2º Trabalho de Modelagem Matemática em Finanças II

Luiz Rodrigo S. de Souza

20 de outubro de 2018

Carregando as libs...

```
library(tidyverse)
library(magrittr)
library(rlang)
```

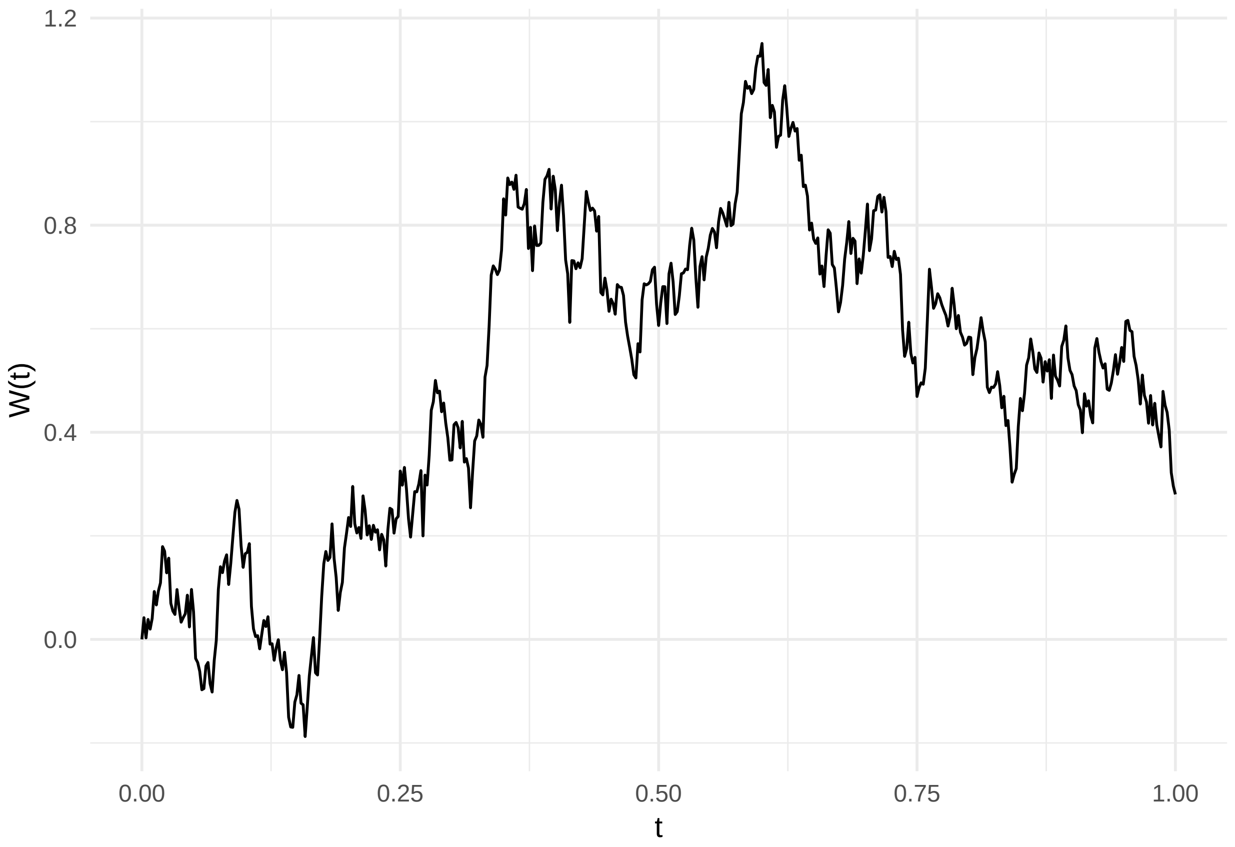
Seção 2: Movimento Browniano

Primeiro, função para gerar m caminhos brownianos avaliados na sequência t.

```
# generates m brownian paths along t
brownian = function(t, m, d = FALSE) {
  dt = diff(t)
  entries = rep(sqrt(dt), each = m) * rnorm(length(dt) * m)
  dB = matrix(entries, byrow = TRUE, nrow = length(dt))
  if (d) {
    dB
  } else {
    rbind(double(m), apply(dB, 2, cumsum))
  }
}
```

