РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Компьютерный практикум

по статистическому данных анализ

Студент: Доре Стевенсон Эдгар

Группа: НКН-бд-01-19

МОСКВА

2023 г.

Лабораторная работа 5.

Построение графиков.

Цель работы:

Основная цель работы — освоить синтаксис языка Julia для построения графиков.

Ход работы:

5.2. Основные пакеты для работы с графиками в Julia

```
using Pkg
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Plotly")
Pkg.add("UnicodePlots")
```

```
  Installed ColorSchemes ──────────────── v3.10.1
  Installed DataValueInterfaces ───────── v1.0.0
  Installed Adapt ─────────────────────── v2.3.0
  Installed FriBidi_jll ───────────────── v1.0.5+6
  Installed Zlib_jll ──────────────────── v1.2.11+18
  Installed Compat ────────────────────── v3.20.0
  Installed FFMPEG ────────────────────── v0.4.0
  Installed StaticArrays ──────────────── v0.12.4
  Installed Requires ──────────────────── v1.1.0
  Installed FreeType2_jll ─────────────── v2.10.1+5
  Installed StatsBase ─────────────────── v0.33.2
Updating `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
  [91a5bcdd] + Plots v1.6.12
Updating `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  [79e6a3ab] + Adapt v2.3.0
  [56f22d72] + Artifacts v1.3.0
  [6e34b625] + Bzip2_jll v1.0.6+5
  [35d6a980] + ColorSchemes v3.10.1
  [34da2185] + Compat v3.20.0
  [d38c429a] + Contour v0.5.5
```

```
# подключаем для использования Plots:
using Plots
```

```
[ Info: Precompiling Plots [91a5bcdd-55d7-5caf-9e0b-520d859cae80]
@ Base loading.jl:1278
```

```
# задание функции:
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
```

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

```
# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
# (шаг задан через указание длины массива):
x = collect(range(-5,10,length=151))
```

```
151-element Array{Float64,1}:
 -5.0
 -4.9
 -4.8
 -4.7
 -4.6
 -4.5
 -4.4
 -4.3
 -4.2
 -4.1
 -4.0
 -3.9
 -3.8
  ⋮
  8.9
  9.0
  9.1
  9.2
  9.3
  9.4
  9.5
  9.6
  9.7
  9.8
  9.9
 10.0
```

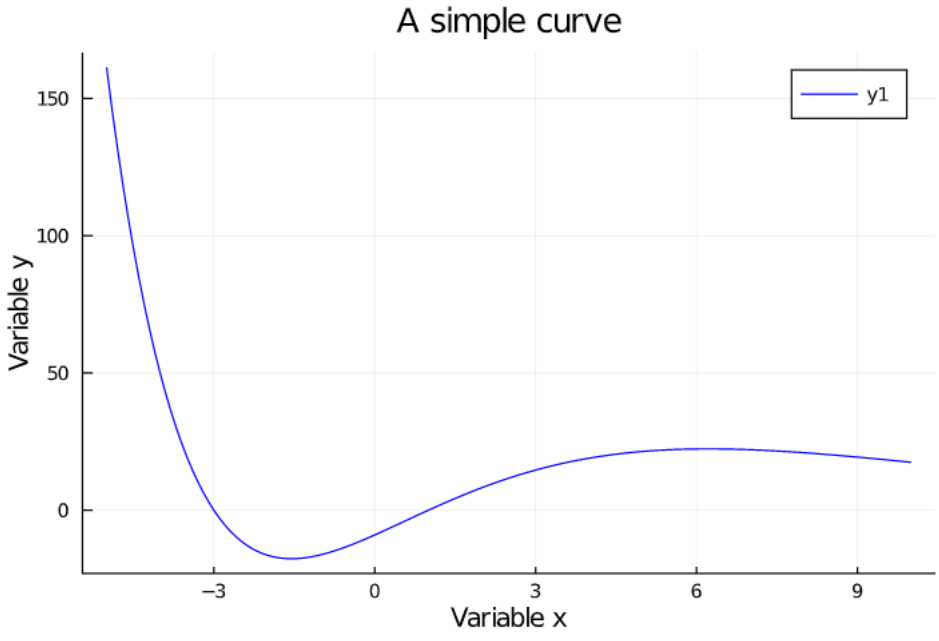
```
# генерирование массива значений y:
y = f(x)
```

```
151-element Array{Float64,1}:
 161.34080653217032
 146.26477779394
 132.19219298833204
 119.06942359634911
 106.8453557470588
  95.47128188475011
  84.9007968362764
  75.08969810741056
  65.99589024347995
  57.57929309575517
  49.80175384104821
  42.62696260773204
  36.02037156694452
   ⋮
  19.531205103994854
  19.355187669047936
  19.176427741119536
  18.995125520095375
  18.811475185747092
  18.62566497768937
  18.437877278854756
  18.248288702120195
  18.05707017974037
  17.86438705526336
  17.6703991776229
  17.475260997120245
```

```
# указывается, что для построения графика используется gr():
gr()
```

```
Plots.GRBackend()
```

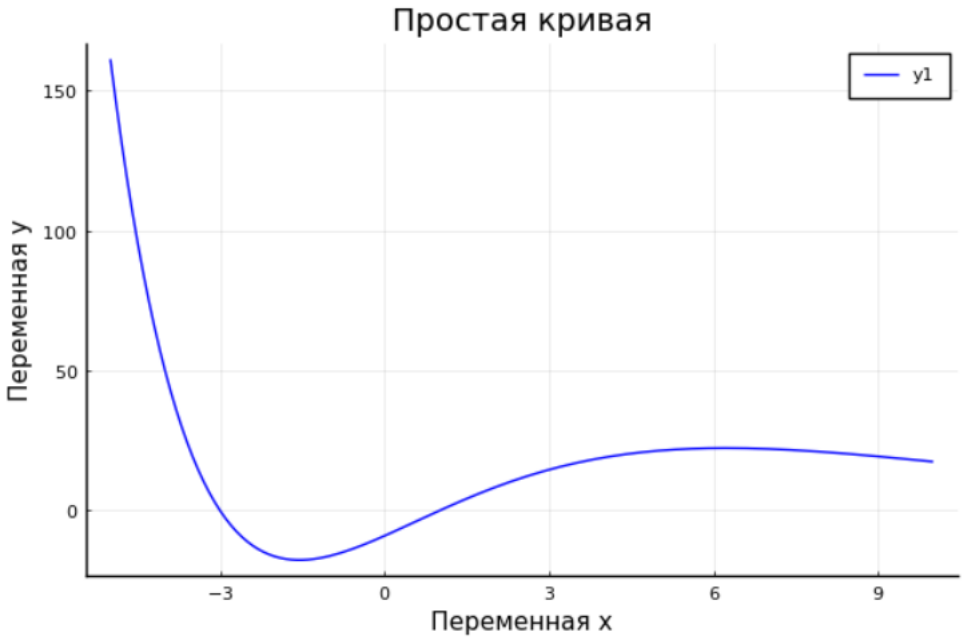
```
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
title="A simple curve",
xlabel="Variable x",
ylabel="Variable y",
color="blue")
```



```
# указывается, что для построения графика используется pyplot():
pyplot()
```

```
[ Info: Precompiling PyPlot [d330b81b-6aea-500a-939a-2ce795aea3ee]
[ @ Base loading.jl:1278
[ Info: Installing matplotlib via the Conda matplotlib package...
[ @ PyCall C:\Users\Admin\.julia\packages\PyCall\BcTLp\src\PyCall.jl:708
[ Info: Running `conda install -y matplotlib` in root environment
[ @ Conda C:\Users\Admin\.julia\packages\Conda\3rPhK\src\Conda.jl:113

Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\Admin\.julia\conda\3

  added / updated specs:
    - matplotlib


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    cycler-0.10.0              |           py38_0          14 KB
    freetype-2.10.3            |       hd328e21_0         467 KB
    icu-58.2                   |       ha925a31_3         9.4 MB
    jpeg-9b                    |       hb83a4c4_2         245 KB
    kiwisolver-1.2.0           |   py38h74a9793_0          56 KB
    libpng-1.6.37              |       h2a8f88b_0         333 KB
    libtiff-4.1.0              |       h56a325e_1         739 KB
    lz4-c-1.9.2                |       hf4a77e7_3         106 KB
    matplotlib-3.3.1           |                0          25 KB
    matplotlib-base-3.3.1      |   py38hba9282a_0         5.1 MB
    olefile-0.46               |             py_0          33 KB
```

```
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
title="Простая кривая",
xlabel="Переменная x",
ylabel="Переменная y",
color="blue")
```



Упражнение:

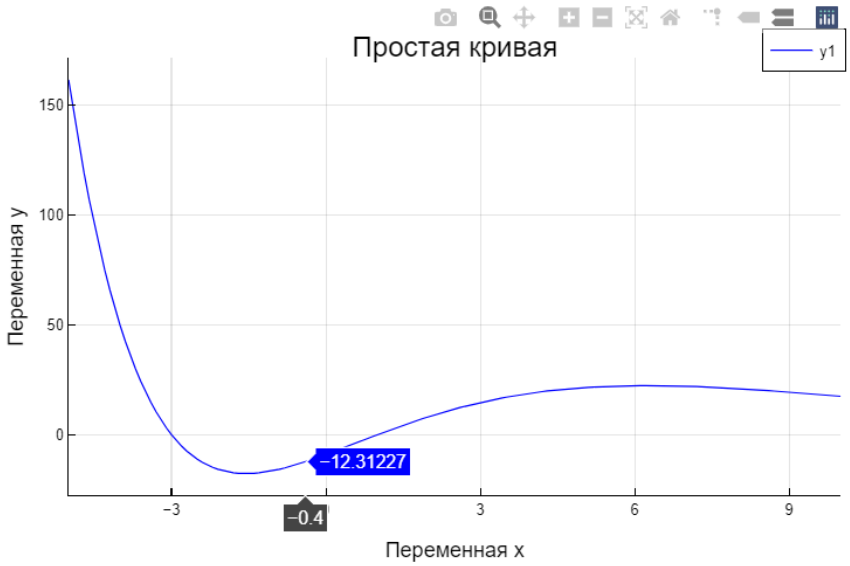
В качестве упражнения постройте график функции $f(x) = (3x2 + 6x - 9)e-0,3x$ при помощи plotly() и unicodeplots()

```
#В качестве упражнения постройте график функции f(x) = (3x2 + 6x - 9)e-0,3x при помощи plotly() и unicodeplots().
plotly()
```

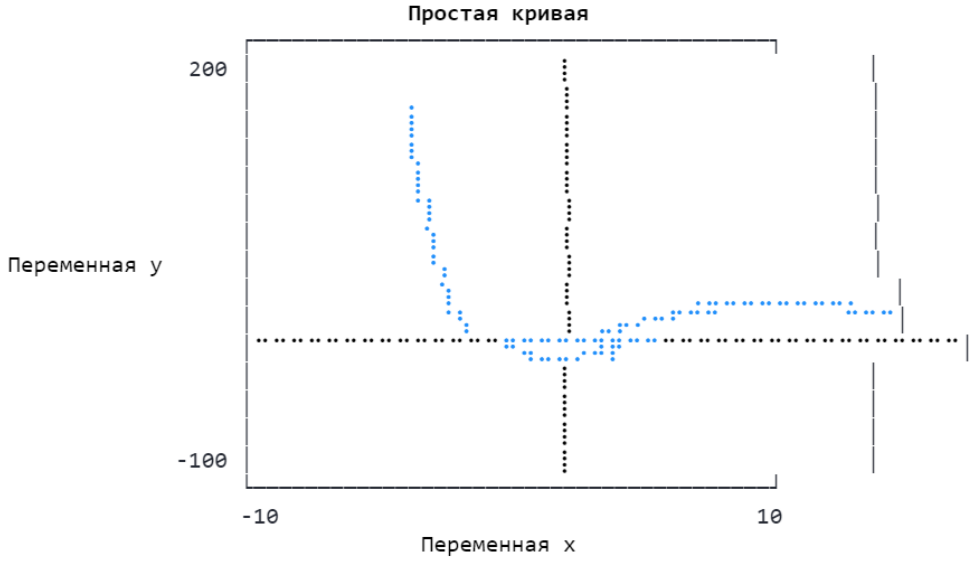
```
[ Info: For saving to png with the Plotly backend PlotlyBase has to be installed.
@ Plots C:\Users\Admin\.julia\packages\Plots\uCh2y\src\backends.jl:372
```

```
Plots.PlotlyBackend()
```

```
plot(x,y,
title="Простая кривая",
xlabel="Переменная x",
ylabel="Переменная y",
color="blue")
```




```
using UnicodePlots
lineplot(x,y,
title="Простая кривая",
xlabel="Переменная x",
ylabel="Переменная y",
color=:blue)
```



5.2.2. Опции при построении графика

Опции при построении графика

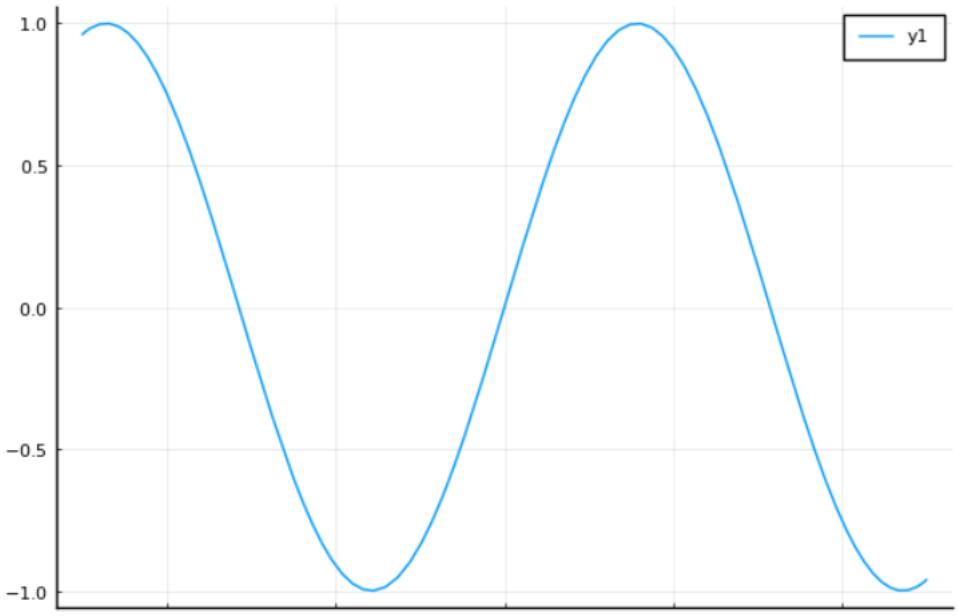
```
# указывается, что для построения графика используется pyplot():
pyplot()
```

```
Plots.PyPlotBackend()
```

```
# задание функции sin(x):
sin_theor(x) = sin(x)
```

