Betting Strategies

This is sourced from STAD70 course practice questions and sample R code taught by professor Sotos. If you have any questions/concerns/comments feel free to email me: cristal.wang111@gmail.com (mailto:cristal.wang111@gmail.com).

1.Kelly criterion in betting

This session will compares two betting strategies: Kelly criterion (optimal fractional betting) and Constant bet of $10 per game. We simulate a series of bets and track wealth over time under each strategy.

1.1.Kelly Criterion

- **Kelly Criterion**: Bet fraction (f^*) of wealth that maximizes expected logarithmic return.
- Maximizing expected log-return ⇔ Maximizing expected logarithm of V_n.
  - Equivalent to maximizing the geometric average of returns.
  - Given a sequence of returns $R_1, R_2, \ldots, R_n$, the **geometric average return (a.k.a. geometric mean)** is: $\text{GeoMean} = \left(\prod_{i=1}^{n}(1+R_i)\right)^{1/n} - 1$.
  - This tells you the equivalent constant rate of return that would result in the same final wealth after n periods.
- **Kelly formula for even-money bets**:

$$f^* = 2p - 1$$

- **General sequence of bets, where: Set ⇒ +a if win and -b if loss**. $P(\text{win}) = p$ and $P(\text{lose}) = q$. The bet is favorable, i.e., $p \cdot a > q \cdot b$. The Kelly criterion optimal fraction to bet is:

$$f^* = \frac{pa - (1-p)b}{ab}$$

  - If you make w wins and $l = n - w$ losses:

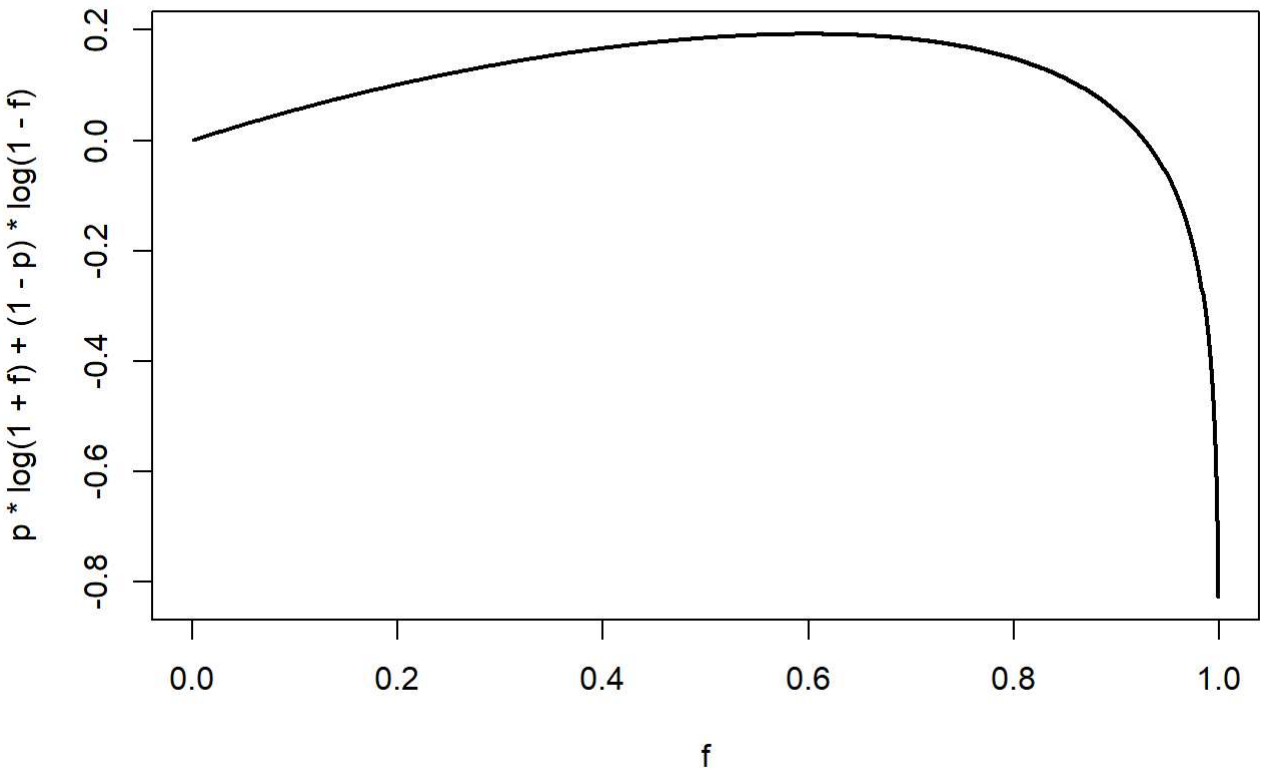
$$\text{Final Wealth} = V_0 \cdot (1+af)^w \cdot (1-bf)^l$$

    The geometric average return per bet is:

$$\text{GeoMean} = \left[(1+af)^w(1-bf)^l\right]^{1/n} - 1$$

This is the expected logarithmic growth rate of wealth, and the Kelly criterion maximizes this.

```
set.seed(1234567890)

p=.8
f=seq(.001,.999,.001)

plot(f, p*log(1+f)+(1-p)*log(1-f), type='l', lwd=2)
```



- Plot shows how expected growth changes with different bet fractions. The **maximum** occurs at the Kelly fraction.
- Plots is the function (n=1):

$$E[\log(1 + f \cdot R)] = p \cdot \log(1 + f) + (1 - p) \cdot \log(1 - f)$$

  Where: You gain `f%` of wealth if you win. You lose `f%` of wealth if you lose.

  - `p = 0.8` : You win a bet with 80% probability.
  - `f` : Range of fractions of wealth to bet, from 0.001 to 0.999.
  - for $p = 0.8$, $f^* = 2p - 1 = 0.6$ (i.e., bet 60% of wealth each time).

1.2.Simulate 500 Bets

- `V0 = 100` : Start with $100. `n = 500` : 500 bets. `p = 0.55` : Win probability is 55%.
- `w.l` : vector of 0s and 1s where 1 = win, 0 = loss.

