered_irises[!,:PetalWidth]
    scatter!(clusters_figure,xvals,yvals,markersize=4)
end
xlabel!("SepalLength")
ylabel!("PetalWidth")
title!("Irises color-coded by cluster")
display(clusters_figure)
```



In [10]:

```
#Построила график, раскрасив кластеры по почтовому индексу:
unique_species = unique(iris[!,:Species])
species_figure = plot(legend = false)
for uspecies in unique_species
    subs = iris[iris[!,:Species].==uspecies,:]
    x = subs[!,:SepalLength]
    y = subs[!,:PetalWidth]
    scatter!(species_figure,x,y)
end
xlabel!("SepalLength")
ylabel!("PetalWidth")
title!("Irises color-coded by species")
display(species_figure)
```



7.4.2. Регрессия (метод наименьших квадратов в случае линейной регрессии)

In [11]:
```
# Часть 1
X = randn(1000, 3)
a0 = rand(3)
y = X * a0 + 0.1 * randn(1000);
```

In [12]:
```
#воспользовавшись подсказской создаю матрицу данных X2
X2 = fill(1, 1000);
#X2 - массив, на первом месте которого еденицы, далее - значения X
X2 = [X2 X]
```

Out[12]:
```
1000×4 Array{Float64,2}:
 1.0  -0.887777  -0.482291  -1.14619
 1.0  -0.446698  -0.688677   0.636682
 1.0  -0.109624  -0.31993    1.03873
 1.0  -0.441695   1.33632   -0.115264
 1.0  -0.476271   0.231994   1.03743
 1.0  -0.546734   0.484826   0.670887
 1.0   0.517417  -0.379723   0.077138
 1.0  -1.55597    1.1243     0.50659
 1.0   0.875176   0.982447   0.886336
 1.0  -2.01575    0.679407  -0.0975508
 1.0   0.688852  -0.146105  -0.389294
 1.0   2.42368    1.12226   -1.37165
 1.0  -0.377736   1.19195   -1.56079
 ⋮
 1.0  -0.192258  -0.198234   0.692979
 1.0   1.74591    1.17189    0.594511
 1.0  -0.775456  -1.29692   -0.237357
 1.0  -1.58995   -0.211807  -0.261809
 1.0   1.0648    -0.312019  -0.0408651
 1.0   1.52532   -1.04859    0.636429
 1.0  -0.603096  -0.283404  -0.676286
 1.0   0.308324  -0.715038  -0.407162
 1.0   0.567375  -0.456013  -0.368623
 1.0   2.54009    0.251281  -1.13064
 1.0   2.16617    0.278155   0.0855147
 1.0   0.668565  -0.166032  -0.841378
```

In [13]:
```
#решаю систему линейных уравнений
x =X2\y
```

Out[13]:
```
4-element Array{Float64,1}:
 -0.0028990699496620758
  0.4276603629719022
  0.2735079322736477
  0.09955421806403335
```

In [14]:
```
#нахожу решение при помощи использования llsq из MultivariateStats.jl
using MultivariateStats
x = llsq(X, y)
#результаты практически аналогичные
```

Out[14]:
```
4-element Array{Float64,1}:
  0.4276603629719025
  0.2735079322736476
  0.0995542180640334
 -0.002899069949662087
```

In [15]:
```
#import Pkg
#Pkg.add("GLM")
using GLM
```

In [16]:
```
#результаты использования регулярной регрессии наименьших квадратов из GLM.jl
DF= DataFrame(y = y, x1 = X[:,1], x2 = X[:,2], x3 = X[:,3])
lm(@formula(y ~ x1 + x2 + x3), DF)
```

Out[16]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Array{Float64,1}},GLM.Dens
ePredChol{Float64,Cholesky{Float64,Array{Float64,2}}}},Array{Float64,2}}

y ~ 1 + x1 + x2 + x3

Coefficients:
```

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	-0.00289907	0.00307493	-0.94	0.3460	-0.00893316	0.00313502
x1	0.42766	0.00304842	140.29	<1e-99	0.421678	0.433642
x2	0.273508	0.00294041	93.02	<1e-99	0.267738	0.279278
x3	0.0995542	0.00292623	34.02	<1e-99	0.0938119	0.105297