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

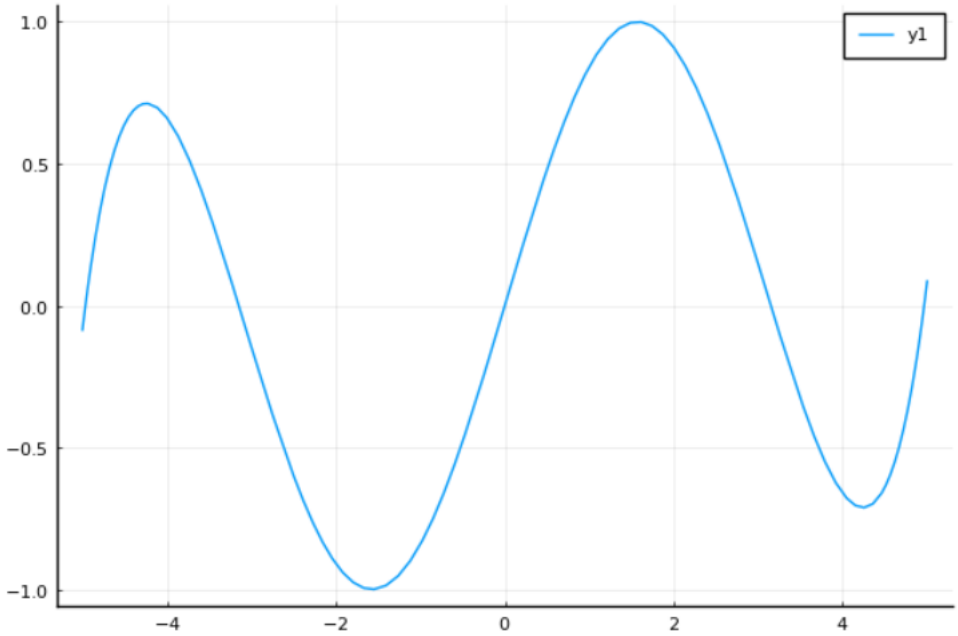
```
# построение графика функции sin(x):
plot(sin_theor)
```



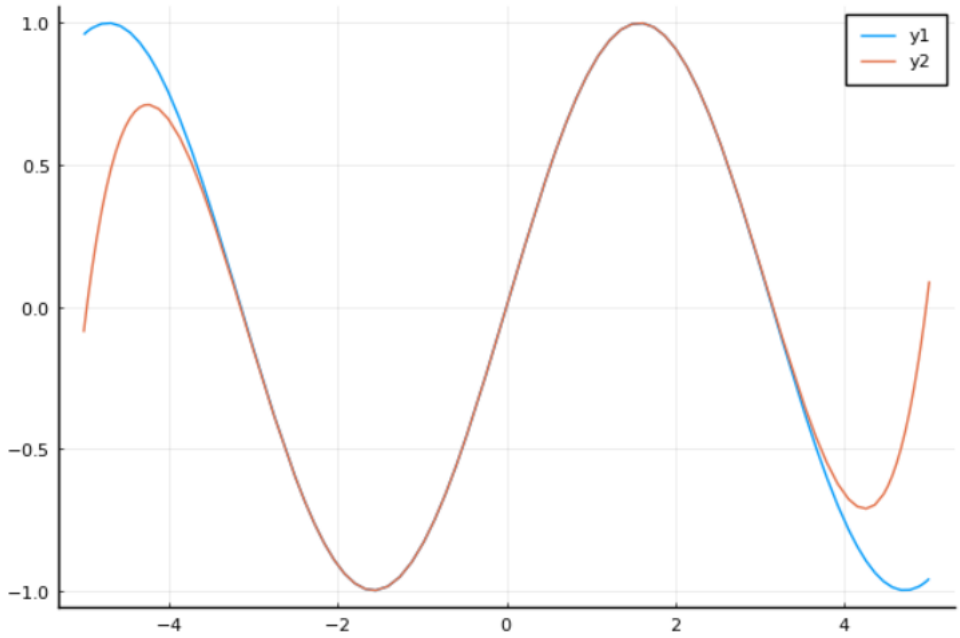
```
# задание функции разложения исходной функции в ряд Тейлора:
sin_taylor(x) = [(-1)^i*x^(2*i+1)/factorial(2*i+1) for i in 0:4] |> sum
```

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

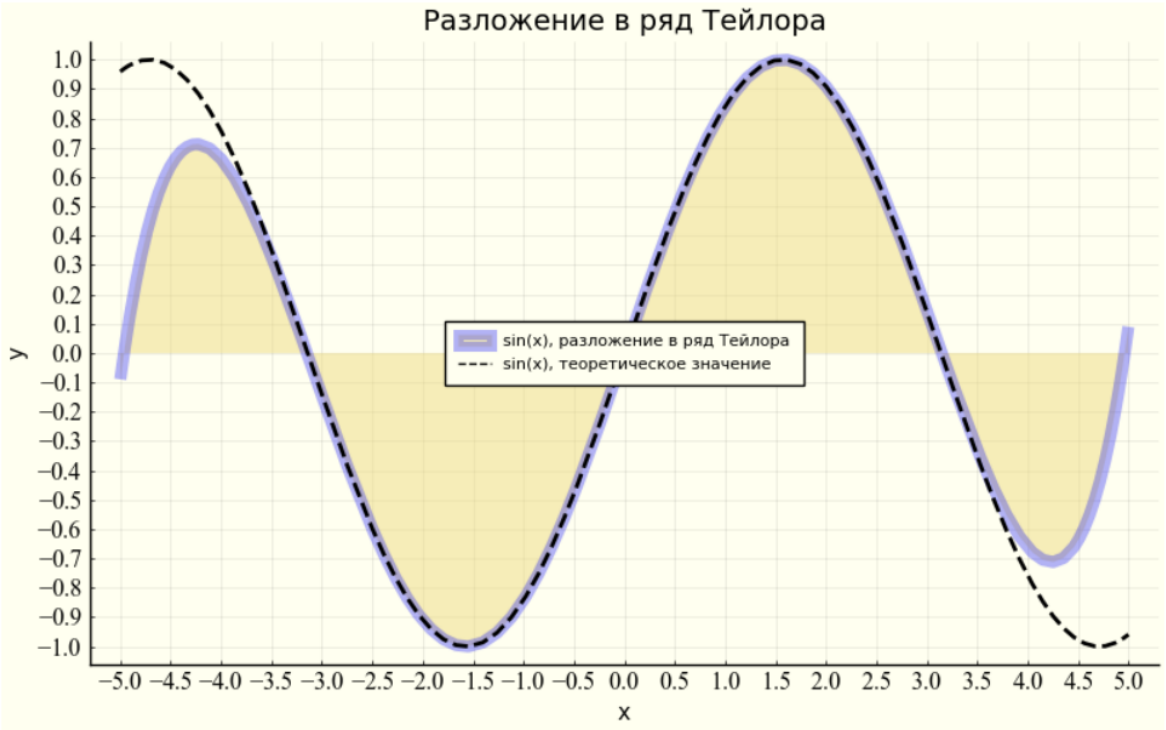
```
# построение графика функции sin_taylor(x):
plot(sin_taylor)
```




```
# построение двух функций на одном графике:
plot(sin_theor)
plot!(sin_taylor)
```



```
plot(
# функция sin(x):
sin_taylor,
# подпись в легенде, цвет и тип линии:
label = "sin(x), разложение в ряд Тейлора",
line=(:blue, 0.3, 6, :solid),
# размер графика:
size=(800, 500),
# параметры отображения значений по осям
xticks = (-5:0.5:5),
yticks = (-1:0.1:1),
xtickfont = font(12, "Times New Roman"),
ytickfont = font(12, "Times New Roman"),
# подписи по осям:
ylabel = "y",
xlabel = "x",
# название графика:
title = "Разложение в ряд Тейлора",
# поворот значений, заданный по оси x:
xrotation = rad2deg(pi/4),
# заливка области графика цветом:
fillrange = 0,
fillalpha = 0.5,
fillcolor = :lightgoldenrod,
# задание цвета фона:
background_color = :ivory
)
plot!(
# функция sin_theor:
sin_theor,
# подпись в легенде, цвет и тип линии:
label = "sin(x), теоретическое значение",
line=(:black, 1.0, 2, :dash))
```




```
# сохранение графика в файле в формате pdf или png:
savefig("taylor.pdf")
savefig("taylor.png")
```

5.2.3. Точечный график

5.2.3.1. Простой точечный график

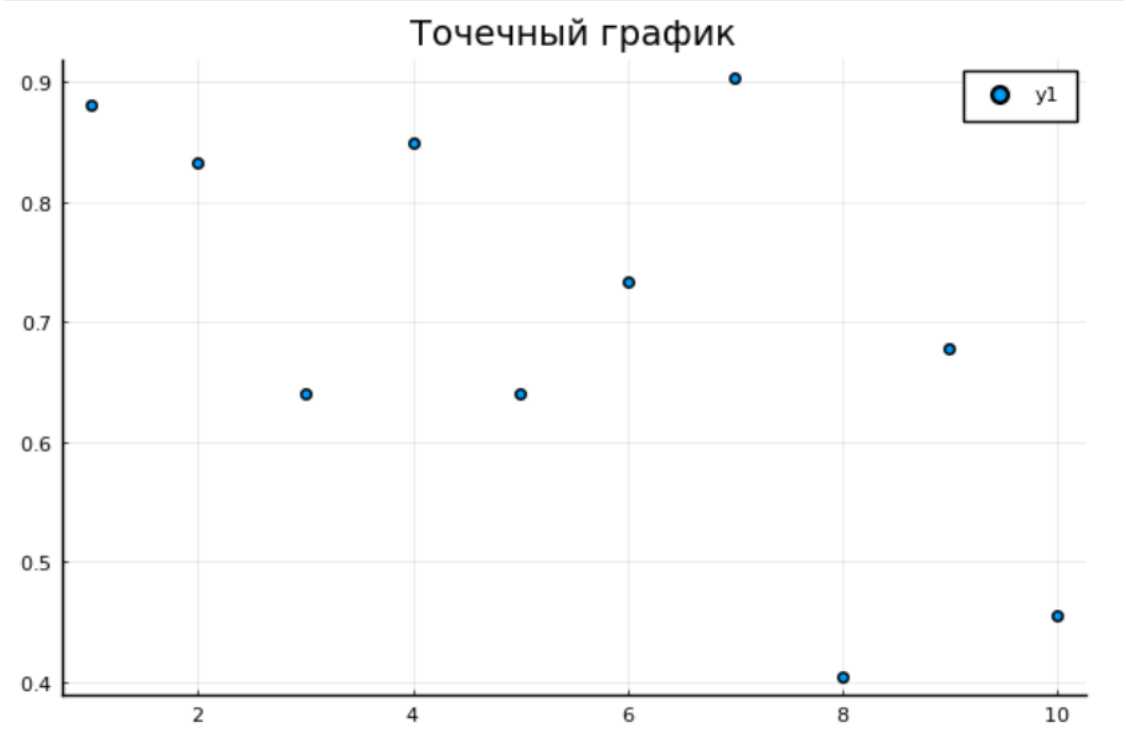
Точечный график

Простой точечный график

```
: # параметры распределения точек на плоскости:
  x = range(1,10,length=10)
  y = rand(10)
```

```
: 10-element Array{Float64,1}:
   0.8805078740697747
   0.8322451814235936
   0.6410473051714543
   0.8495112936749583
   0.6406281152318136
   0.7343732511691772
   0.9032881385286169
   0.40385374049309686
   0.6772472911247718
   0.4551320745344023
```

```
: # параметры построения графика:
  plot(x, y,
  seriestype = :scatter,
  title = "Точечный график"
  )
```



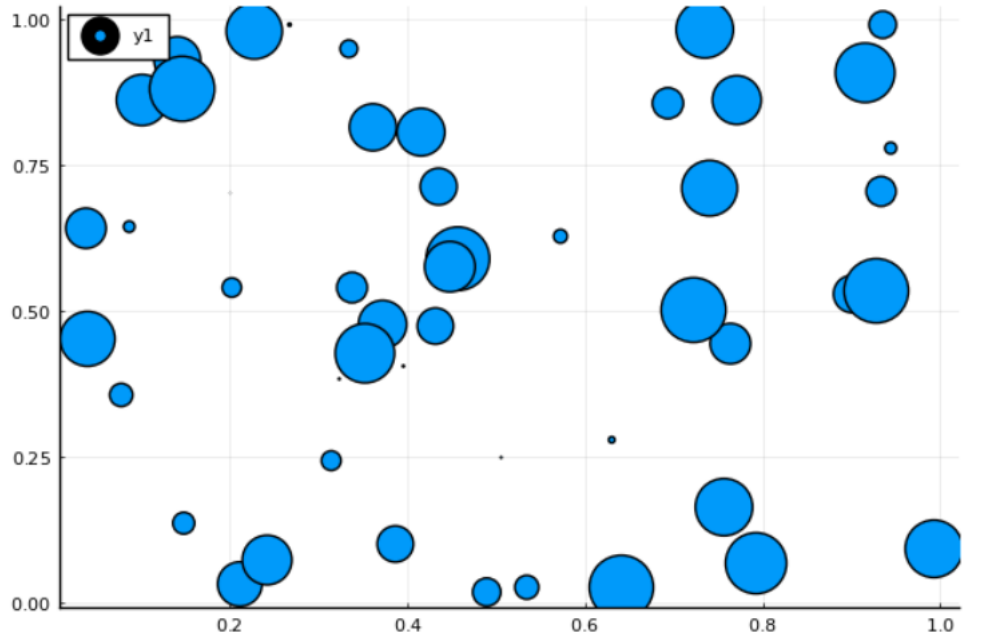
5.2.3.2. Точечный график с кодированием значения размером точки

Точечный график с кодированием значения размером точки

```
]: # параметры распределения точек на плоскости:
   n = 50
   x = rand(n)
   y = rand(n)
   ms = rand(50) * 30
```

```
]: 50-element Array{Float64,1}:
    0.5694203150545385
   25.722806771598957
   18.178179144153255
    8.647741892592332
   16.52205011190044
   26.697223328455912
   22.997880447548873
   21.654249474651376
   12.275534237191705
   19.955480622462105
   24.97329066624787
    9.766059395836614
   13.719763030383398
    ⋮
   28.777592966299302
   28.914358158348847
   21.908878177311927
    2.825731129206719
    6.126559837673424
    7.860545784723616
   10.372236637274593
   26.631392183822356
   16.236752233829485
   12.641862459329808
   16.278136950692225
    0.13147011651388985
```

```
# параметры построения графика:
scatter(x, y, markersize=ms)
```



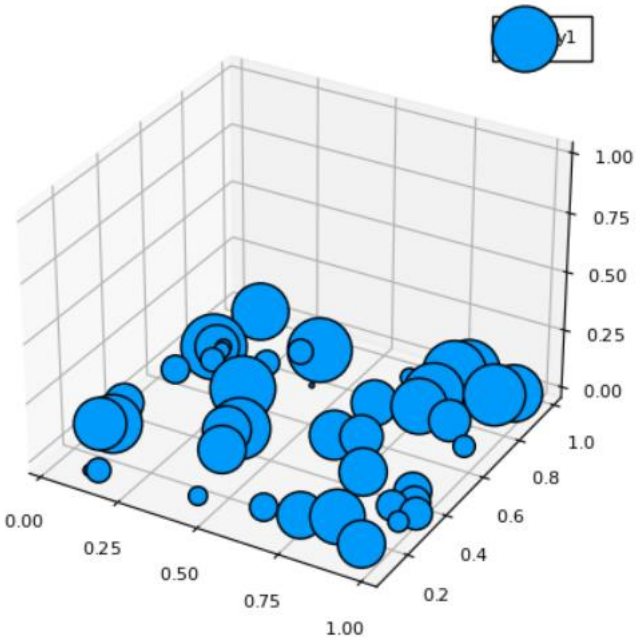
5.2.3.3. 3-мерный точечный график с кодированием значения размером точки

