# Cognitive Modeling LC3

*Harm Manders, 10677186*

---

## Temporal Difference Learning

Temporal difference learning is een uitbreiding op het Rescola Wagner model zodat ook verloop van tijd tussen stimulus en respons kan worden gemodelleerd, de formule voor TD learning is:

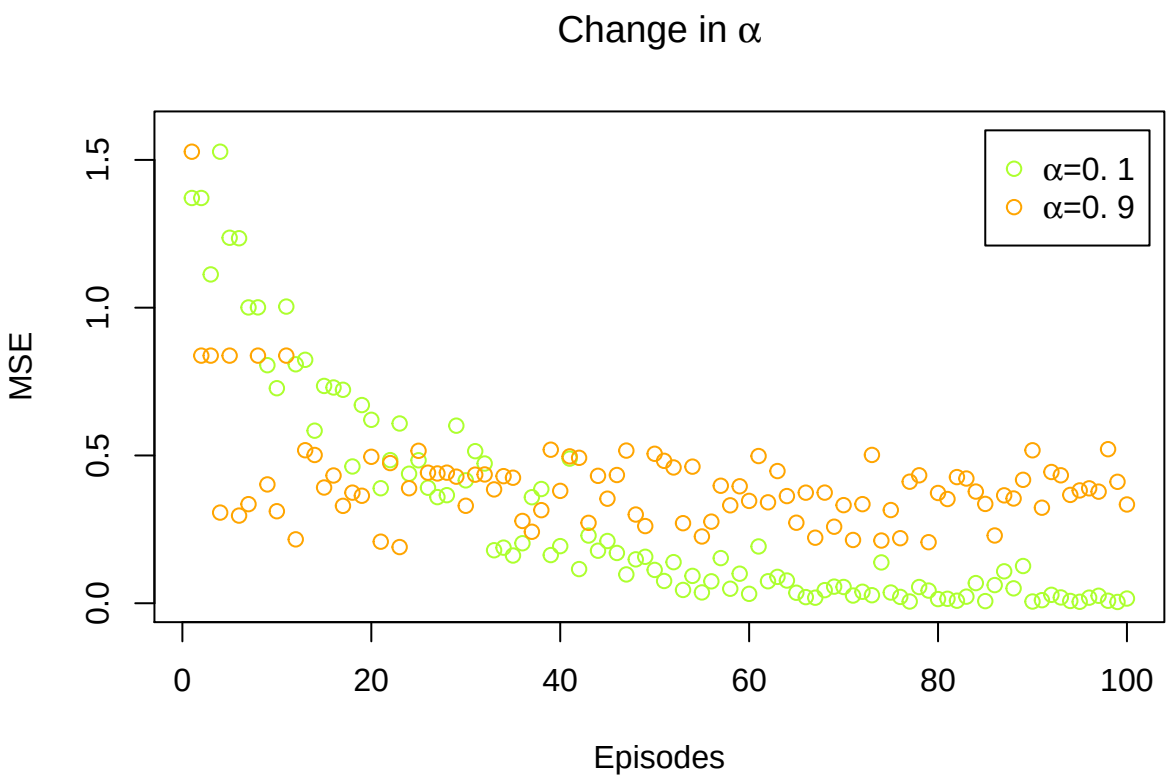
$$V(s_t) = V(s_t) + \alpha \cdot [r_{t+1} + \gamma \cdot V(s_{t+1} - V(s_t))]$$

