Macroeconomía Internacional

Francisco Roldán
IMF

June 2021

Cuándo se paga la deuda?

- Defaults soberanos coinciden con
  - Aumentos de la tasa de interés (riesgo país)
  - Recesiones

- Objetivo estudiar la dinámica conjunta de
  1. Deuda
  2. Tasas de interés
  3. Producto
  4. Cuenta corriente

Cuándo se paga la deuda?

- Defaults soberanos coinciden con
  - Aumentos de la tasa de interés (riesgo país)
  - Recesiones

- Objetivo   estudiar la dinámica conjunta de
  1. Deuda
  2. Tasas de interés
  3. Producto
  4. Cuenta corriente

Cuándo se paga la deuda?

- Defaults soberanos coinciden con
  - Aumentos de la tasa de interés (riesgo país)
  - Recesiones

- Objetivo estudiar la dinámica conjunta de
  1. Deuda
  2. Tasas de interés
  3. Producto
  4. Cuenta corriente

Arellano, C. (2008): "Default Risk and Income Fluctuations in Emerging Economies," *American Economic Review*, 98, 690–712.

Arellano (2008)

Replicación

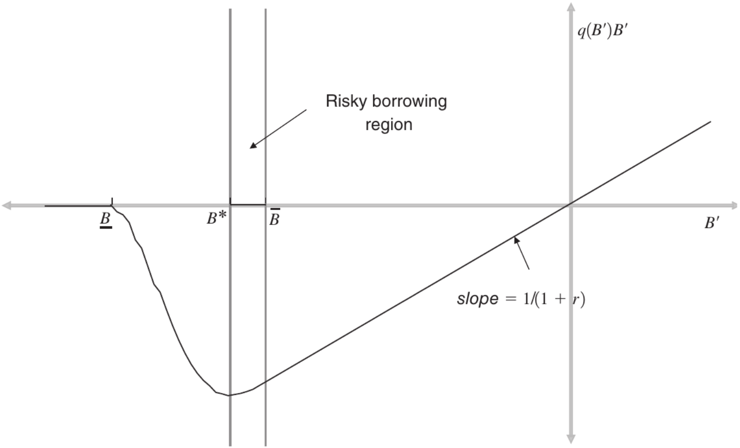


FIGURE 2. TOTAL RESOURCES BORROWED

Plan

Vamos a proceder en etapas

1. Problema de un agente con ingreso aleatorio y mercados incompletos
    - Solamente un activo (deuda) libre de riesgo
    - Escritura recursiva, ecuación de Bellman
    - Encontrar la función de valor via `vfi`
    - Distinto de McCall: un control continuo

2. Agregar default
    - Como McCall: hay una elección entre dos opciones en cada período
    - Complicación: ahora el precio de la deuda depende de la probabilidad de default

3. Reinterpretar
    - Consumo/ahorro del agente $\iff$ Endeudamiento de la economía pequeña y abierta

Plan

Vamos a proceder en etapas

1. Problema de un agente con ingreso aleatorio y mercados incompletos
   - Solamente un activo (deuda) libre de riesgo
   - Escritura recursiva, ecuación de Bellman
   - Encontrar la función de valor via `vfi`
   - Distinto de McCall: un control continuo

2. Agregar default
   - Como McCall: hay una elección entre dos opciones en cada período
   - Complicación: ahora el precio de la deuda depende de la probabilidad de default

3. Reinterpretar
   - Consumo/ahorro del agente $\iff$ Endeudamiento de la economía pequeña y abierta

Plan

Vamos a proceder en etapas

1. Problema de un agente con ingreso aleatorio y mercados incompletos
   - Solamente un activo (deuda) libre de riesgo
   - Escritura recursiva, ecuación de Bellman
   - Encontrar la función de valor via `vfi`
   - Distinto de McCall: un control continuo

2. Agregar default
   - Como McCall: hay una elección entre dos opciones en cada período
   - Complicación: ahora el precio de la deuda depende de la probabilidad de default

3. Reinterpretar
   - Consumo/ahorro del agente $\iff$ Endeudamiento de la economía pequeña y abierta

Plan

Vamos a proceder en etapas

1. Problema de un agente con ingreso aleatorio y mercados incompletos
   - Solamente un activo (deuda) libre de riesgo
   - Escritura recursiva, ecuación de Bellman
   - Encontrar la función de valor via `vfi`
   - Distinto de McCall: un control continuo

2. Agregar default
   - Como McCall: hay una elección entre dos opciones en cada período
   - Complicación: ahora el precio de la deuda depende de la probabilidad de default