In [17]:
```
# Часть 2
X = rand(100);
y = 2X + 0.1 * randn(100);
```

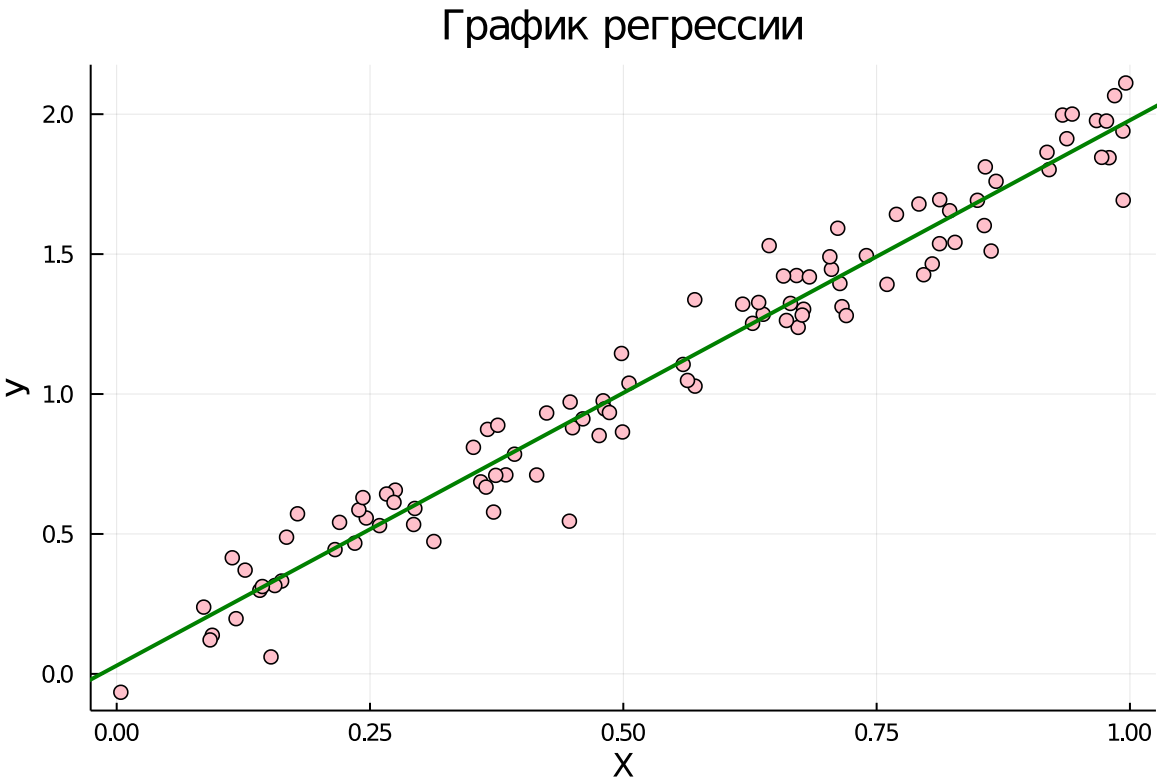
In [18]:
```
# Функция линейной регрессии:
function find_best_fit(xvals,yvals)
    meanx = mean(xvals)
    meany = mean(yvals)
    stdx = std(xvals)
    stdy = std(yvals)
    r = cor(xvals,yvals)
    a = r*stdy/stdx
    b = meany - a*meanx
    return a,b
end
```

Out[18]:
```
find_best_fit (generic function with 1 method)
```

In [19]:
```
a,b = find_best_fit(X,y)
ynew = a * X .+ b

scatter(X, y, title="График регрессии", xlabel="X", ylabel="У", legend=false, colo
Plots.abline!(a,b, color="green", lw = 2)
```

Out[19]:



7.4.3. Модель ценообразования биномиальных опционов

In [20]:

```
S = 100 # начальная цена акции
n = 10000 #количество периодов
T = 1 #– длина биномиального дерева в годах
h = T / n # длина одного периода;
sigma = 0.3 #волатильность акции
r = 0.08 #годовая процентная ставка
u = exp(r*h + sigma*(h^0.5))
d = exp(r*h - sigma*(h^0.5))
p = (exp(r*h) - d)/(u - d);
```

