CUNY MSDA - IS609 Project

Ben Arancibia
Xingjia Wu
Shipra Ahuja

Project # 1

Chapter 5 Section 3 Project 3

Craps is a dice game in which players make wages on the outcome of the roll, or a series of rolls, of a pair of dice.

- Question: Craps - Construct and perform a Monte Carlo simulation of the popular casino game of craps.
- There are two basic bets in craps, pass and don't pass. In the pass bet, you wager that the shooter will win; in the don't pass bet, you wager that the shooter will lose. Conduct of the game:
  - Roll a 7 or 11 on the first roll: Shooter wins (pass bets win and don't pass bets lose)
  - Roll a 12 on the first roll: Shooter loses (boxcars; pass and don't pass bets lose)
  - Roll a 2 or 3 on the first roll: Shooter loses (pass bets lose, don't pass bets win)
  - Roll 4, 5, 6, 8, 9, 10 on the first roll: This becomes the point. The object then becomes to roll the point again before rolling a 7.
- The shooter continues to roll the dice until the point or a 7 appears. Pass bettors win if the shooter rolls the point again before rolling a 7. Don't pass bettors win if the shooter rolls a 7 before rolling the point again.

Craps involves the rolling of two dice. The assumption is that the dice are fair and the outcomes of the rolls are independent.

- Mathematically you can solve for the possibilities

Initial Roll	Probability of Winning	Probability in Decimal
4	3/36 x 3/9	0.027778
5	4/36 x 4/10	0.044444
6	5/36 x 5/11	0.063131
7	6/36	0.166667
8	5/36 x 5/11	0.063131
9	4/36 x 4/10	0.044444
10	3/36 x 3/9	0.027778
11	2/36	0.055556
	Total:	0.492929

Probability the Shooter wins = 49.29%
Probability the Shooter loses = 50.71%

Write an algorithm and code it in the computer language of your choice. Run the simulation to estimate the probability of winning a pass bet and the probability of winning a don’t pass bet. Which is the better bet?

- Wrote algorithm in R
  - Code included in the report
- The better bet is house bets
  - Pass bets: 49.29%
  - Don't Pass bets: 47.93%

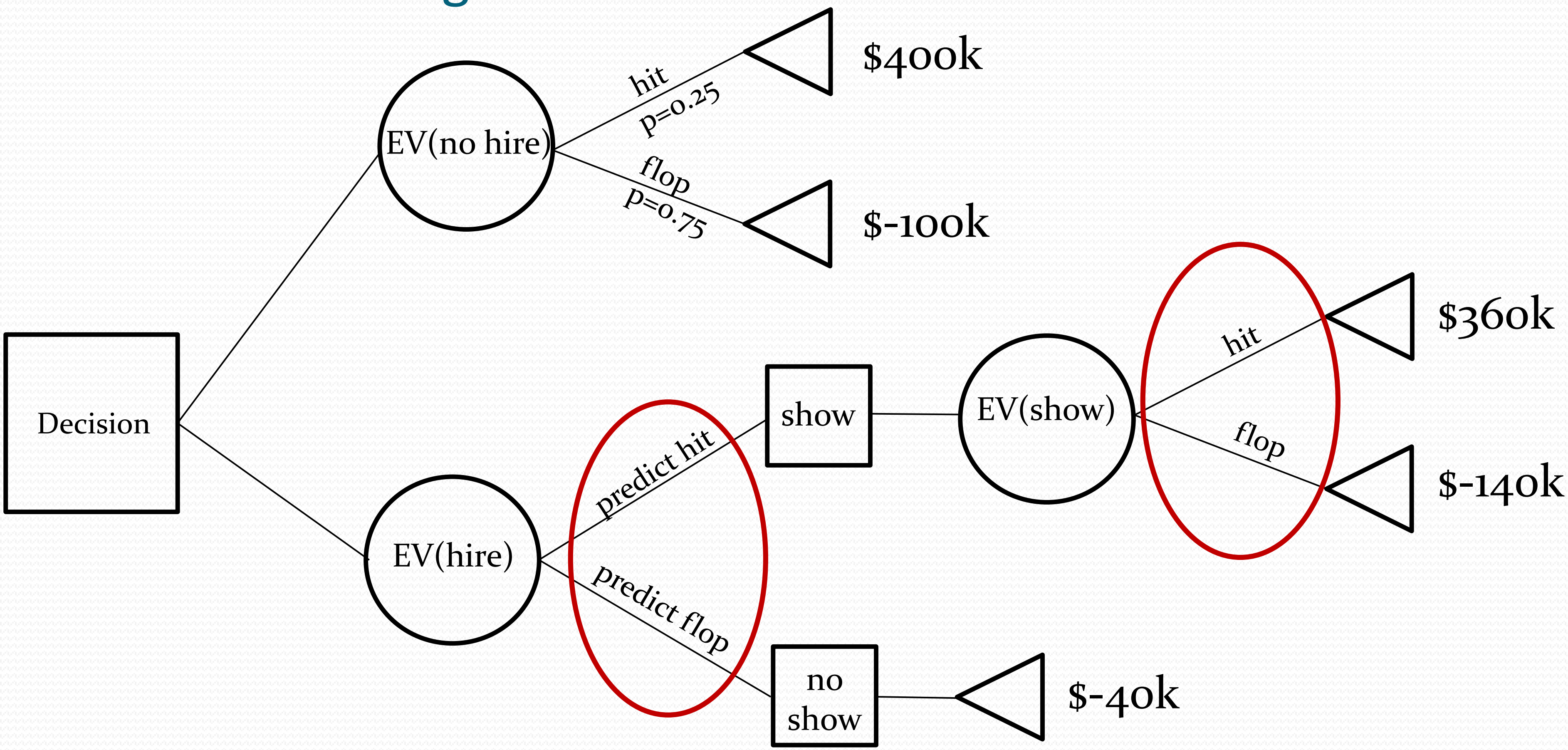
Project # 2

Chapter 9 Section 3 Project 4

Problem

The NBC TV network earns an average of $400,000 from a hit show and loses an average of $100,000 on a flop (a show that cannot hold its rating and must be canceled). If the network airs a show without a market review, 25% turn out to be hits, and 75% are flops. For $40,000, a market research firm can be hired to help determine whether the show will be a hit or a flop. If the show is actually going to be a hit, there is a 90% chance that the market research firm will predict a hit. If the show is going to be a flop, there is an 80% chance that the market research will predict the show to be a flop. Determine how the network can maximize its profits over the long haul.