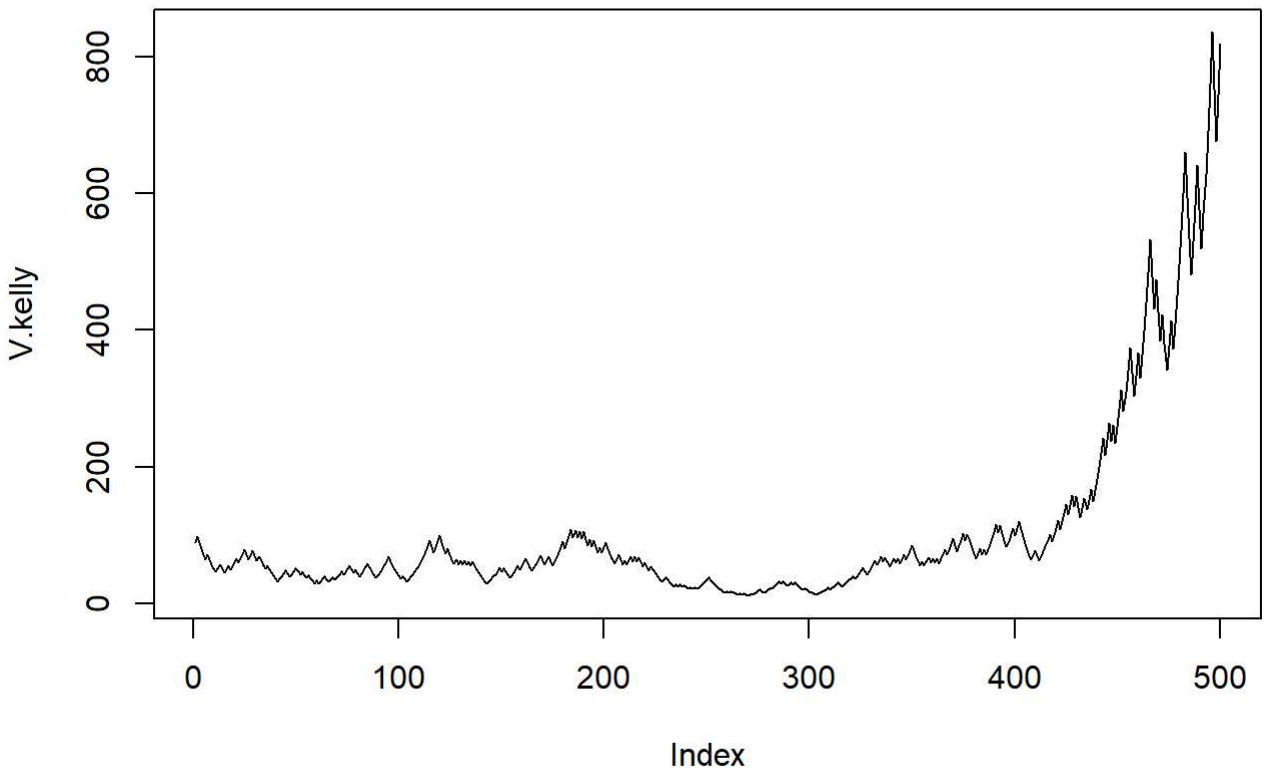
```
V0=100; n=500; p=.55
w.l=rbinom(n,1,p)
```

1.2.1.Kelly Betting Strategy

- Kelly fraction (even-money bets): **f=2p−1**=2(0.55)−1=0.1
- Wealth evolves multiplicatively: Multiply by **(1+f)** for each win. Multiply by **(1-f)** for each loss.

This models **exponential wealth growth** based on outcome history.

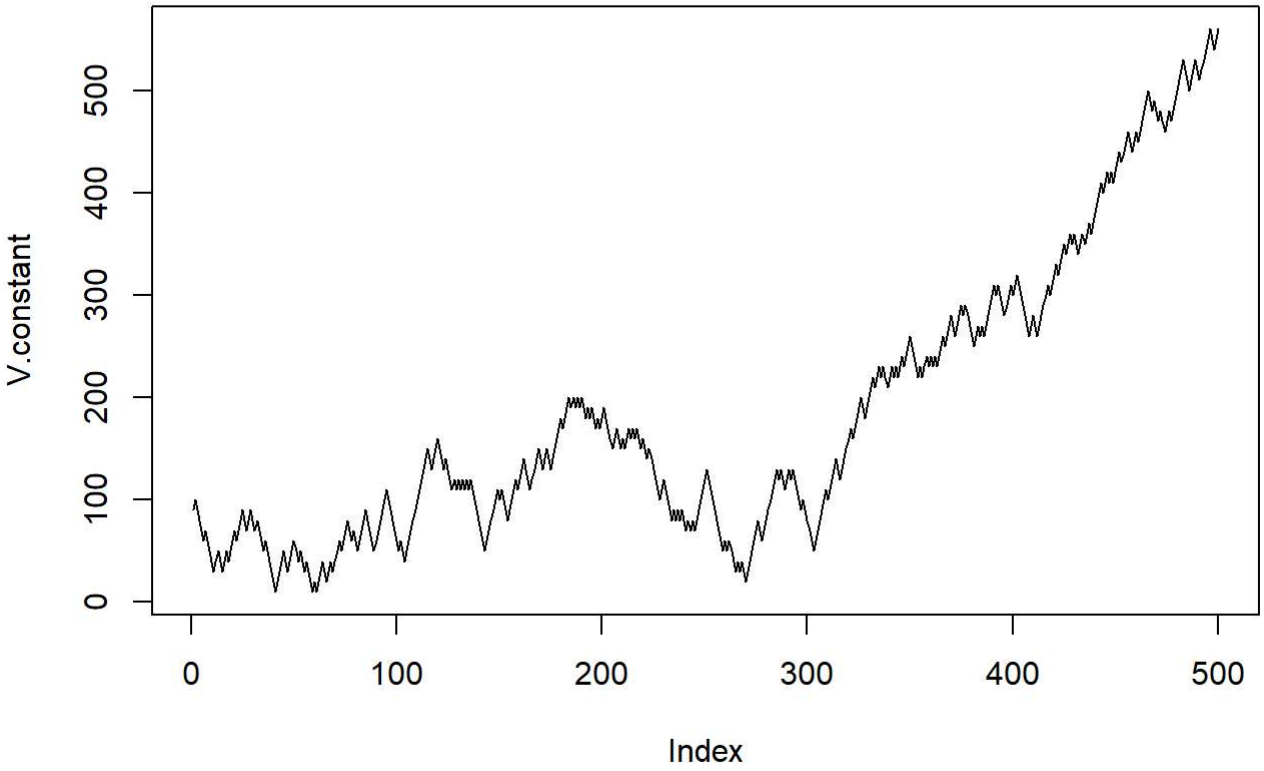
```
f=2*p-1
wins=cumsum(w.l)
losses=(1:n)-wins
V.kelly = V0*(1-f)^losses*(1+f)^wins
plot(V.kelly, type='l')
```



1.2.2.Constant Bet Strategy ($10 per bet)