Figura 1: Caminho browniano discretizado

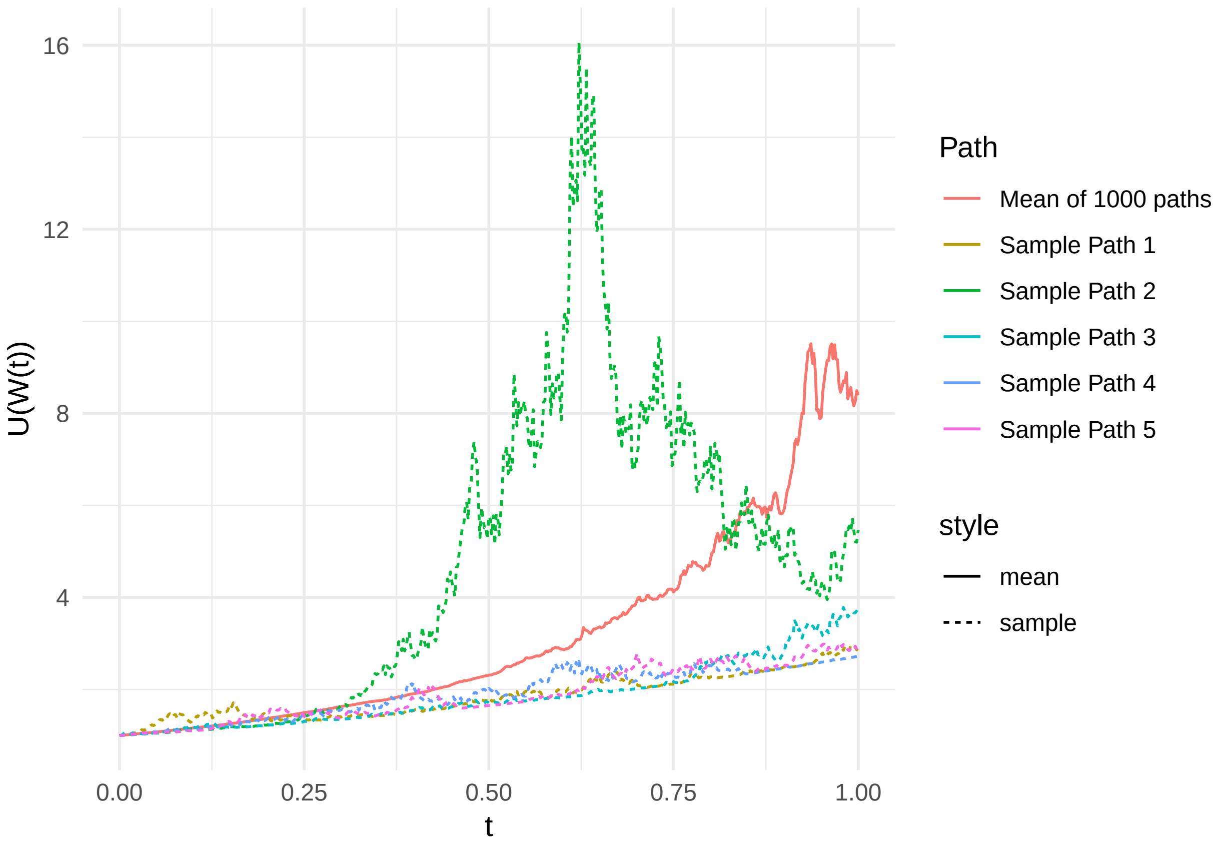
```
t = seq(0, 1, by = 1/500)
W = brownian(t, 1)
data.frame(t = t, W = W) %>%
  ggplot(aes(x = t, y = W)) + geom_path() + ylab("W(t)") +
  theme_minimal()
```



Agora vamos simular o comportamento de $u(t) = exp(t + \frac{W(t)}{2})$.

Figura 2: U(t) na média de 1000 runs e alguns runs específicos

```
t = seq(0, 1, by = 1/500)
m = 1000
W = brownian(t, m)

df = apply(W[, 1:5], 2, function(W) exp(t + W^2/2)) %>% as.data.frame
names(df) = paste0("Sample Path ", 1:5)
df$`Mean of 1000 paths` = apply(W, 2, function(W) exp(t + W^2/2)) %>% rowMeans
df$t = t

df %>% gather(Path, y, -t) %>%
  mutate(style = if_else(startsWith(Path, "Mean"), "mean", "sample")) %>%
  ggplot(aes(x = t, y = y, color = Path, linetype = style)) + geom_path() +
  ylab("U(W(t))") + theme_minimal()
```



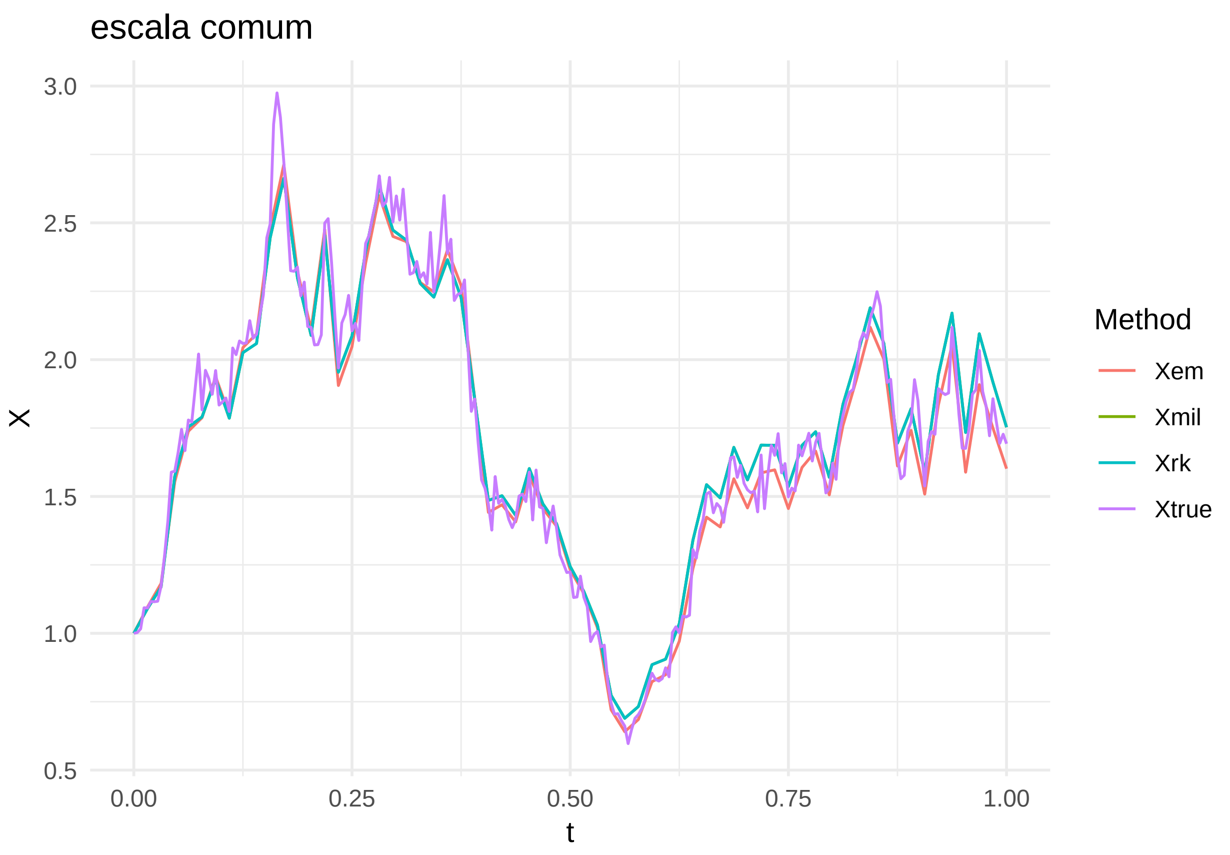
Seções 4 e 6: Os métodos Euler-Maruyama, Milstein e Runge-Kutta

```
# generates m integral paths of dX = fdt + gdW along t
# using euler-maruyama method
eulerMaruyama = function(f, g, t, x0, m = NULL, dB = NULL) {
  dt = diff(t)

  if (is.null(dB)) {
    dB = brownian(t, m, d = TRUE)
  }

  X = rbind(x0, dB)

  for (i in 1:length(dt)) {
    X[i+1,] = X[i,] + f(t[i], X[i,])*dt[i] + g(t[i], X[i,])*dB[i,]
  }

  X
}

# weak euler-maruyama: uses dX
weakEM = function(f, g, t, x0, m = NULL, dB = NULL) {
  dt = diff(t)

  if (is.null(dB)) {
    dB = brownian(t, m, d = TRUE)
  }

  X = rbind(x0, dB)

  for (i in 1:length(dt)) {
    X[i+1,] = X[i,] + f(t[i], X[i,])*dt[i] + g(t[i], X[i,])*sqrt(dt[i])*sign(dB[i,])
  }

  X
}


# generates m integral paths of dX = fdt + gdW along t
# using milstein's method
milstein = function(f, g, gx, t, x0, m = NULL, dB = NULL) {
  dt = diff(t)

  if (is.null(dB)) {
    dB = brownian(t, m, d = TRUE)
  }

  X = rbind(x0, dB)

  for (i in 1:length(dt)) {
    X[i+1,] = X[i,] + f(t[i], X[i,])*dt[i] + g(t[i], X[i,])*dB[i,] +
      .5*g(t[i], X[i,])*gx(t[i], X[i,])*(dB[i,]*dB[i,] - dt[i])
  }

  X
}

# generates m integral paths of dX = fdt + gdW along t
# using strong order 1 runge-kutta
stochasticRK = function(f, g, t, x0, m = NULL, dB = NULL) {
  dt = diff(t)

  if (is.null(dB)) {
    dB = brownian(t, m, d = TRUE)
  }

  X = rbind(x0, dB)

  for (i in 1:length(dt)) {
    gi = g(t[i], X[i,])
    X[i+1,] = X[i,] + f(t[i], X[i,])*dt[i] + gi*dB[i,] +
      .5*(g(t[i], X[i,] + gi*sqrt(dt[i])) - gi)*(dB[i,]*dB[i,] - dt[i]) / sqrt(dt[i])
  }

  X
}
```