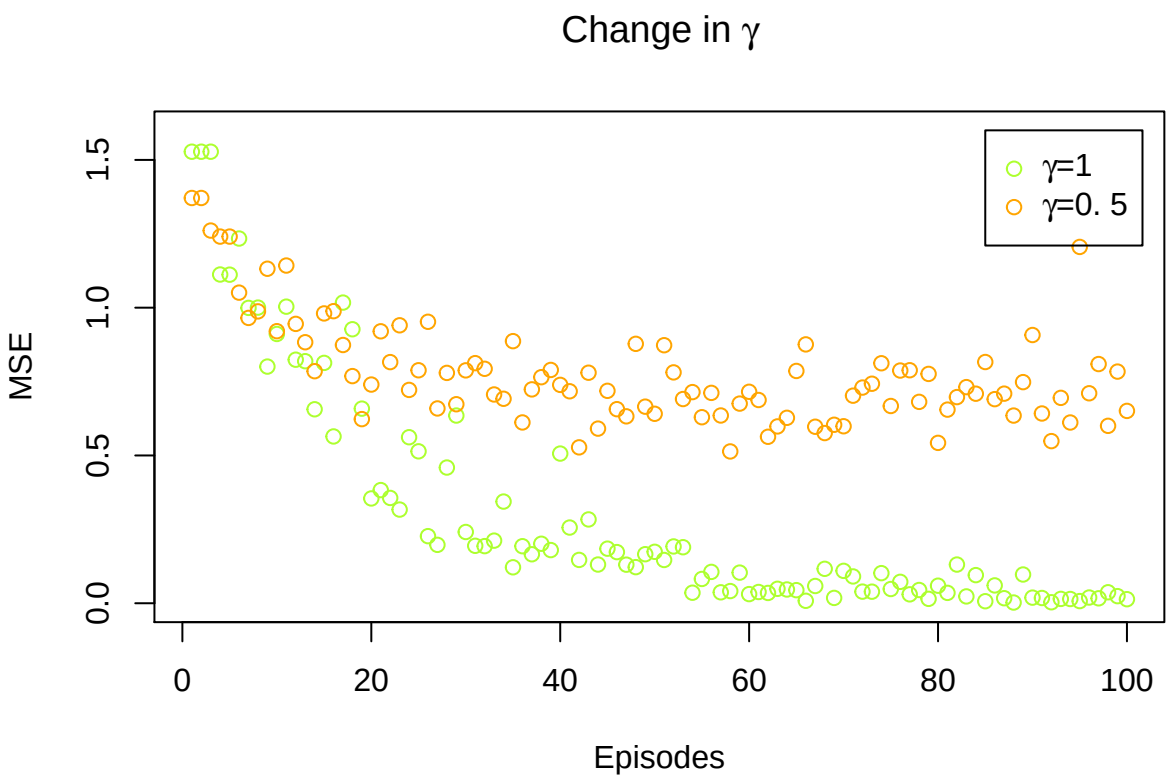
Hierbij is $V(s_t)$ de verwachte waarde bij state $s_t$. De V's in de van alle states worden geinitialiseerd op 0 maar zullen groeien naarmate er meer trials worden gedaan.
$\alpha$ is de learning rate, dit bepaald de stapgrootte van $V(s_t)$.
$\gamma$ is de discout parameter. Deze heeft een invloed op hoe zwaar de volgende state mee telt. $r$ is de reward die wordt gegeven als een goal state is bereikt.

### Parameters bij Temporal Difference Learning

De $\alpha$ heeft invloed op de snelheid waarmee de acquisitiecurve daalt. Bij hogere $\alpha$'s ziet men een snellere daling maar wel een uiteindelijk hogere MSE. Lagere $\alpha$'s dalen langzamer maar halen dan ook een betere score.