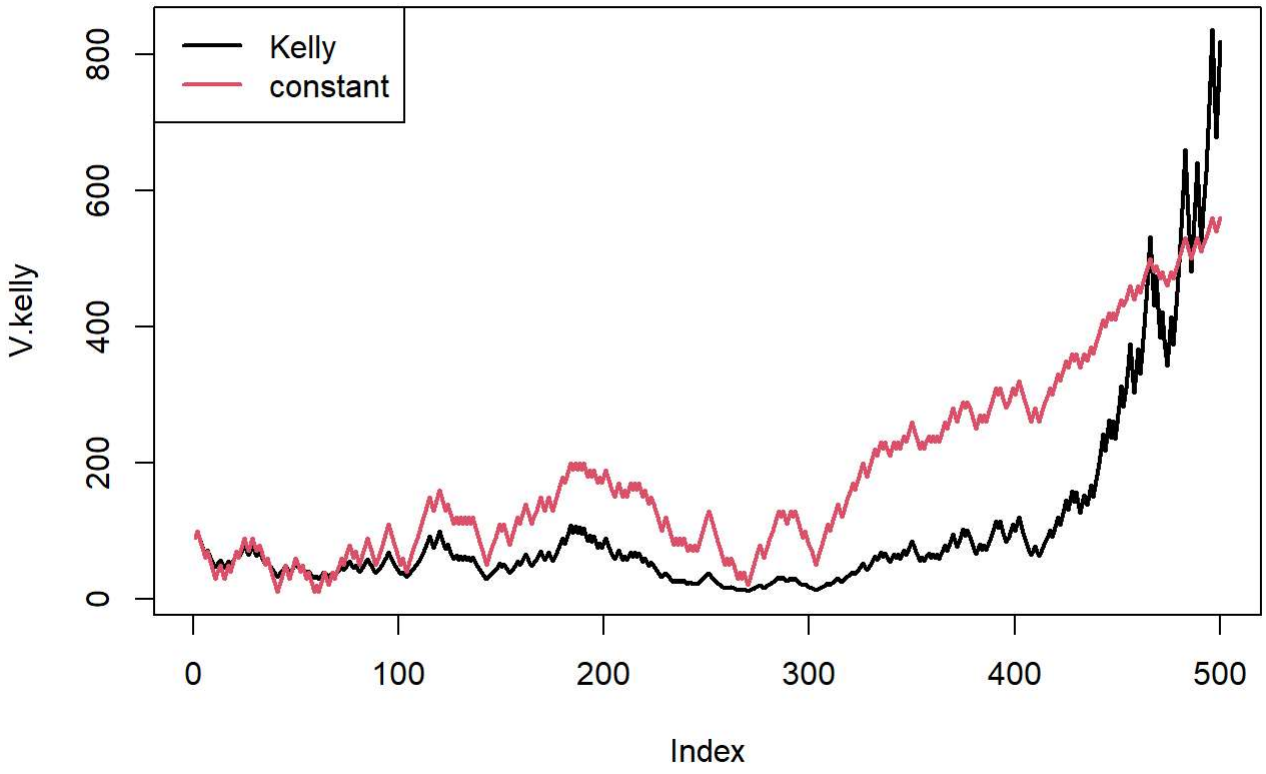
- Win: gain $10 → `+10` ; Loss: lose $10 → `-10` . Track total wealth over time with `cumsum` .

```
Constant_bet_Amount = 10

V.constant=V0+cumsum(Constant_bet_Amount*(2*w.l-1))
plot(V.constant, type='l') #Plot how wealth evolves under constant $10 bets.
```



1.2.3.Comparison Plot

```
plot(V.kelly, type='l', lwd=2, ylim = range(c(V.constant,V.kelly))) # wealth with Kelly be
tting
lines(V.constant, col=2, lwd=2) # wealth with constant $10 bet
legend("topleft", lwd=2, col=1:2, c("Kelly","constant") )
```



2.Kelly criterion in investing

- Consider a situation with:
  - Risk-free asset with return r_f.
  - Risky asset with return R, where

$$E[R] = \mu, \quad Var[R] = \sigma^2$$

- Strategy: Invest fraction f of wealth into the risky asset and the remaining $(1 - f)$ into the risk-free asset.

Kelly Optimization : Find f that maximizes the logarithm of wealth.

- The expected log growth rate is approximated by:

$$E\left[\log\left(\frac{V_n}{V_0}\right)\right] \approx \log(1 + r_f) + f\frac{(\mu - r_f)}{(1 + r_f)} - f^2\frac{\sigma^2 + (\mu - r_f)^2}{2(1 + r_f)^2}$$

- The Kelly-optimal fraction f^* is:

$$f^* \approx (1 + r_f)\frac{(\mu - r_f)}{\sigma^2}$$

  - $\mu - r_f$: Excess return of the risky asset.
  - σ^2: Variance of the risky asset (risk penalty).

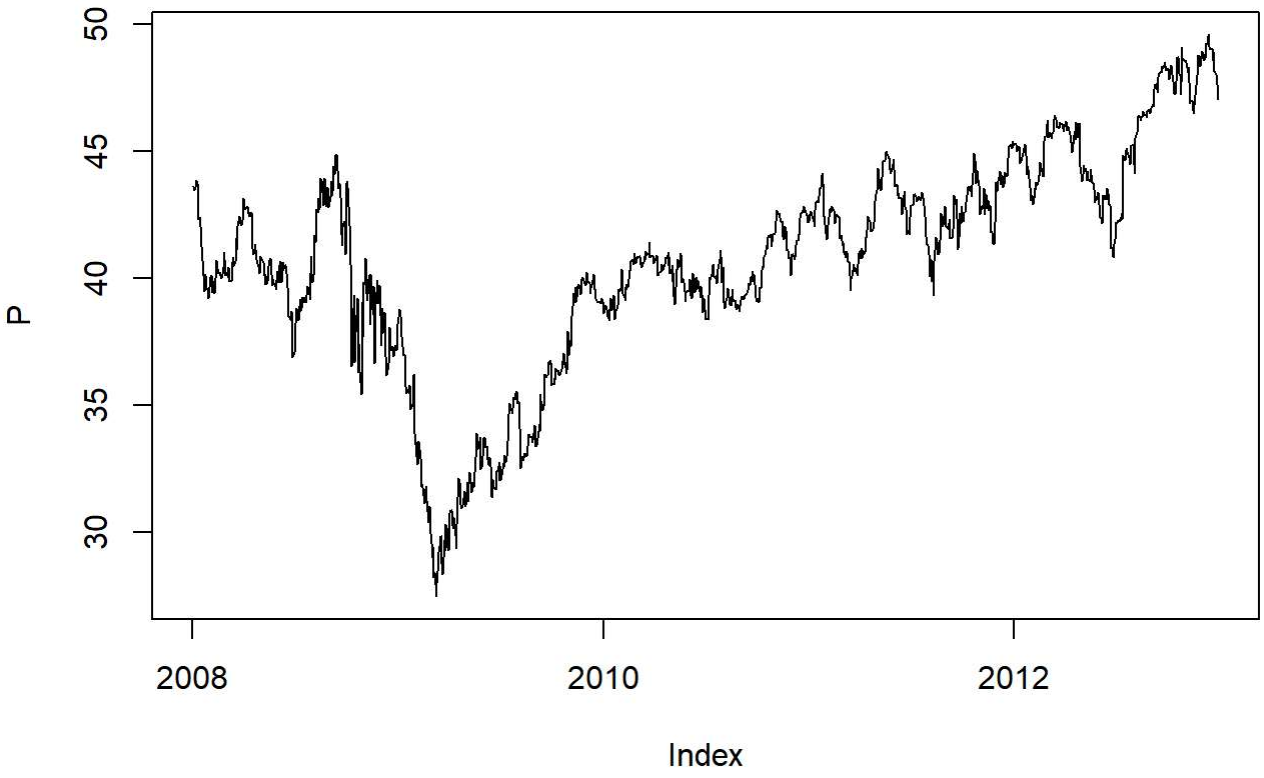
```
library(tseries); library(zoo)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

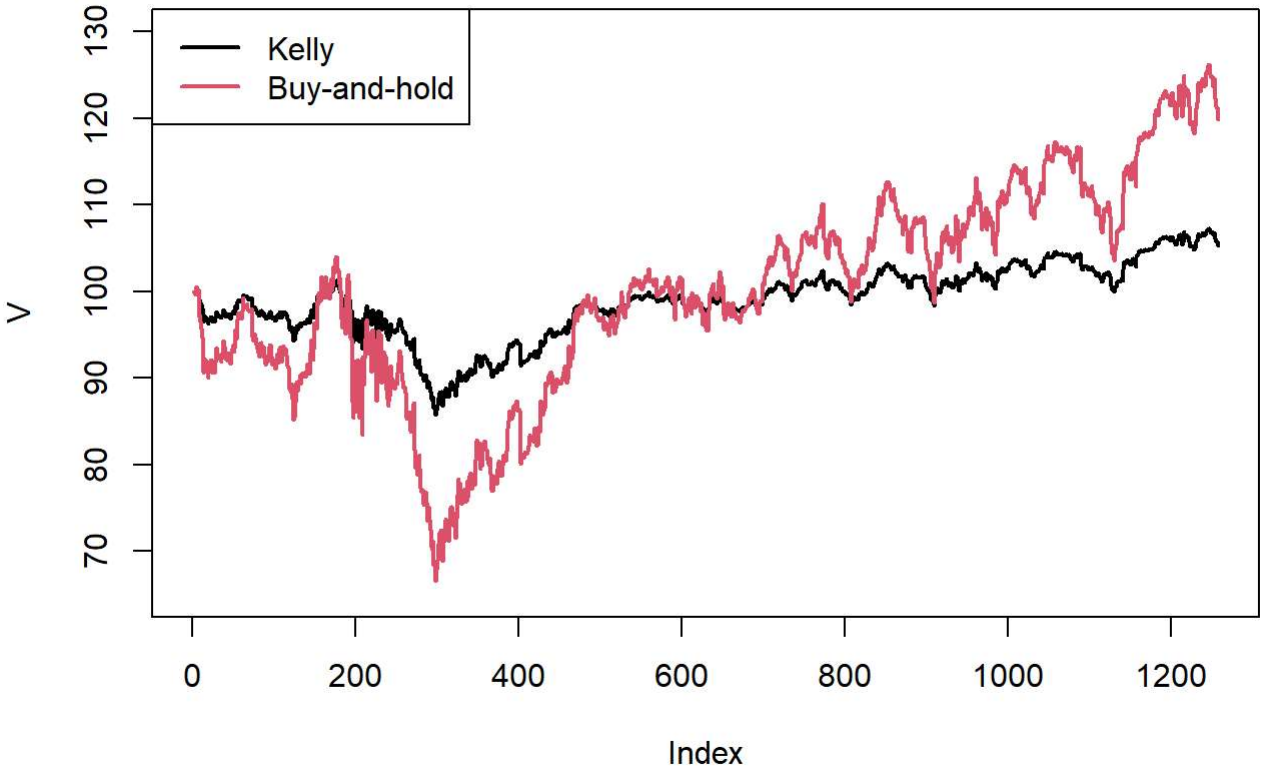
```
### 1.Download Adjusted Close Prices
start.date=as.Date("2008-01-01"); end.date=as.Date("2012-12-31")
P=get.hist.quote(instrument = "PG", start = start.date, end=end.date, quote = "AdjClose",
retclass = "zoo", quiet = T)
plot(P) # Plot Adjusted Close Prices
```