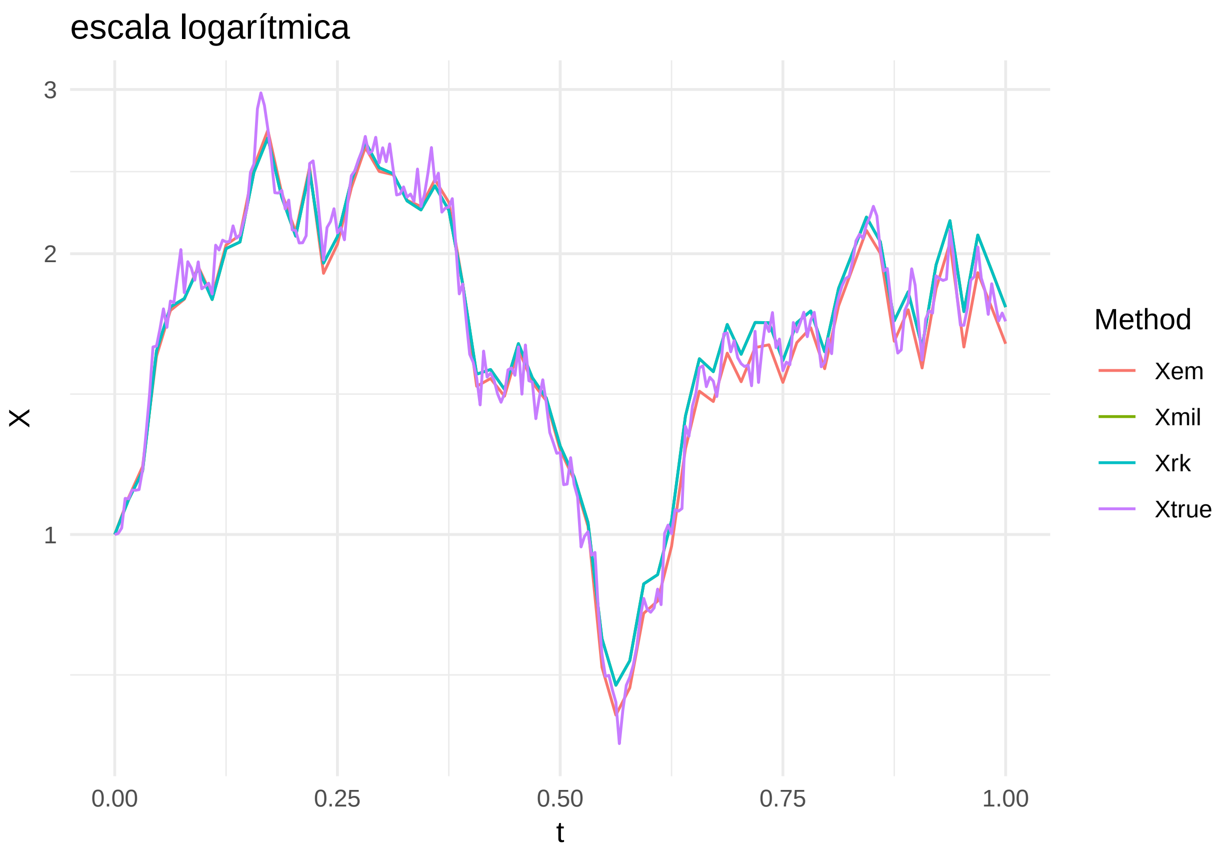
Figura 3: Solução real e aproximada por EM/Milstein/Runge-Kutta

Usando $dt = 2^{-8}$ para a solução real e $dt = 2^{-6}$ para EM/Milstein.

```
lambda = 2
mu = 1
X0 = 1
t = seq(0, 1, by = 2^(-8))
f = function(t, X) lambda*X
g = function(t, X) mu*X
gx = function(t, X) mu

sumeach = function(x, n) {
  matrix(x, byrow = TRUE, ncol = n) %>% rowSums()
}

W = brownian(t, 1)
dW = diff(W)
Xtrue = X0 * exp((lambda - mu*mu/2)*t + mu*W)

t2 = seq(0, 1, by = 2^(-6))
Xem = eulerMaruyama(f, g, t2, X0, dB = matrix(sumeach(dW, 4), ncol = 1))
Xwem = weakEM(f, g, t2, X0, dB = matrix(sumeach(dW, 4), ncol = 1))
Xmil = milstein(f, g, gx, t2, X0, dB = matrix(sumeach(dW, 4), ncol = 1))
Xrk = stochasticRK(f, g, t2, X0, dB = matrix(sumeach(dW, 4), ncol = 1))

left_join(data.frame(t, Xtrue), data.frame(t = t2, Xem, Xmil, Xrk)) %>%
  gather(Method, y, -t) %>%
  filter(!is.na(y)) %>%
  ggplot(aes(x = t, y = y, color = Method)) + geom_path() +
  ylab("X") + theme_minimal() +
  ggtitle("escala comum")
```



```
left_join(data.frame(t, Xtrue), data.frame(t = t2, Xem, Xmil, Xrk)) %>%
  gather(Method, y, -t) %>%
  filter(!is.na(y)) %>%
  ggplot(aes(x = t, y = y, color = Method)) + geom_path() +
  scale_y_continuous(trans = "log10") +
  ylab("X") + theme_minimal() +
  ggtitle("escala logarítmica")
```