3. Reinterpretar
   - Consumo/ahorro del agente $\Longleftrightarrow$ Endeudamiento de la economía pequeña y abierta

Cuando te creen

Problema de fluctuación de ingresos

Situación

- Un agente tiene una dotación aleatoria y_t distribuida $F(y'|y)$
- Preferencias: utilidad u, descuento β
- Puede comprar y vender un activo libre de riesgo b
- Límite de deuda $\underline{b}$

$$V_0 = \max_{c_t, b_t} \quad \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t) \right]$$

$$\text{sujeto a} \quad c_t = y_t + \frac{1}{1+r} b_t - b_{t-1}$$

$$b_t \leq \underline{b}$$

Problema de fluctuación de ingresos

Situación

- Un agente tiene una dotación aleatoria y_t distribuida $F(y'|y)$
- Preferencias: utilidad u, descuento β
- Puede comprar y vender un activo libre de riesgo b
- Límite de deuda $\underline{b}$

$$
\begin{aligned}
V_0 = \max_{c_t, b_t} \quad & \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t) \right] \\
\text{sujeto a} \quad & c_t = y_t + \frac{1}{1+r} b_t - b_{t-1} \\
& b_t \leq \underline{b}
\end{aligned}
$$

Escritura recursiva

$$V_t = \max_{c_{t+s}, b_{t+s}} \mathbb{E}_t \left[\sum_{s=0}^{\infty} \beta^s u(c_{t+s}) \right] = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[\max_{c_{t+1+s}, b_{t+1+s}} \sum_{s=0}^{\infty} \beta^s u(c_{t+1+s}) \right]$$

$$\text{sujeto a } \; c_{t+s} = y_{t+s} + \frac{1}{1+r} b_{t+s} - b_{t+s-1}$$

$$b_{t+s} \leq \underline{b}$$

Así que

$$V_t = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[V_{t+1} \right]$$

$$\text{sujeto a } \; c_t = y_t + \frac{1}{1+r} b_t - b_{t-1}$$

$$b_t \leq \underline{b}$$

Escritura recursiva

$$V_t = \max_{c_{t+s}, b_{t+s}} \mathbb{E}_t \left[\sum_{s=0}^{\infty} \beta^s u(c_{t+s}) \right] = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[\max_{c_{t+1+s}, b_{t+1+s}} \sum_{s=0}^{\infty} \beta^s u(c_{t+1+s}) \right]$$

$$\text{sujeto a } \; c_{t+s} = y_{t+s} + \frac{1}{1+r} b_{t+s} - b_{t+s-1}$$

$$b_{t+s} \leq \underline{b}$$

Así que

$$V_t = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[V_{t+1} \right]$$

$$\text{sujeto a } \; c_t = y_t + \frac{1}{1+r} b_t - b_{t-1}$$

$$b_t \leq \underline{b}$$

Escritura recursiva

$$V_t = \max_{c_{t+s}, b_{t+s}} \mathbb{E}_t \left[\sum_{s=0}^{\infty} \beta^s u(c_{t+s}) \right] = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[\max_{c_{t+1+s}, b_{t+1+s}} \sum_{s=0}^{\infty} \beta^s u(c_{t+1+s}) \right]$$

$$\text{sujeto a } \; c_{t+s} = y_{t+s} + \frac{1}{1+r} b_{t+s} - b_{t+s-1}$$

$$b_{t+s} \leq \underline{b}$$

Así que

$$V_t = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[V_{t+1} \right]$$

$$\text{sujeto a } \; c_t = y_t + \frac{1}{1+r} b_t - b_{t-1}$$

$$b_t \leq \underline{b}$$

Escritura recursiva

$$V_t = \max_{c_{t+s}, b_{t+s}} \mathbb{E}_t \left[\sum_{s=0}^{\infty} \beta^s u(c_{t+s}) \right] = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[\max_{c_{t+1+s}, b_{t+1+s}} \sum_{s=0}^{\infty} \beta^s u(c_{t+1+s}) \right]$$

$$\text{sujeto a } \; c_{t+s} = y_{t+s} + \frac{1}{1+r} b_{t+s} - b_{t+s-1}$$

$$b_{t+s} \leq \underline{b}$$

Así que

$$V_t = \max_{c_t, b_t} u(c_t) + \beta \mathbb{E}_t \left[V_{t+1} \right]$$

$$\text{sujeto a } \; c_t = y_t + \frac{1}{1+r} b_t - b_{t-1}$$

$$b_t \leq \underline{b}$$

Escritura recursiva

Ec. de Bellman

$$
\begin{aligned}
v(b,y) &= \max_{c,b'} u(c) + \beta \mathbb{E}\left[v(b',y')|y\right] \\
\text{sujeto a} \quad c + b &= y + \frac{1}{1+r} b' \\
b' &\leq \underline{b} \\
y' &\sim F(\cdot|y)
\end{aligned}
$$


21:29
12°5
LN+
VIVO
DANIEL HEYMANN
MACROECONOMISTA
HABLA EL MENTOR ARGENTINO DE MARTÍN GUZMÁN
#larepregunta
LA REPREGUNTA
CÓN QUE SE PAGA LA DEUDA?
imgflip.com

