```julia
using Fuzzy
```

```julia
using Plots
```

```julia
using PlutoUI
```

```
distance = 0.0:0.10101010101010101:10.0
```

```julia
distance = range(0, stop=10, length=100)
```

```
dist =
 Dict("far" ⇒ TriangularMF(6, 10, 10), "too close" ⇒ TriangularMF(0, 0, 4), "close" ⇒
```

```julia
dist = Dict(
        "too close" => TriangularMF(0, 0, 4),
        "close" => TrapezoidalMF(2, 4, 6, 8),
    "far" => TriangularMF(6, 10, 10)
        )
```

```
dist_chart =

 Dict("names" ⇒ 1×3 Matrix{String}:          , "values" ⇒ [[0.0, 0.0, 0.0, 0.0, 0.0,
                 "far"  "too close"  "close"
```

```julia
dist_chart = chart_prepare(dist, distance)
```

```
speed = 0.0:0.25252525252525254:25.0
```

```julia
speed = range(0, stop=25, length=100)
```

```
SP =
 Dict("slow" ⇒ TrapezoidalMF(3, 5, 7, 9), "too slow" ⇒ TrapezoidalMF(0, 0, 2, 4), "opti
```

```julia
SP = Dict(
        "too slow" => TrapezoidalMF(0, 0, 2, 4),
        "slow" => TrapezoidalMF(3, 5, 7, 9),
    "optimum" => TrapezoidalMF(8, 10, 12, 14),
    "fast" => TrapezoidalMF(13, 15, 17, 19),
    "too fast" => TrapezoidalMF(18, 20, 22, 24),
        )
```

```
SP_chart =
 Dict("names" ⇒ 1×5 Matrix{String}:                                   , "values" ⇒ [[0.0,
                 "slow"  "too slow"  "optimum"  "fast"  "too fast"
```

```julia
SP_chart = chart_prepare(SP, speed)
```

```
brake = 0.0:0.25252525252525254:25.0
```

```julia
brake = range(0, stop=25, length=100)
```

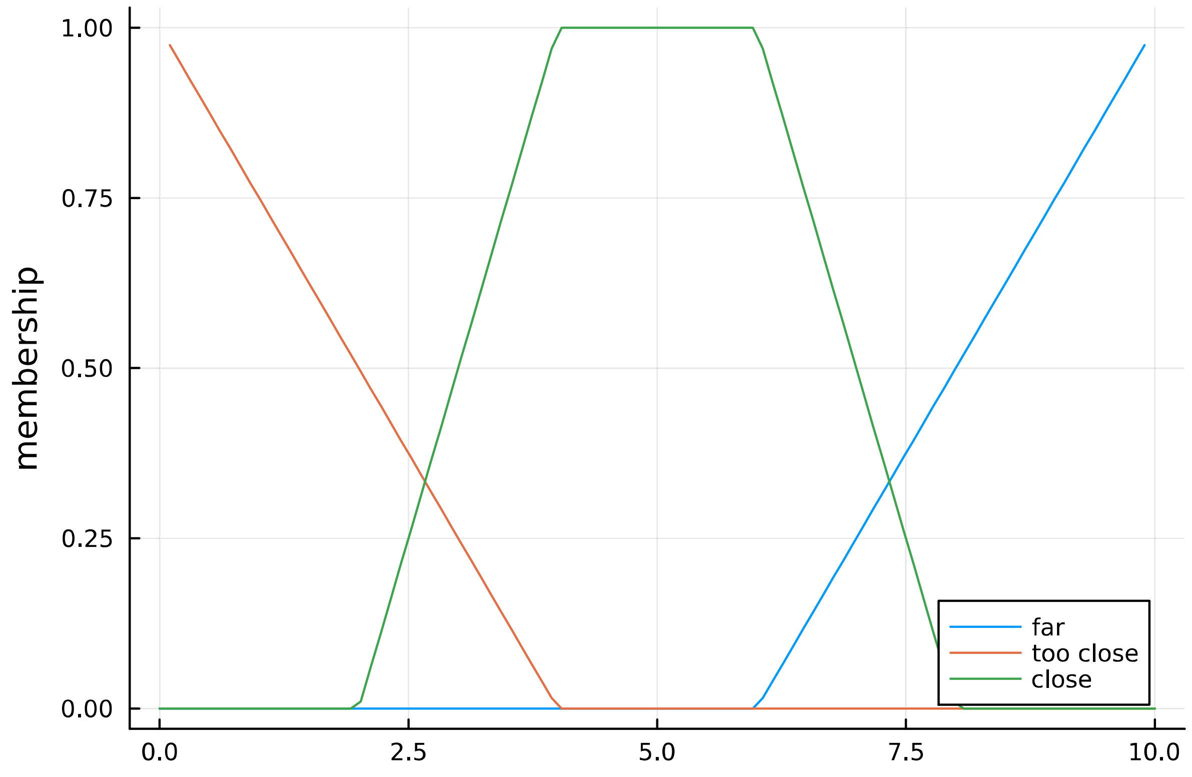
```
b_force =
  Dict("dec_slightly" ⇒ TriangularMF(4, 8, 12), "dec_greatly" ⇒ TriangularMF(0, 4, 8), "
```

```
b_force = Dict(
        "dec_greatly" => TriangularMF(0, 4, 8),
    "dec_slightly" => TriangularMF(4, 8, 12),
    "no_reaction" => TriangularMF(8, 12, 16),
    "inc_slightly" => TriangularMF(12, 16, 20),
    "inc_greatly" => TriangularMF(16, 20, 24)
        )
```

```
b_force_chart =
  Dict("names" ⇒ 1×5 Matrix{String}:
   "dec_slightly"  "dec_greatly"  "inc_slightly"  "no_reaction"  "inc_gre
```