Decision Tree Diagram



Building network using gRain

```r
suppressWarnings(suppressMessages(library(gRain)))

hf <- c("hit", "flop")
phf <- c("p.hit", "p.flop")

# Specify the Conditional Probability Tables
show <- cptable(~show, values=c(25, 75), levels=hf)
predict <- cptable(~predict|show, values= c(0.9, 0.1, 0.2, 0.8), levels=phf)

# Compile plist
plist <- compileCPT(list(show, predict))
summary(plist)
```

```
## $show
## show
##  hit flop
## 0.25 0.75
##
## $predict
##         show
## predict  hit flop
##   p.hit  0.9  0.2
##   p.flop 0.1  0.8
```

```r
# Build the network
net <- grain(plist)
```

Query network (1)

- The probability that the market review predicts a hit

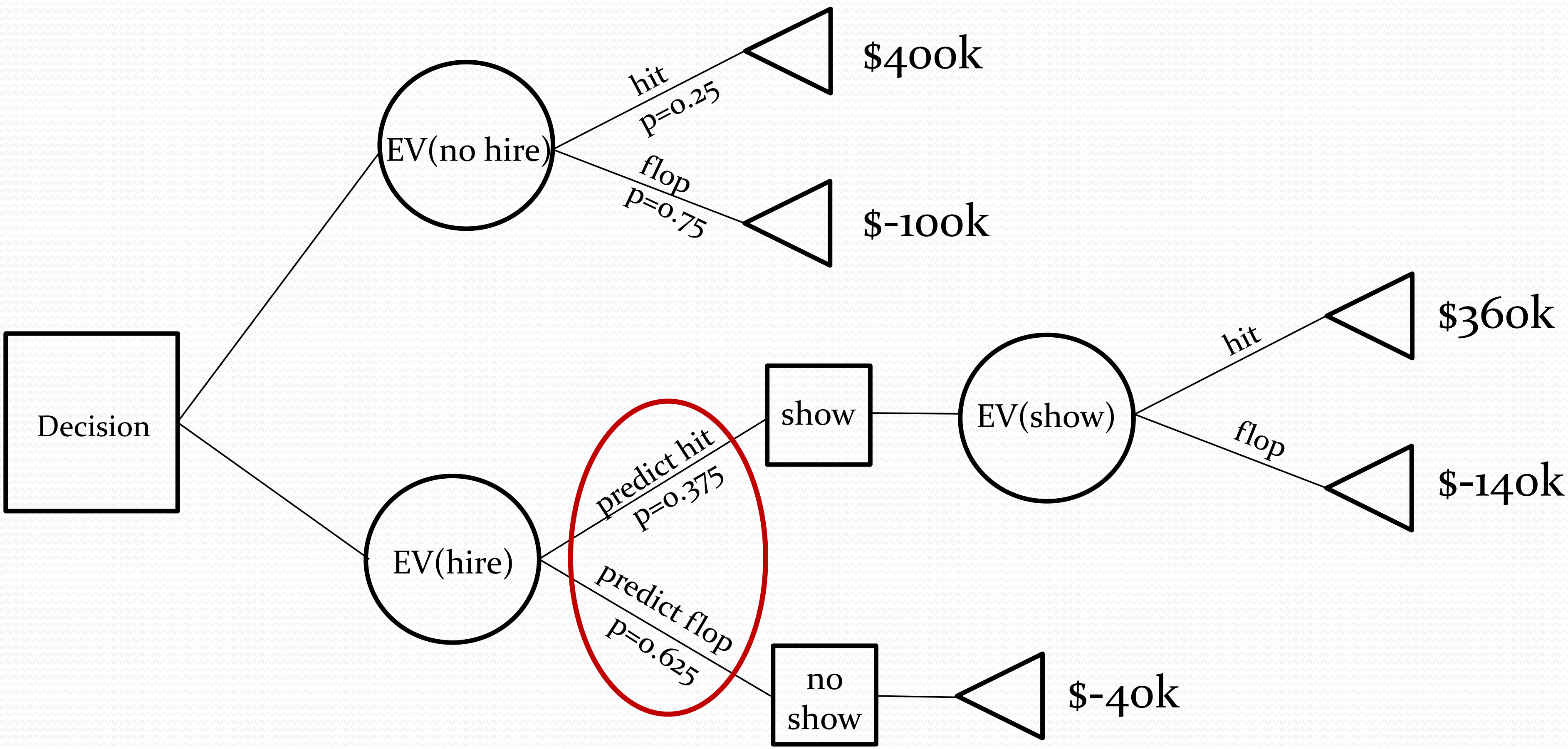
```
q1 <- setFinding(net, nodes="predict", states=c("p.hit"))
(phit <- pFinding(q1))
```

```
## [1] 0.375
```

- The probability the market review predicts a flop

```
q2 <- setFinding(net, nodes="predict", states=c("p.flop"))
(pflop <- pFinding(q2))
```

```
## [1] 0.625
```