Escritura recursiva

Restricción de presupuesto

$$c + b = y + \frac{1}{1+r} b'$$

$$b = y - c + \frac{1}{1+r} b''$$

$$b = y - c + \frac{1}{1+r} \left(y' - c' + \frac{1}{1+r} b'' \right)$$

$$b = y - c + \frac{1}{1+r} (y' - c') + \frac{1}{(1+r)^2} b''$$

$$b = \sum_{i=0}^{\infty} \frac{y^{(j)} - c^{(j)}}{(1+r)^j}$$

Escritura recursiva

Restricción de presupuesto

$$c + b = y + \frac{1}{1+r} b'$$

$$b = y - c + \frac{1}{1+r} b''$$

$$b = y - c + \frac{1}{1+r} \left(y' - c' + \frac{1}{1+r} b'' \right)$$

$$b = y - c + \frac{1}{1+r} (y' - c') + \frac{1}{(1+r)^2} b''$$

$$b = \sum_{i=0}^{\infty} \frac{y^{(j)} - c^{(j)}}{(1+r)^j}$$

Escritura recursiva

Restricción de presupuesto

$$c + b = y + \frac{1}{1+r} b'$$

$$b = y - c + \frac{1}{1+r} b''$$

$$b = y - c + \frac{1}{1+r}\left(y' - c' + \frac{1}{1+r} b''\right)$$

$$b = y - c + \frac{1}{1+r}\left(y' - c'\right) + \frac{1}{(1+r)^2} b''$$

$$b = \sum_{i=0}^{\infty} \frac{y^{(j)} - c^{(j)}}{(1+r)^j}$$

Escritura recursiva

Restricción de presupuesto

$$c + b = y + \frac{1}{1+r} b'$$

$$b = y - c + \frac{1}{1+r} b''$$

$$b = y - c + \frac{1}{1+r} \left(y' - c' + \frac{1}{1+r} b'' \right)$$

$$b = y - c + \frac{1}{1+r} (y' - c') + \frac{1}{(1+r)^2} b''$$

$$b = \sum_{i=0}^{\infty} \frac{y^{(j)} - c^{(j)}}{(1+r)^j}$$

Escritura recursiva

Restricción de presupuesto

$$c + b = y + \frac{1}{1+r} b'$$

$$b = y - c + \frac{1}{1+r} b''$$

$$b = y - c + \frac{1}{1+r} \left(y' - c' + \frac{1}{1+r} b'' \right)$$

$$b = y - c + \frac{1}{1+r} (y' - c') + \frac{1}{(1+r)^2} b''$$

$$b = \sum_{i=0}^{\infty} \frac{y^{(j)} - c^{(j)}}{(1+r)^j}$$

Jerarquía de tipos

```julia
abstract type Deuda
end
mutable struct NoDefault <: Deuda
  β::Float64
  γ::Float64
  r::Float64

  bgrid::Vector{Float64}
  ygrid::Vector{Float64}
  Py::Matrix{Float64}

  v::Dict{Symbol, Matrix{Float64}}
end
```

- Vamos a declarar funciones generales que sirvan para cualquier *subtipo* de `Deuda`
- Notación `<:` indica que declaramos `NoDefault` como un subtipo de `Deuda`
- Dos nuevos tipos: `Dict` y `Symbol`

Multiple dispatch

Si `x::NoDefault`, y

```
f(dd::Deuda) = sum(dd.bgrid)
```

entonces `f(x)` devuelve la suma de los valores posibles para b.

Pero si también está definida

```
f(dd::NoDefault) = dd.γ
```

entonces `f(x)` devuelve el parámetro de aversión al riesgo

`Dict` y `Symbol`

- Un `Dict` es un conjunto de elementos (como un `Vector`).
- Cada elemento tiene un nombre en vez de un número $\to$ sin orden
- Siempre aparecen declarados como `Dict{K,T}`

```
v::Dict{Symbol, Matrix{Float64}}
v = Dict(:a => A, :b => B)
```