```
### 2.Calculate Log Returns & Kelly Criterion
R=diff(log(P))
mu=mean(R)
s2=var(R)
(f=mu/s2) # optimal Kelly fraction
```

```
##           Adjusted
## Adjusted 0.3603411
```

```
### 3.Simulate Investment Growth Using Kelly Fraction
V=100; # Start with $100 capital.
Prices=coredata(P)
NetReturns=exp( diff(log(Prices)))-1 # Actual returns, not log returns.
for(i in 2:length(P)){
  # Simulates portfolio value by Kelly strategy, reinvest f fraction each day.
  V[i]=V[i-1]* (1+f*NetReturns[i-1])
}
### 3.1.Equivalently (but faster)
# V=100*cumprod(1+f*c(0,NetReturns))


### 4.Plot Kelly vs. Buy-and-Hold Strategy
plot(V,type='l',lwd=2, ylim = c(65,130)) # wealth with Kelly investing
lines(V[1]*exp(cumsum(c(0,NetReturns))), col=2,lwd=2) # Investing everything at the start
and holding throughout.
legend("topleft", lwd=2, col=1:2, c("Kelly","Buy-and-hold") )
```



3.Static and Dynamic Kelly

The “Efficient Markets Hypothesis” presumes that returns are unpredictable. For this session, however, you will assume that returns are predictable and they actually follow an AR(1) process

$$(R_t - \mu) = \varphi(R_{t-1} - \mu) + \varepsilon_t, \quad \varepsilon_t \sim^{iid} N(0, \sigma^2)$$

The model dictates that the **conditional mean of the next return is a linear function of the current return**, i.e.

$$\mathbb{E}[R_t - \mu | R_{t-1}] = \varphi(R_{t-1} - \mu)$$

Based on this observation, one can modify the Kelly criterion for investing that we saw in class to allow for dynamically changing fractions, based on the conditional mean & variance of returns at each point. Remember that the Kelly-optimal fraction for i.i.d. returns with mean μ, variance σ^2 and constant risk-free rate r_f:

$$f^* \approx (1 + r_f)\frac{\mu - r_f}{\sigma^2}$$

In the **dynamic setting**, substitute μ and σ^2 with the **conditional mean and variance of the return**

For AR(1) returns, replace μ and σ^2 with **conditional moments**:

$$f^*_{t-1} \approx (1+r_f)\frac{\mathbb{E}[R_t|R_{t-1}] - r_f}{\mathbb{V}[R_t|R_{t-1}]}$$

- here: $\mathbb{E}[R_t|R_{t-1}]$: Conditional mean (uses AR(1) dynamics). $\mathbb{V}[R_t|R_{t-1}]$: Conditional variance.

3.1.Simulating the Return Paths

Simulate $n = 1,000$ paths of $m = 252$ daily returns each, from an AR(1) model with:

- $R_0 = 0$. $\varphi = 0.25$ (mean-reversion strength).
- $\mu = 0.05/252$. $\sigma_\varepsilon = 0.4/\sqrt{252}$. $r_f = 0.02/252$

The simulated paths are created by:

```
n=1000; m=252
phi=.25 #speed of mean reversion
mu=.05/252 #average daily return
sigma=.4/sqrt(252)
rf=.02/252
V0=1 #initial wealth
R_0 = 0 #initial return


# --- Generate returns ---#
set.seed(1234567890)
R=matrix(0,m,n) #return matrix: rows = time, cols = simulations
Z=matrix(rnorm(n*m),m,n) # Zt~N(0,1): i.i.d.standard normal noise

### 1.Initialize returns at stationary distribution
##### 1.1.Initialize first return using R0
R[1,]=mu+(R_0-mu)*phi+sigma*Z[1,]

##### 1.2.Simulate returns using AR(1) mean-reversion model
for(i in 2:m){
  R[i,]=mu+(R[i-1,]-mu)*phi+sigma*Z[i,]
}
# To avoid the loop (& speed up computation) you can use the filter() function:
# R = apply( mu*(1-phi) + sigma*Z, 2, filter, filter=phi, method="recursive")
```

The result is `R` : a 252×1000 matrix of simulated return paths.

```
dim(R)
```

```
## [1]  252 1000
```

3.2.Static Kelly Strategy

Apply the static Kelly criterion based on: Stationary mean: $\mathbb{E}[R_t] = \mu$ and Stationary variance: $\mathbb{V}[R_t] = \frac{\sigma_\varepsilon^2}{(1-\varphi^2)}$ of AR(1) process, and Calculate the resulting wealth process for each path.
Plot the wealth of the strategy on the first path, and the histogram of the final log-wealths across all paths

- Formula:

$$f^* = \frac{(1+r_f)(\mu - r_f)}{\sigma_b^2}$$

  - σ_b: long-run volatility, accounting for AR(1). Fixed f^* applied to each day
- Wealth path:

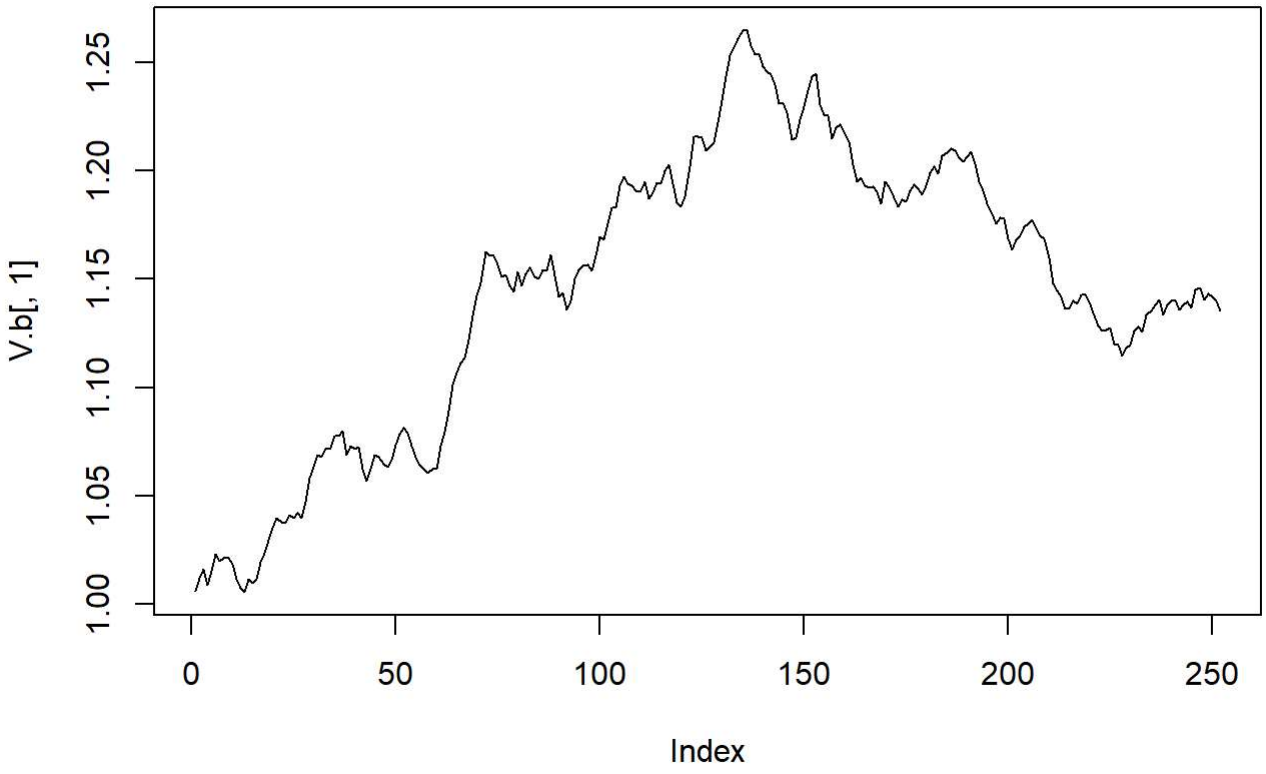
$$V_t = V_{t-1} \times (1 + r_f + f^*(R_t - r_f))$$

This strategy does not react to short-term trends, just uses long-run expectations.

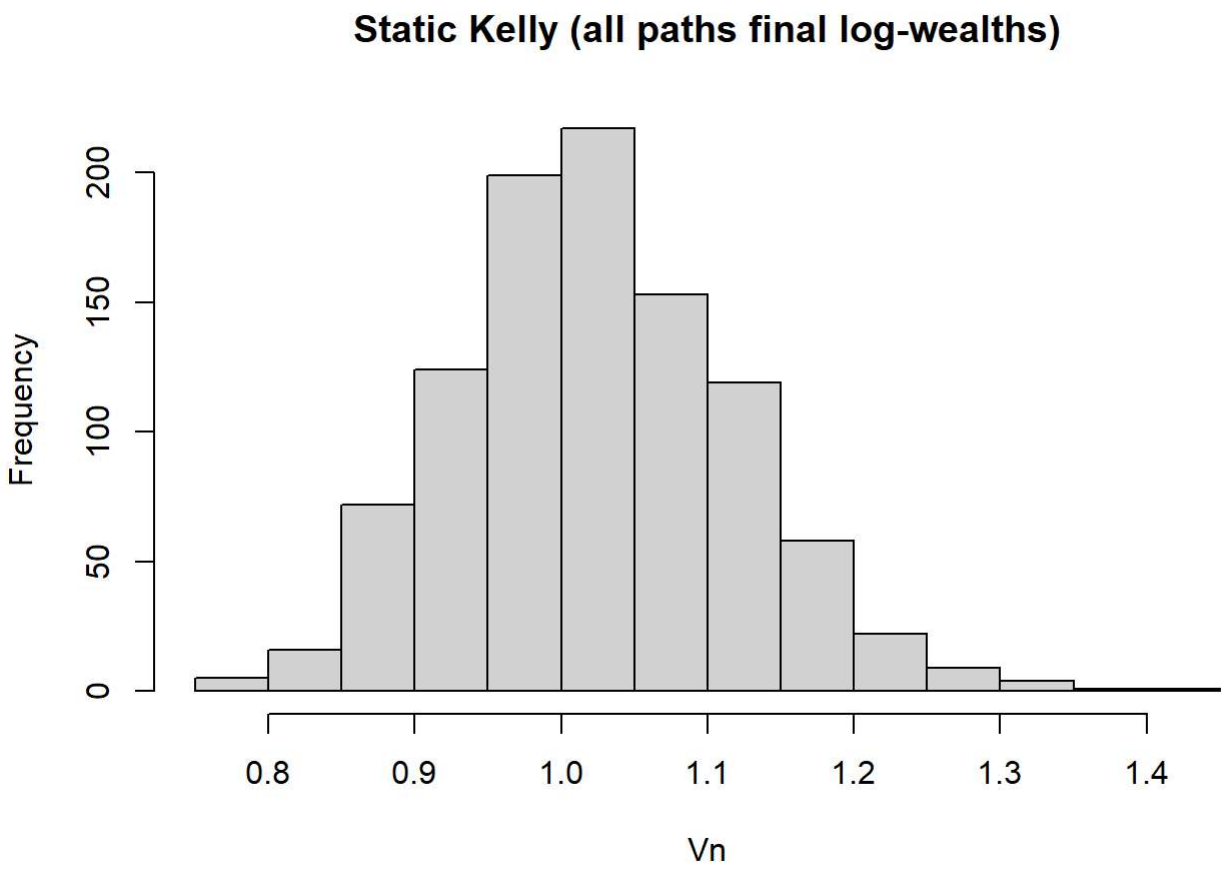
```
### 1.long-run volatility
sigma.b=sigma/sqrt(1-phi^2)

### 2.Static Kelly fraction
f.b=(1+rf)*(mu-rf)/(sigma.b^2)
print(paste("Static Kelly fraction f =",f.b))
```

```
## [1] "Static Kelly fraction f = 0.175795200892857"
```

```
### 3.Wealth process under static Kelly for each path
V.b=matrix(0,m,n)
V.b[1,]=V0*(1+rf+f.b*(R[1,]-rf))
for(i in 2:m){
  V.b[i,]=V.b[i-1,]*(1+rf+f.b*(R[i,]-rf))
}
# To avoid the loop use
# V.b=apply( 1+rf+f.b*(R-rf), 2, cumprod)

### 4.Plot the wealth of the strategy on the first path
plot(V.b[,1], type='l')
```




```
### 5.histogram of final log-wealths across all paths
hist( V.b[m,], main = "Static Kelly (all paths final log-wealths)", xlab="Vn" )
```



Using the “static” Kelly criterion, we do not really take advantage of the predictability of the process, just of the fact that the asset returns are on average better than the risk-free rate.

The average final wealth over 1 year is a modest 1.0246512.