3-мерный точечный график с кодированием значения размером точки

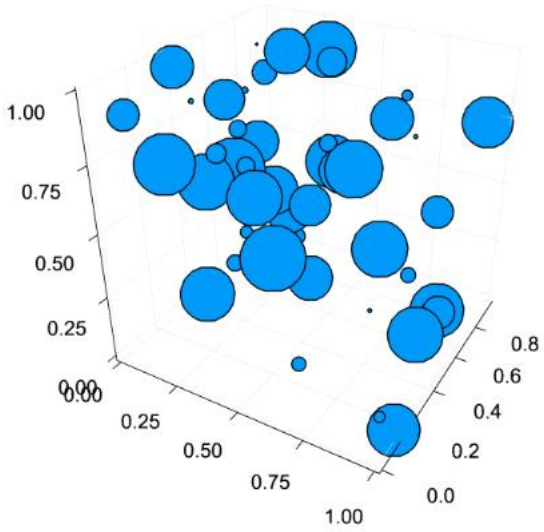
```
# параметры распределения точек в пространстве:
n = 50
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(50) * 30
```

```
50-element Array{Float64,1}:
 29.508841919040325
 16.892457082397456
 21.424282250084882
  2.223366932863955
 27.710940991166687
 12.630253226268241
 11.018835916522214
 28.653538939886328
 22.21288103484033
 17.561861914628885
  2.5533032958242807
 29.333132178028677
  8.167064712116453
  ⋮
 23.709015693618024
 11.360257551828418
  1.6226841170225592
 25.511151199709673
 24.753240548137235
 14.582901192432905
 26.26967902064449
 28.04776648629304
 13.840096656597055
  6.8016949454395785
  2.1592703106769173
 27.735687701682583
```

```
# параметры построения графика:
scatter(x, y, z, markersize=ms)
```



Если переключить на plotly, получится:



5.2.4. Аппроксимация данных

Аппроксимация данных

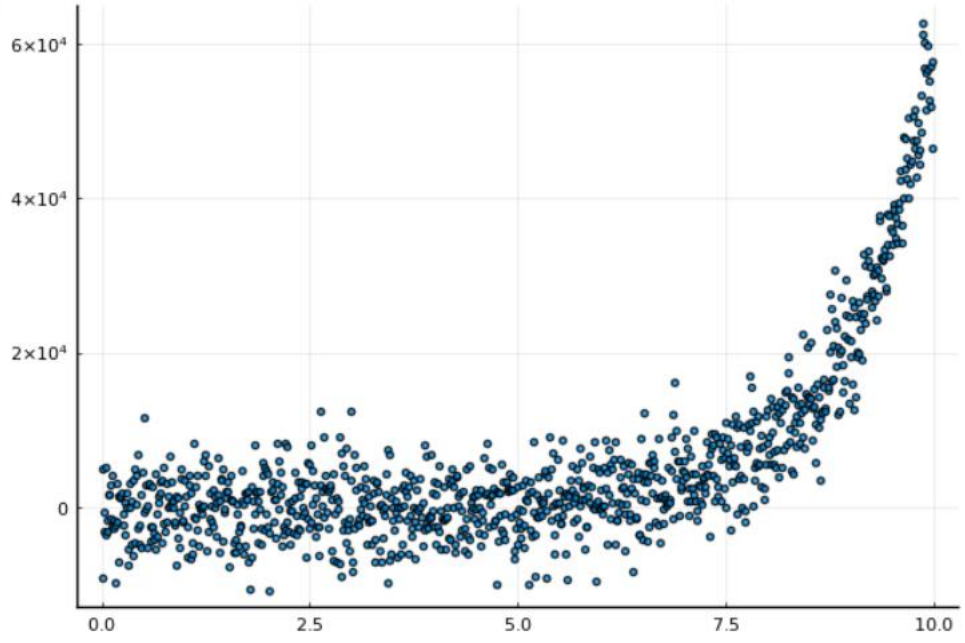
```
# массив данных от 0 до 10 с шагом 0.01:
x = collect(0:0.01:9.99)
```

```
1000-element Array{Float64,1}:
 0.0
 0.01
 0.02
 0.03
 0.04
 0.05
 0.06
 0.07
 0.08
 0.09
 0.1
 0.11
 0.12
 ⋮
 9.88
 9.89
 9.9
 9.91
 9.92
 9.93
 9.94
 9.95
 9.96
 9.97
 9.98
 9.99
```

```
# экспоненциальная функция со случайным сдвигом значений:
y = exp.(ones(1000)+x) + 4000*randn(1000)
```

```
1000-element Array{Float64,1}:
  4953.4417835565355
 -9240.207155618644
  -619.9871661932408
 -3114.6320841906604
 -3679.0040388279103
  5086.275800711875
 -3099.7624448310394
 -1749.7344251232323
  1669.8696082773658
  1701.656756167586
  2933.149937285367
 -1567.2199782150585
  4036.027049370335
     ⋮
 56911.48210179193
 60184.878794022705
 51574.00992904276
 56330.16333436031
 56724.414039701456
 59780.922529211704
 52867.86366485634
 55342.3235293029
 57191.94759474847
 52012.474856541776
 57848.02203877742
 46610.3255066568
```

```
# построение графика:
scatter(x,y,markersize=3,alpha=.8,legend=false)
```



```
# определение массива для нахождения коэффициентов полинома:
A = [ones(1000) x x.^2 x.^3 x.^4 x.^5]
```

```
1000×6 Array{Float64,2}:
 1.0  0.0    0.0       0.0          0.0             0.0
 1.0  0.01   0.0001    1.0e-6       1.0e-8          1.0e-10
 1.0  0.02   0.0004    8.0e-6       1.6e-7          3.2e-9
 1.0  0.03   0.0009    2.7e-5       8.1e-7          2.43e-8
 1.0  0.04   0.0016    6.4e-5       2.56e-6         1.024e-7
 1.0  0.05   0.0025    0.000125     6.25e-6         3.125e-7
 1.0  0.06   0.0036    0.000216     1.296e-5        7.776e-7
 1.0  0.07   0.0049    0.000343     2.401e-5        1.6807e-6
 1.0  0.08   0.0064    0.000512     4.096e-5        3.2768e-6
 1.0  0.09   0.0081    0.000729     6.561e-5        5.9049e-6
 1.0  0.1    0.01      0.001        0.0001          1.0e-5
 1.0  0.11   0.0121    0.001331     0.00014641      1.61051e-5
 1.0  0.12   0.0144    0.001728     0.00020736      2.48832e-5
 ⋮                                                 ⋮
 1.0  9.88  97.6144  964.43      9528.57        94142.3
 1.0  9.89  97.8121  967.362     9567.21        94619.7
 1.0  9.9   98.01    970.299     9605.96        95099.0
 1.0  9.91  98.2081  973.242     9644.83        95580.3
 1.0  9.92  98.4064  976.191     9683.82        96063.5
 1.0  9.93  98.6049  979.147     9722.93        96548.7
 1.0  9.94  98.8036  982.108     9762.15        97035.8
 1.0  9.95  99.0025  985.075     9801.5         97524.9
 1.0  9.96  99.2016  988.048     9840.96        98015.9
 1.0  9.97  99.4009  991.027     9880.54        98509.0
 1.0  9.98  99.6004  994.012     9920.24        99004.0
 1.0  9.99  99.8001  997.003     9960.06        99501.0
```

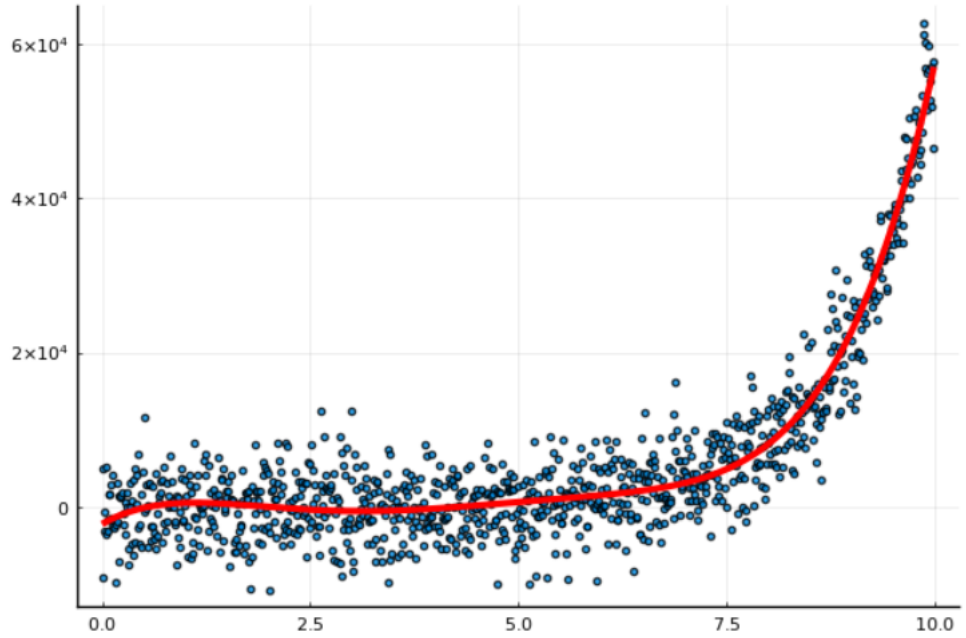
```
# решение матричного уравнения:
c = A\y
```

```
6-element Array{Float64,1}:
 -2280.3969461577512
  6301.329448416751
 -4831.1908280674115
  1469.870028484856
  -194.12620963315086
     9.513426189163557
```

```
# построение полинома:
F = c[1]*ones(1000) + c[2]*x + c[3]*x.^2 + c[4]*x.^3 + c[5]*x.^4 + c[6]*x.^5
```

```
1000-element Array{Float64,1}:
 -2280.3969461577512
 -2217.8653028266726
 -2156.291105590166
 -2095.665604970794
 -2035.980097853088
 -1977.2259273693865
 -1919.3944827856726
 -1862.4771993874167
 -1806.46555836541
 -1751.3510867016082
 -1697.1253570549668
 -1643.7799876472832
 -1591.306642149033
     ⋮
 51840.361354603316
 52298.44791979191
 52759.910838087555
 53224.768715790706
 53693.04022563202
 54164.744106884114
 54639.8991654783
 55118.52427411708
 55600.638372389134
 56086.260466882144
```

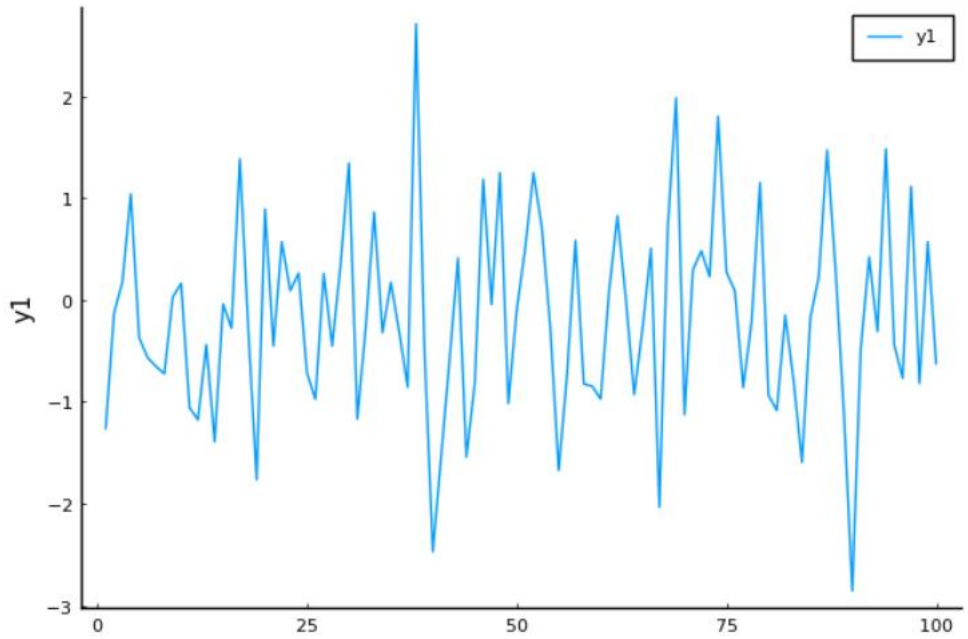
```
# построение графика аппроксимирующей функции:
plot!(x,F,linewidth=3, color=:red)
```



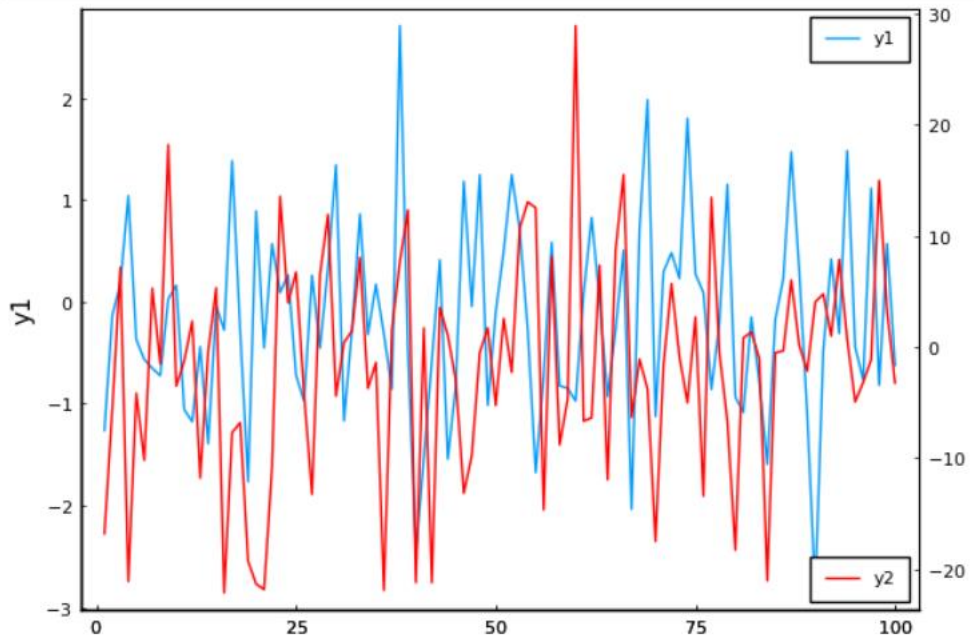
5.2.5. Две оси ординат

Две оси ординат

```
# пример случайной траектории
# (заданы обозначение траектории, легенда вверху справа, без сетки)
plot(randn(100),
ylabel="y1",
leg=:topright,
grid = :off,
)
```



```
# пример добавления на график второй случайной траектории
# (задано обозначение траектории и её цвет, легенда снизу справа, без сетки)
# задана рамка графика
plot!(twinx(), randn(100)*10,
c=:red,
ylabel="y2",
leg=:bottomright,
grid = :off,
box = :on,
# size=(600, 400)
)
```



5.2.6. Полярные координаты

Полярные координаты

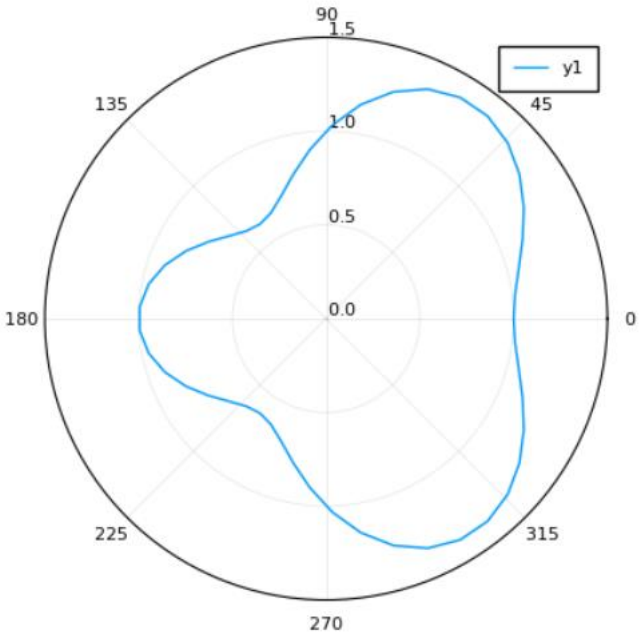
```
# функция в полярных координатах:
r(θ) = 1 + cos(θ) * sin(θ)^2
```