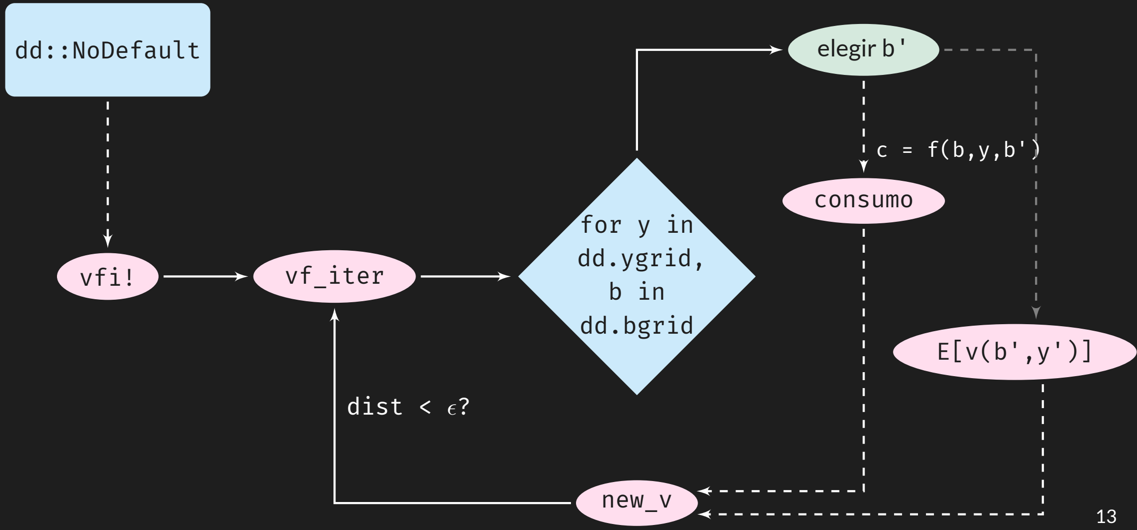
Constructor

Default

```
function NoDefault(; β = 0.953, γ = 2, r = 0.017, \rho = 0.945,
    \eta = 0.025, Nb = 200, Ny = 21, bmax = 0.5)

  ychain = tauchen(Ny, ρ, η, 0, 2)

  Py = ychain.p
  ygrid = exp.(ychain.state_values)

  bgrid = range(0, bmax, length=Nb)

  v = Dict(key => zeros(Nb, Ny) for key in [:v, :c, :b])

  return NoDefault(β, γ, r, bgrid, ygrid, Py, v)
end
```

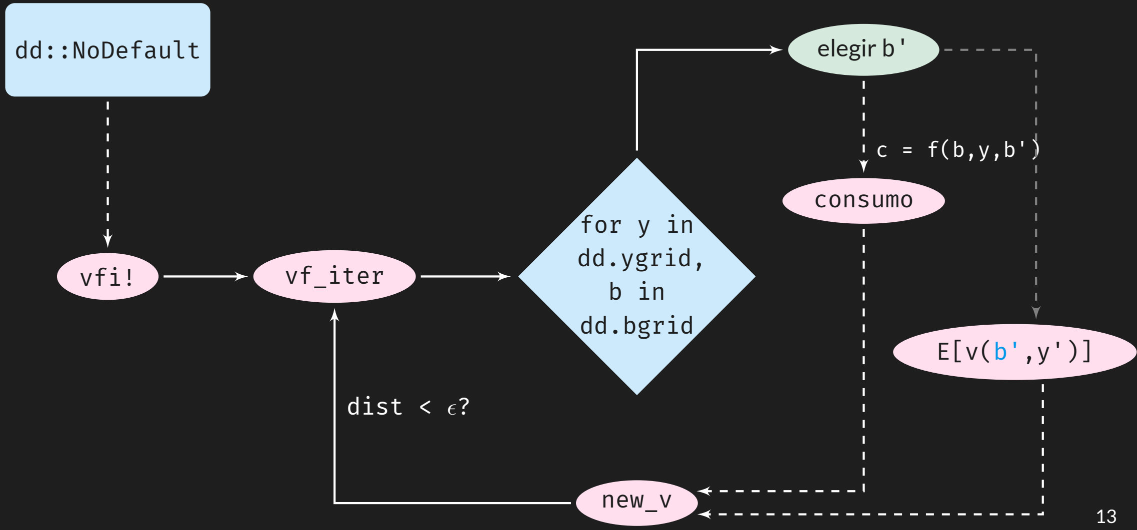
Pseudo-código

Default



Pseudo-código

Default



Funciones básicas

```julia
function u(c, dd::Deuda)
  if dd.γ == 1
    return log(c)
  else
    return c^(1-dd.γ) / (1-dd.γ)
  end
end

budget_constraint(bpv, qv, bv, yv) = yv - bv + qv*bpv

debtprice(bpv, py, itp_v, dd::Deuda) = 1/(1+dd.r)
```

Loop

```julia
function vfi!(dd::Deuda; tol=1e-4, maxiter = 2000)
  iter, dist = 0, 1+tol
  new_v = Dict(key => similar(val) for (key, val) in dd.v)
  while iter < maxiter && dist > tol
    iter += 1

    vf_iter!(new_v, dd)

    dist = maximum([ norm(new_v[key] - dd.v[key]) / (1+norm(dd.v[key]))
       for key in keys(dd.v) ])
    norm_v = 1+maximum([norm(dd.v[key]) for key in keys(dd.v)])

    print("Iteration $iter: dist = $(@sprintf("%0.3g", dist)) at |v| =
       $(@sprintf("%0.3g", norm_v))\n")
    update_v!(new_v, dd)
  end
end
```

Iterador

Default

```julia
itp(dd::Deuda, key) = interpolate((dd.bgrid, dd.ygrid), dd.v[key], Gridded(Linear()))

function vf_iter!(new_v, dd::Deuda)
  itp_v = Dict(key => itp(dd, key)) for key in keys(dd.v))
  for (jb, bv) in enumerate(dd.bgrid), (jy, yv) in enumerate(dd.ygrid)
    py = dd.Py[jy, :]

    b_opt, c_opt, v_opt = optim_value(bv, yv, py, itp_v, dd, optim=true)

    new_v[:b][jb, jy] = b_opt
    new_v[:c][jb, jy] = c_opt
    new_v[:v][jb, jy] = v_opt
  end
end
```