In [21]:

```
traj = []
j = 0
append!(traj, S)
for i=1:n
    k = rand()
    if k>0.5
        append!(traj,S*u^(i-j)*d^j)
    else
        append!(traj, S*u^(i-j)*d^(j+1))
        j = j+1
    end
end
```

In [22]:

```
#Траектория курса акций
using Plots
plot(traj, title="Траектория курса акций", xlabel="Длина биномиального дерева", yl
```

Out[22]:

In [23]:

```
#функция createPath, которая создает траекторию цены акции с учетом начальных пара
function createPath(S::Int64, T::Int64, n::Int64, sigma::Float64, r::Float64)
    h = T / n
    u = exp(r*h + sigma*(h^0.5))
    d = exp(r*h - sigma*(h^0.5))
    p = (exp(r*h) - d)/(u - d)

    traj = []
    j = 0
    append!(traj, S)
    for i=1:n
        k = rand()
        if k>0.5
            append!(traj,S*u^(i-j)*d^j)
        else
            append!(traj, S*u^(i-j)*d^(j+1))
            j = j+1
        end
    end
    return traj
end
```

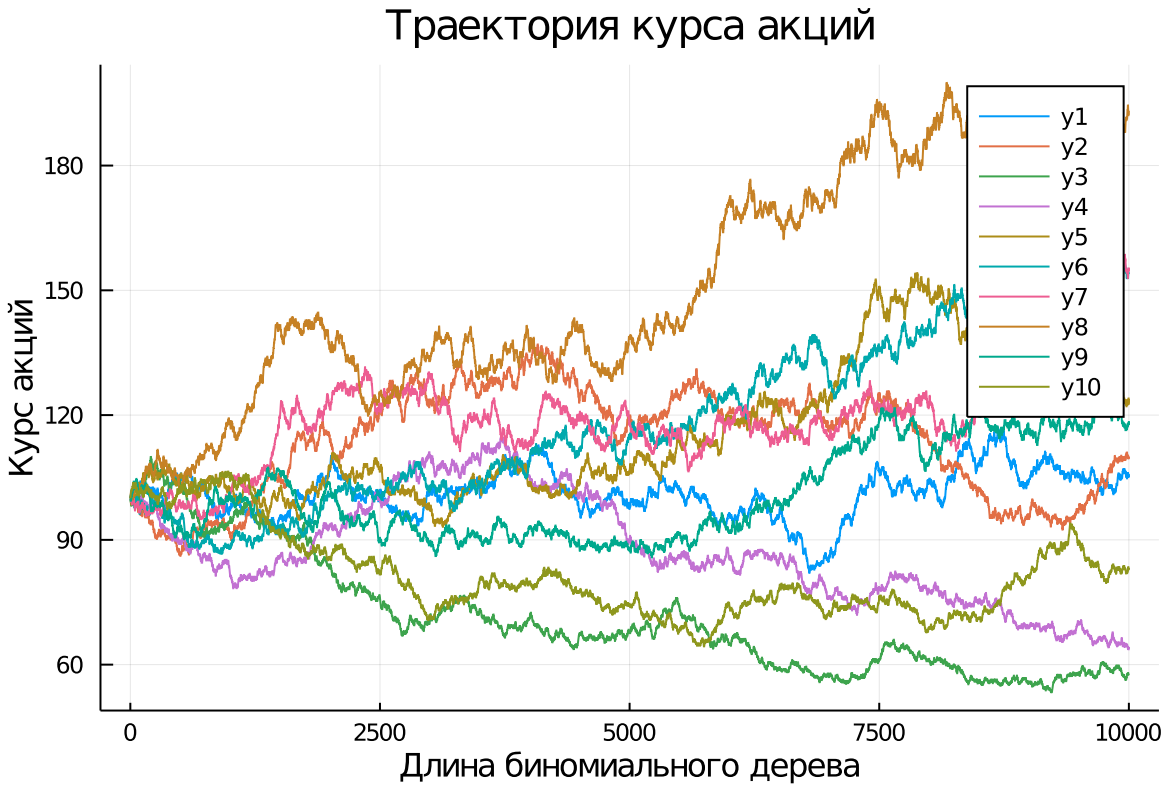
Out[23]:

```
createPath (generic function with 1 method)
```