Seção 5: Convergência forte e fraca

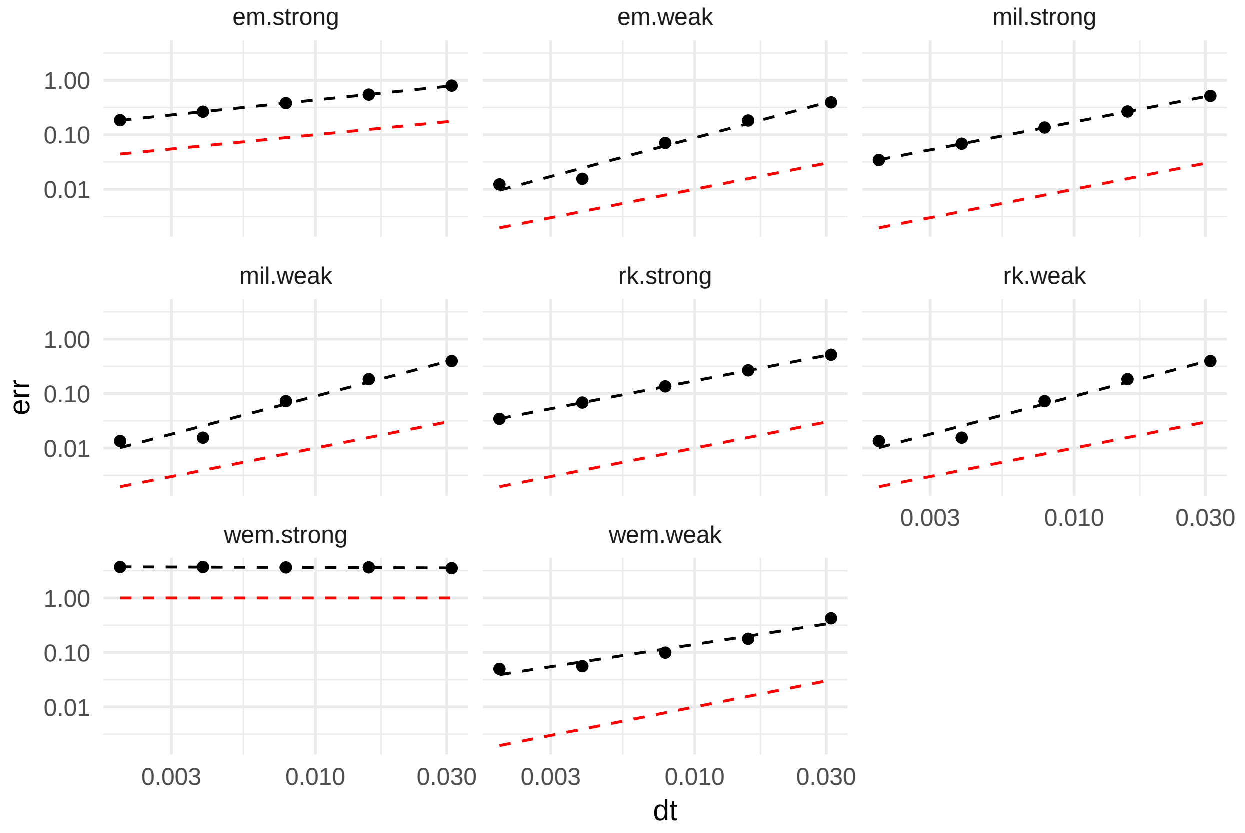
Primeiro, gero 50000 caminhos brownianos em várias resoluções diferentes

```r
t = seq(0, 1, by = 2^(-9))
m = 50000
dW = brownian(t, m, d = TRUE)
W = rbind(double(m), apply(dW, 2, cumsum))
Xtrue = X0 * exp((lambda - mu*mu/2)*t + mu*W)

# gera uma lista de runs de tamanhos 512 x 1000, 256 x 1000, ..., 32 x 1000
# a partir dos mesmos runs dW 512 x 1000
dWs = map(1:5, function(i) {
  apply(dW, 2, partial(sumeach, n = 2^(i - 1)))
})
```