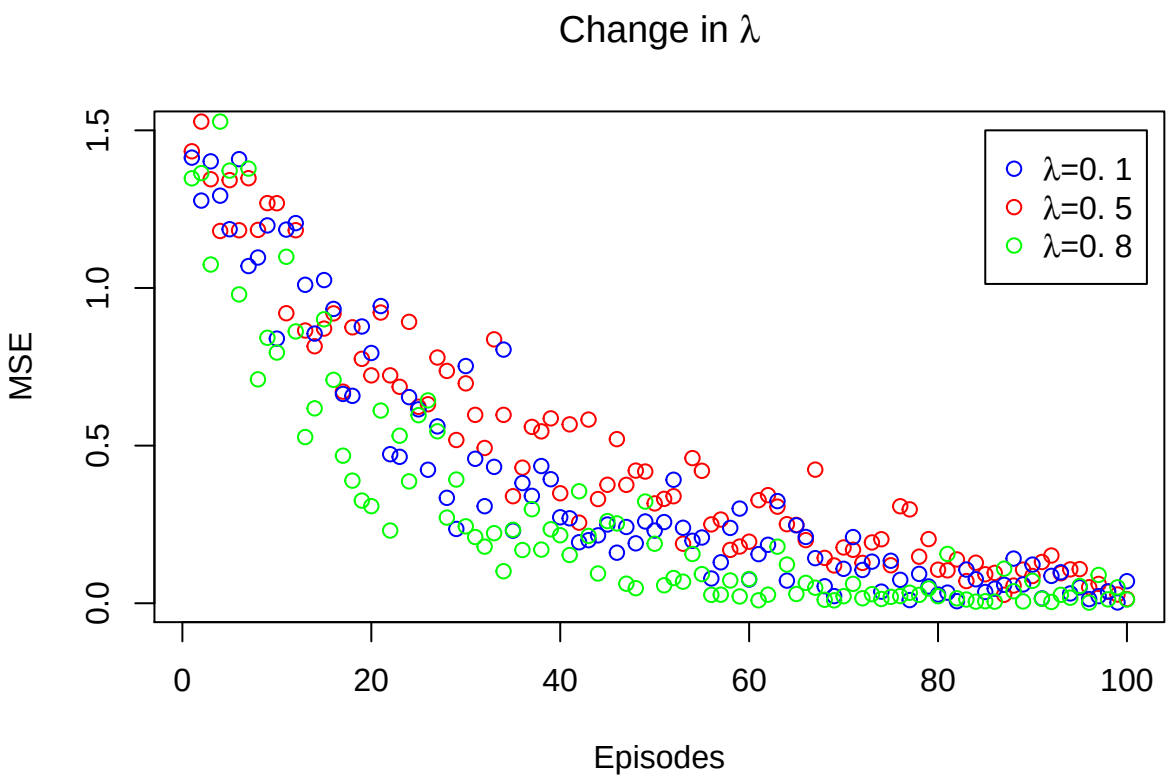
De $\gamma$ speelt een rol in hoe zwaar de volgende state wordt meegewogen in de berekening van de huidige state. Hierdoor zal een hogere $\gamma$ zorgen voor een lagere MSE en vise versa.