In [24]:

```
#создала 10 различных тракекторий на одном графике. Первую траекторию нарисовала о
p = plot(createPath(S, T, n, sigma, r))
for i in 1:9
plot!(p, createPath(S, T, n, sigma, r), title="Траектория курса акций",
        xlabel="Длина биномиального дерева", ylabel="Курс акций")
end
p
```

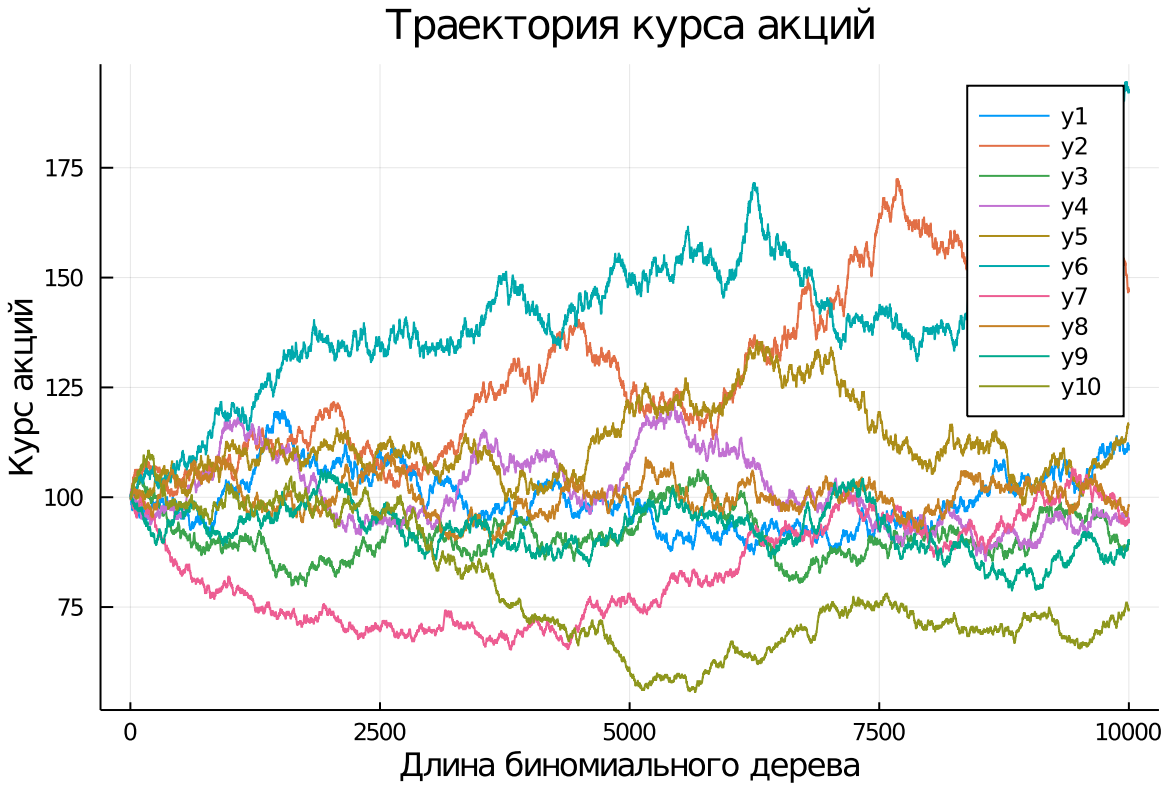
Out[24]:



In [25]:

```
#сделала распараллеливание генерации траекторий
p = plot(createPath(S, T, n, sigma, r))
Threads.@threads for i in 1:9
plot!(p, createPath(S, T, n, sigma, r), title="Траектория курса акций",
        xlabel="Длина биномиального дерева", ylabel="Курс акций")
end
p
```

Out[25]:



In [ ]:

In [ ]: