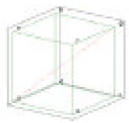


📕 Table of Contents


```
begin
        using PlutoUI  , Plots   ,DataFrames   ,HypertextLiteral   ,LaTeXStrings
,Symbolics   ,LinearAlgebra   ,RowEchelon   ,CairoMakie
        using Plots: plot,plot!,text,scatter,scatter!,theme,surface,surface!,gr
        gr()
        theme(:bright)
        @htl("""<script src="https://cdn.bootcdn.net/ajax/libs/mathjax/3.2.0/es5/tex-svg-full.min.js"></script>
        """)
        PlutoUI.TableOfContents()
 end
```

ch02 sec2.5 向量的长度

Outcomes

- A 计算n 维空间中点的距离
- B 计算向量的长度
- C 标准化一个向量

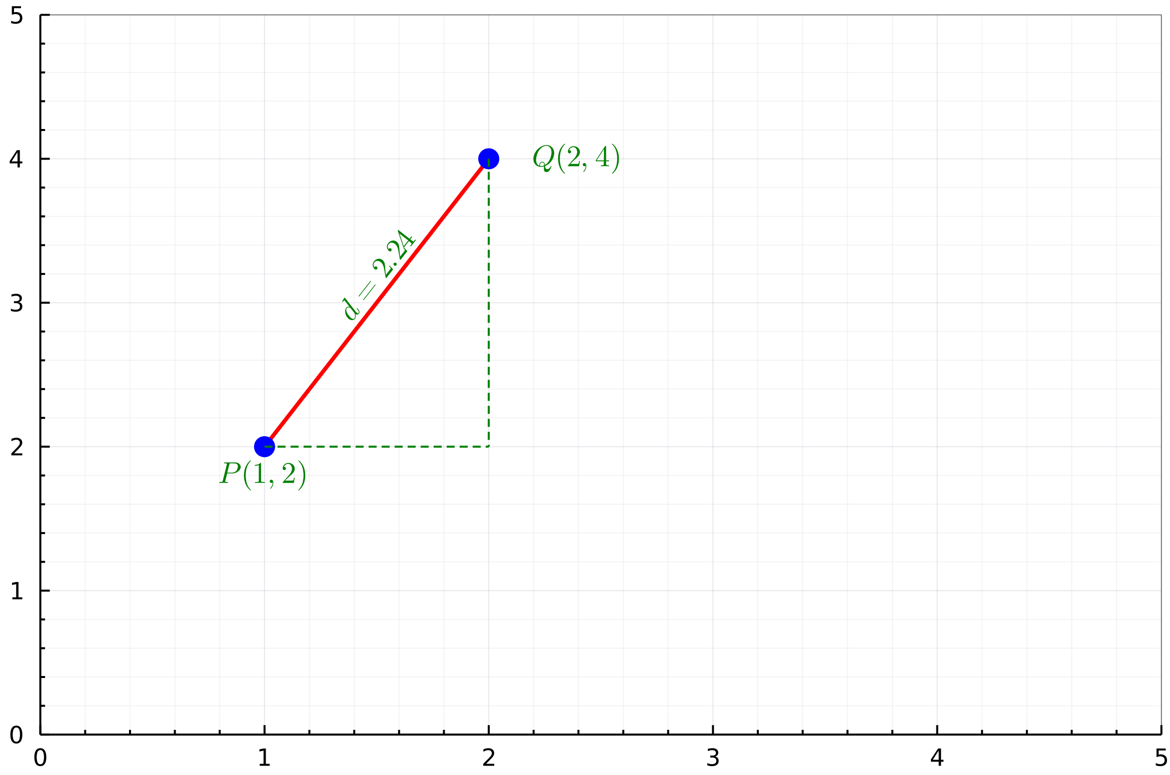
在物理中, 我们可以借助数字度量各种物理性质, 比如可以度量身高,可以度量体重, 这种度量的一般形式总结起来就是长度.

对于一个数学对象-向量也可以度量它的长度, 在同一个多维度空间下度量向量长度是有意义的. 这种度量反映的是向量的一个性质.

```
md"""
在物理中, 我们可以借助数字度量各种物理性质, 比如可以度量身高,可以度量体重, 这种度量的一般形式总结起来就是长度.

对于一个数学对象-向量也可以度量它的长度, 在同一个多维度空间下度量向量长度是有意义的.  这种度量反映的是向量的一个性质.
"""
```

二维空间中两个点之间的距离

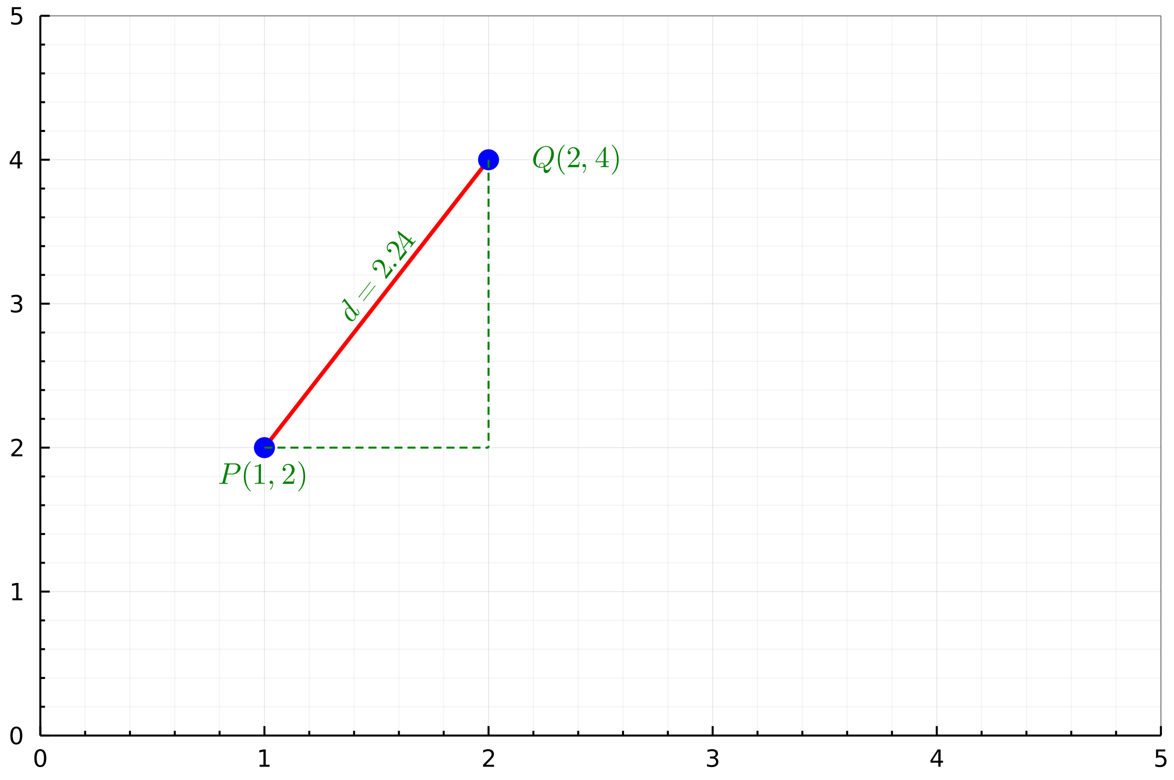
如下图:



二维空间中两点间的距离表示为:

$$d(P,Q)=\sqrt{|p_1-q_1|^2+|p_2-q_2|^2}=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2}$$

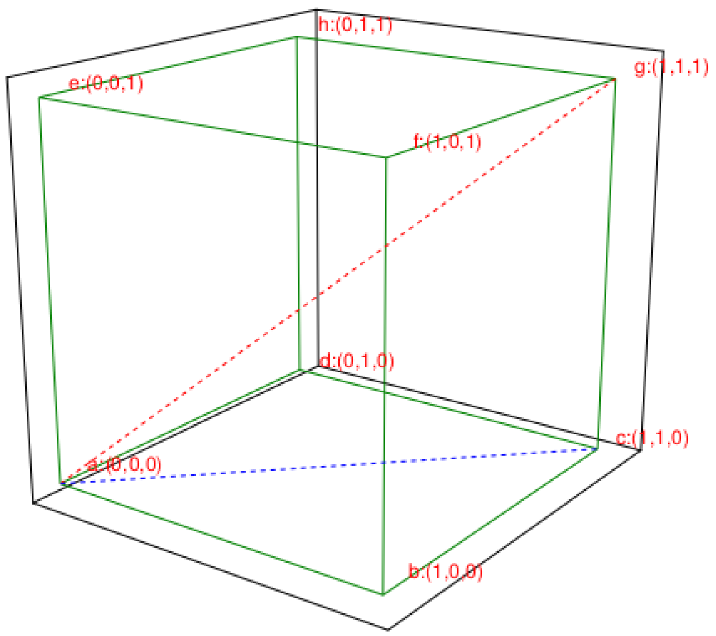
```
md"""
## 二维空间中两个点之间的距离

如下图:

 $(store["2points"])

二维空间中两点间的距离表示为:

$d(P,Q)=\sqrt{|p_1-q_1|^2+|p_2-q_2|^2}=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2}$

"""
```




```
let

    P,Q=[1,2],[2,4]
    d=round(dist(P,Q),digits=2)
    ann=[
        (1,1.8,text(L"P(1,2)",pointsize=10,color=:green)),
        (2.4,4,text(L"Q(2,4)",pointsize=10,color=:green)),
         (1.5,3.2,text(L"d=%$(d)",pointsize=10,color=:green,rotation=52)),

    ]



    plot([P[1],Q[1]],[P[2],Q[2]],label=false,lw=2,color=:red,ann=ann)
    p1=scatter!([P[1],Q[1]],[P[2],Q[2]],mc=:blue,label=false,xlims=(0,5),ylims=
(0,5),frame=:semi)
    plot!([P[1],2],[P[2],2],label=false,lw=1,color=:green,ls=:dash)
    plot!([Q[1],2],[Q[2],2],label=false,lw=1,color=:green,ls=:dash)

    save("2points",p1)

end
```

三维空间中两个点之间的距离

如下图:



三维维空间中两点间的距离表示为:

$$d(P,Q)=\sqrt{|p_1-q_1|^2+|p_2-q_2|^2+|p_3-q_3|^2}=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+(p_3-q_3)^2}$$

例如图中a,g 点的距离可计算为:

```
md"""
## 三维空间中两个点之间的距离

如下图:

  $(store["3points"])

三维维空间中两点间的距离表示为:

$d(P,Q)=\sqrt{|p_1-q_1|^2+|p_2-q_2|^2+|p_3-q_3|^2}=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+(p_3-q_3)^2}$

例如图中$a,g$ 点的距离可计算为:

"""
```

```
1.7320508075688772
```

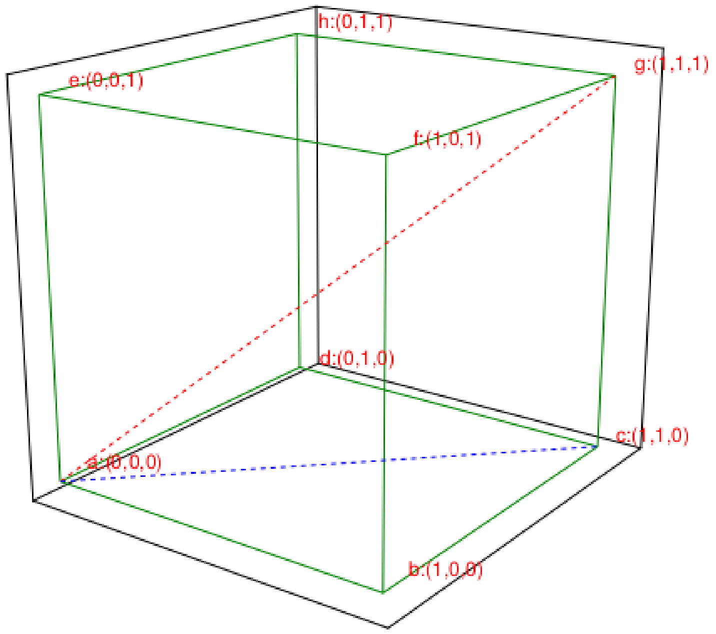
这就是欧几里得定律(勾股定律), 欧式定律在更高维度的空间下仍然是适用的.

Definition

n 维空间中两点之间的距离公式:

$$d(P,Q)=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2}$$

```
md"""
这就是欧几里得定律(勾股定律), 欧式定律在更高维度的空间下仍然是适用的.

!!! definition

	$n$ 维空间中两点之间的距离公式:

	$$d(P,Q)=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2}$$
"""
```




```
let
    fig=makeCubic()
    figure=fig[1]
    ax=fig[2]
    lines!(ax,lag[:,1],lag[:,2],lag[:,3],color=:red,linestyle=:dash,linewidth=1)
    lines!(ax,lac[:,1],lac[:,2],lac[:,3],color=:blue,linestyle=:dash,linewidth=1)
    save("3points",figure)

end
```

Example

example 4

$$R^4$$

空间中的两点$p=(1,2,-4,6)$ 和$Q=(2,3,-1,0)$ 之间的距离为:

$$d(P,Q)=\sqrt{(1-2)^2+(2-3)^2+(-4-(-1))^2+(6-0)^2}=\sqrt{47}$$

```
md"""

!!! example
    example 4

    $R^4$ 空间中的两点$p=(1,2,-4,6)$ 和$Q=(2,3,-1,0)$ 之间的距离为:

    $d(P,Q)=\sqrt{(1-2)^2+(2-3)^2+(-4-(-1))^2+(6-0)^2}=\sqrt{47}$
"""
```

Example

example 5 描述一下R^3 空间中到两点$p=(1,2,3)$ 和$Q=(0,1,2)$ 距离为相等的点的数学性质:

设这个点为Z,坐标向量表示为:(z_1,z_2,z_3)

到两点的距离相同,所以有:

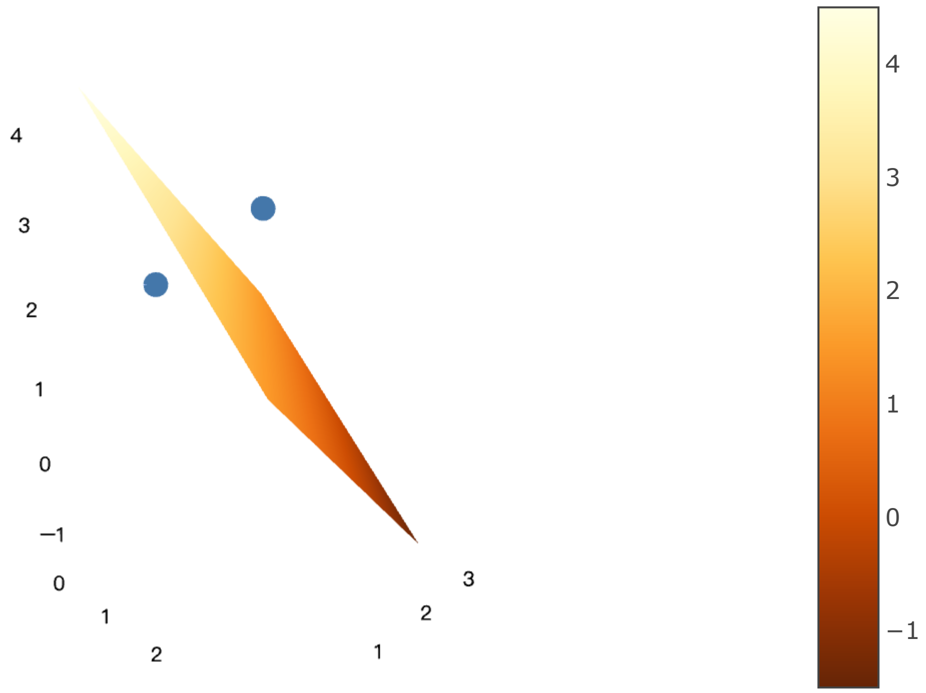
$$\sqrt{(z_1-1)^2+(z_2-2)^2+(z_3-3)^2}=\sqrt{(z_1-1)^2+(z_2-1)^2+(z_3-)^2}$$

展开化简得到:

$$2p_1+2p_2+2p_3=9$$

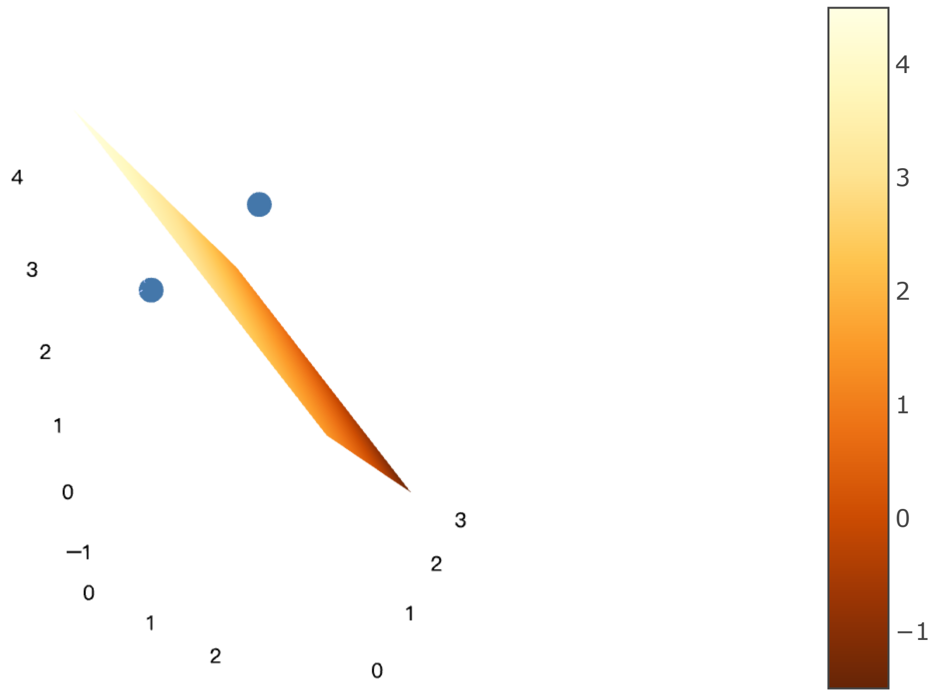
所以三维空间中到两点距离相等的点是一个平面集合.