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

```
# полярная система координат:
θ = range(0, stop=2π, length=50)
```

```
0.0:0.1282282715750936:6.283185307179586
```

```
# график функции, заданной в полярных координатах:
plot(θ, r.(θ),
proj=:polar,
lims=(0,1.5)
)
```



5.2.7. Параметрический график

5.2.7.1. Параметрический график кривой на плоскости

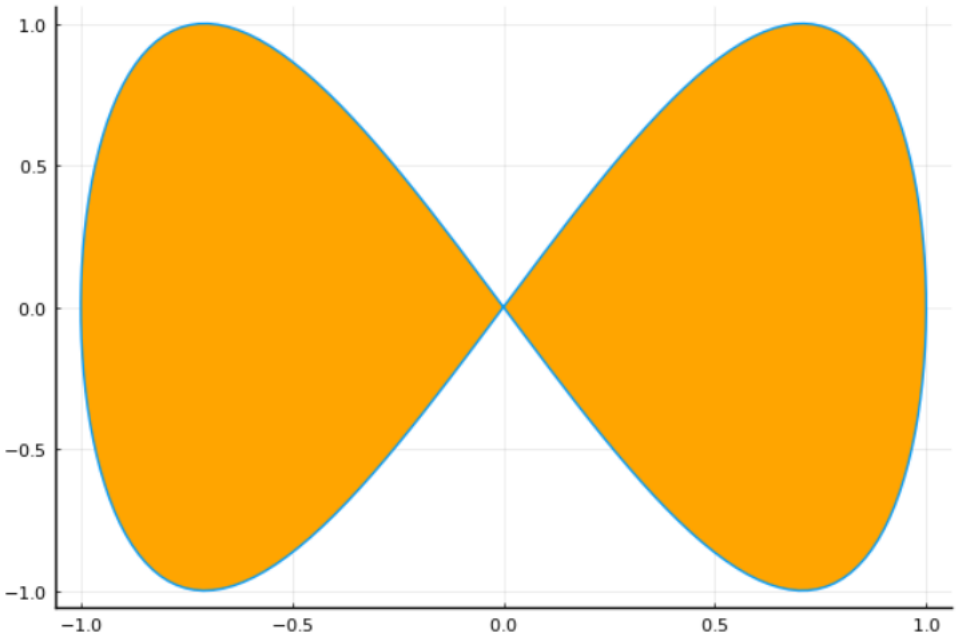
Параметрический график

Параметрический график кривой на плоскости

```
# параметрическое уравнение:
X(t) = sin(t)
Y(t) = sin(2t)
```

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

```
# построение графика:
plot(X, Y, 0, 2π, leg=false, fill=(0,:orange))
```



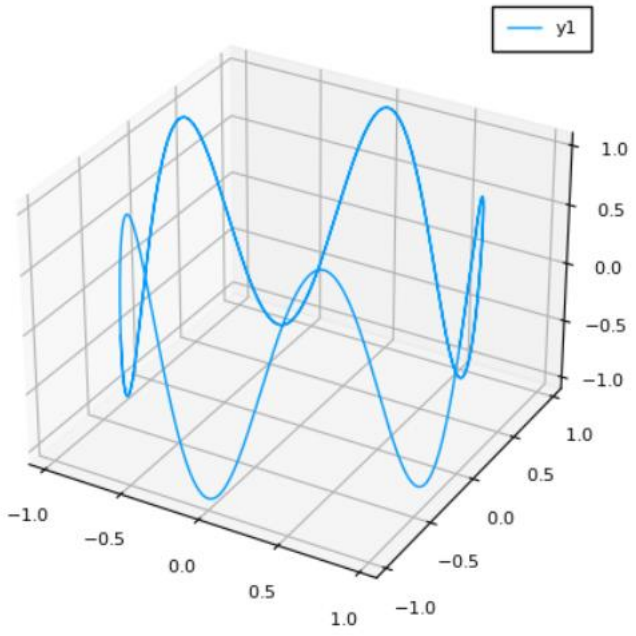
5.2.7.2. Параметрический график кривой в пространстве

Параметрический график кривой в пространстве

```
# параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
```

```
1000-element Array{Float64,1}:
  0.0
  0.05002915670857863
  0.09993301616303926
  0.1495865949143112
  0.19886553633749857
  0.24764642208113616
  0.2958070811665539
  0.34322689596321565
  0.38978710427372076
  0.4353710967719027
  0.4798647090490943
  0.5231565075371343
  0.5651380685920271
  ⋮
 -0.7284337177222215
 -0.6932457523964256
 -0.6563215637377048
 -0.6177536278090856
 -0.5776385374133922
 -0.5360767601780119
 -0.49317238693531235
 -0.4490328710287044
 -0.40376875919744315
 -0.3574934147139688
 -0.3103227334673841
 -0.26237485370392877
```

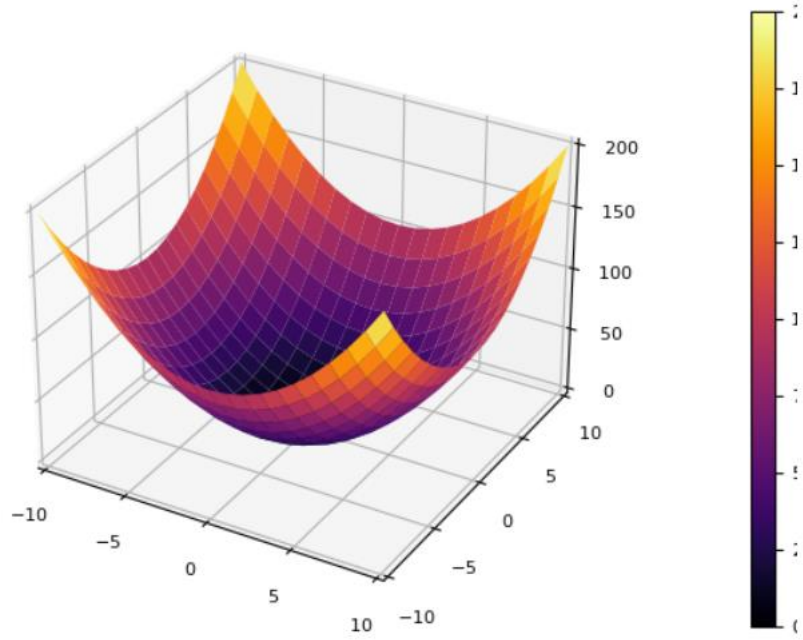
```
# построение графика:
plot(x, y, z)
```



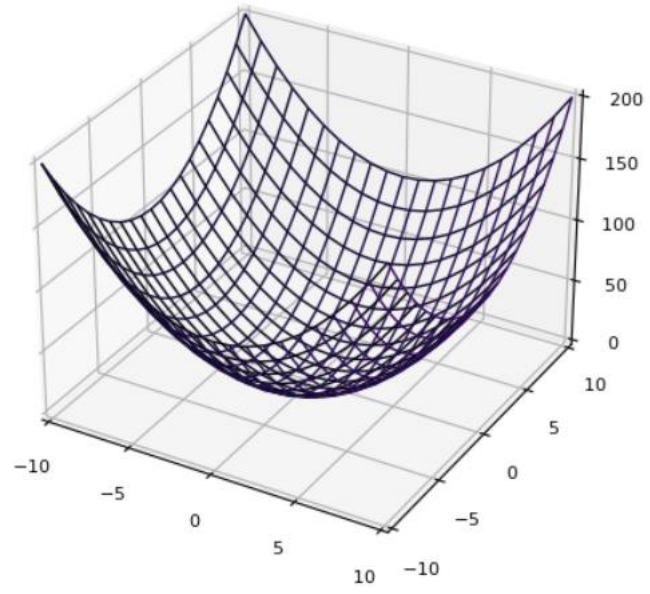
5.2.8. График поверхности

График поверхности

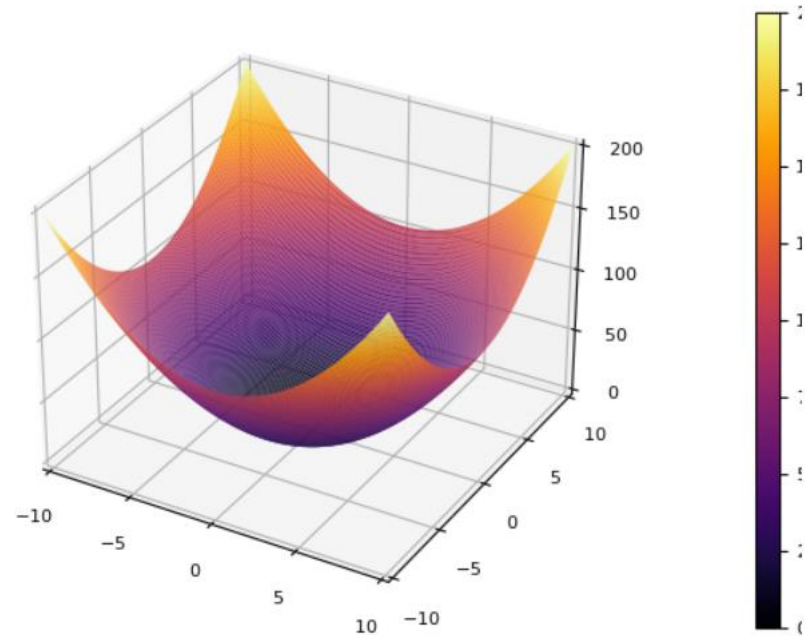
```
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
surface(x, y, f)
```



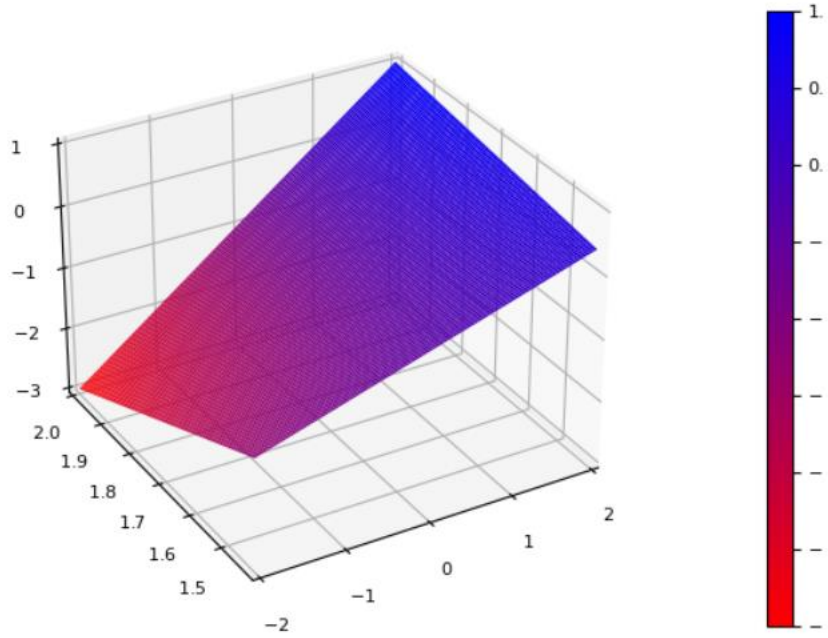

```
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
plot(x, y, f,
linetype=:wireframe
)
```



```
#Можно задать параметры сглаживания
f(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
plot(x, y, f,
linetype = :surface
)
```



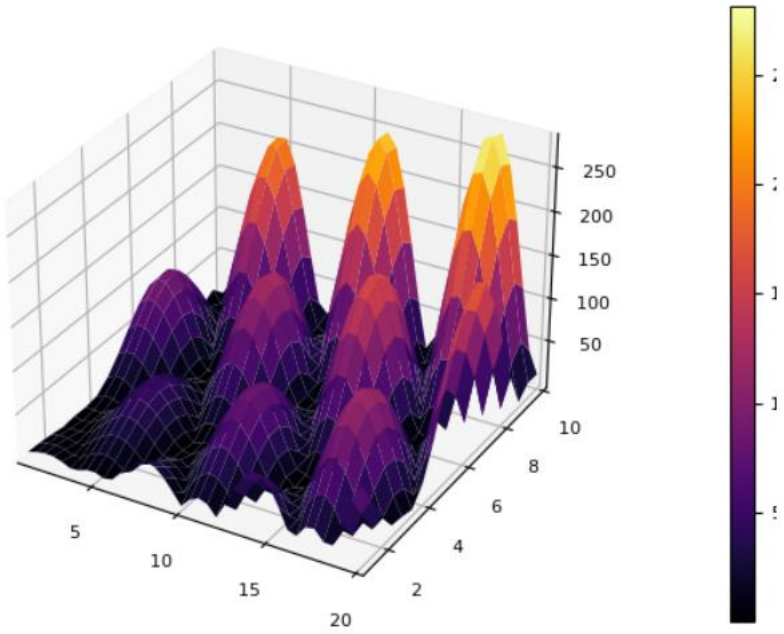

```
#Можно задать определённый угол зрения
x=range(-2,stop=2,length=100)
y=range(sqrt(2),stop=2,length=100)
f(x,y) = x*y-x-y+1
plot(x,y,f,
linetype = :surface,
c=cgrad([:red,:blue]),
camera=(-30,30),
)
```



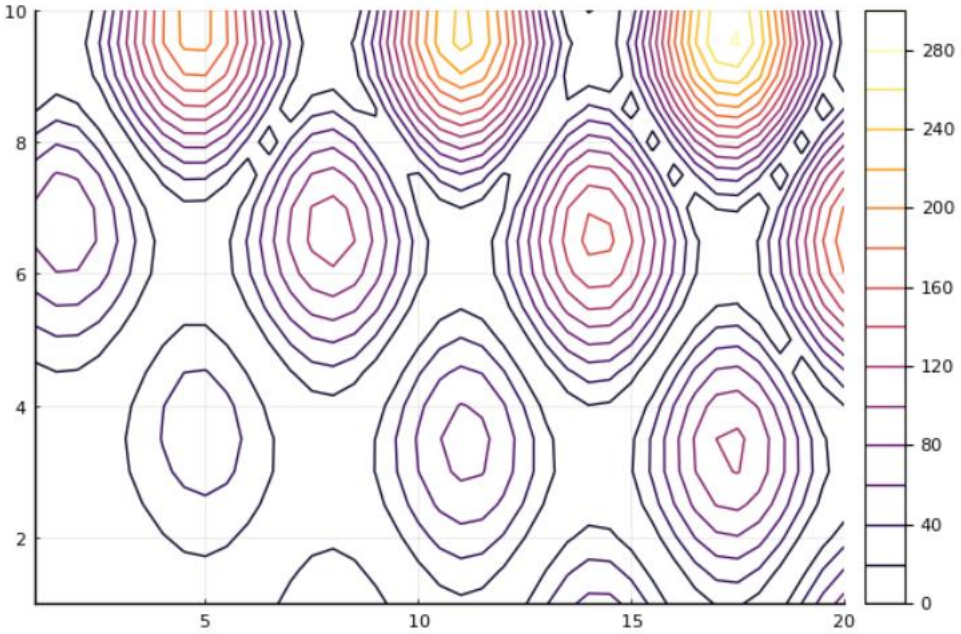
5.2.9. Линии уровня

Линии уровня

```
#Рассмотрим поверхность, заданную функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|
x = 1:0.5:20
y = 1:0.5:10
g(x, y) = (3x + y ^ 2) * abs(sin(x) + cos(y))
plot(x,y,g,
linetype = :surface,
)
```