Evaluar la v

```julia
function expect_v(bpv, py, itp_v, dd::Deuda)
  Ev = 0.0
  for (jyp, ypv) in enumerate(dd.ygrid)
    prob = py[jyp]
    Ev += prob * itp_v[:v](bpv, ypv)
  end
  return Ev
end

function eval_value(bpv, bv, yv, py, itp_v, dd::Deuda)
  qv = debtprice(bpv, py, itp_v, dd)
  c = budget_constraint(bpv, qv, bv, yv)
  vp = expect_v(bpv, py, itp_v, dd)
  return u(c, dd) + dd.β * vp
end
```

Optimizador trucho

```julia
function optim_value2(bv, yv, py, itp_v, dd::Deuda)
  v, b_opt, c_opt = -Inf, 0.0, 0.0
  for (jbp, bpv) in enumerate(dd.bgrid)
    qv = debtprice(bpv, py, itp_v, dd)
    new_v = eval_value(bpv, bv, yv, py, itp_v, dd)

    if new_v > v
      v = new_v
      c_opt = budget_constraint(bpv, qv, bv, yv)
      b_opt = bpv
    end
  end
  return b_opt, c_opt, v
end
```

Optimizador de verdad

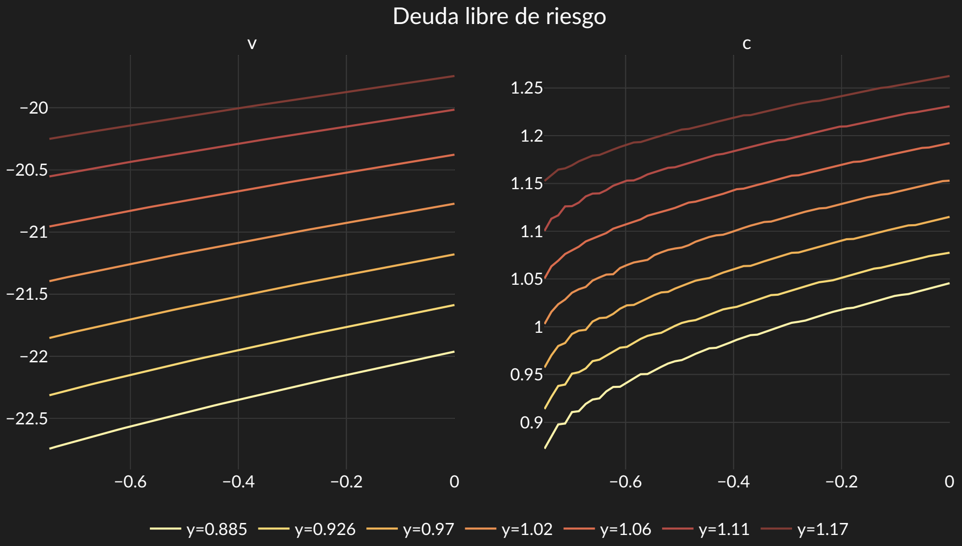
```julia
function optim_value1(bv, yv, py, itp_v, dd::Deuda)
  obj_f(bpv) = -eval_value(bpv, bv, yv, py, itp_v, dd)

  res = Optim.optimize(obj_f, 0.0, maximum(dd.bgrid))

  b_opt = res.minimizer
  qv = debtprice(b_opt, py, itp_v, dd)
  c_opt  = budget_constraint(b_opt, qv, bv, yv)
  new_v  = -res.minimum

  return b_opt, c_opt, new_v
end
```

Elegir el optimizador

```julia
function optim_value(bv, yv, py, itp_v, dd::Deuda; optim=true)
  if optim
    return optim_value1(bv, yv, py, itp_v, dd)
  else
    return optim_value2(bv, yv, py, itp_v, dd)
  end
end
```

Cuando te prestan

Con default



Cuando no te creen (y hacen bien)

Dos cambios

Para agregar default,

- Especificar qué pasa cuando el agente decide no pagar la deuda

$$y^d = h(y) = \min\left\{y,\ 0.969\mathbb{E}\left[y\right]\right\}$$

Exclusión de mercados de capital *por un tiempo* $\rightarrow \theta$

- Especificar el precio de la deuda

$$q(b', y) = \frac{1}{1+r}\mathbb{E}\left[1 - d(b', y')|y\right]$$

Dos cambios

Para agregar default,

- Especificar qué pasa cuando el agente decide no pagar la deuda

$$y^d = h(y) = \min\left\{y,\ 0.969\mathbb{E}\left[y\right]\right\}$$

Exclusión de mercados de capital *por un tiempo* $\to \theta$

- Especificar el precio de la deuda

$$q(b', y) = \frac{1}{1+r}\mathbb{E}\left[1 - d(b', y')|y\right]$$

Dos cambios

Para agregar default,

- Especificar qué pasa cuando el agente decide no pagar la deuda

$$y^d = h(y) = \min\left\{y,\ 0.969\mathbb{E}\left[y\right]\right\}$$

Exclusión de mercados de capital *por un tiempo* $\to \theta$

- Especificar el precio de la deuda

$$q(b', y) = \frac{1}{1+r}\mathbb{E}\left[1 - d(b', y')|y\right]$$

Bellmans

- Elegir default o repago

$$\mathcal{V}(b, y) = \max \left\{ v^R(b, y), v^D(y) \right\}$$

- En repago, elegir emisión

$$v^R(b, y) = \max_{c, b'} u(c) + \beta \mathbb{E} \left[\mathcal{V}(b', y') | y \right]$$
$$\text{sujeto a } c + b = y + q(b', y) b'$$