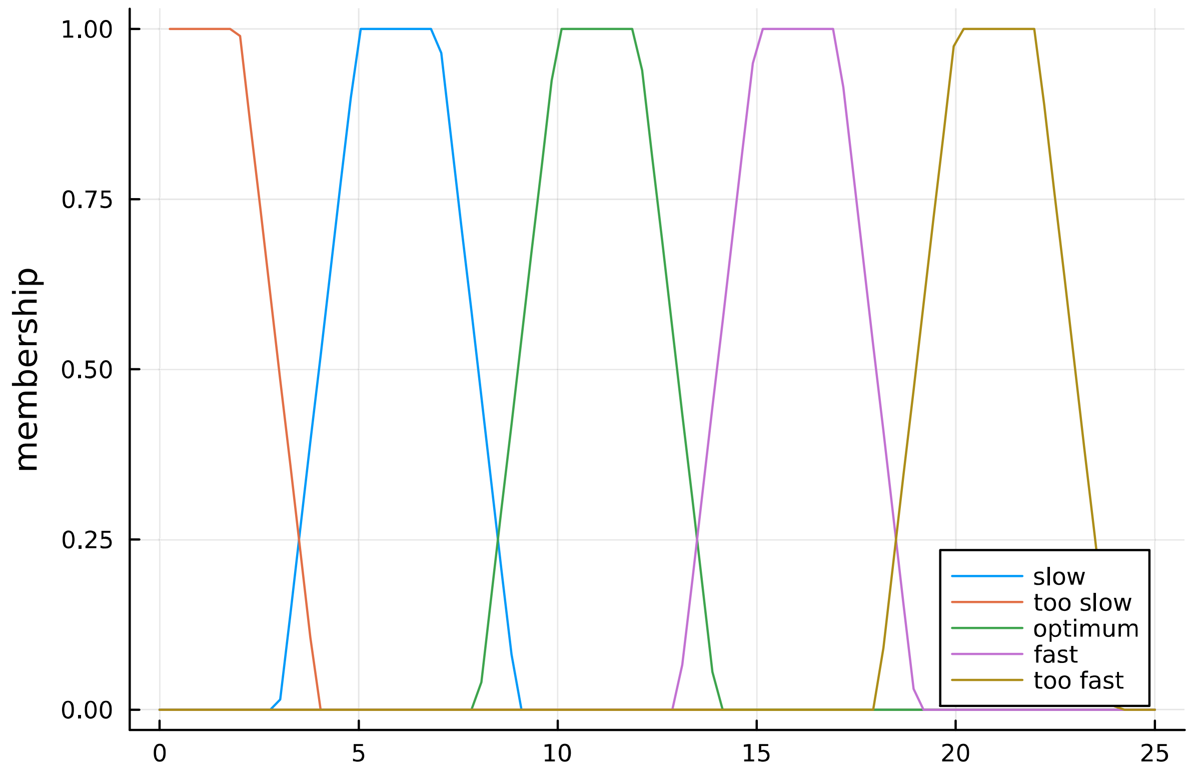
```
b_force_chart = chart_prepare(b_force, brake)
```

```
p1_distance =
```