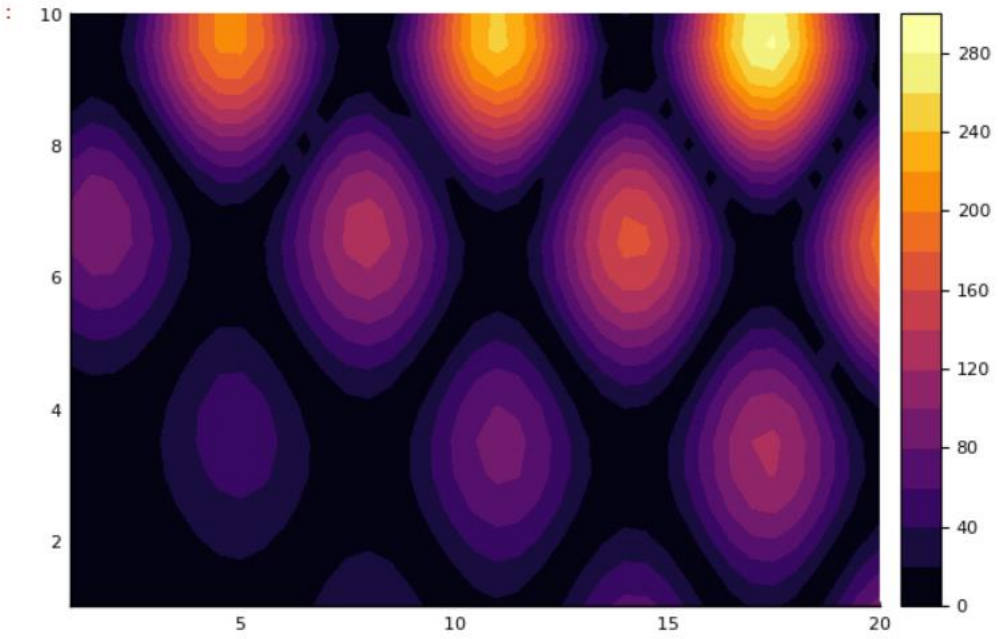



```
#Линии уровня можно построить, используя проекцию значений исходной функции на плоскость:
contour(x, y, g)
```




```
sys:1: UserWarning: The following kwargs were not used by contour: 'label'
```

```
#Можно дополнительно добавить заливку цветом:
p = contour(x, y, g,
fill=true)
plot(p)
```



5.2.10. Векторные поля

Векторные поля

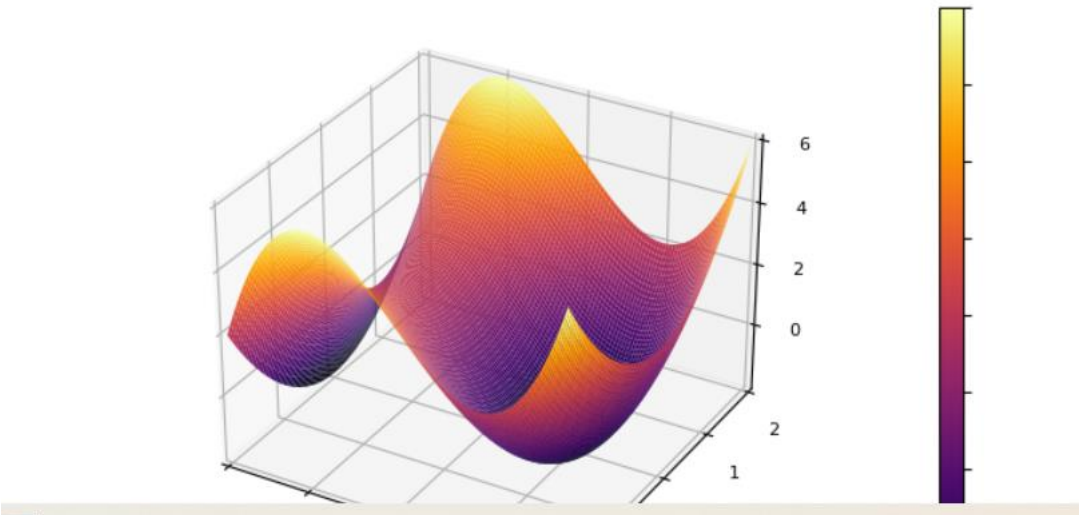
```
# определение переменных:
X0 = range(-2, stop=2, length=100)
Y0 = range(-2, stop=2, length=100)
```

```
-2.0:0.04040404040404041:2.0
```

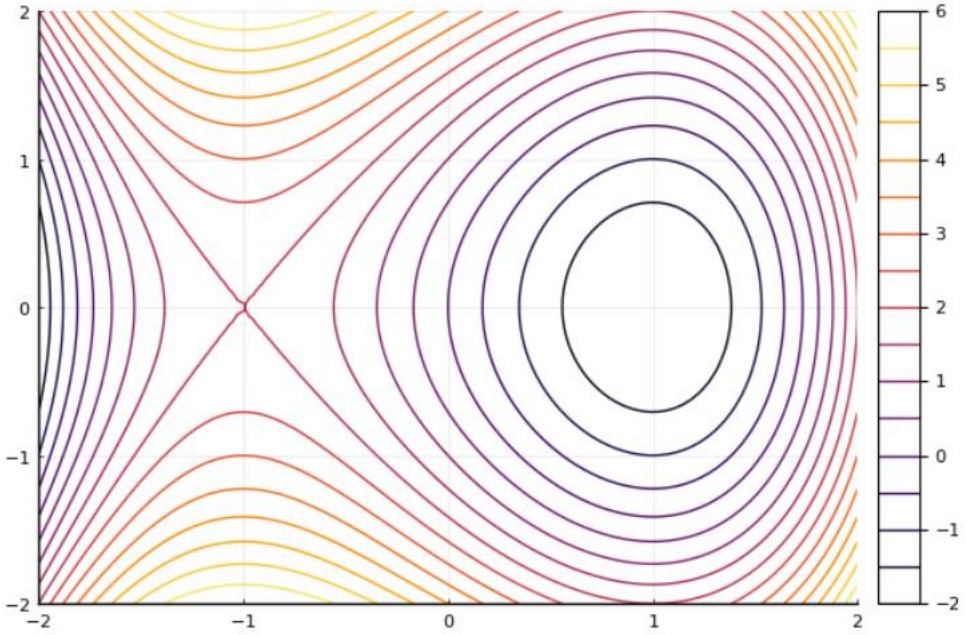
```
# определение функции:
h(x, y) = x^3 - 3x + y^2
```

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

```
# построение поверхности:
plot(X0,Y0,h,
linetype = :surface
)
```



```
# построение линий уровня:
contour(X0, Y0, h)
```



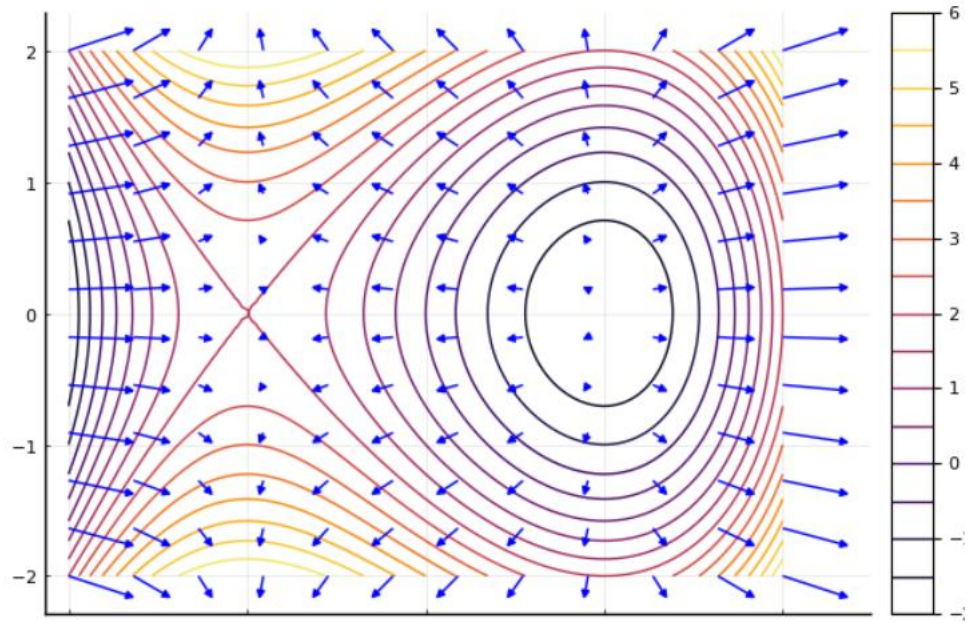

```
# градиент:
x = range(-2, stop=2, length=12)
y = range(-2, stop=2, length=12)
```

```
-2.0:0.36363636363636365:2.0
```

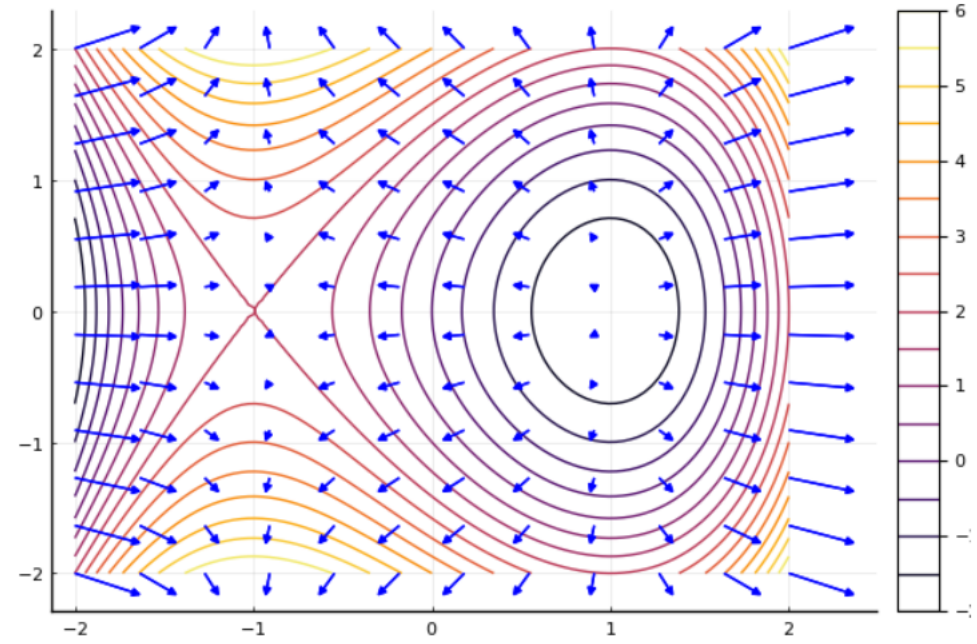
```
# производная от исходной функции:
dh(x, y) = [3x^2 - 3; 2y] / 25
```

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

```
# построение векторного поля:
quiver!(x, y', quiver=dh, c=:blue)
```



```
# построение векторного поля:
quiver!(x, y', quiver=dh, c=:blue)
```




```
sys:1: UserWarning: The following kwargs were not used by contour: 'label'
```

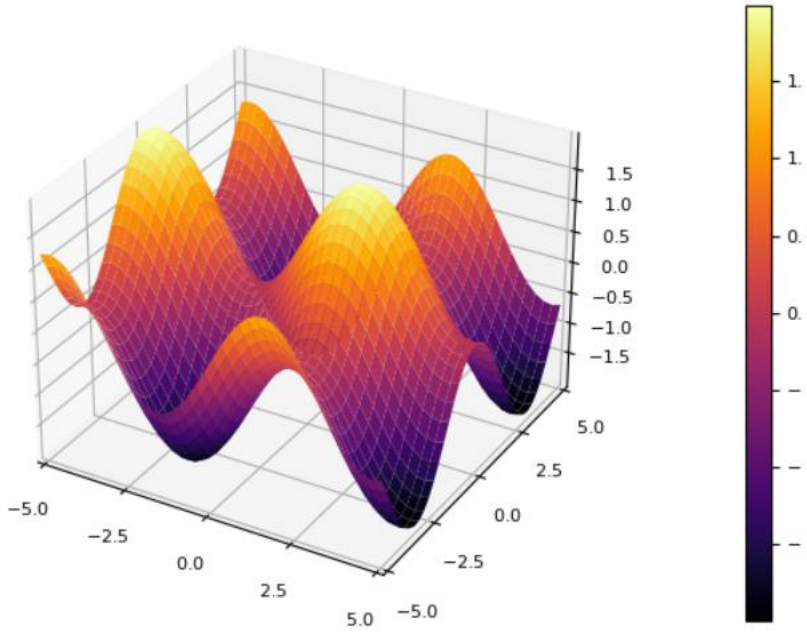
5.2.11. Анимация

5.2.11.1. Gif-анимация

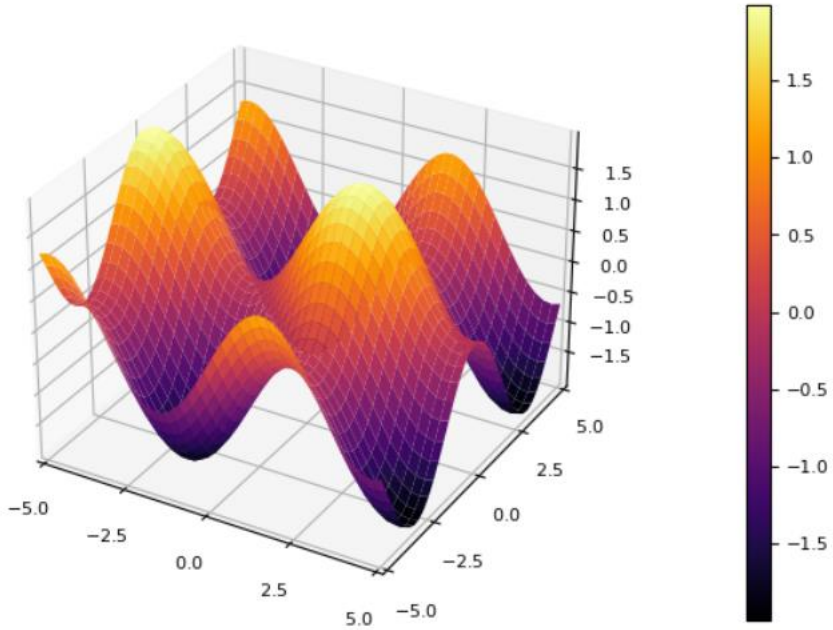
Анимация

Gif-анимация

```
# построение поверхности:
i = 0
X0 = Y0 = range(-5,stop=5,length=40)
surface(X0, Y0, (x,y) -> sin(x+10sin(i))+cos(y))
```

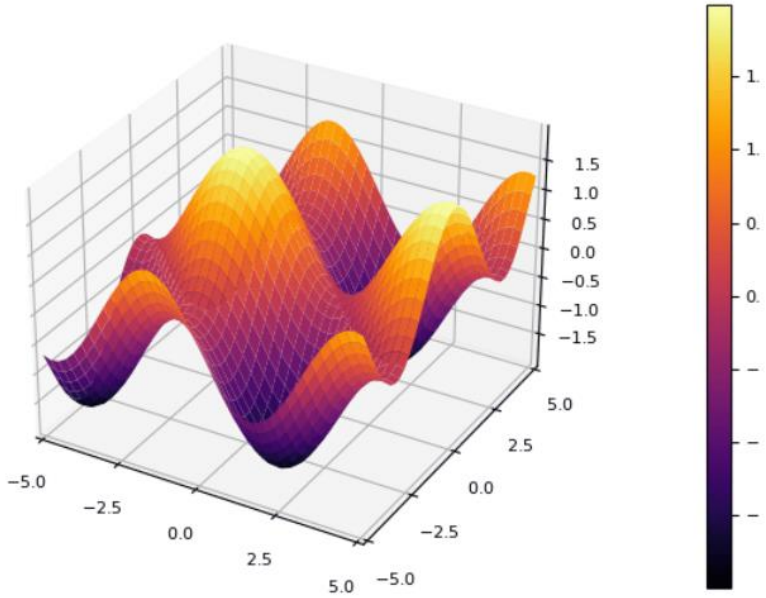


```
: # анимация:
  X0 = Y0 = range(-5,stop=5,length=40)
  @gif for i in range(0,stop=2π,length=100)
  surface(X0, Y0, (x,y) -> sin(x+10sin(i))+cos(y))
  end
```




```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\tmp.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\uCh2y\src\animation.jl:104
```