- En default, nada que elegir

$$v^D(y) = u(h(y)) + \beta \mathbb{E} \left[\theta \mathcal{V}(0, y') + (1 - \theta) v^D(y') | y \right]$$

Bellmans

- Elegir default o repago

$$\mathcal{V}(b, y) = \max \left\{ v^R(b, y), v^D(y) \right\}$$

- En repago, elegir emisión

$$v^R(b, y) = \max_{c, b'} u(c) + \beta \mathbb{E} \left[\mathcal{V}(b', y') | y \right]$$

$$\text{sujeto a } c + b = y + q(b', y) b'$$

- En default, nada que elegir

$$v^D(y) = u(h(y)) + \beta \mathbb{E} \left[\theta \mathcal{V}(0, y') + (1 - \theta) v^D(y') | y \right]$$

Bellmans

- Elegir default o repago

$$\mathcal{V}(b, y) = \max \left\{ v^R(b, y), v^D(y) \right\}$$

- En repago, elegir emisión

$$v^R(b, y) = \max_{c, b'} u(c) + \beta \mathbb{E} \left[\mathcal{V}(b', y') | y \right]$$

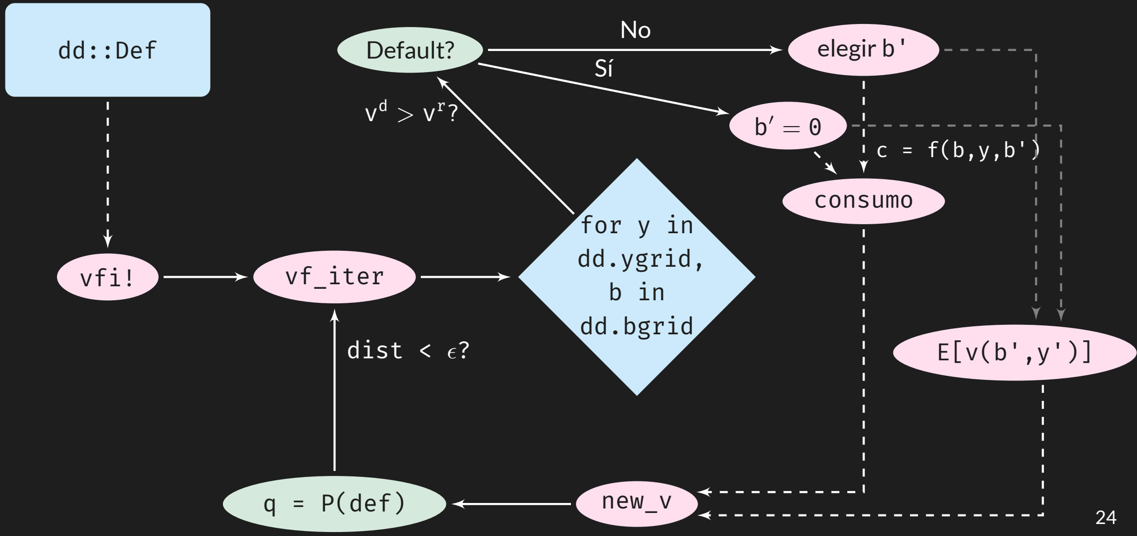
$$\text{sujeto a } c + b = y + q(b', y) b'$$

- En default, nada que elegir

$$v^D(y) = u(h(y)) + \beta \mathbb{E} \left[\theta \mathcal{V}(0, y') + (1 - \theta) v^D(y') | y \right]$$

Pseudo-código

NoDefault



Jerarquía de tipos

```
mutable struct Def <: Deuda
  β::Float64
  γ::Float64
  r::Float64
  \theta::Float64
  \kappa::Float64

  bgrid::Vector{Float64}
  ygrid::Vector{Float64}

  Py::Matrix{Float64}
  v::Dict{Symbol, Matrix{Float64}}
end
```

- `Def` es otro subtipo de `Deuda`
- No tiene los mismos campos que `NoDefault`

  Cuidado!
- v sigue siendo un `Dict`, pero esta vez vamos a poner más cosas ahí

Constructor

◂ NoDefault

```julia
function Def(; β = 0.953, γ = 2, r = 0.017, \theta = 0.282, \kappa = 0.18,
    \rho = 0.945, \eta = 0.025, Nb = 200, Ny = 21, bmax = 0.5)

  ychain = tauchen(Ny, ρ, η, 0, 2)

  Py = ychain.p
  ygrid = exp.(ychain.state_values)

  bgrid = range(0, bmax, length=Nb)

  v = Dict(key => zeros(Nb, Ny) for key in
    [:v, :R, :D, :prob, :cR, :cD, :b, :q])

  return Def(β, γ, r, \theta, \kappa, bgrid, ygrid, Py, v)
end
```

Envolventes!