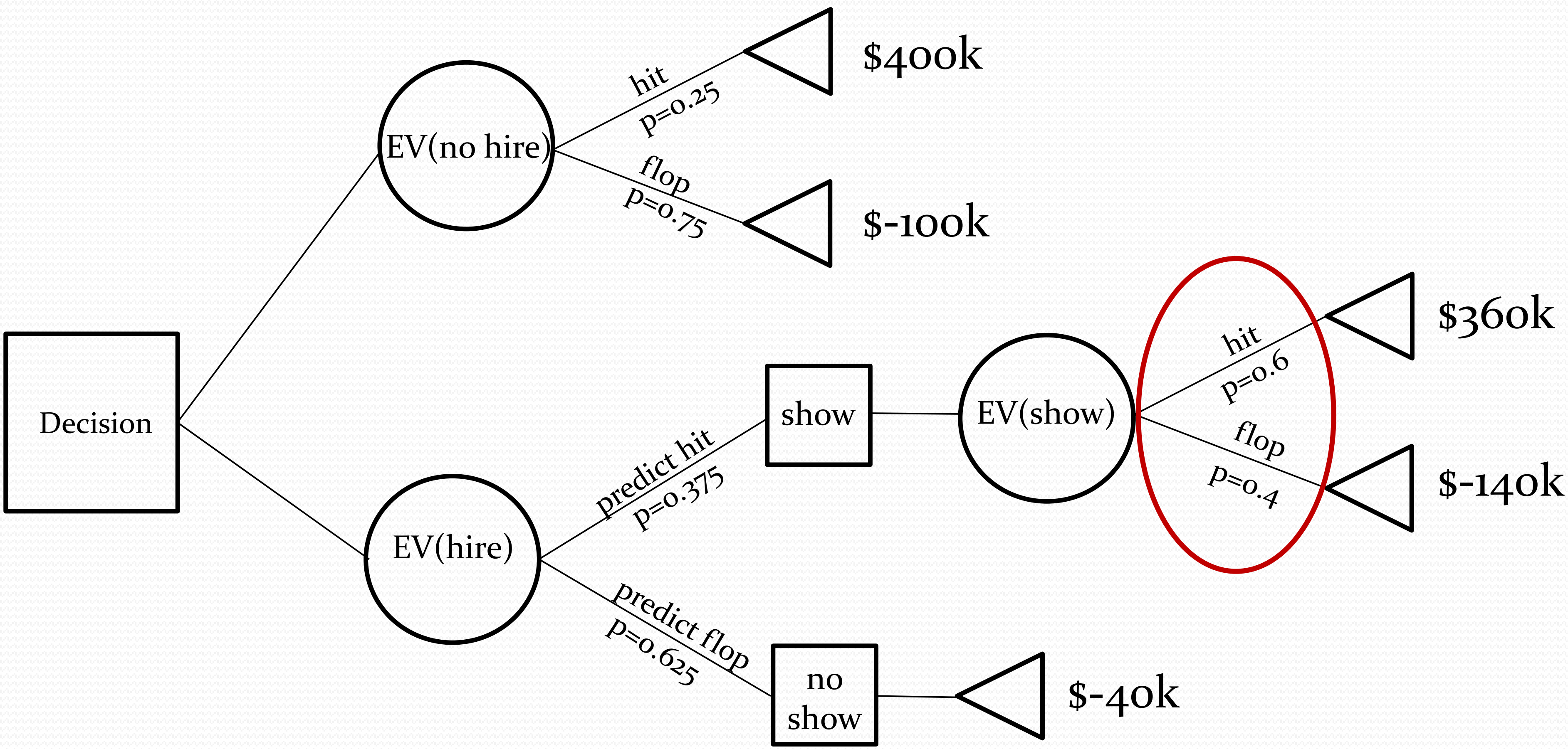

Decision
EV(no hire)
hit
p=0.25
$400k
flop
p=0.75
$-100k
EV(hire)
predict hit
p=0.375
show
EV(show)
hit
$360k
flop
$-140k
predict flop
p=0.625
no show
$-40k

Query network (2)

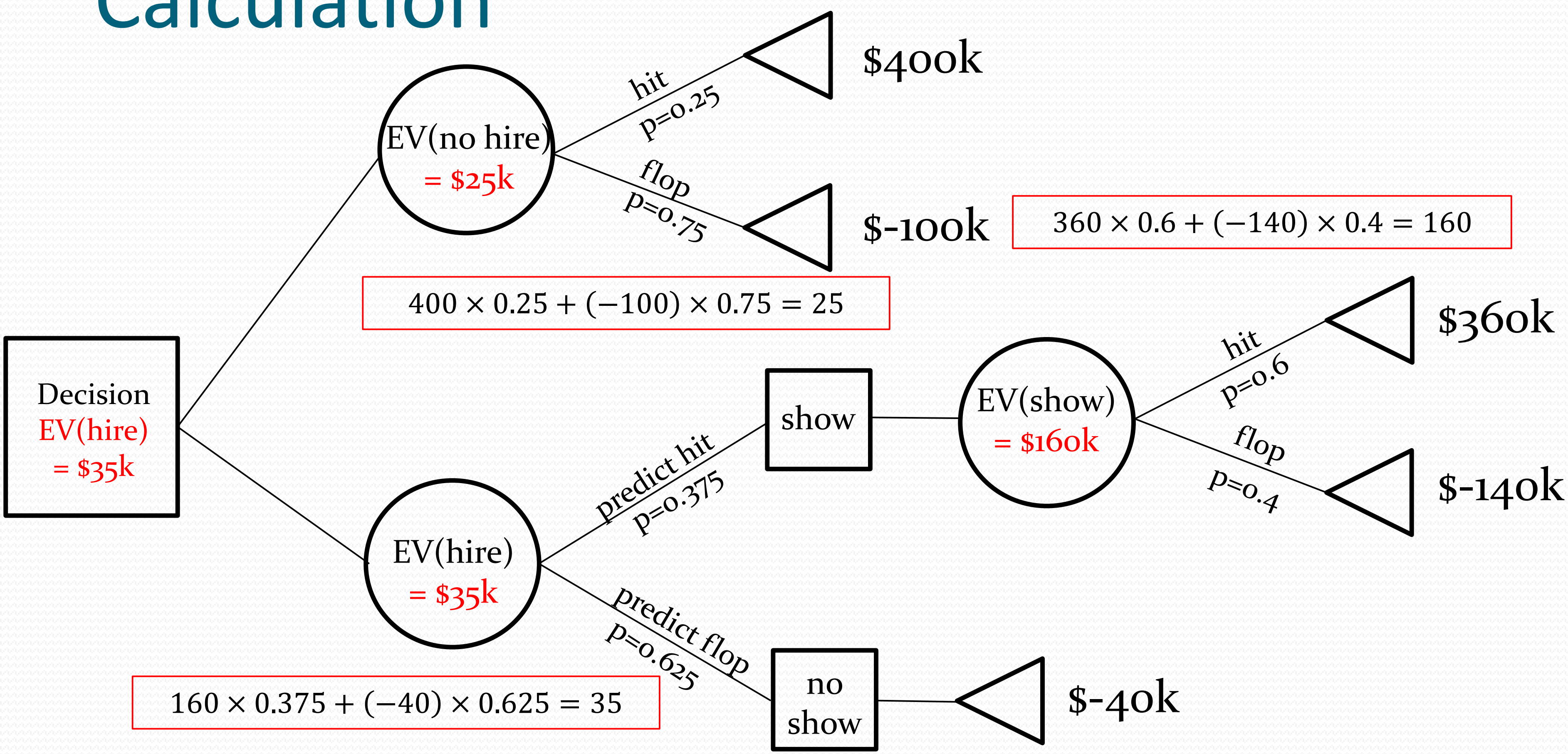
- The probability that a show actually turns out to be a hit or a flop when the market review predicts a hit

```
(querygrain(q1, nodes = "show", type="marginal"))
```

```
## $show
## show
##  hit flop
##  0.6  0.4
```