如图:




```
md"""

!!! example
    example 5
    描述一下$R^3$ 空间中到两点$p=(1,2,3)$ 和$Q=(0,1,2)$ 距离为相等的点的数学性质:



设这个点为$Z$,坐标向量表示为:$(z_1,z_2,z_3)$

到两点的距离相同,所以有:

```

```
$\sqrt{(z_1 -1)^2 +(z_2 -2)^2 +(z_3-3)^2} =\sqrt{(z_1-1)^2 +(z_2 -1)^2 +(z_3 -)^2}$

展开化简得到:

$2p_1+2p_2+2p_3=9$

所以三维空间中到两点距离相等的点是一个平面集合.

如图:

$(store["face"])


"""
```




```
let
    plotly()
    P,Q=[1,2,3],[0,1,2]
    xspan,yspan=0:0.02:3,0:0.02:3
    f(x,y)=(9-2x-2y)/2
    zspan=[f(x,y) for x in xspan ,y in yspan]
    scatter([P[1],Q[1]],[P[2],Q[2]],[P[3],Q[3]],label=false,ms=2)
    face=surface!(xspan,yspan,zspan, label=false)
    save("face",face)
end
```

向量的长度

Definition

在R^n 空间中的坐标向量u: $\begin{bmatrix} u_1\\ \vdots \\u_n \end{bmatrix}$ 的长度定义为: $||u||$,由公式:

$$||u||=\sqrt{{u_1}^2+\cdots+{u_n}^2}$$

计算. 称为量值, 或者叫做范数, 这里实际应该叫做欧几里得范数

向量长度的性质:

Props

- $||u|| \geq 0$
- $||u|| = 0$,有且只有 $u = 0$
- $||ku|| = |k|\ ||u||$

Definition

单位向量

当一个向量u 的长度为 $||u|| = 1$ 时, 这个向量就称为单位向量

一个向量除以它的长度就可以得到单位向量

$$v = \frac{1}{||u||}u$$

v 向量与u 向量方向相同,长度不同, v 的长度为 1

```
md"""
##  向量的长度

!!! definition

    在$R^n$ 空间中的坐标向量u:$\begin{bmatrix} u_1\\ \vdots \\u_n \end{bmatrix}$ 的长度定义为: $||u||$,由公式:

    $||u||=\sqrt{{u_1}^2+\cdots+{u_n}^2}$ 计算. 称为量值, 或者叫做范数, 这里实际应该叫做欧几里得范数


向量长度的性质:

!!! props

    -  $||u|| \geq  0$
```

```
    - $||u||  =  0$,有且只有 $u=0$
    - $||ku|| =  |k| \ ||u||$


!!! definition

    单位向量

    当一个向量$u$ 的长度为 $||u||=1$ 时, 这个向量就称为单位向量



一个向量除以它的长度就可以得到单位向量

$v=\frac{1}{||u||}u$

 $v$ 向量与$u$ 向量方向相同,长度不同,  $v$ 的长度为 1
"""
```