- Opción 1

$$\mathcal{V}(b,y) = \max\left\{v^R(b,y), v^D(y)\right\}$$

- Opción 2

$$\mathcal{P}(b,y) = \frac{\exp(v^D(y)/\kappa)}{\exp(v^R(b,y)/\kappa) + \exp(v^D(y)/\kappa)}$$

$$\mathcal{V}(b,y) = \mathcal{P}(b,y)v^D(y) + (1 - \mathcal{P}(b,y))\, v^R(b,y)$$

- Opción 1 = Opción 2 con $\mathcal{P}(b,y) = 1_{v^D(y) > v^R(b,y)}$

Envolventes!

- Opción 1

$$\mathcal{V}(b,y) = \max\left\{v^R(b,y), v^D(y)\right\}$$

- Opción 2

$$\mathcal{P}(b,y) = \frac{\exp(v^D(y)/\kappa)}{\exp(v^R(b,y)/\kappa) + \exp(v^D(y)/\kappa)}$$
$$\mathcal{V}(b,y) = \mathcal{P}(b,y)v^D(y) + (1 - \mathcal{P}(b,y))\, v^R(b,y)$$

- Opción 1 = Opción 2 con $\mathcal{P}(b,y) = 1_{v^D(y) > v^R(b,y)}$

Envolventes!

- Opción 1

$$\mathcal{V}(b,y) = \max\left\{v^R(b,y), v^D(y)\right\}$$

- Opción 2

$$\mathcal{P}(b,y) = \frac{\exp(v^D(y)/\kappa)}{\exp(v^R(b,y)/\kappa) + \exp(v^D(y)/\kappa)}$$
$$\mathcal{V}(b,y) = \mathcal{P}(b,y)v^D(y) + (1-\mathcal{P}(b,y))\, v^R(b,y)$$

- Opción 1 = Opción 2 con $\mathcal{P}(b,y) = 1_{v^D(y) > v^R(b,y)}$

Multiple dispatch para todos y todas

- `vfi!` ✓
- `vf_iter` ✗ $\rightarrow$ separar consumo en repago y default, guardar en v^R
- `optim_value` (1 y 2) ✓
- `debtprice` ✗ $\rightarrow$ reemplazar por la probabilidad de default
- `eval_value` ✓
- `expect_v` ✓
- `u` ✓
- `update_defprob!` $\rightarrow$ opción 1 vs opción 2
- `value_default` $\rightarrow$ para calcular v^D
- `defcost` $\rightarrow$ para calcular $h(y)$

Iterador

NoDefault

```julia
function vf_iter!(new_v, dd::Deuda)
  itp_v = Dict(key => itp(dd, key)) for key in keys(dd.v))
  for (jy, yv) in enumerate(dd.ygrid)
    py = dd.Py[jy, :]
    for (jb, bv) in enumerate(dd.bgrid)
      b_opt, c_opt, v_opt = optim_value(bv, yv, py, itp_v, dd)
      new_v[:b][jb, jy]   = b_opt
      new_v[:cR][jb, jy]  = c_opt
      new_v[:R][jb, jy]   = v_opt
      new_v[:q][jb, jy]   = debtprice(bv, py, itp_v, dd)
    end
    c_opt, v_opt = value_default(yv, py, itp_v, dd)
    new_v[:cD][:, jy] .= c_opt
    new_v[:D][:, jy]  .= v_opt
  end
  update_defprob!(new_v, dd)
end
```

Actualizar la probabilidad de default

```
function update_defprob!(v, dd::Def)
  for (jy, yv) in enumerate(dd.ygrid), (jb, bv) in enumerate(dd.bgrid)
    vD = v[:D][jb, jy]
    vR = v[:R][jb, jy]

    if dd.\kappa >= 0.001
      prob = exp(vD/dd.\kappa) / (exp(vD/dd.\kappa) + exp(vR/dd.\kappa))
    else
      prob = (vD > vR)
    end

    v[:v][jb, jy] = prob * vD + (1-prob) * vR
    v[:prob][jb, jy] = prob
  end
end
```