Figura 4: Testes de convergência fraca e forte para os métodos de SDEs

Pontilhado: slope encontrado por mínimos quadrados. Pontilhado vermelho: slope teórico

```r
tb = map_dfr(1:5, function(i) {
  dt = 2^(i-10)
  t = seq(0, 1, by = dt)
  Xem = eulerMaruyama(f, g, t, X0, dB = dWs[[i]])
  Xwem = weakEM(f, g, t, X0, dB = dWs[[i]])
  Xmil = milstein(f, g, gx, t, X0, dB = dWs[[i]])
  Xrk = stochasticRK(f, g, t, X0, dB = dWs[[i]])

  data.frame(
    dt = dt,
    em.strong = mean(abs(tail(Xtrue, 1) - tail(Xem, 1))),
    em.weak = abs(exp(lambda) - mean(tail(Xem, 1))),
    wem.strong = mean(abs(tail(Xtrue, 1) - tail(Xwem, 1))),
    wem.weak = abs(exp(lambda) - mean(tail(Xwem, 1))),
    mil.strong = mean(abs(tail(Xtrue, 1) - tail(Xmil, 1))),
    mil.weak = abs(exp(lambda) - mean(tail(Xmil, 1))),
    rk.strong = mean(abs(tail(Xtrue, 1) - tail(Xrk, 1))),
    rk.weak = abs(exp(lambda) - mean(tail(Xrk, 1)))
  )
}) %>%
  gather(type, err, -dt) %>%
  group_by(type) %>%
  mutate(
    b = coef(lm(log(err) ~ log(dt)))[1],
    a = coef(lm(log(err) ~ log(dt)))[2],
    err_hat = exp(b)*dt^a
  )

  ggplot(tb) + theme_minimal() + scale_x_continuous(trans = "log10") +
  scale_y_continuous(trans = "log10") +
  geom_point(aes(x = dt, y = err)) +
  geom_path(aes(x = dt, y = err_hat), linetype = "dashed") +
  geom_path(aes(x = dt,
                y = case_when(
                  type == "em.strong" ~ dt^.5,
                  type == "wem.strong" ~ 1,
                  TRUE ~ dt
                )),
            color = "red", linetype = "dashed") +
  facet_wrap(~ type)
```



Seção 7: Estabilidade linear

Figura 5.a: Teste mean squares

```
T = 20
dts = c(.25, .5, 1)
m = 50000

lambda = -3
mu = sqrt(3)
X0 = 1
f = function(t, X) lambda*X
g = function(t, X) mu*X

map(dts, function(dt) {
  t = seq(0, T, dt)
  data.frame(t = t) %>%
    mutate(!!sym(paste0("dt = ", dt)) := rowMeans(eulerMaruyama(f, g, t, X0, m)^2))
}) %>%
  reduce(left_join) %>%
  gather(run, y, -t) %>%
  filter(!is.na(y)) %>%
  ggplot(aes(x = t, y = y, color = run)) + geom_path() +
  ylab("E[X²]") + theme_minimal() + scale_y_continuous(trans = "log10") +
  ggtitle("Mean square: lambda = -3, mu = sqrt(3)")
```