Change in α
MSE
0.0
0.5
1.0
1.5
0
20
40
60
80
100
Episodes
α=0. 1
α=0. 9



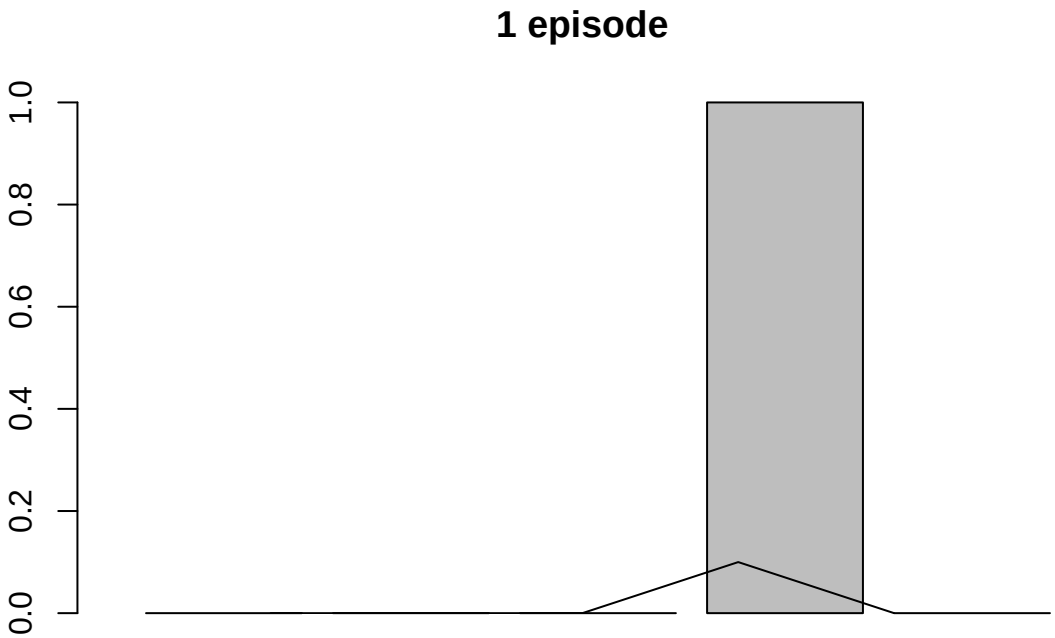

### Eligibility Traces

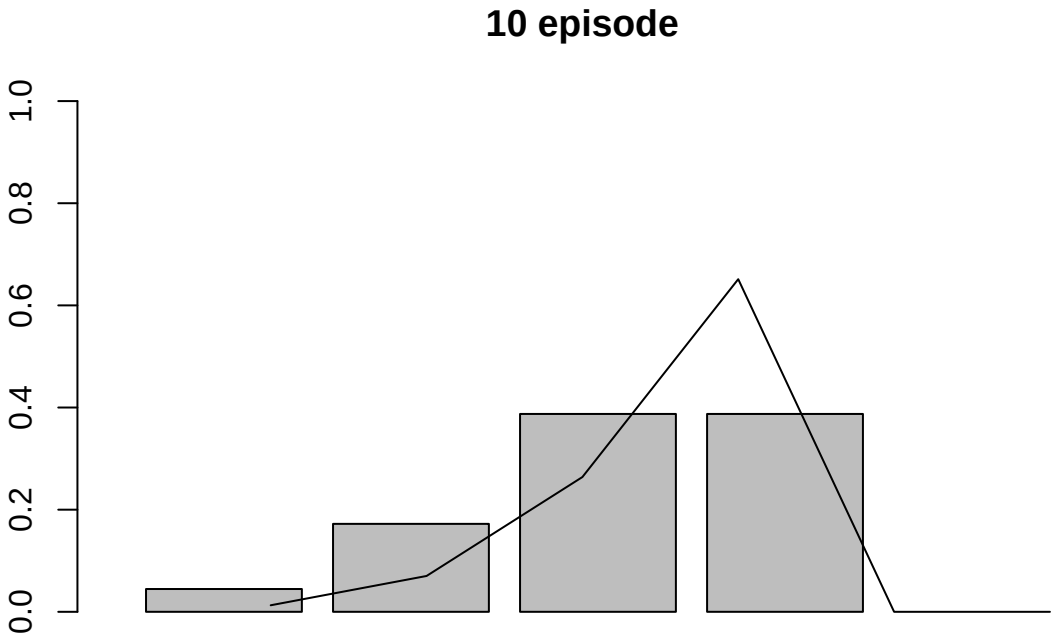
Door het gebruik van Eligibility Traces kan je tijdens het doen van de trials bijhouden wat de impact zal zijn op voorgaande states bij het bereiken van een goal state. Doordat alle states die je hebt bezocht een eligibility value hebben kunnen alle states de verwachtingswaarde updaten zodra er een goal is bereikt. Hoe langer de agent niet meer op een state is geweest hoe lager de eligibility waarde deze state heeft.
Je ziet hier dat een hogere $\lambda$ zorgt voor een snelere daling en een lagere MSE.