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

```
begin
        store=Dict()

        function save(key::String, dict)
            store[key]=dict
        end

        function read(key::String)
            return  store[key]
        end


        function vec_plot(v1,v2,ls=:solid)
            v11,v12=v1[1],v1[2]
            v21,v22=v2[1],v2[2]
            return plot([v11,v21],[v12,v22],label=false, arrow=true, lw=2,ls=ls)
        end

        function vec_plot!(v1,v2,ls=:solid)
            v11,v12=v1[1],v1[2]
            v21,v22=v2[1],v2[2]
            return plot!([v11,v21],[v12,v22],label=false, arrow=true, lw=2,ls=ls)
        end

        function vec_plot3d(v1,v2,ls=:solid)
            v11,v12,v13=v1[1],v1[2],v1[3]
            v21,v22,v23=v2[1],v2[2],v2[3]
            return plot([v11,v21],[v12,v22],[v13,v23],label=false, lw=1,ls=ls)
        end

        function vec_plot3d!(v1,v2,ls=:solid)
            v11,v12,v13=v1[1],v1[2],v1[3]
            v21,v22,v23=v2[1],v2[2],v2[3]
            return plot!([v11,v21],[v12,v22],[v13,v23],label=false, lw=1,ls=ls)
        end

        function dist(p,q)
            length=size(p)
            arr=[(abs(p[i]-q[i]))^2 for i in 1:length[1]]
            return sqrt(sum(arr))
        end


end
```

getRotz (generic function with 1 method)

```
begin
    yshift=0.1
    a,b,c,d=[0 0 0],[1 0 0],[1 1 0],[0 1 0]
    e,f,g,h=[0 0 1],[1 0 1],[1 1 1],[0 1 1]
    ma,lowerplane,upperplane=[a;b;c;d;e;f;g;h],[a;b;c;d;a],[e;f;g;h;e]
    lab,lae,lbf,lcg=[a;b],[a;e],[b;f],[c;g]
    ldh,lag,lad,lbh,lac=[d;h],[a;g],[a;d],[b;h],[a;c]


    function makeCubic()
     figure = Figure()
    ax = Axis3(figure[1, 1], aspect = :data, perspectiveness =0.3, elevation =
0.1*π,azimuth=-0.3*π, viewmode=:fit)
    hidedecorations!(ax)
    #ax.protrusions = (0, 0, 0, 20)

    lines!
(ax,lowerplane[:,1],lowerplane[:,2],lowerplane[:,3],linewidth=1,color=:green)
    lines!
(ax,upperplane[:,1],upperplane[:,2],upperplane[:,3],linewidth=1,color=:green)
    lines!(ax,lae[:,1],lae[:,2],lae[:,3],linewidth=1,color=:green)
    lines!(ax,lbf[:,1],lbf[:,2],lbf[:,3],linewidth=1,color=:green)
    lines!(ax,lcg[:,1],lcg[:,2],lcg[:,3],linewidth=1,color=:green)
    lines!(ax,ldh[:,1],ldh[:,2],ldh[:,3],linewidth=1,color=:green)
    #lines!(ax,lag[:,1],lag[:,2],lag[:,3],color=:red,linestyle=:dash,linewidth=1)
     # arrows!([Point3f(0,0,0)],[Vec3f(0.97,0.97,0.97)],arrowsize =
0.04,arrowcolor=:red,linecolor=:red,linestyle=:dot)  #定义矢量的方法

    text!("a:(0,0,0)", position =(0,0+yshift,0), align = (:left,
:baseline),textsize=12,color=:red)
    text!("b:(1,0,0)", position = (1,0+yshift,0), align = (:left,
:baseline),textsize=12,color=:red)
    text!("c:(1,1,0)", position =(1,1+yshift,0), align = (:left,
:center),textsize=12,color=:red)
    text!("d:(0,1,0)", position =(0,1+yshift,0), align = (:left,
:center),textsize=12,color=:red)

    text!("e:(0,0,1)", position = (0,0+yshift,1), align = (:left,
:baseline),textsize=12,color=:red)
    text!("f:(1,0,1)", position = (1,0+yshift,1), align = (:left,
:baseline),textsize=12,color=:red)
    text!("g:(1,1,1)", position = (1,1+yshift,1), align = (:left,
:baseline),textsize=12,color=:red)
    text!("h:(0,1,1)", position = (0,1+yshift,1), align = (:left,
:baseline),textsize=12,color=:red)
    return figure,ax  #返回绘图对象和坐标轴系统对象
  end

  function dist2(vec1,vec2)
     return sqrt((vec2[1]-vec1[1])^2+(vec2[2]-vec1[2])^2+(vec2[3]-vec1[3])^2)
  end

  function slope(vec1,vec2)
     return [vec2[1]-vec1[1] vec2[2]-vec1[2] vec2[3]-vec1[3]]
  end

    function getRotz(theta::Float64)
```

```
        return [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0;0 0 1]
    end


end
```