Decision
EV(no hire)
hit
p=0.25
$400k
flop
p=0.75
$-100k
EV(hire)
predict hit
p=0.375
show
EV(show)
hit
p=0.6
$360k
flop
p=0.4
$-140k
predict flop
p=0.625
no show
$-40k

Calculation



Conclusion

Since the expected profit of hiring market review (35k) is higher than not hiring (25k), the network can hire a market review to maximize its profits over the long haul.

Project # 3

Chapter 11 – Section 5 – Project 5

Problem Description

- Analyze the spread of a communicable disease depicted by ordinary differential equation below using various methods
  - $dN/dt = 0.25N(10-N)$

- Qualitative Graphical Analysis
  - Phase Lines for First & Second Derivatives
  - Solution Curves
  - Slope Field Plot

- Actual Solution
  - Separation of Variables technique

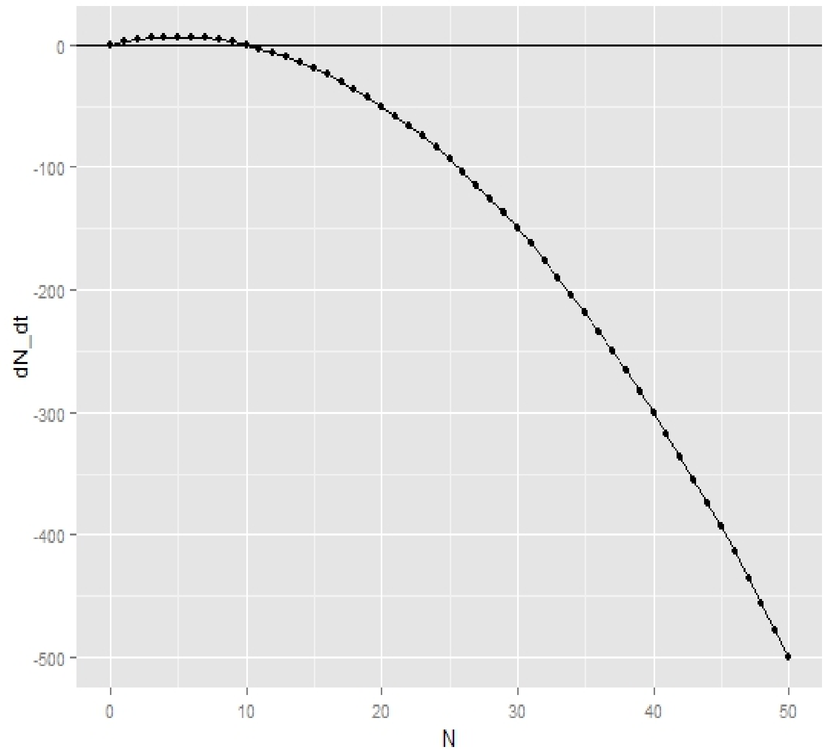
- Numerical Methods
  - Euler's Method
  - Runge -Kutta Method

Autonomous Differential Equation Points of Equilibrium

- Autonomous Differential Equation
  - $dN/dt = 0.25N(10-N)$
- When $dN/dt = 0$
  - Differential Equation – “Autonomous Differential Equation”
- Points at which $dN/dt=0$
  - Rest Points or Equilibrium Points
- $dN/dt = 0$ when
  - $N=0$
  - $N=10$
- $N= 0$ and $N=10$ are Equilibrium Points