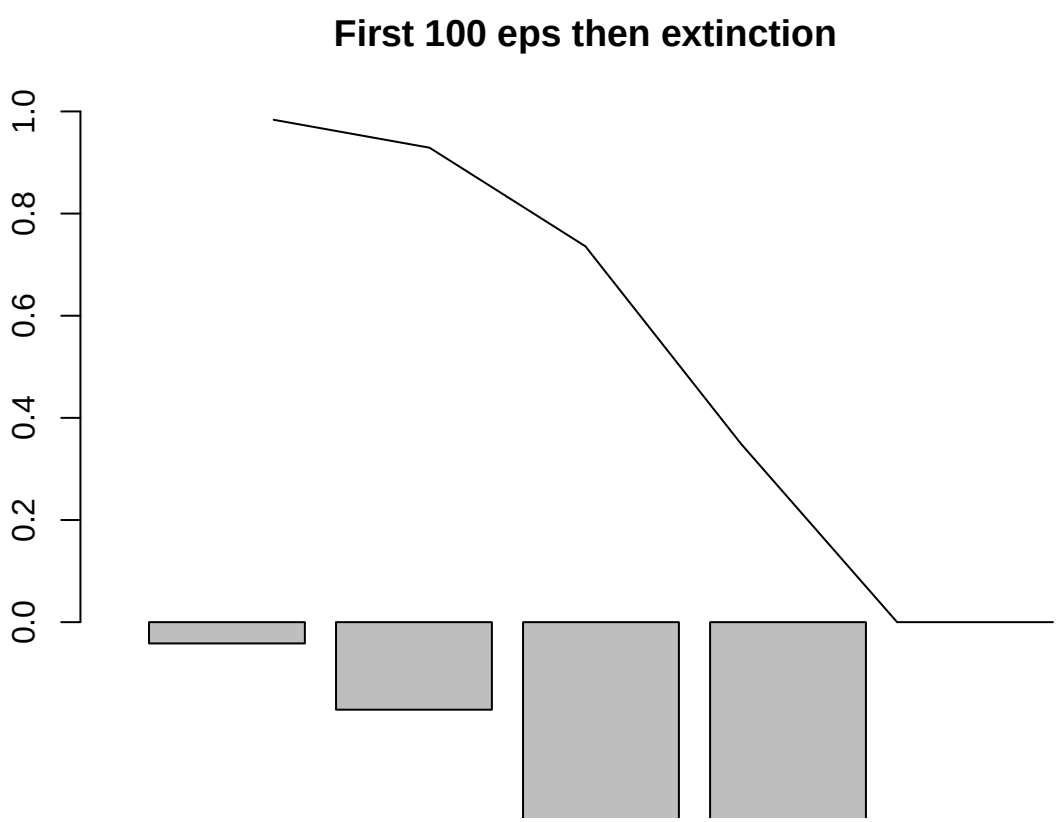

### Prediction Errors

De prediction error is het verschil tussen de huidige verwachtingswaarde in een staat en de geupdate verwachtingswaarde in de staat. Na 1 episode wordt alleen de V in de laaste state gupdate en is daar de error maximaal aangezien de reward 1 is en de verwachting 0. Daarna wordt V in die state geupdate waardoor in latere episodes de error lager zal worden.


1 episode
0.0
0.2
0.4
0.6
0.8
1.0


10 episode
0.0
0.2
0.4
0.6
0.8
1.0


Als er eerst wordt getraind met reward en er daarna geen reward meer komt zal er extinctie plaatsvinden. De prediction error wordt dan negatief.




Uit onderzoek is gebleken dat de temporal predition error vergeleken kunnenworden met bepaalde gebieden in het brein. Dit zijn de dopamine neuronen,het ventrale tegmentale gebied en de substantia nigra. Al deze 3 gebiedenhebben te maken met het beloningssysteem en dan specifiek het systeem waardopamine aan te pas komt. De temporale voorspellings error word ook gebruiktin onderzoeken naar schitzofrenie

**Q1**

Zie appendix voor functie

**Q2**

Voor $\alpha = 0.1$ en $\gamma = 1$: $V(E) = 0.1$
Voor $\alpha = 0.6$ en $\gamma = 1$: $V(E) = 0.6$

**Q3**

```
# Alpha = 0.1
round(td_zero(0.1,1,100),4)
```

```
## [1] 0.0000 0.0704 0.2450 0.4344 0.6475 0.8749 0.0000
```

```
# Alpha = 0.6
round(td_zero(0.6,1,100),4)
```

```
## [1] 0.0000 0.1134 0.4482 0.7626 0.9712 0.9979 0.0000
```

Je ziet dat de V hoger bij 1 ligt aan de rechter kant bij een hogere $\alpha$

### Q4

Best gevonden $\alpha$ is 0.1, best gevonden $\gamma$ is 1.

```
min_mse = Inf
for (a in seq(0,1,0.1)) {
  for (g in seq(0,1,0.1)) {
    V_td = td_zero(a,g,100)
    mse = V_mse(V_td)
    if (mse < min_mse) {
      min_mse = mse
      best_a = a
      best_g = g
    }
  }
}
print(best_a)
```

```
## [1] 0.2
```

```
print(best_g)
```

```
## [1] 1
```

### Q5 en Q6

Plots die zijn gebruikt in uitleg.