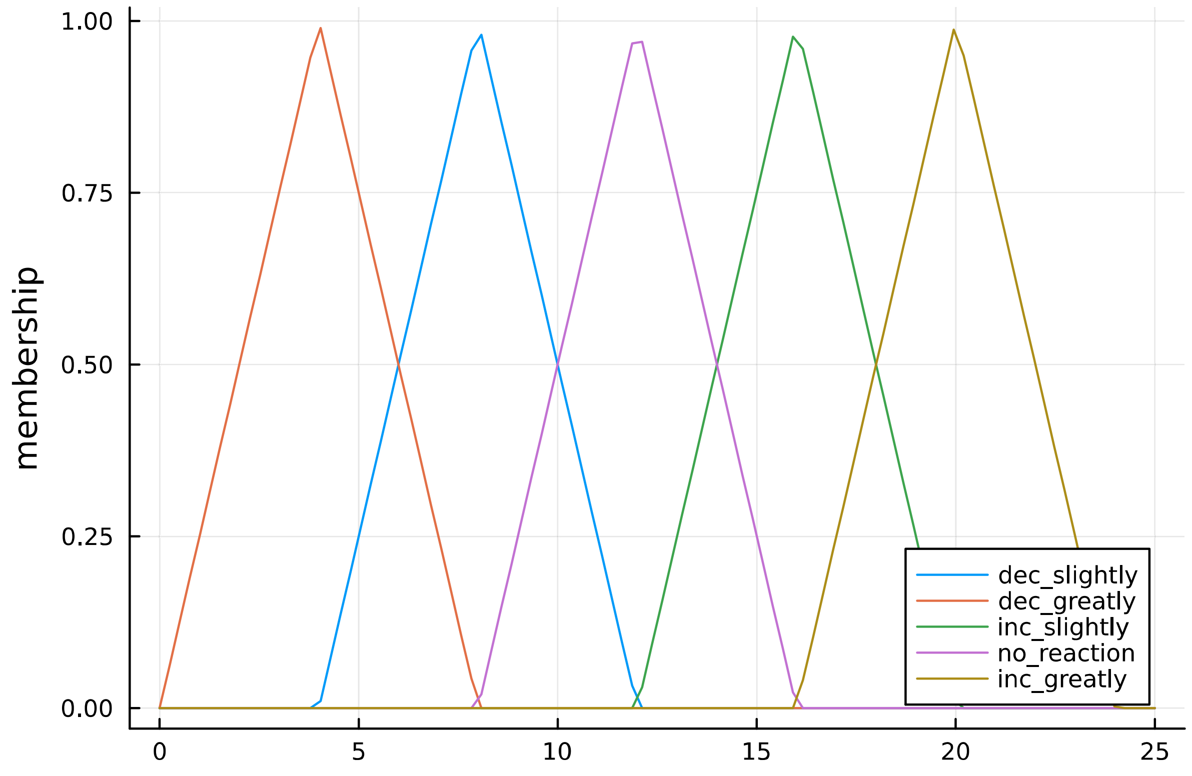
```
p1_distance = plot(distance, dist_chart["values"], ylabel="membership",
label=dist_chart["names"], legend=:bottomright)
```

p2_speed =



```
p2_speed = plot(speed, SP_chart["values"], ylabel="membership",
label=SP_chart["names"], legend=:bottomright)
```

p3_brake =



```
p3_brake = plot(brake, b_force_chart["values"], ylabel="membership",
label=b_force_chart["names"], legend=:bottomright)
```

```
rule_1 =  Rule(["too close"], "inc_greatly", "MAX")
```

```
rule_1 = Rule(["too close"], "inc_greatly", "MAX")
```

```
rule_2 =  Rule(["close", "too fast"], "inc_slightly", "MAX")
 • rule_2 = Rule(["close", "too fast"], "inc_slightly", "MAX")

rule_3 =  Rule(["close", "optimum"], "inc_slightly", "MAX")
 • rule_3 = Rule(["close", "optimum"], "inc_slightly", "MAX")

rule_4 =  Rule(["far", "optimum"], "no_reaction", "MAX")
 • rule_4 = Rule(["far", "optimum"], "no_reaction", "MAX")

rule_5 =  Rule(["far", "slow"], "dec_slightly", "MAX")
 • rule_5 = Rule(["far", "slow"], "dec_slightly", "MAX")

rule_6 =  Rule(["far", "too slow"], "inc_slightly", "MIN")
 • rule_6 = Rule(["far", "too slow"], "inc_slightly", "MIN")

rules =
 [Rule(["too close"], "inc_greatly", "MAX"), Rule(["close", "too fast"], "inc_slightly",
 • rules = [rule_1, rule_2, rule_3, rule_4, rule_5, rule_6]

fis =
 FISMamdani([Dict("far" ⇒ TriangularMF(6, 10, 10), "too close" ⇒ TriangularMF(0, 0, 4),
 • fis = FISMamdani([dist, SP], b_force, rules)

NaN
 • eval_fis(fis, [dist1, speed1])

 0.0
 • @bind speed1 Slider(0.:25.; default=0, show_value=true)

 0.0
 • @bind dist1 Slider(0.:10. ,default=0, show_value=true)

 • Enter cell code...
```