```
mean( V.b[m,] )
```

```
## [1] 1.024651
```

3.3.Dynamic Kelly Strategy

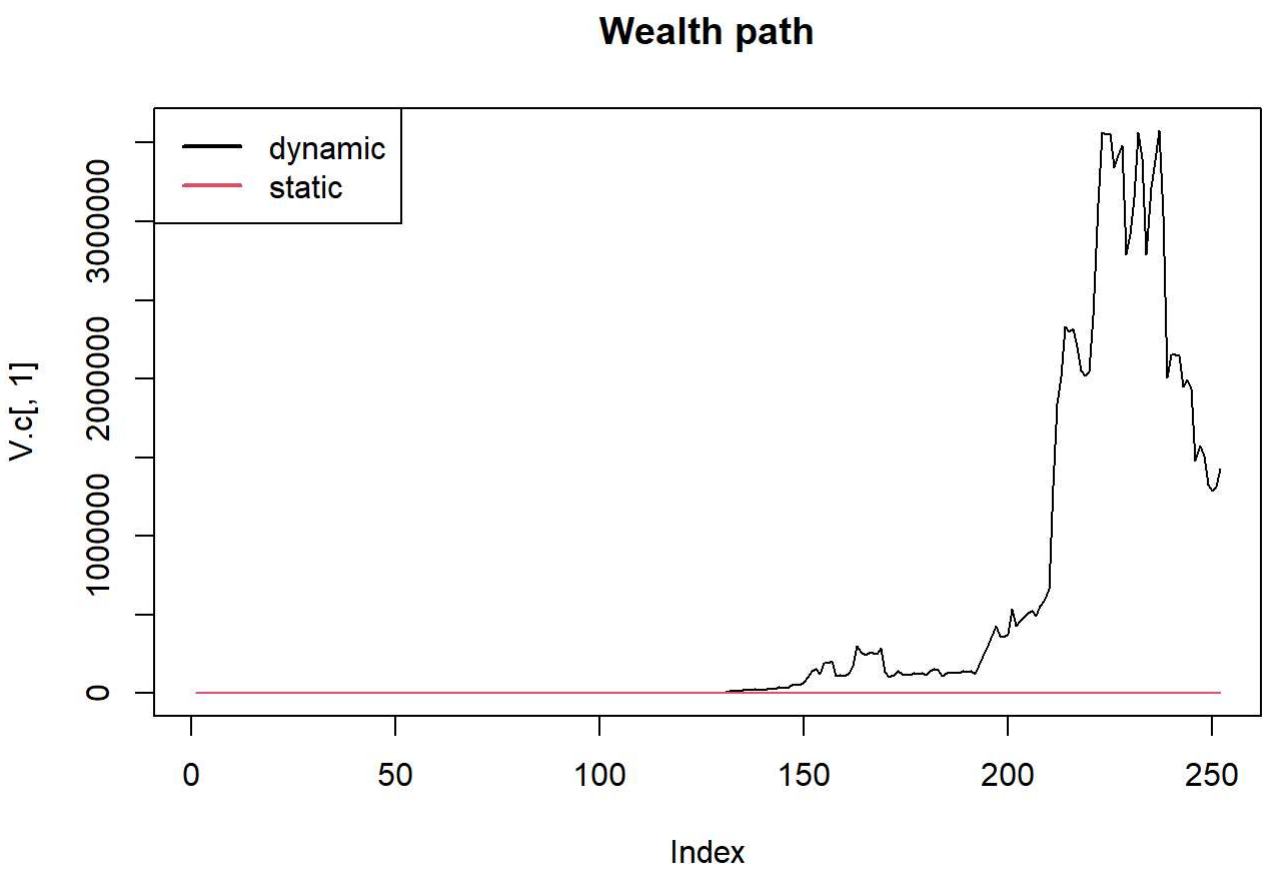
Apply the dynamic Kelly criterion and calculate the resulting wealth process for each path.

Plot the first path and the histogram of the final log-wealths across all paths, and compare them to those of the previous part.

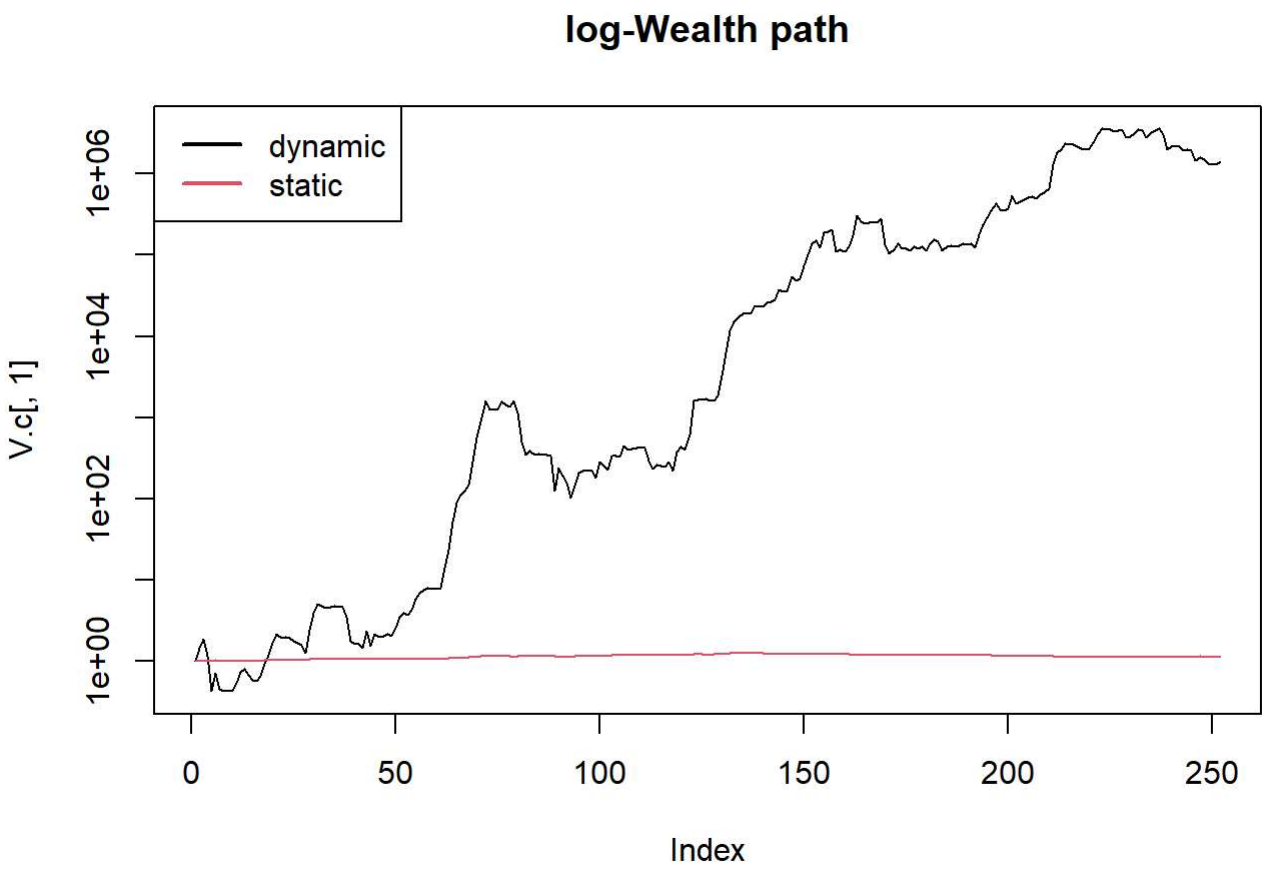
```
n=1000; m=252; phi=.25 #speed of mean reversion
mu=.05/252; sigma=.4/sqrt(252); rf=.02/252; V0=1 #initial wealth
R_0 = 0 #initial return