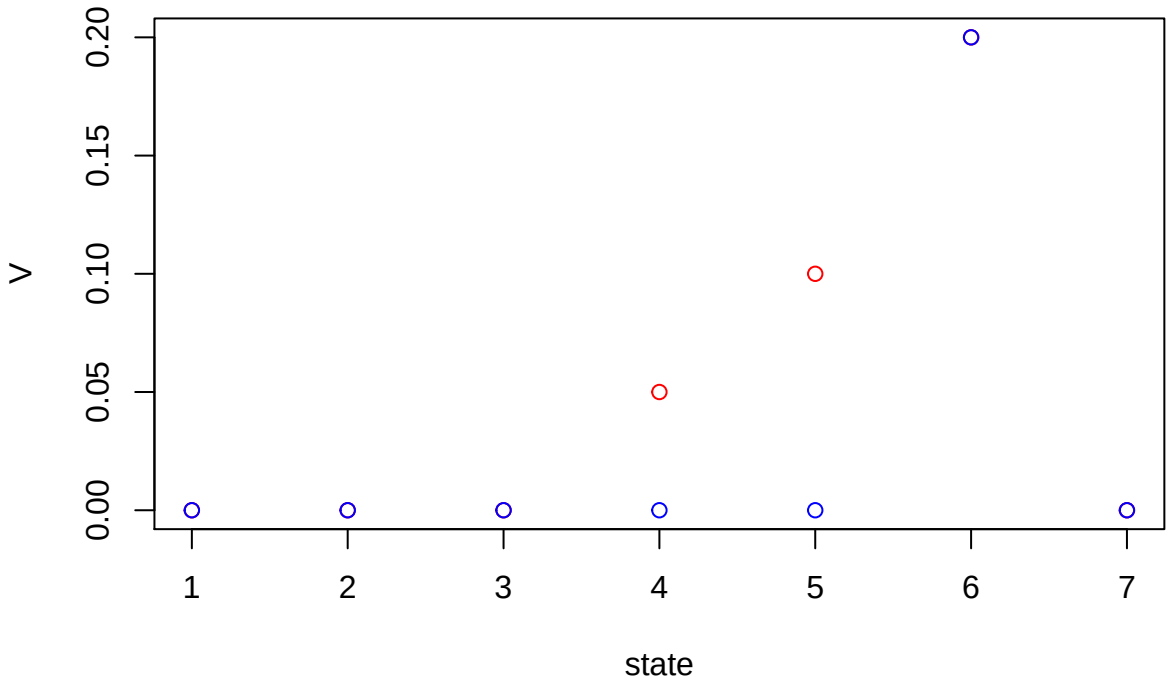
```
# Q5
mse1 = mse_change(0.1, 1, 100)
mse2 = mse_change(0.9, 1, 100)
plot(mse1, col="greenyellow", ylim = c(0,1.6), main=TeX('Change in $\\alpha$'))
points(mse2, col='orange')
legend(85,1.6, legend=c(TeX("$\\alpha$=0.1"),TeX('$\\alpha$=0.9')),
       col=c('greenyellow','orange'), pch=1)
```

```
# Q6
mse1 = mse_change(0.1, 1, 100)
mse2 = mse_change(0.1, 0.5, 100)
plot(mse1, col="greenyellow", ylim = c(0,1.6), main=TeX('Change in $\\gamma$'))
points(mse2, col='orange')
legend(85,1.6, legend=c(TeX("$\\gamma$=1"),TeX('$\\gamma$=0.5')),
       col=c('greenyellow','orange'), pch = 1)
```

### Q7

Door randomheid van de coin flip komt er niet altijd een mooie uitkomst.
Je ziet dat als ze allebei naar de goede goal lopen dat een hogere lambda direct meerdere states update en lambda 0 niet.

```
l1 = td_lambda(0.2, 1, .5, 1)
l2 = td_lambda(0.2, 1, 0, 1)
plot(l1$V, col='red', ylab='V', xlab = 'state')
points(l2$V, col='blue')
```




```
# Lambda = 0.5
l1$E
```

```
## [1] 0.00 0.00 0.00 0.25 0.50 1.00 0.00
```

```
# Lambda = 1
l2$E
```

```
## [1] 0 0 0 0 0 1 0
```