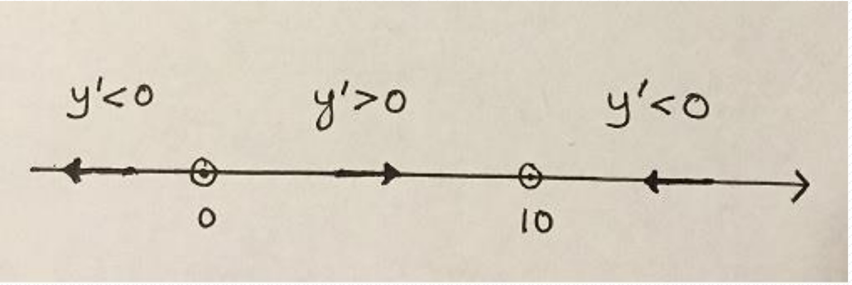
Rate of spread of the disease

- Plot shows the rate at which the communicable disease spreads
- Equilibrium points are N=0 and N=10 since dN/dt=0 at these points
- Rate of change is fastest at N=5 as dN/dt is max at N=5

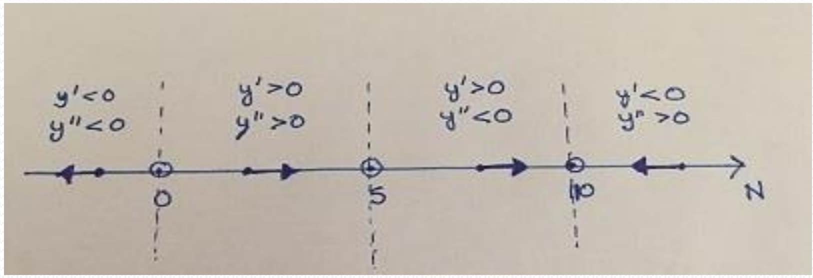


Qualitative Graphical Method
Phase Lines

Phase Lines (Using First-Derivative)



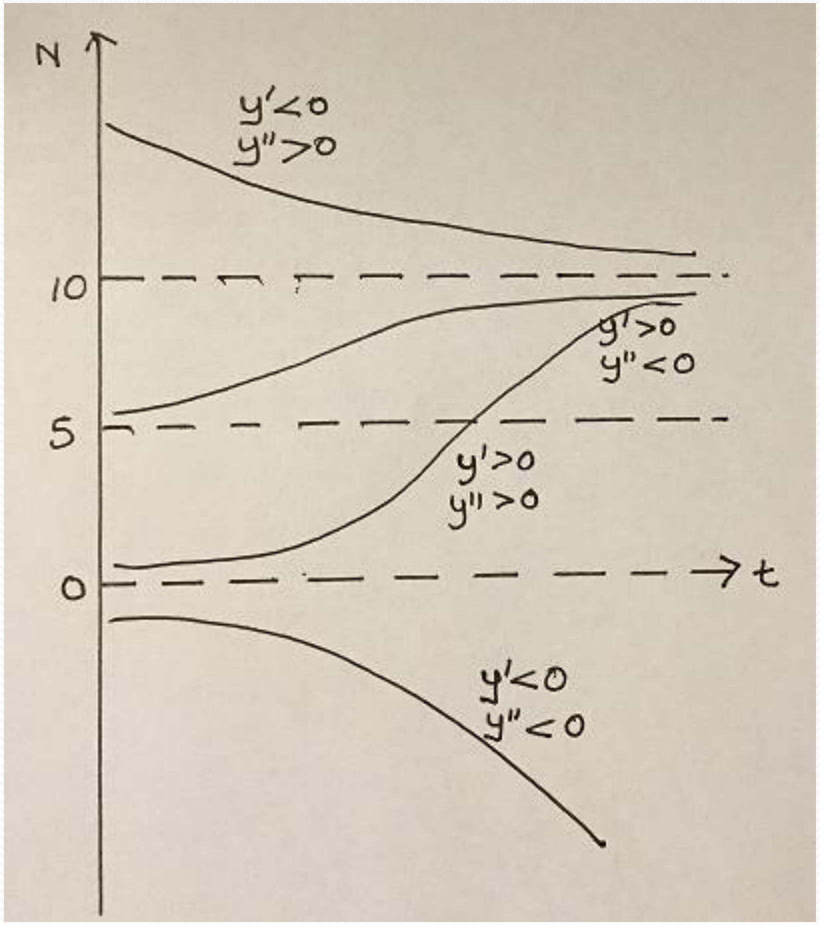
Phase Lines (Using Second-Derivative)



- Arrows go away from N=0 (Unstable Equilibrium)
- Arrows lead to N=10 (Stable Equilibrium)

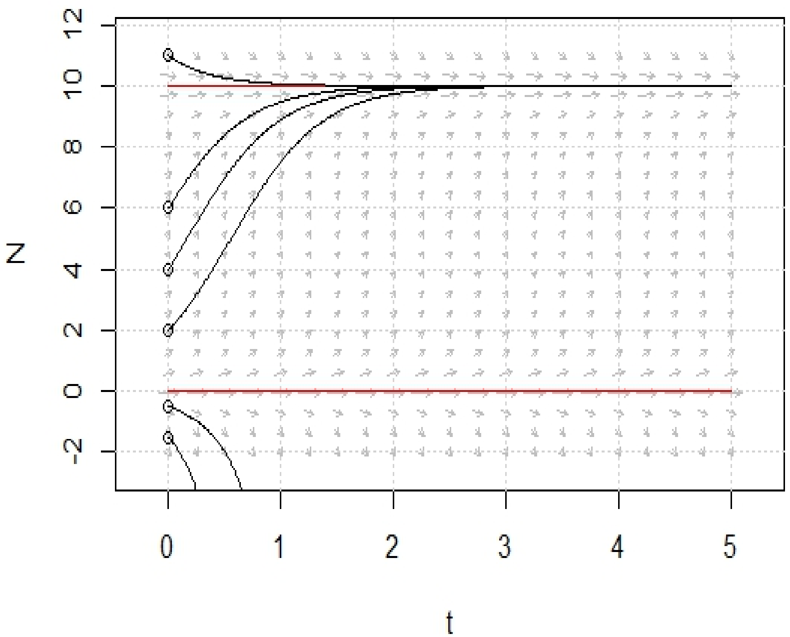
Solution Curves

- Trajectories of solution curves move away from N=0 (Unstable Equilibrium)
- Trajectories of solution curves move towards N=10 (Stable Equilibrium)



Slope Fields

- Equilibrium Points N=0 and N=10 are shown with red lines
- Solution Curves are shown for some initial conditions
  - N<0
  - 0<N<10
  - N>10
- Solution curves move towards N=10 (Stable)
- Solution Curves move away from N=0 (Unstable)



Stability of Equilibrium Points

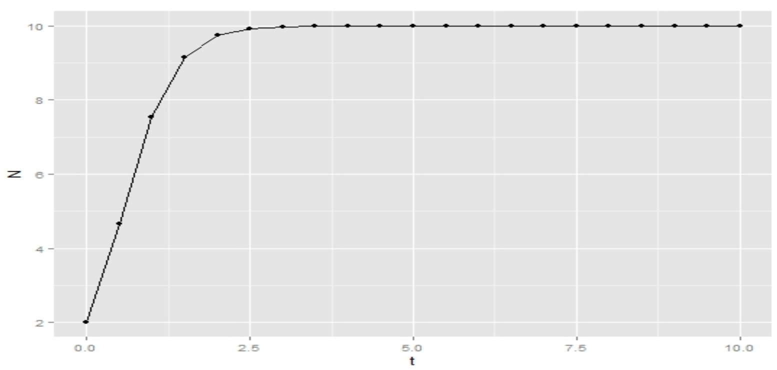
- Equilibrium points can be stable or unstable
- Stability can be determined by looking at
  - Phase Lines
    - Arrows moving away from equilibrium points - UNSTABLE equilibrium point (N=0)
    - Arrows moving towards equilibrium points - STABLE equilibrium point(N=10)
  - Solution Curves Trajectories
    - Solution Curves moving towards Equilibrium points (STABLE - N=10) and moving away from equilibrium points (UNSTABLE – N=0)
  - Slope Fields
    - Small line segments move towards equilibrium points (STABLE – N=10) and away from equilibrium points(UNSTABLE – N=0)

Actual Solution
Separation of Variables Technique

- Actual Solution
  - Integrate the differential equation dN/dt = 0.25N(10-N) using separation of variables technique