### 1.Initialize welath matrix
V.c=matrix(1,m,n)

### 2.Dynamic Kelly fraction
##### 2.1.First Time Step
mu_i=(R_0-mu)*phi
(f.c_i=(1+rf)*mu_i/(sigma^2)) # if f<0 -> short asset
```

```
## [1] -0.0781312
```

```
V.c[1,]=V0*(1+rf+f.c_i*(R[1,]-rf))

##### 2.2.Iterative Update
for(i in 2:m){
  # Dynamic Kelly fraction
  mu_i=(R[i-1,]-mu)*phi
  f.c_i=(1+rf)*mu_i/(sigma^2) # changing fraction
  V.c[i,]=V.c[i-1,]*(1+rf+f.c_i*(R[i,]-rf))
}

####### 2.2.b.Alternative, avoid loop by: Vectorized Dynamic Kelly ######
# mu_i=(rbind(rep(R_0,n),R[-m,])-mu)*phi # Time-vary expected return (mu_i)
# f.c_i=(1+rf)*mu_i/(sigma^2+mu_i^2)  # Dynamic Kelly fraction f.c_i
# V.c=V0*apply( 1+rf+f.c_i*(R-rf), 2, cumprod) # Wealth paths by cumulative product

### 3.Plot wealth of 1st path
plot(V.c[,1], type='l', main = "Wealth path")
lines(V.b[,1], type='l', col=2)
legend( "topleft", lwd = 2, col = 1:2, c("dynamic","static"))
```




```
#### 3.1.Plot log-wealth, for better comparison.
plot(V.c[,1], type='l', log = "y", main = "log-Wealth path")
lines(V.b[,1], type='l', col=2)
legend( "topleft", lwd = 2, col = 1:2, c("dynamic","static"))
```



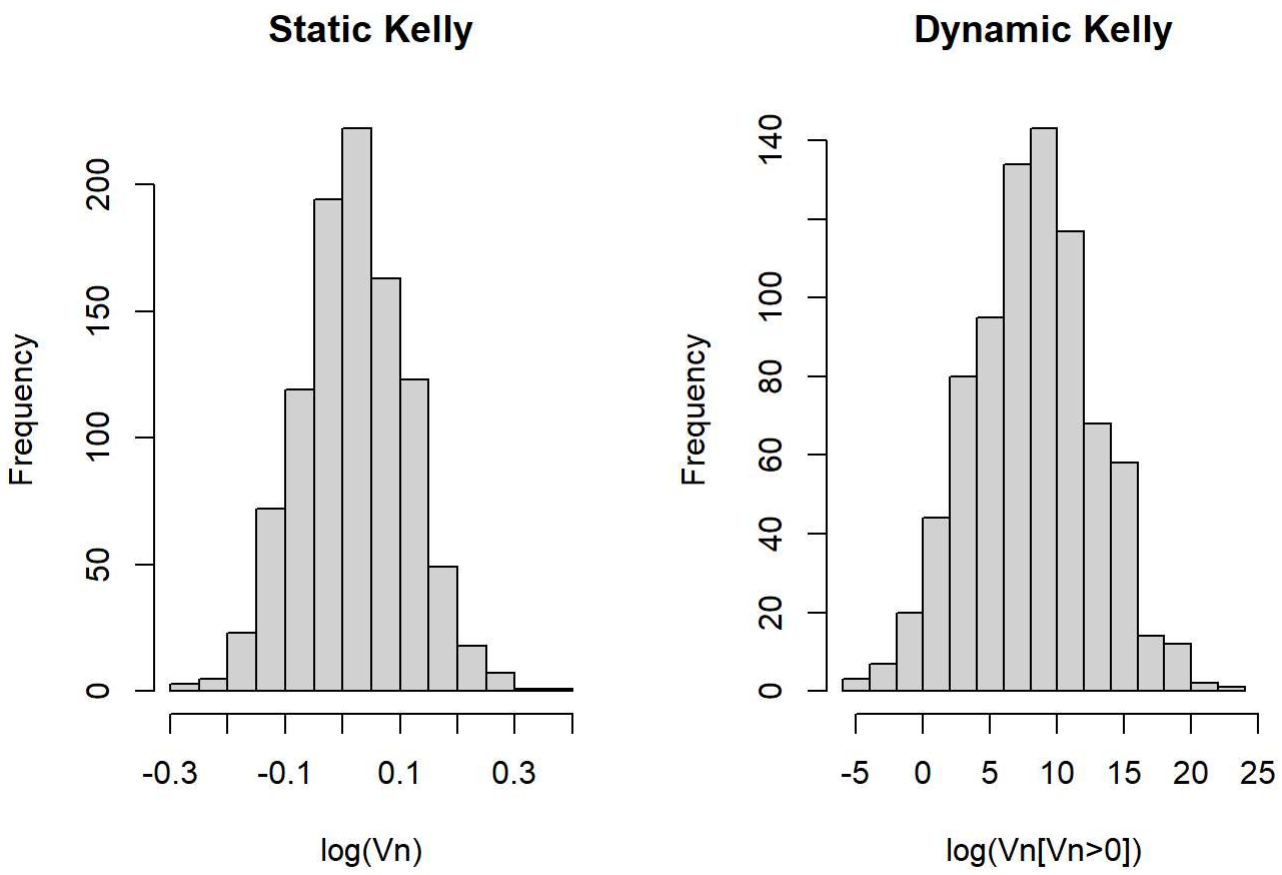
- The dynamic Kelly wealth is growing much faster than the static one. That's because it can take advantage of the predictability of the price on each day, and even when the price goes down, by using a "negative" fraction (i.e. shorting the asset).
- However, **it is possible for the dynamic Kelly wealth to become *negative* if the optimal fraction is >1** (i.e., you use leverage (https://en.wikipedia.org/wiki/Leverage_(finance))).

```
mean( apply( V.c<=0, 2, any) )*100 # 2 means across columns.
```

```
## [1] 22.4
```

- In fact, 22.4% of the dynamic Kelly paths had negative wealth (i.e. ruin/bankruptcy) at some point.
- But for those paths that did not go negative, the dynamic strategy led to much higher terminal wealth.