Precio de la deuda

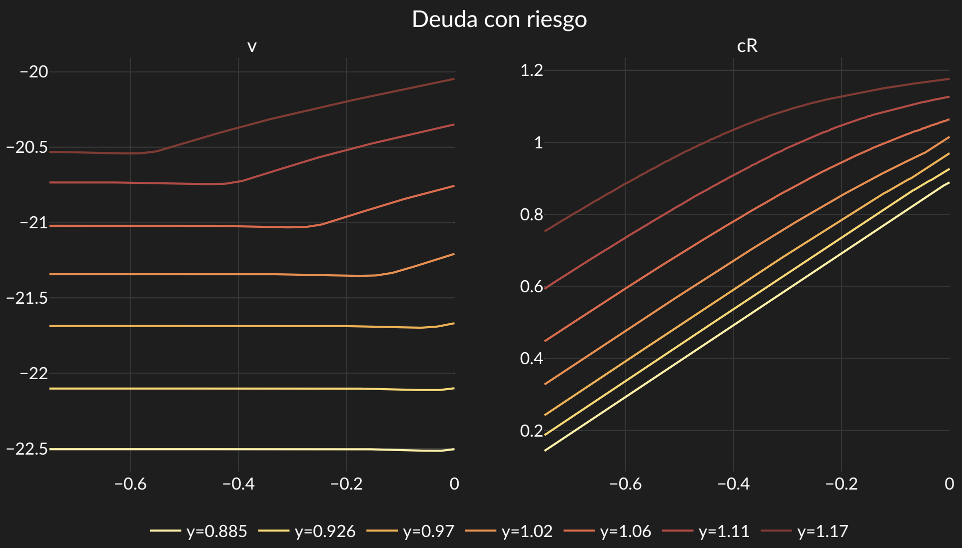
```julia
function debtprice(bpv, py, itp_v, dd::Def)
  defprob = 0.0
  for (jyp, ypv) in enumerate(dd.ygrid)
    prob = py[jyp]
    defprob += prob * itp_v[:prob](bpv, ypv)
  end

  return (1-defprob) / (1+dd.r)
end
```

Valor en default

```
defcost(yv, dd::Def) = min(yv, 0.969)

function value_default(yv, py, itp_v, dd::Def)
  c = defcost(yv, dd)

  Ev = 0.0
  for (jyp, ypv) in enumerate(dd.ygrid)
    prob = py[jyp]
    Ev += prob * ( dd.\theta * itp_v[:v](0.0, ypv) + (1-dd.\theta) * itp_v[:D](0.0, ypv)
  end

  v = u(c, dd) + dd.β * Ev
  return c, v
end
```

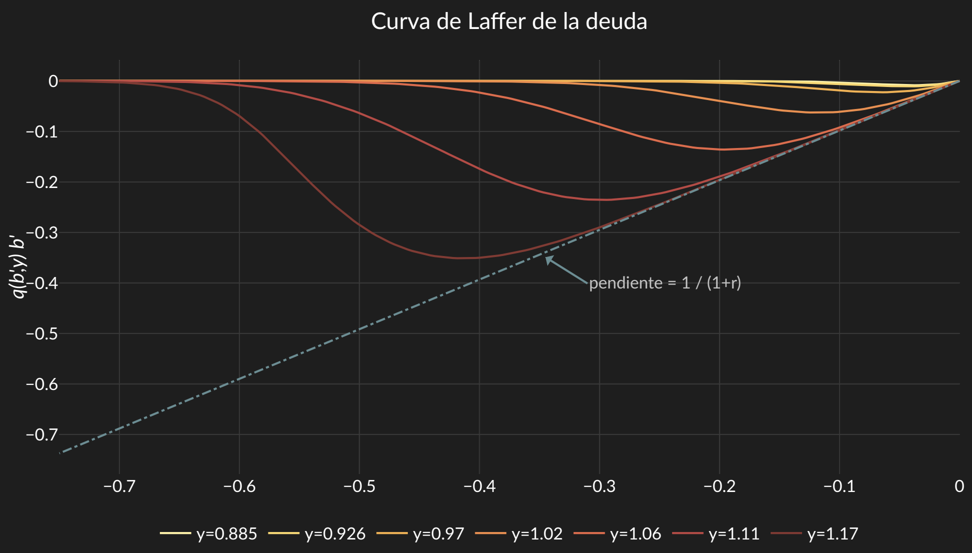
Cuando no te prestan

◂ Sin default



Cuando no te prestan

Paper



Cierre

Cierre

Vimos

- Problema de fluctuación de ingresos
  - Interpolar la función de valor
  - Un control continuo
- Agregar default
  - Costos de default
  - Precio de la deuda

Dejé afuera

- Simulador
- Estática comparada
- Otros costos de default

... en los códigos que subí

Cierre

Vimos

- Problema de fluctuación de ingresos
  - Interpolar la función de valor
  - Un control continuo
- Agregar default
  - Costos de default
  - Precio de la deuda

Dejé afuera

- Simulador
- Estática comparada
- Otros costos de default

. . . en los códigos que subí

Cómo se come una torta?

Situación

- Un agente tiene una torta de tamaño K
- Preferencias standard, si come c_t de torta, recibe utilidad $u(c_t)$ y queda $K_t - c_t$
- El agente descuenta el futuro con β

$$v(k) = \max_{c,k'} u(c) + \beta v(k')$$

$$\text{sujeto a} \quad c + k' = k$$