$$N = \frac{10Ke^{2.5t}}{Ke^{2.5t}-1}$$

  - For N(0) = 2



- Actual Solution for N(0)=2

$$N = \frac{-2.5e^{2.5t}}{-0.25e^{2.5t}-1}$$

- Actual Solution for N(0)=7

$$N = \frac{-23.3e^{2.5t}}{-2.33e^{2.5t}-1}$$

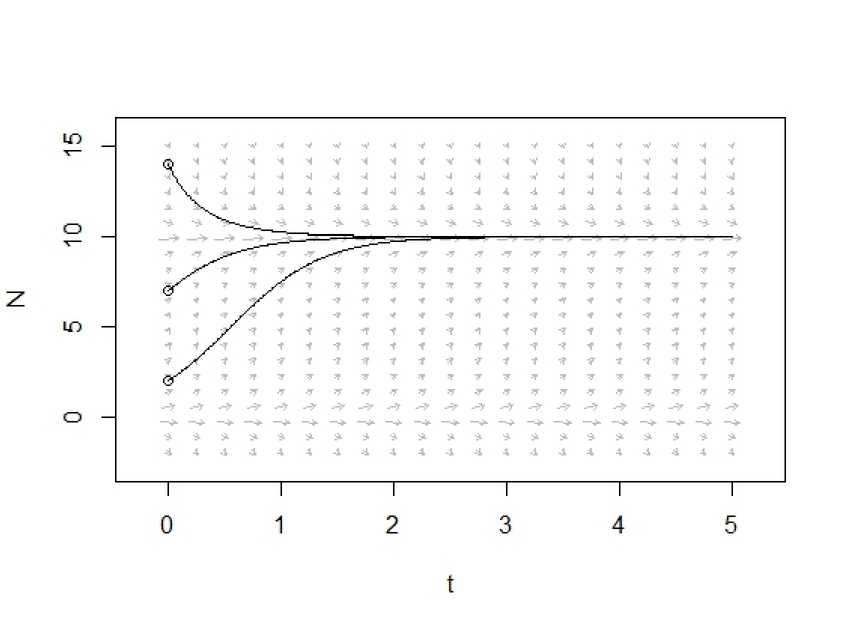
- Actual Solution for N(0)=14

$$N = \frac{35e^{2.5t}}{3.5e^{2.5t}-1}$$

Comparison
Qualitative Plot with Actual Solution

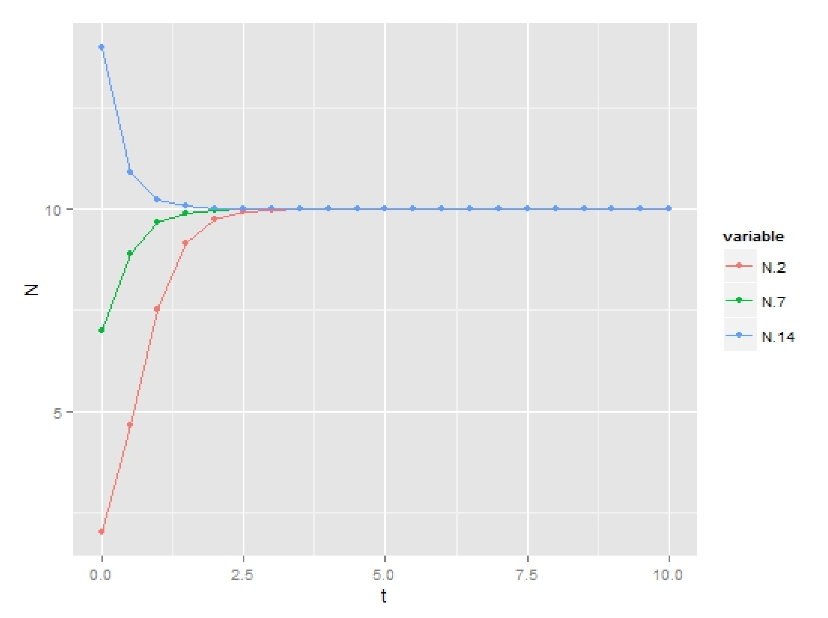
Qualitative Plot

N(0)=2, N(0)=7, N(0)=14



Actual Solution Plot

N(0)=2, N(0)=7, N(0)=14



Numerical Methods
Euler's Method

```
# Function to compute numerical solution using Euler's method


eulers <- function(h)        # Accept step size

{
  N <- 0
  start <- 0                 # Start of interval
  end <- 10                  # End of Interval
  N0 <- 2                    # Initial Value
  nsteps <- (end-start)/h    # Compute number of steps required
  N[1] <- N0                 # Put intial value in first position of array

  t <- seq(start,end,by=h)   # Generate sequence of time(x-axis)

  for(i in 1:nsteps)         # Loop to generate data points
  {
    N[i+1] <- N[i] + ((0.25*N[i])*(10-N[i]))*h
  }

  df <- data.frame(cbind(t,N))

  return(df)
}
```

- Differential Equation $dN/dt=0.25N(10-N)$ has been solved using Euler's Method
- Step Size $h=0.1$ and $h=1$ has been used to compute the solution

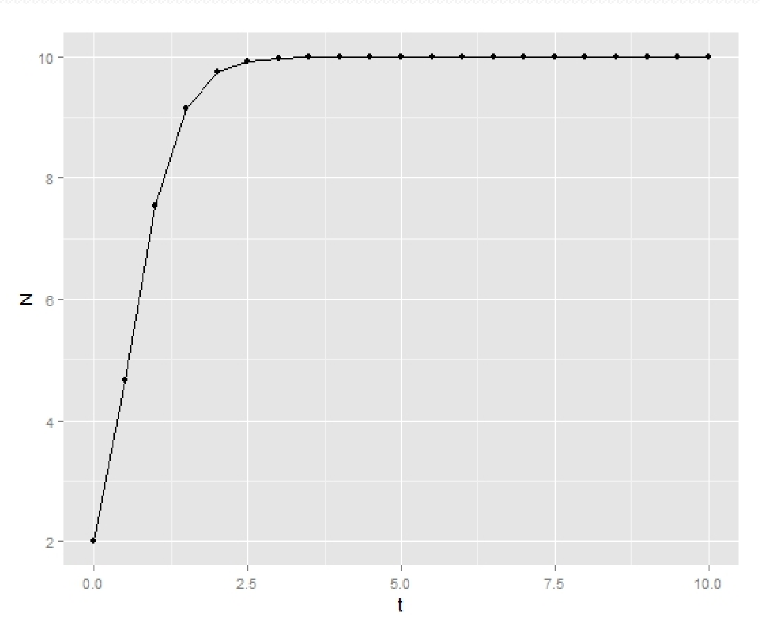
Comparison
Euler's Method with Actual Solution

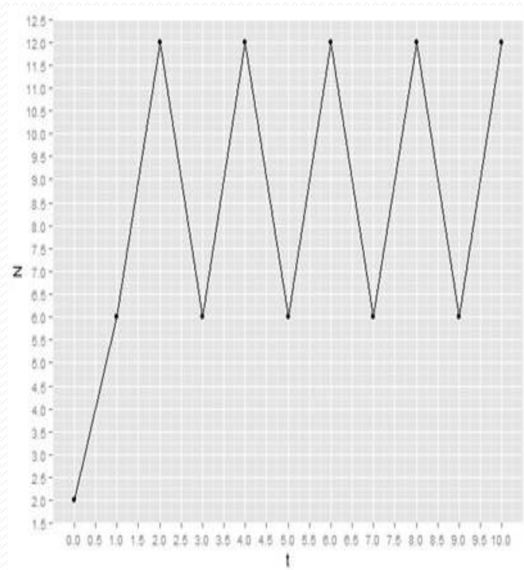
Actual Solution

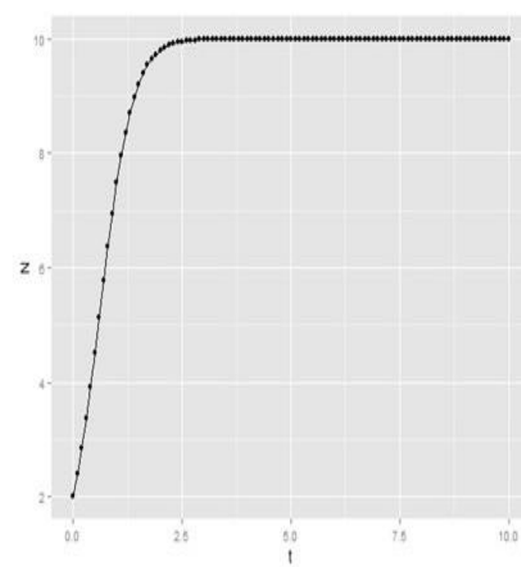
- Reaches equilibrium(N=10) at t=7.5
- Matches Euler's Solution with h=0.1
- No match with Euler's Solution for h=1

Euler's Solution

- Larger step size(h=1)
  - Solution does not reach equilibrium
- Smaller step size (h=0.1)
  - Reaches equilibrium(N=10) at t=6.5