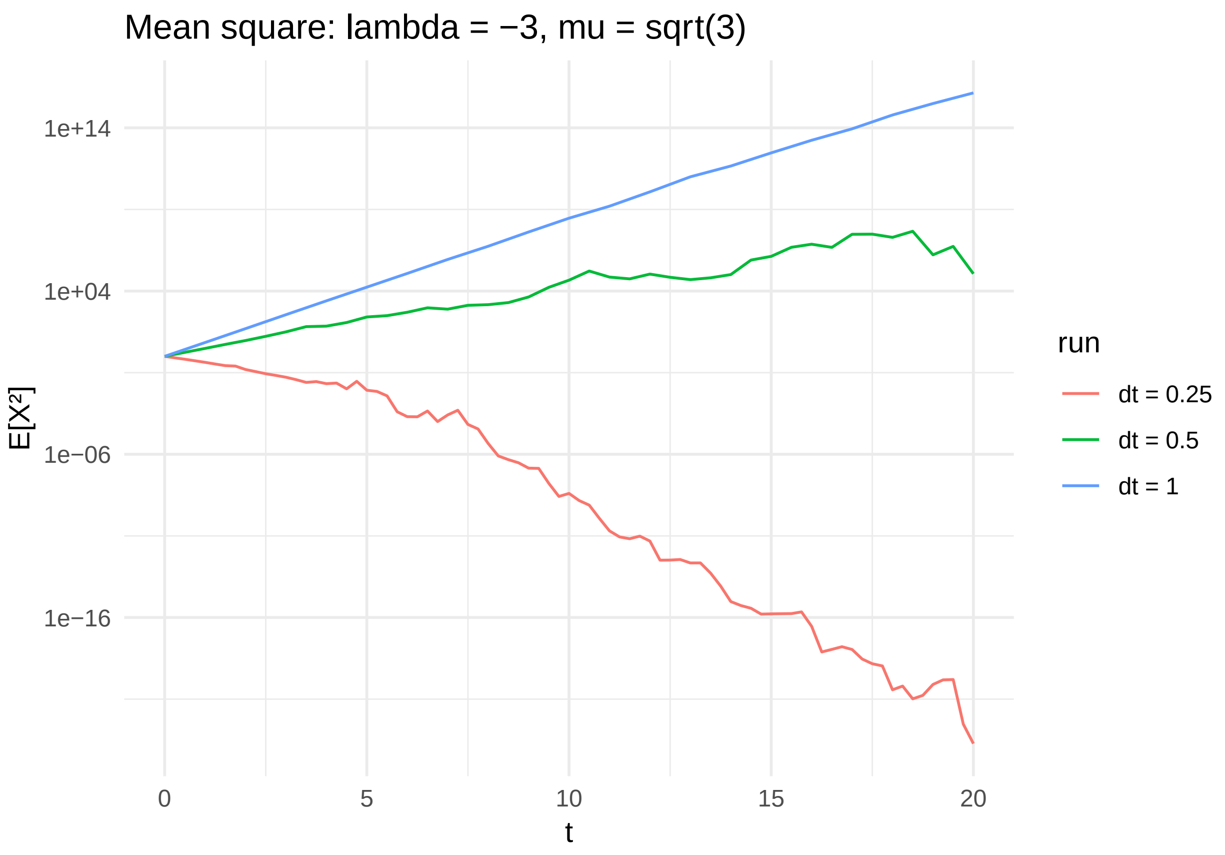
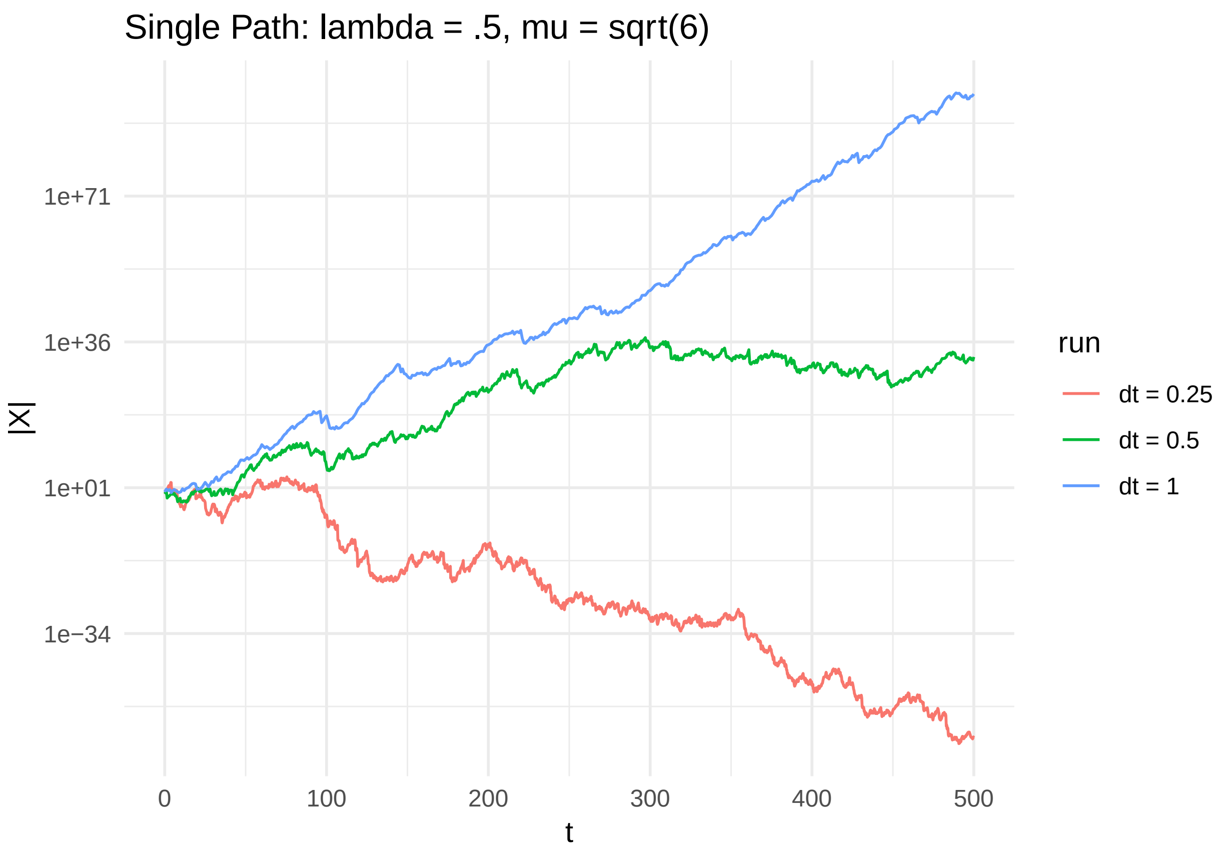