Out[87]:

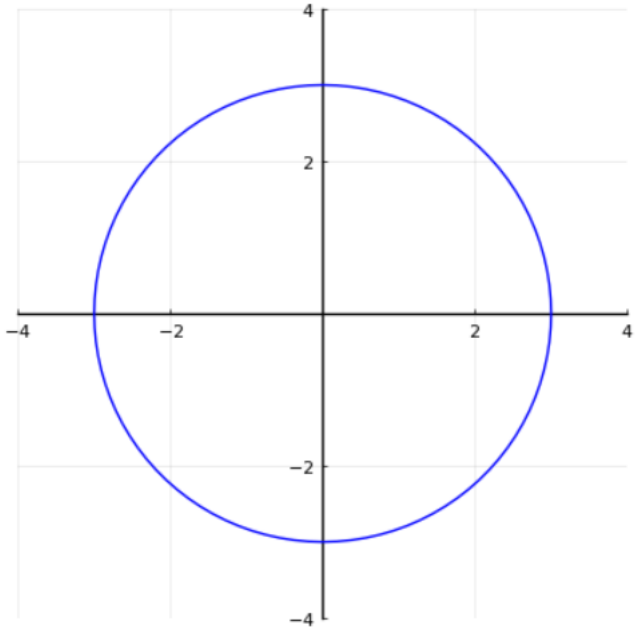


5.2.11.2. Гипоциклоида

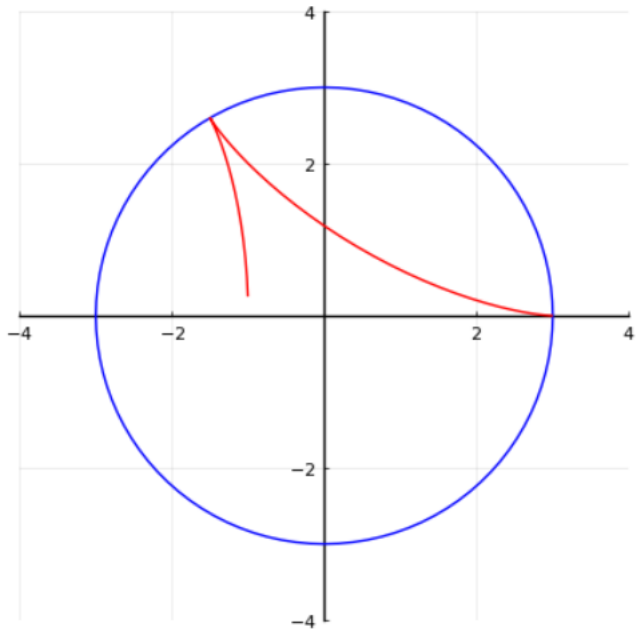
Гипоциклоида

```
# радиус малой окружности:
r0 = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100
# массив значений угла ϑ:
# theta from 0 to 2pi ( + a little extra)
θ = collect(0:2*π/100:2*π+2*π/100)
# массивы значений координат:
X01 = r0*k*cos.(θ)
Y01 = r0*k*sin.(θ)
# задаём оси координат:
plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
```

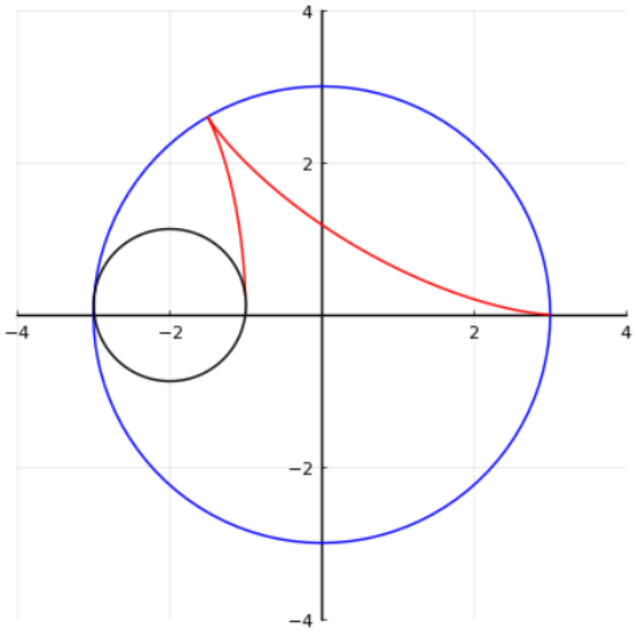

```
# большая окружность:
plot!(plt, X01,Y01, c=:blue, legend=false)
```



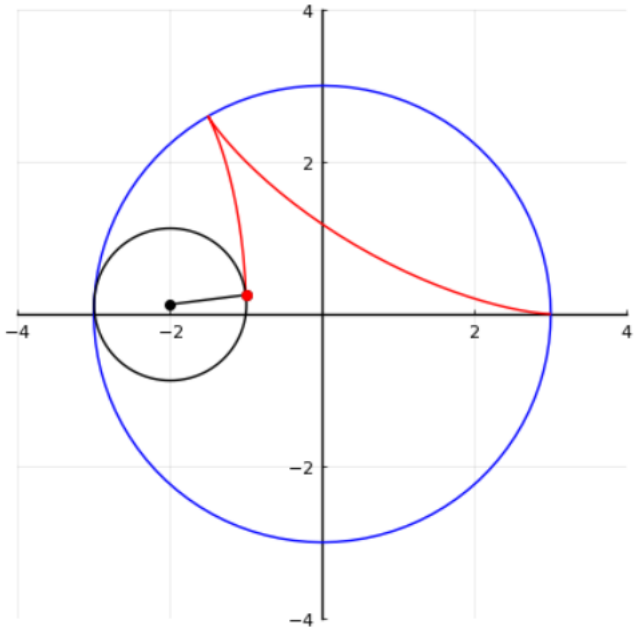
```
#Для частичного построения гипоциклоиды будем менять параметр t
i = 50
t = θ[1:i]
# гипоциклоида:
x = r0*(k-1)*cos.(t) + r0*cos.((k-1)*t)
y = r0*(k-1)*sin.(t) - r0*sin.((k-1)*t)
plot!(x,y, c=:red)
```



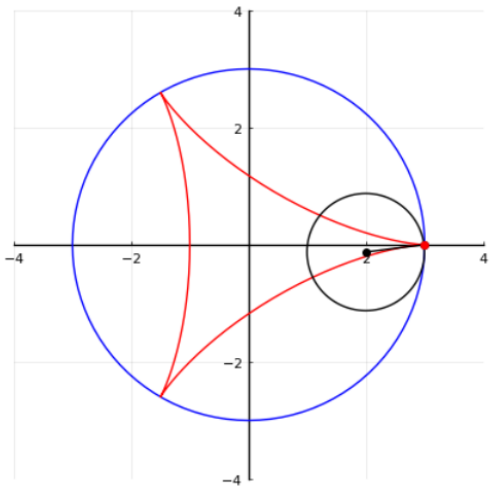

```
# малая окружность:
xc = r0*(k-1)*cos(t[end]) .+ r0*cos.(θ)
yc = r0*(k-1)*sin(t[end]) .+ r0*sin.(θ)
plot!(xc,yc,c=:black)
```



```
# радиус малой окружности:
xl = transpose([r0*(k-1)*cos(t[end]) x[end]])
yl = transpose([r0*(k-1)*sin(t[end]) y[end]])
plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
```



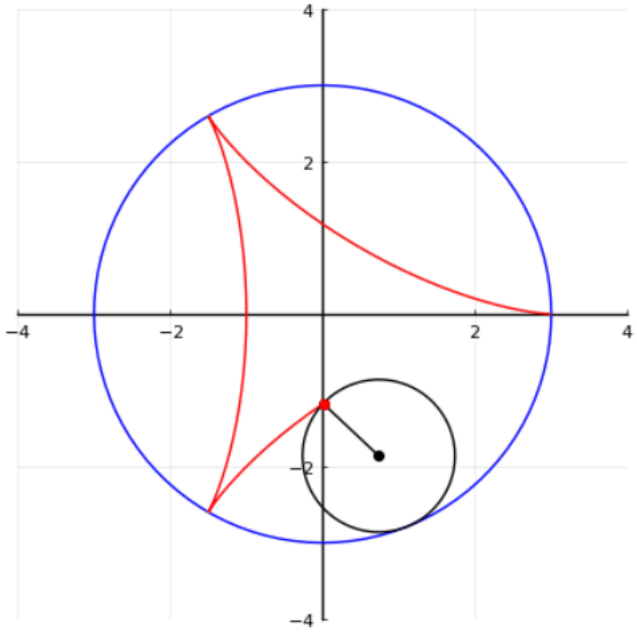

```
#В конце сделаем анимацию получившегося изображения
anim = @animate for i in 1:n
# задаём оси координат:
plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1,legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X01,Y01, c=:blue, legend=false)
t = θ[1:i]
# гипоциклоида:
x = r0*(k-1)*cos.(t) + r0*cos.((k-1)*t)
y = r0*(k-1)*sin.(t) - r0*sin.((k-1)*t)
plot!(x,y, c=:red)
# малая окружность:
    xc = r0*(k-1)*cos(t[end]) .+ r0*cos.(θ)
    yc = r0*(k-1)*sin(t[end]) .+ r0*sin.(θ)
plot!(xc,yc,c=:black)
# радиус малой окружности:
xl = transpose([r0*(k-1)*cos(t[end]) x[end]])
yl = transpose([r0*(k-1)*sin(t[end]) y[end]])
plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
end
```




```
Animation("C:\\Users\\Admin\\AppData\\Local\\Temp\\jl_XrXJIV", ["000001.png", "000002.png", "000003.png", "000004.png", "00000
5.png", "000006.png", "000007.png", "000008.png", "000009.png", "000010.png"  …  "000091.png", "000092.png", "000093.png", "000
094.png", "000095.png", "000096.png", "000097.png", "000098.png", "000099.png", "000100.png"])
```

```
#Сохраним анимацию в gif-файл
gif(anim,"hypocycloid.gif")
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\hypocycloid.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\uCh2y\src\animation.jl:104
```



5.2.12. Errorbars

Errorbars

```
# подключение пакета Statistics:
import Pkg
Pkg.add("Statistics")
using Statistics
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

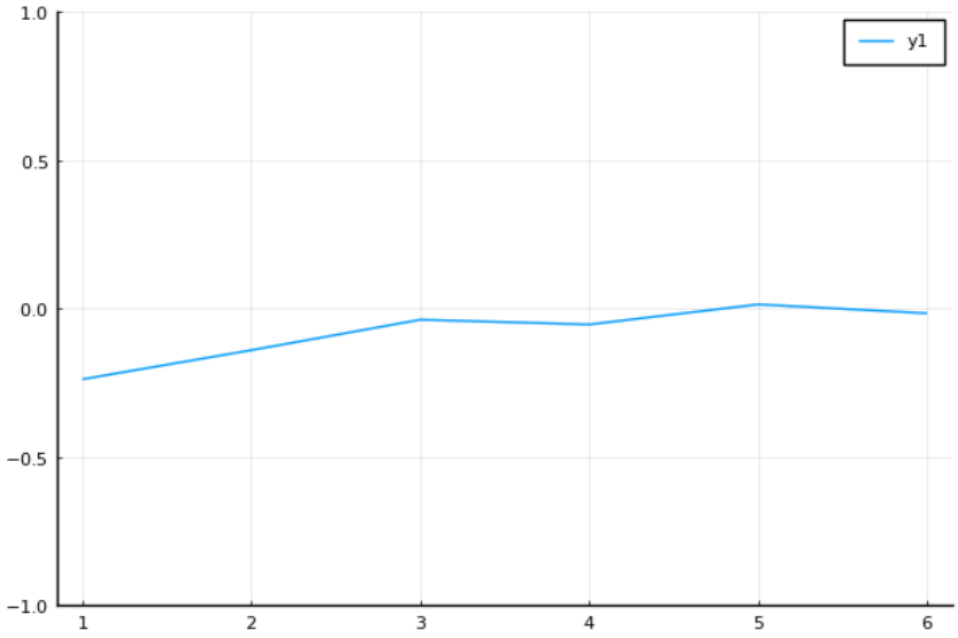
```
#Зададим массив значений:
sds = [1, 1/2, 1/4, 1/8, 1/16, 1/32]
```

```
6-element Array{Float64,1}:
 1.0
 0.5
 0.25
 0.125
 0.0625
 0.03125
```

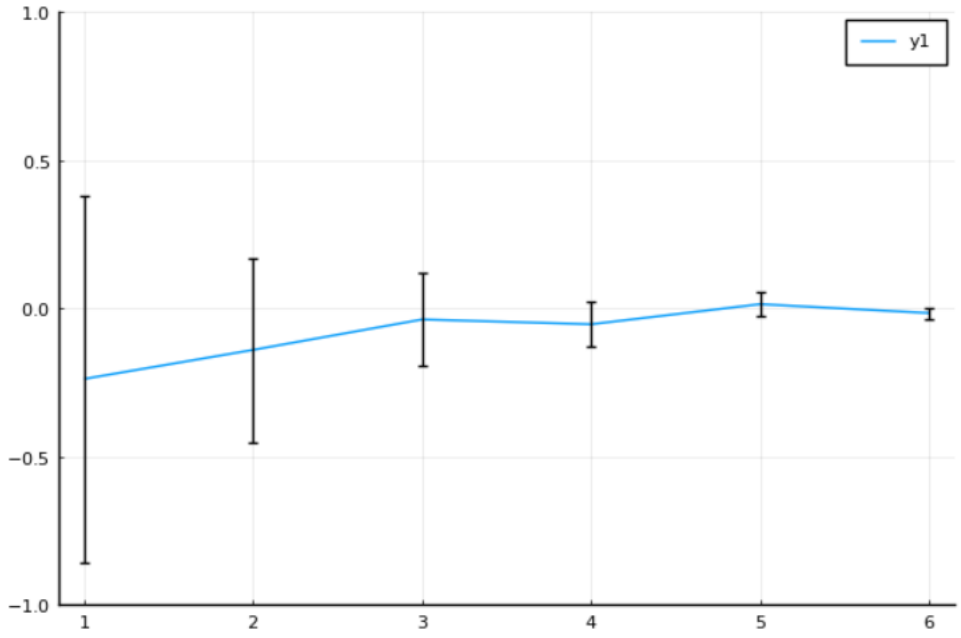
```
#Затем сгенерируем массив ошибок (отклонений от исходных значений):
n = 10
y = [mean(sd*randn(n)) for sd in sds]
errs = 1.96 * sds / sqrt(n)
```