Numerical Methods
Runge-Kutta Method

```
##########################################################################
# Function to compute numerical solution using 4th order Runge Kutta method
##########################################################################

rk4 <- function(f,h)

  {
    start <- 0                  # Start of interval
    end <- 10                   # End of Interval
    t0 <- 0                     # Initial value of t
    N0 <- 2                     # Initial Value of N
    nsteps <- (end-start)/h     # Compute number of steps required for a given step size

    vt <- double(nsteps + 1)    # Initialize vector vt
    vN <- double(nsteps + 1)    # Initialize vector vN

    vt[1] <- t <- t0            # Put initial value of t in 1st position of array vt
    vN[1] <- N <- N0            # Put initial value of N in 1st position of array vN

    # Loop computing k1,k2,k3,k4 using Runge Kutta method

    for(i in 1:nsteps)
      {

        k1 <- f(t, N)
        k2 <- f(t + 0.5*h, N + (k1*0.5*h))
        k3 <- f(t + 0.5*h, N + (k2*0.5*h))
        k4 <- f(t+h, N + (k3*h))

        vt[i + 1] <- t <- t0 + i*h
        vN[i+1] <- N <- N + ((1/6)*(k1+(2*k2)+(2*k3)+k4)*h)

      }
        cbind(vt, vN)
}
```

- Differential Equation $dN/dt=0.25N(10-N)$ has been solved using 4th order Runge-Kutta Method
- Step Size h=0.1 and h=1 has been used to compute the solution

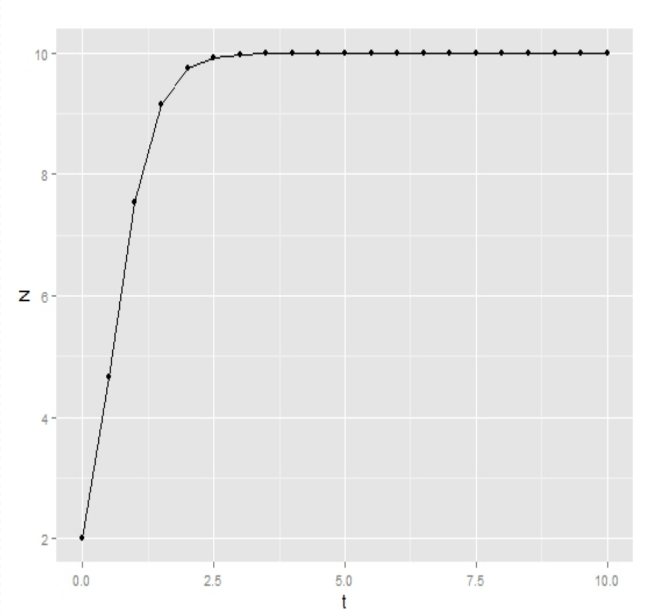
Comparison
Runge-Kutta Method with Actual Solution

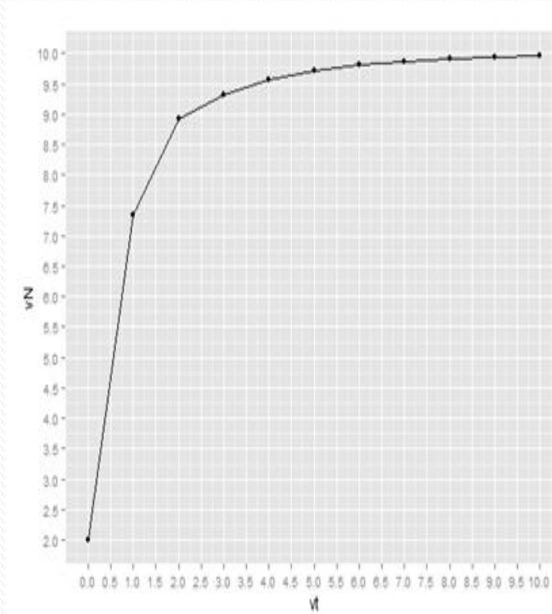
Actual Solution

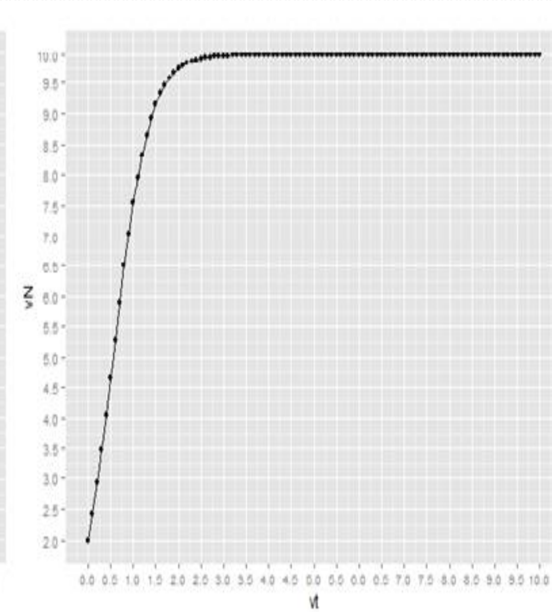
- Reaches equilibrium(N=10) at t=7.5
- Matches Runge-Kutta's Solution with h=0.1
- Fairly matches Runge-Kutta Solution for h=1

Runge-Kutta Solution

- Larger step size(h=1)
  - Reaches equilibrium(N=10) **very slowly** at t=36
- Smaller step size (h=0.1)
  - Reaches equilibrium(N=10) **quickly** at t=7.5







Conclusion

- All methods below for analysis of ordinary differential equation indicate N=10 as a point of stable equilibrium and N=0 as a point of unstable equilibrium
  - Qualitative - Phase Lines & Solution Curves
  - Slope Field Plot
  - Actual Solution using Separation of Variables
  - Euler's Method
  - Runge Kutta Method
- Relative Error between Actual Solution vs Solution from Numerical Methods
  - Error is very small when step size is small
  - Error increases as step size increases
- Runge -Kutta Method vs Euler's Method
  - More accurate than Euler's method
  - Much closer to actual solution
  - Relative Error between Actual Solution and Runge Kutta solution is negligible compared to Euler's Method

Thank You

Questions?