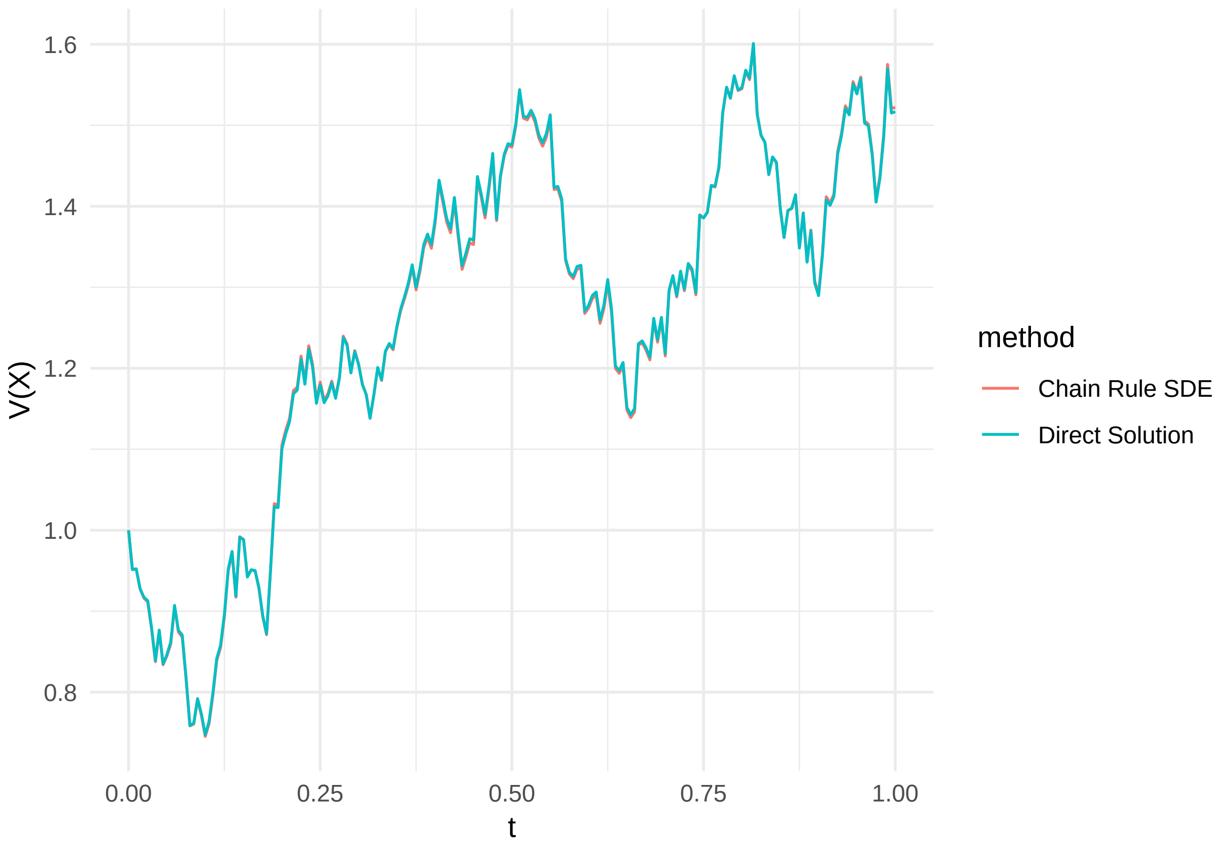
Figura 5.b: Teste de estabilidade assintótica

```r
T = 500
dts = c(.25, .5, 1)
m = 1

lambda = .5
mu = sqrt(6)
X0 = 1
f = function(t, X) lambda*X
g = function(t, X) mu*X

map(dts, function(dt) {
  t = seq(0, T, dt)
  data.frame(t = t) %>%
    mutate(!!sym(paste0("dt = ", dt)) := as.vector(abs(eulerMaruyama(f, g, t, X0, m))))
}) %>%
  reduce(left_join) %>%
  gather(run, y, -t) %>%
  filter(!is.na(y)) %>%
  ggplot(aes(x = t, y = y, color = run)) + geom_path() +
  ylab("|X|") + theme_minimal() + scale_y_continuous(trans = "log10") +
  ggtitle("Single Path: lambda = .5, mu = sqrt(6)")
```



Seção 8: Regra da cadeia estocástica

```
alpha = 2
beta = 1
t = seq(0, 1, by = 1/200)
dW = brownian(t, 1, d = TRUE)
X0 = 1

f1 = function(t, X) alpha - X
g1 = function(t, X) beta * sqrt(X)

f2 = function(t, V) (4*alpha - beta^2)/(8*V) - .5*V
g2 = function(t, V) .5*beta

direct = sqrt(eulerMaruyama(f1, g1, t, X0, dB = dW))
chainrule = eulerMaruyama(f2, g2, t, X0, dB = dW)

bind_rows(
  data.frame(t = t, method = "Direct Solution", y = direct),
  data.frame(t = t, method = "Chain Rule SDE", y = chainrule)
) %>%
  ggplot(aes(x = t, y = y, color = method)) + geom_path() +
  ylab("V(X)") + theme_minimal()
```



```r
print(paste0("Diferença máxima: ", max(abs(direct - chainrule))))
```

```
## [1] "Diferença máxima: 0.00573575093683787"
```