```
6-element Array{Float64,1}:
 0.6198064213930023
 0.3099032106965012
 0.1549516053482506
 0.0774758026741253
 0.03873790133706265
 0.019368950668531323
```

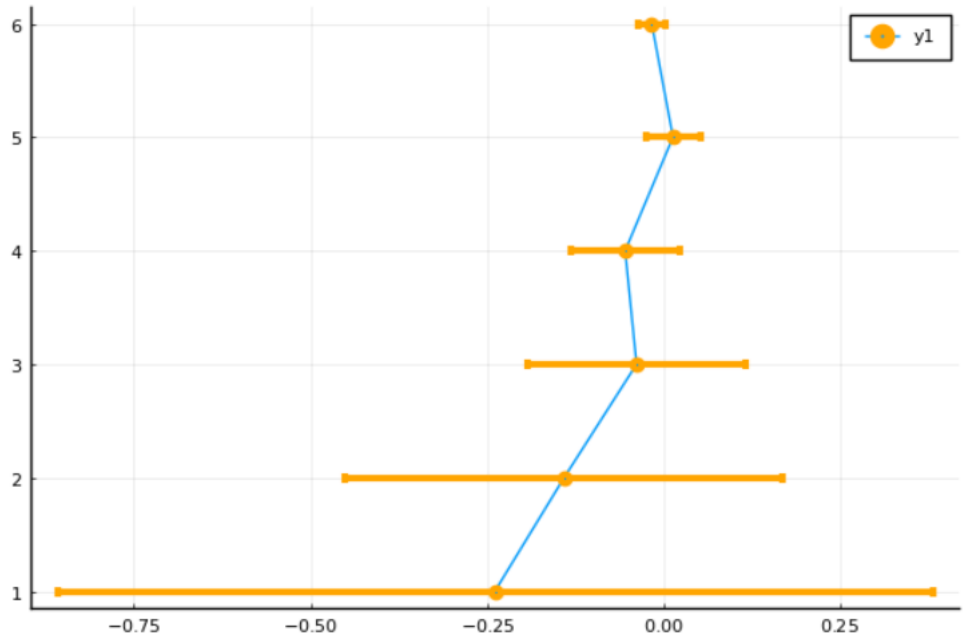
```
#Построим график исходных значений:
plot(y,
ylims = (-1,1),
)
```



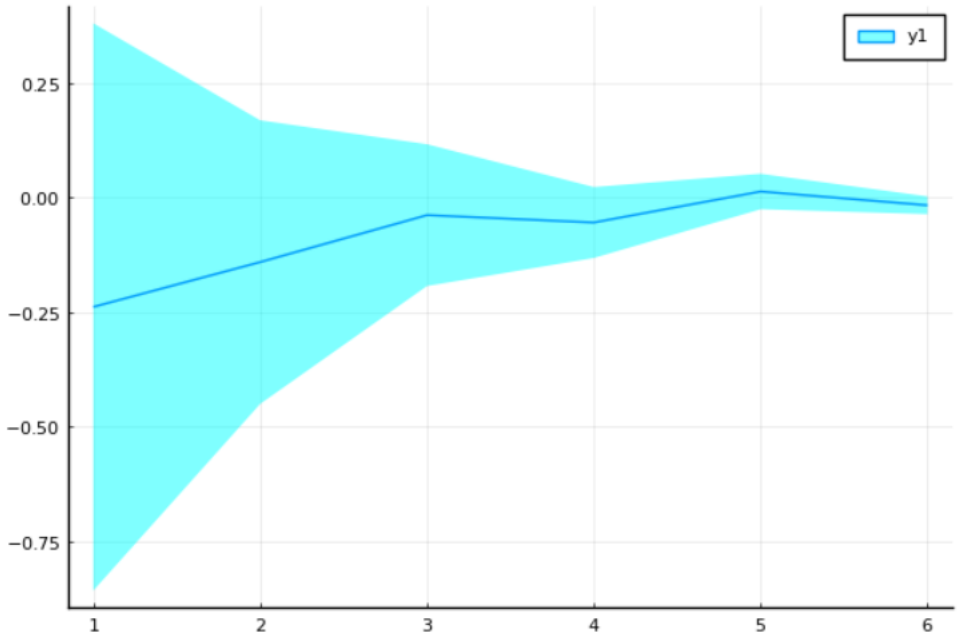
```
#Построим график отклонений от исходных значений:
plot(y,
ylims = (-1,1),
err = errs
)
```



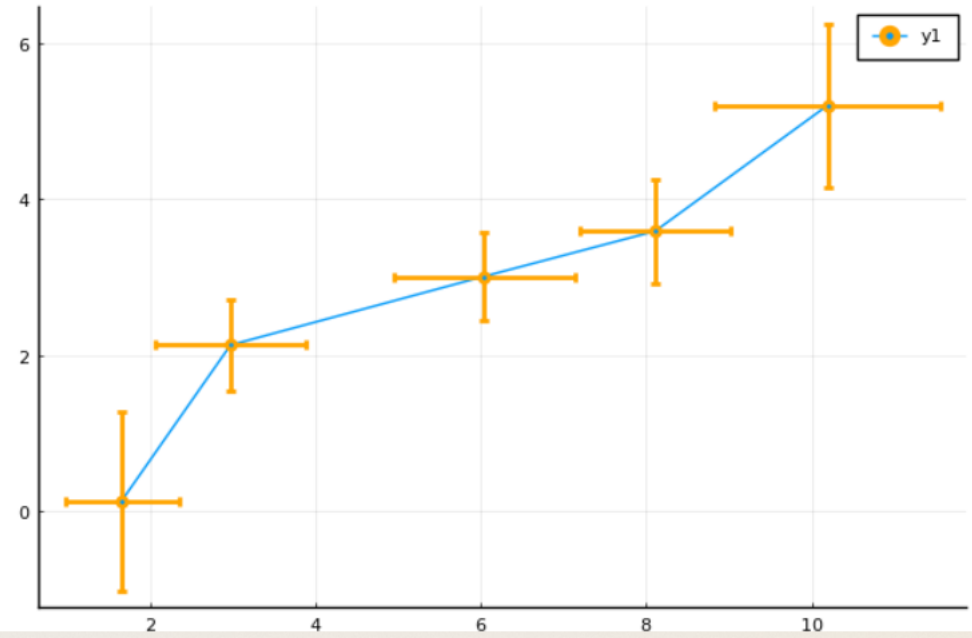

```
#Повернём график:
plot(y, 1:length(y),
xerr = errs,
marker = stroke(3,:orange)
)
```



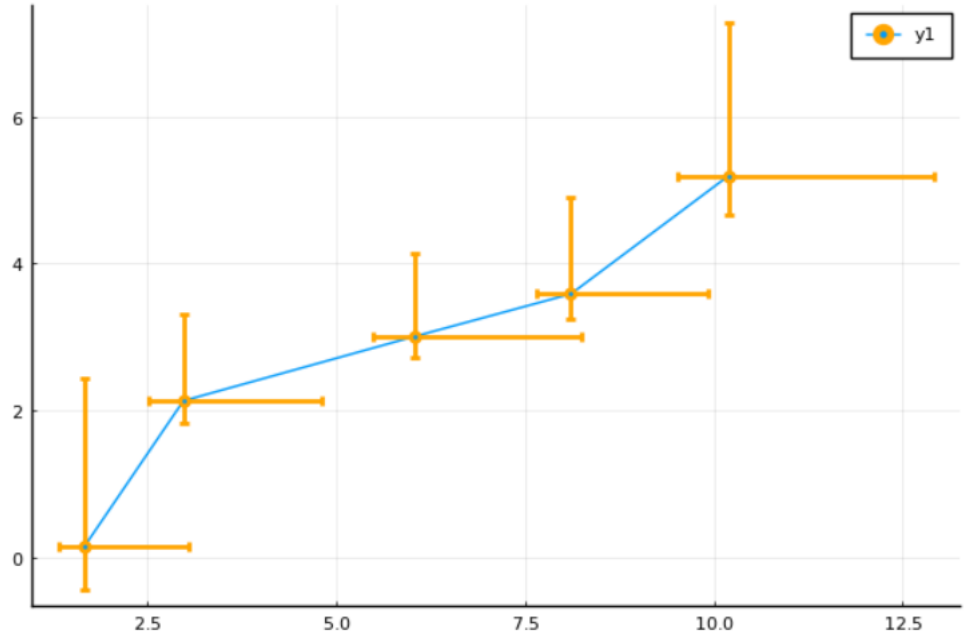
```
#Заполним область цветом:
plot(y,
ribbon=errs,
fill=:cyan
)
```




```
#Можно построить график ошибок по двум осям:
n = 10
x = [(rand()+1) .* randn(n) .+ 2i for i in 1:5]
y = [(rand()+1) .* randn(n) .+ i for i in 1:5]
f(v) = 1.96std(v) / sqrt(n)
xerr = map(f, x)
yerr = map(f, y)
x = map(mean, x)
y = map(mean, y)
plot(x, y,
xerr = xerr,
yerr = yerr,
marker = stroke(2, :orange)
)
```



```
#Можно построить график асимметричных ошибок по двум осям:
plot(x, y,
xerr = (0.5xerr,2xerr),
yerr = (0.5yerr,2yerr),
marker = stroke(2, :orange)
)
```



5.2.13. Использование пакета Distributions

Использование пакета Distributions

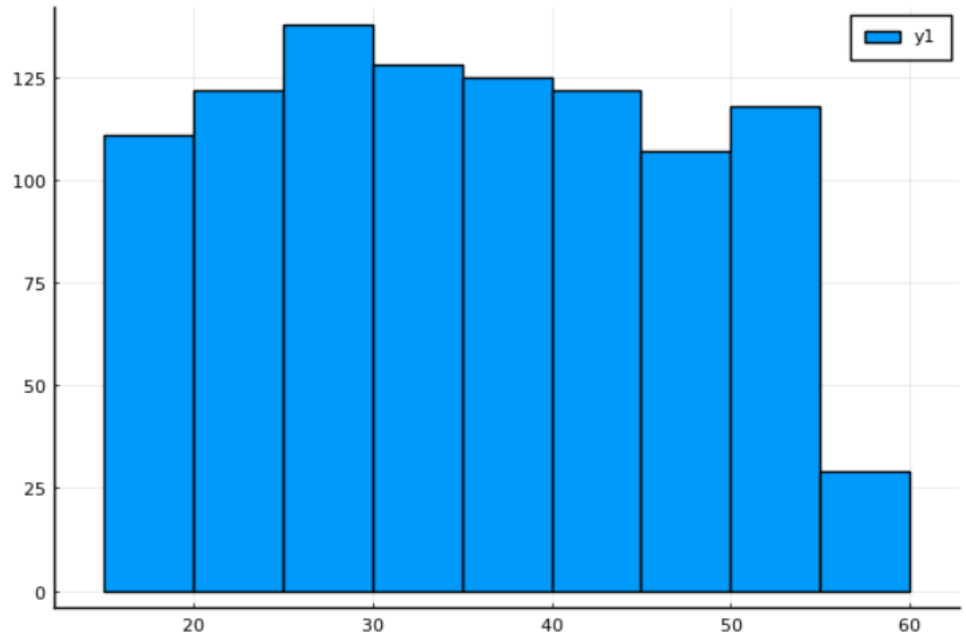
```
#Подгружаем пакет распределений:
import Pkg
Pkg.add("Distributions")
using Distributions
```

```
  Resolving package versions...
  Installed Rmath ──────────────────────── v0.6.1
  Installed OpenSpecFun_jll ────────────── v0.5.3+4
  Installed QuadGK ─────────────────────── v2.4.1
  Installed FillArrays ─────────────────── v0.9.7
  Installed Rmath_jll ──────────────────── v0.2.2+1
  Installed StatsFuns ──────────────────── v0.9.5
  Installed SpecialFunctions ───────────── v0.10.3
  Installed Distributions ──────────────── v0.24.0
  Installed PDMats ─────────────────────── v0.10.1
  Installed CompilerSupportLibraries_jll ─ v0.3.4+0
Updating `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
  [31c24e10] + Distributions v0.24.0
Updating `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  [e66e0078] + CompilerSupportLibraries_jll v0.3.4+0
  [31c24e10] + Distributions v0.24.0
  [1a297f60] + FillArrays v0.9.7
  [efe28fd5] + OpenSpecFun_jll v0.5.3+4
  [90014a1f] + PDMats v0.10.1
  [1fd47b50] + QuadGK v2.4.1
  [79098fc4] + Rmath v0.6.1
  [f50d1b31] + Rmath_jll v0.2.2+1
  [276daf66] + SpecialFunctions v0.10.3
  [4c63d2b9] + StatsFuns v0.9.5
  [4607b0f0] + SuiteSparse
[ Info: Precompiling Distributions [31c24e10-a181-5473-b8eb-7969acd0382f]
[ @ Base loading.jl:1278
```

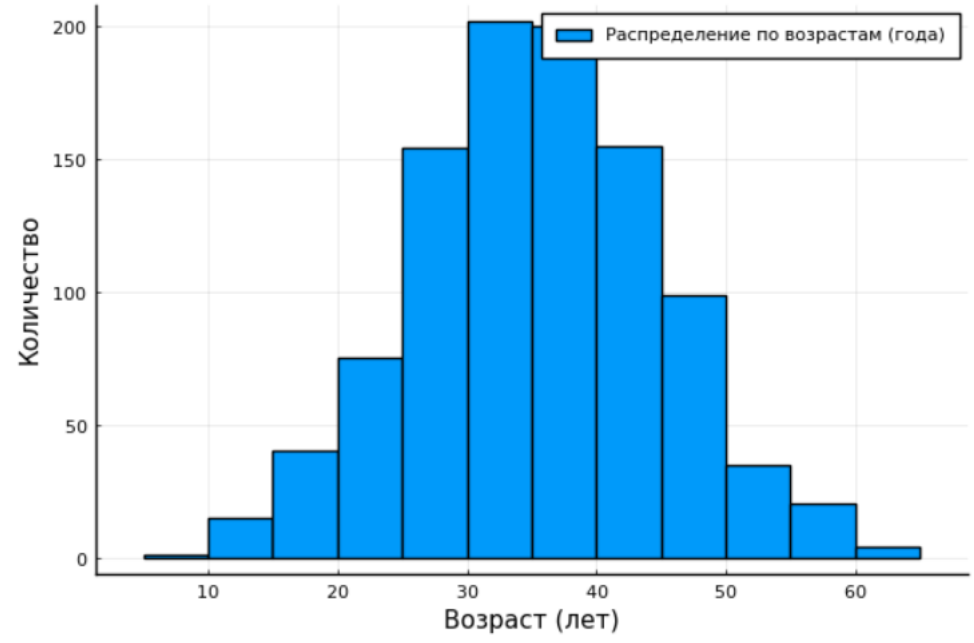
```
#Подгружаем pyplot():
pyplot()
```

```
Plots.PyPlotBackend()
```

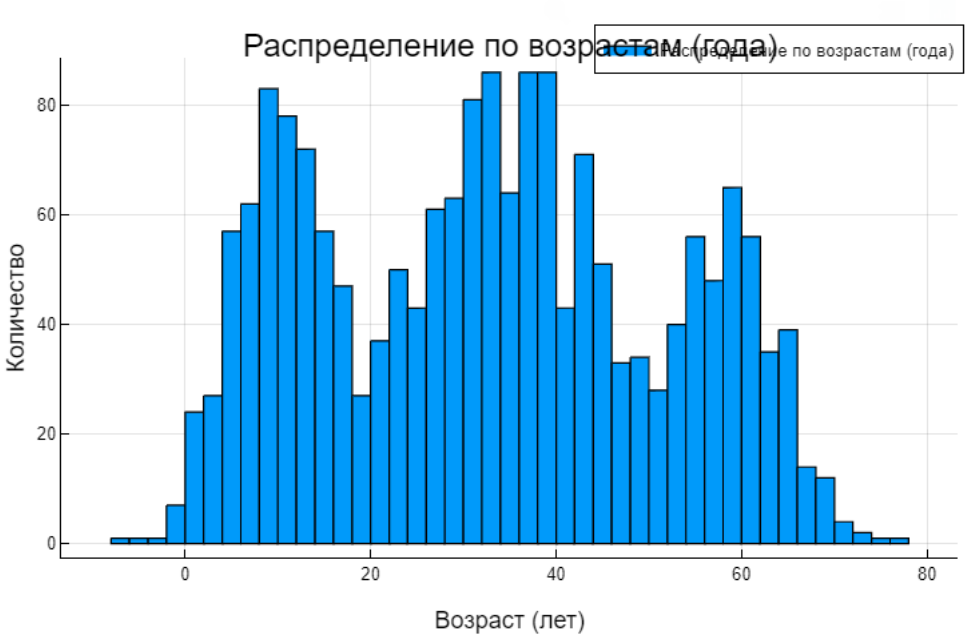
```
#Задаём массив случайных чисел:
ages = rand(15:55,1000)
#Строим гистограмму (рис. 5.41):
histogram(ages)
```