### Q8

Gekozen voor 100 episodes ipv 200 zodat verschil duidelijker is.
Hogere $\lambda$ zorgt voor lagere MSE en sneller daling van MSE.

```
# Q8
mse1 = mse_lambda_change(0.05, 1, .2, 100)
mse2 = mse_lambda_change(0.05, 1, .5, 100)
mse3 = mse_lambda_change(0.05, 1, .8, 100)
plot(mse1, col='red', ylim=c(0,1.5))
points(mse2, col='blue')
points(mse3, col='green')
title(TeX('Change in $\\lambda$'))
legend(85,1.5, legend=c(TeX("$\\lambda$=0.1"),TeX('$\\lambda$=0.5'), TeX("$\\lambda$=0.8")),
       col=c('blue','red', 'green'), pch = 1)
```

```
# Q9
out = td_pe_zero(0.1,1,1)
```

```r
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('1 episode')
lines(out$V, type='l')

# Q10
out = td_pe_zero(0.1,1,10)
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('10 episode')
lines(out$V, type='l')

out = td_pe_zero(0.1,1,100)
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('100 episode')
lines(out$V, type='l')

# Q11
out = td_pe_zero(0.1,1,100)
out = td_pe_ext(0.1,1,10, out$V)
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('First 100 eps then extinction')
lines(out$V, type='l')

# Q12
out = td_pe_zero(0.9,1,5)
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('Increase of alpha')
lines(out$V, type='l')
# Andere optie is om eligibility traces toe te passen en hoge lamda te kiezen.

# Q13
## Weird gedrag in functie. Als reward wordt omgeswitcht is V hoog ander niet.
out = td_pe_chance(0.1,1,100)
barplot(out$PE, ylim=c(0,1), xlim=c(0,6))
title('Chance of 50% that coin doesnt fall')
lines(out$V, type='l')
```

## Appendix: Functions used

```r
td <- function(R, V, t, step, a, g) {
  d = (R[t+step] + g * V[t+step] - V[t])
  V[t] = V[t] + a * d
  return(V)
}
td_zero <- function(a, g, e) {
  R = c(rep(0,6), 1)
  V = rep(0,7)
  for (i in 1:e) {
    t = 4
    while (1 < t & t < length(V)) {
      step_t = sample(c(-1,1), 1);
      V = td(R,V,t,step_t,a,g)
      t = t + step_t
```

```
    }
  }
  return(V)
}

td_lambda <- function(a, g, l, e) {
  R = c(rep(0,6), 1)
  V = rep(0,7)

  for (i in 1:e) {
    E = rep(0,7)
    t = 4
    while (1 < t && t < length(V)) {
      step_t = sample(c(-1,1), 1)
      # step_t = 1
      E = g * l * E
      E[t] = 1
      d = R[t+step_t] + g * V[t+step_t] - V[t]
      V = V + a * d * E
      t = t + step_t
    }
  }
  return(list(V=V, E=E))
}
V_mse <- function(V_td) {
  V_real = c(0,1,2,3,4,5,0)/6
  mse = sum((V_real - V_td) ^ 2)
  return(mse)
}

mse_change <- function(a, g, e) {
  mses = c()
  for (e in (1:e)) {
    V_td = td_zero(a,g,e)
    mses[e] = V_mse(V_td)
  }
  return(mses)
}

mse_lambda_change <- function(a, g, l, e) {
  mses = c()
  for (e in (1:e)) {
    V_td = td_lambda(a,g,l,e)$V
    mses[e] = V_mse(V_td)
  }
  return(mses)
}

td_pe_zero <- function(a, g, e) {
  R = c(rep(0,4), 1)
  V = rep(0,5)
  for (i in 1:e) {
    t = 1
```

```
    PE = rep(0,5)
    while (t < length(V)) {
      PE[t] = R[t+1] + g * V[t+1] - V[t]
      V[t] = V[t] + a * PE[t]
      # V = td(R,V,t,step_t,a,g)
      t = t + 1
    }
  }
  return(list(V=V, PE=PE))
}

td_pe_ext <- function(a, g, e, V) {
  R = c(rep(0,4), 0)
  # V = rep(0,5)
  for (i in 1:e) {
    t = 1
    PE = rep(0,5)
    while (t < length(V)) {
      PE[t] = R[t+1] + g * V[t+1] - V[t]
      V[t] = V[t] + a * PE[t]
      # V = td(R,V,t,step_t,a,g)
      t = t + 1
    }
  }
  return(list(V=V, PE=PE))
}

td_pe_chance <- function(a, g, e) {
  R = c(0,0,0,0,1)
  # R = c(0,0,0,0,0)
  V = rep(0,5)
  for (i in 1:e) {
    coin = sample(c(0,1),1)
    if (coin) {
      R = c(0,0,0,0,0)
      # R = c(0,0,0,0,1)
    }
    t = 1
    PE = rep(0,5)
    while (t < length(V)) {
      PE[t] = R[t+1] + g * V[t+1] - V[t]
      V[t] = V[t] + a * PE[t]
      t = t + 1
    }
  }
  return(list(V=V, PE=PE))
}
```