```
### 4.histogram of the final log-wealths across all paths & Comparison
par(mfrow=c(1,2))
hist(log(V.b[m,]), main = "Static Kelly", xlab = "log(Vn)" )
hist(log(V.c[m, V.c[m,]>0]), main = "Dynamic Kelly", xlab = "log(Vn[Vn>0])" )
```



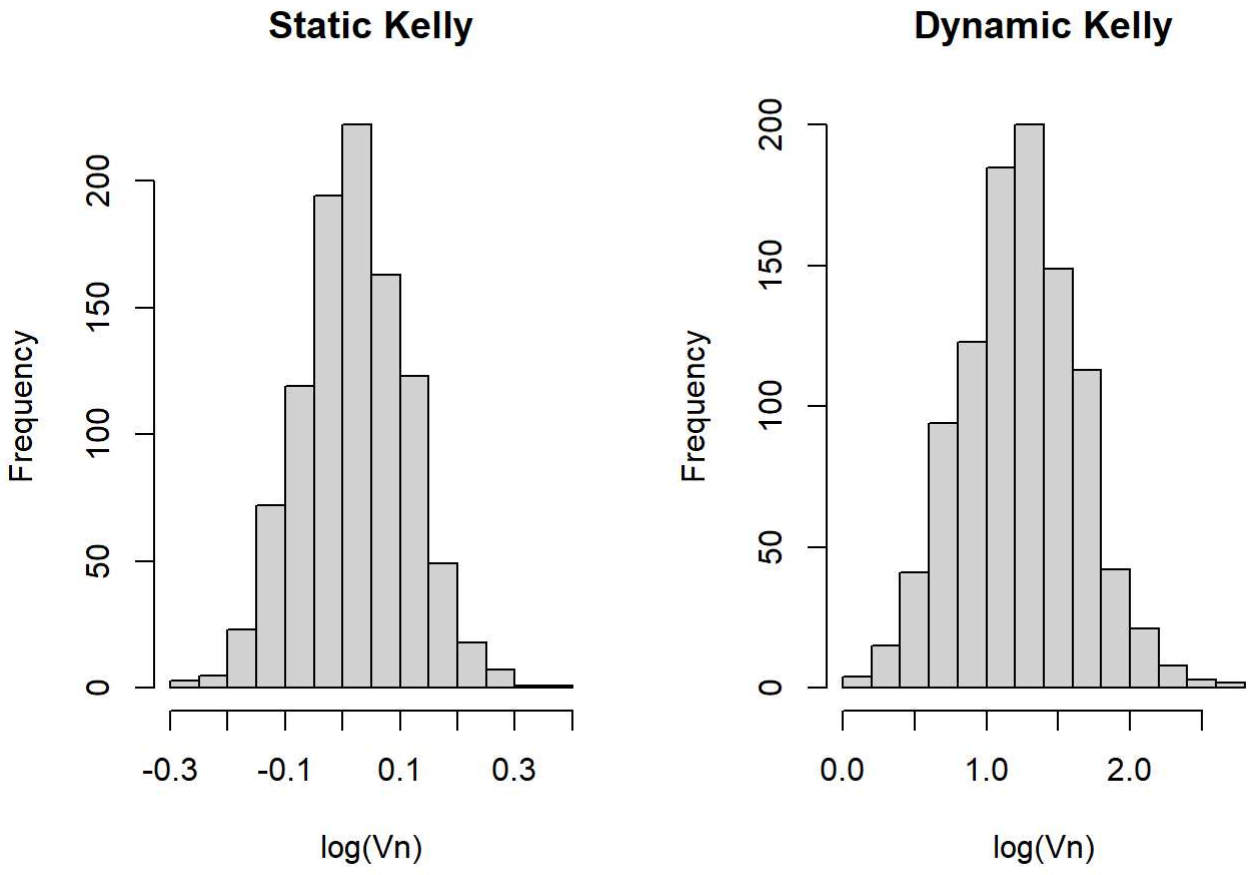
3.4.Constrained Dynamic Kelly

One can moderate and improve the dynamic strategy by **constraining the invested fraction to be always** $|f^*| < 1$ **(i.e. avoid leverage).** This technically removes the risk of ruin while still giving superior results:

- `f.c_i= pmax( pmin( (1+rf)*mu_i/(sigma^2), 1) , -1)` : Applies a constraint so that the investment fraction stays between -1 and 1. The constrained dynamic strategy still outperforms the static one on average but with less risk.
    - `min(max(...))` ensures that `f.c_i` lies within this range. This prevents allocation to risky assets from exceeding these bounds.
        - `f.c_i` is set to `-1` if the fraction goes below -1 (shorting more than the total wealth).
        - It is capped at `+1` if it exceeds 1 (no more than the total wealth invested in risky assets).

```
n=1000; m=252; phi=.25 #speed of mean reversion
mu=.05/252; sigma=.4/sqrt(252); rf=.02/252; V0=1 #initial wealth
R_0 = 0 #initial return
```

```
### 1.Initialize
V.c.constr=matrix(1,m,n)

### 2.First Time Step (Initial Wealth Update)
mu_i=(R_0-mu)*phi
f.c_i= min( max( (1+rf)*mu_i/(sigma^2), -1 ), 1)
V.c[1,]=V0*(1+rf+f.c_i*(R[1,]-rf))

### 3.Subsequent Time Steps (Iterative Wealth Update)
for(i in 2:m){
  # Dynamic Kelly fraction
  mu_i=(R[i-1,]-mu)*phi
  f.c_i= pmax( pmin( (1+rf)*mu_i/(sigma^2), 1) , -1)
  V.c.constr[i,]=V.c.constr[i-1,]*(1+rf+f.c_i*(R[i,]-rf))
}

### 4.Comparison of Static vs. Constrained Dynamic Kelly
par(mfrow=c(1,2))
hist(log(V.b[m,]), main = "Static Kelly", xlab = "log(Vn)" )
hist(log(V.c.constr[m,]), main = "Dynamic Kelly", xlab = "log(Vn)" )
```



3.4.1.General Constraints (a<f<b)

One can moderate and improve the dynamic strategy by constraining the invested fraction to be always $a < f^* < b$ (i.e. avoid leverage).

```
# Parameters a and b for the new constraints
a = -0.5  # lower bound of the investment fraction
b = 0.7    # upper bound of the investment fraction

# Initialize the wealth matrix for the constrained strategy
V.c.constr = matrix(1, m, n)

# First time step
mu_i = (R_0 - mu) * phi
f.c_i = pmin( pmax( (1 + rf) * mu_i / (sigma^2), a), b)  # Constrained between a and b
V.c.constr[1, ] = V0 * (1 + rf + f.c_i * (R[1, ] - rf))

# Loop over subsequent time steps
for(i in 2:m){
  mu_i = (R[i-1, ] - mu) * phi
  f.c_i = pmin( pmax( (1 + rf) * mu_i / (sigma^2), a), b)  # Constrained between a and b
  V.c.constr[i, ] = V.c.constr[i-1, ] * (1 + rf + f.c_i * (R[i, ] - rf))
}

# Plotting the results
par(mfrow = c(1, 2))
hist(log(V.b[m, ]), main = "Static Kelly", xlab = "log(Vn)")
hist(log(V.c.constr[m, ]), main = "Constrained Dynamic Kelly", xlab = "log(Vn)")
```