```
:
#Задаём нормальное распределение и строим гистограмму:
d=Normal(35.0,10.0)
ages = rand(d,1000)
histogram(
ages,
label="Распределение по возрастам (года)",
xlabel = "Возраст (лет)",
ylabel= "Количество"
)
```



```
: #Далее применим для построения нескольких гистограмм распределения людей по
  #возрастам на одном графике plotly():
  plotly()
  d1=Normal(10.0,5.0);
  d2=Normal(35.0,10.0);
  d3=Normal(60.0,5.0);
  N=1000;
  ages = (Float64)[];
  ages = append!(ages,rand(d1,Int64(ceil(N/2))));
  ages = append!(ages,rand(d2,N));
  ages = append!(ages,rand(d3,Int64(ceil(N/3))));
  histogram(
  ages,
  bins=50,
  label="Распределение по возрастам (года)",
  xlabel = "Возраст (лет)",
  ylabel= "Количество",
  title = "Распределение по возрастам (года)"
  )
```

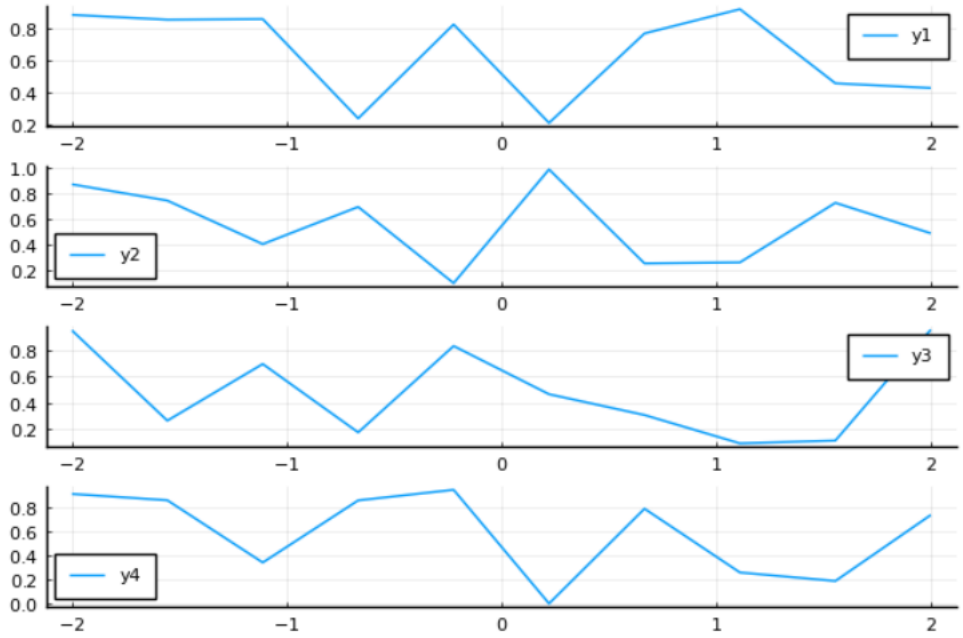


5.2.14. Подграфики

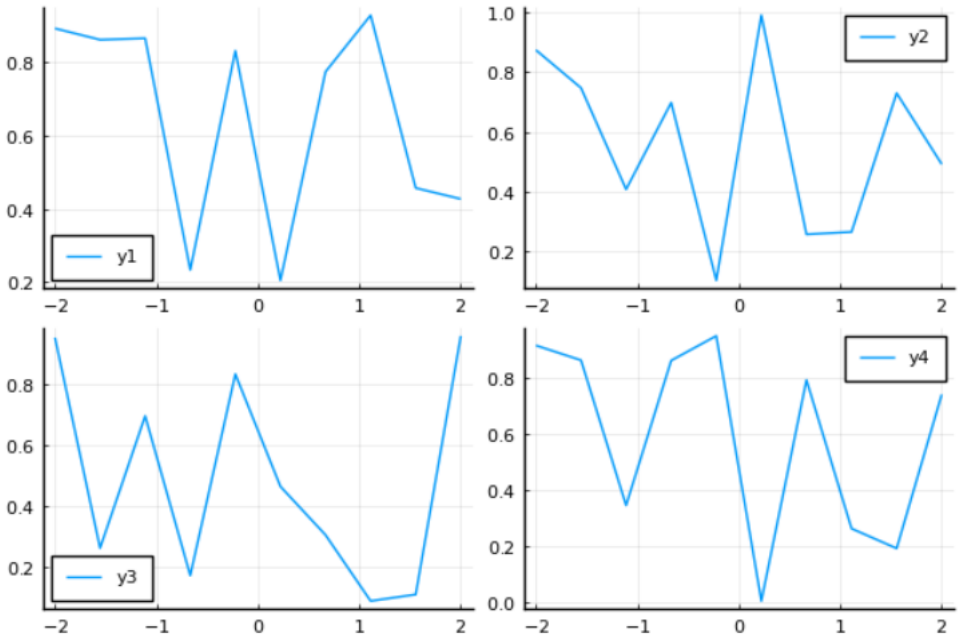
```
# подгружаем pyplot():
pyplot()
```

```
Plots.PyPlotBackend()
```

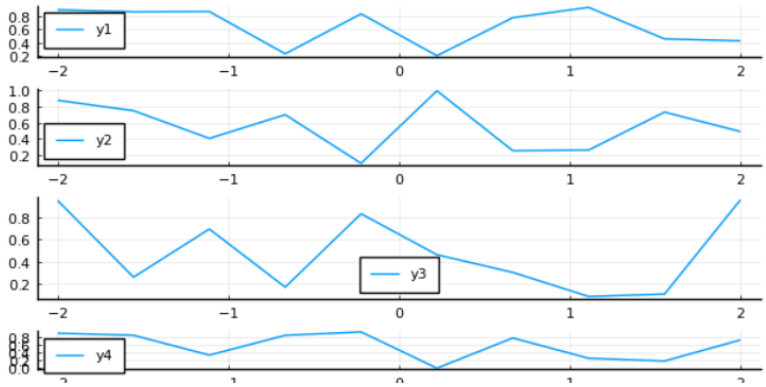
```
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
layout=(4,1)
)
```



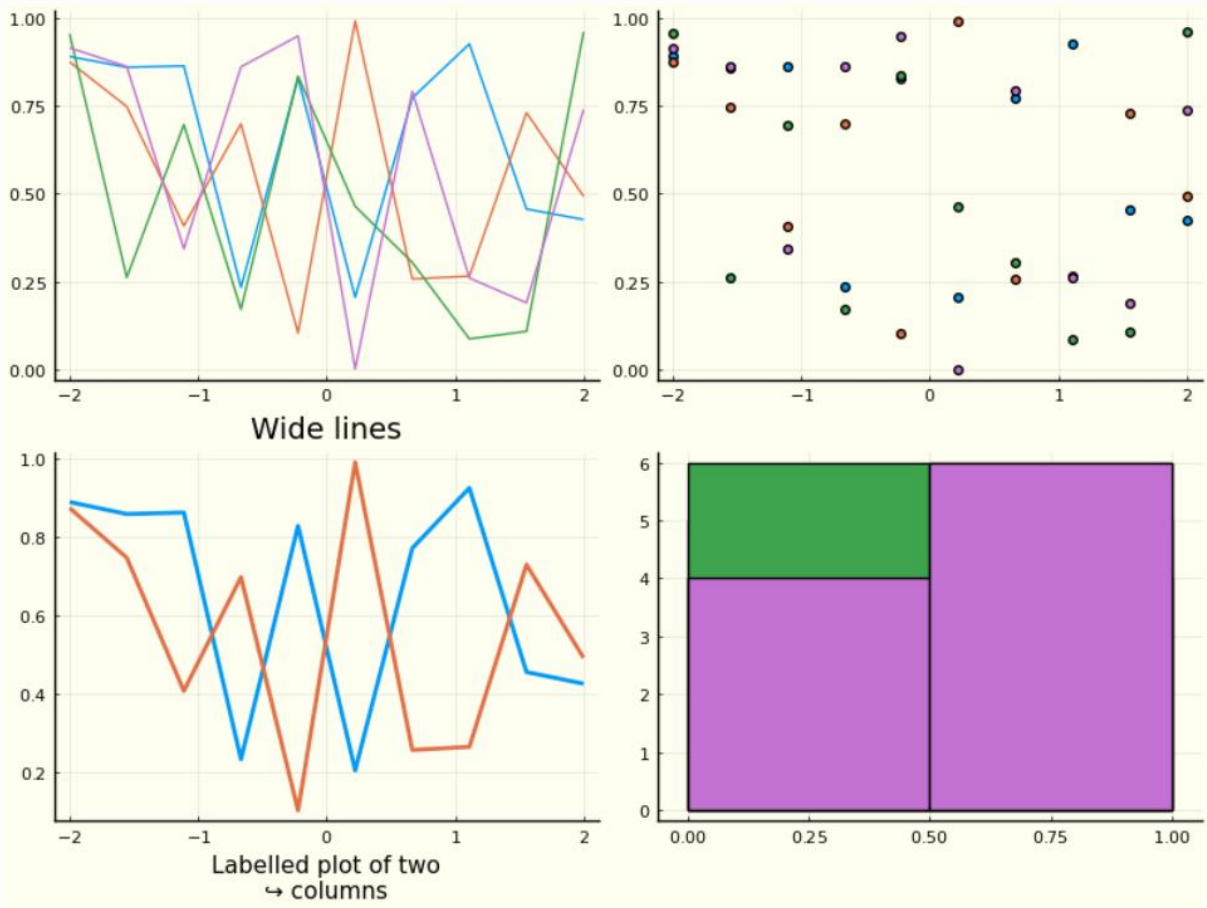

```
#Для автоматического вычисления сетки необходимо передать layout целое число:
plot(x,y,
layout=4
)
```



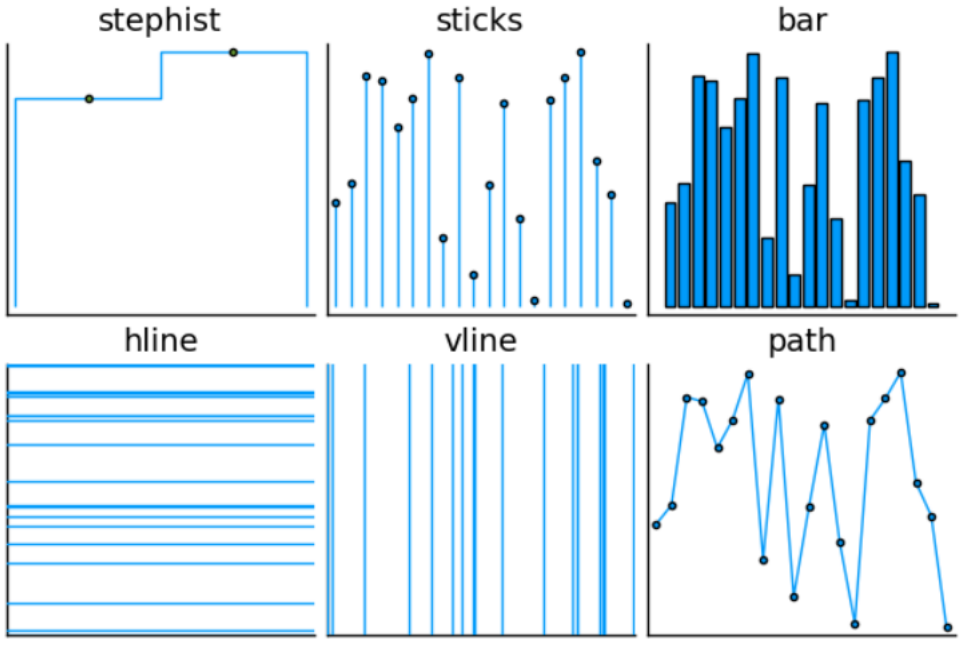
```
#Аргумент heights принимает в качестве входных данных массив с долями желаемых высот. Если в сумме дроби не составляют 1,0,
#то некоторые подзаголовки могут отображаться неправильно:
plot(x,y,
size=(600,300),
layout = grid(4,1,heights=[0.2,0.3,0.4,0.15])
)
```



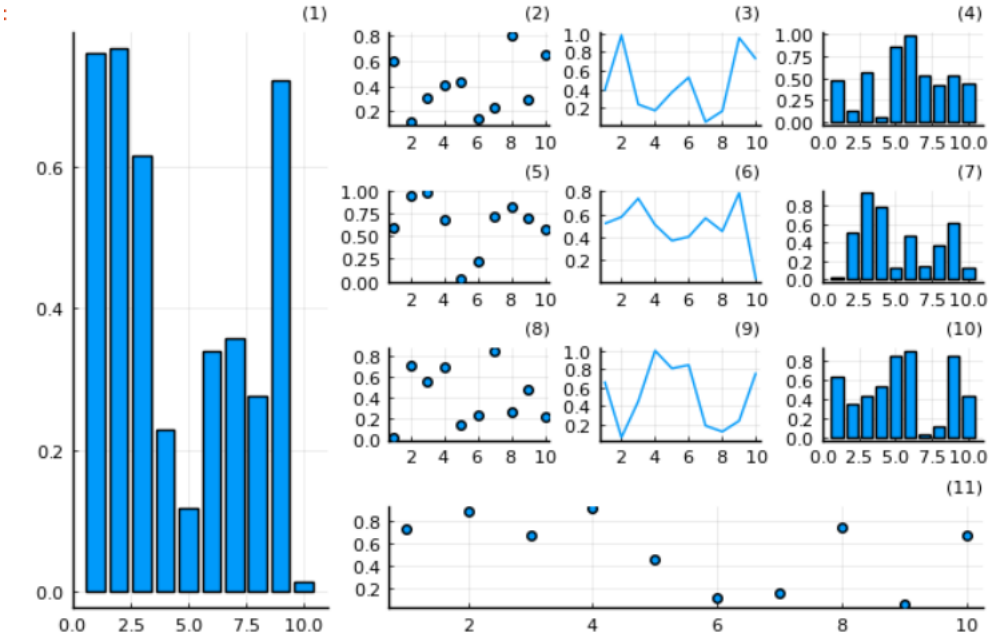

```
#Можно сгенерировать отдельные графики и объединить их в один, например, в сетке 2 × 2:
# график в виде линий:
p1 = plot(x,y)
# график в виде точек:
p2 = scatter(x,y)
# график в виде линий с оформлением:
p3 = plot(x,y[:,1:2],xlabel="Labelled plot of two
↪ columns",lw=2,title="Wide lines")
# 4 гистограммы:
p4 = histogram(x,y)
plot(
p1,p2,p3,p4,
layout=(2,2),
legend=false,
size=(800,600),
background_color = :ivory
)
```




```
#Разнообразные варианты представления данных:
pyplot()
seriestypes = [:stephist, :sticks, :bar, :hline, :vline, :path]
titles =["stephist" "sticks" "bar" "hline" "vline" "path"]
plot(rand(20,1), st = seriestypes,
    layout = (2,3),
    ticks=nothing,
    legend=false,
    title=titles,
    m=3)
```



```
#Применение макроса @layout наиболее простой способ определения сложных макетов.
#Точные размеры могут быть заданы с помощью фигурных скобок, в противном случае
#пространство будет поровну разделено между графиками:
l = @layout [ a{0.3w} [grid(3,3)
b{0.2h} ]]
plot(
rand(10,11),
layout = l, legend = false, seriestype = [:bar :scatter :path],
title = ["($i)" for j = 1:1, i=1:11], titleloc = :right, titlefont = font(8)
)
```



Вывод:

Получал Навыки на синтаксис языка Julia для построения графиков