CHAPTER 7

RISK MANAGEMENT BY STRATEGY

TRADING STRATEGIES

FROM BEGINNER TO PROFESSIONAL

versopropfirm.com

Risk Management by Strategy

Risk management is not just an additional tool; it is the very heart that keeps a trader alive. As the old Wall Street adage says: "Novice traders worry about being right, professional traders worry about how much they can lose."

The Three Pillars of Risk Management

1. Risk Per Trade (Risk per Operation)
2. Portfolio Risk (Portfolio Risk)
3. Systemic Risk (Systemic Risk)

Risk Management for Intraday Trading Unique

Characteristics of Intraday Risk:

• Extreme Speed: Losses can accumulate in minutes

• High Leverage: Typically 4:1 to 10:1

• Temporal Concentration: All risk in market hours

• Variable Liquidity: Spreads can expand substantially

Intraday Risk Management Framework:

1. 1% Drawdown Rule:

```
classPortfolioRiskBudget:
def__init__(self, total_capital):
self.total_capital = total_capital
self.risk_buckets={
'growth_stocks':{'allocation':0.40,'max_vol':0.25},
'value_stocks':{'allocation':0.30,'max_vol':0.20},
'defensive_stocks':{'allocation':0.20,'max_vol':0.15},
'alternatives':{'allocation':0.10,'max_vol':0.30}
}
defallocate_capital(self):
allocations ={}
for bucket, params in self.risk_buckets.items():
capital_allocation = self.total_
capital * params['allocation']
max_portfolio_contribution = params['max_vol']* params['allocation']
allocations[bucket]={'capital': capital_allocation,'max_vol_contribution':
max_portfolio_contribution}
return allocations
defmonitor_risk_budget(self, actual_volatilities):
"""
Monitor whether each bucket is within its risk budget
 """
 violations =[]
for bucket, actual_vol in actual_volatilities.items():
max_allowed = self.risk_buckets[bucket]['max_vol']
```

2. **Dynamic Stop Losses:**

ATR-Based Stops: 1.5 x ATR(14)

• Stop Distance: =1.5 x ATR(14) from the market

• Advantage: Adapts to market volatility

• Disadvantage: Can generate premature stops in volatile markets

Support/Resistance Stops:

• Stop =2-3 pips beyond key level

• Advantage: Based on market structure

• Disadvantage: Does not consider current volatility

Time-Based Stops:

• Auto-close after 2 hours without progress

python

```python
def calculate_intraday_position_size(account_balance, entry_price, stop_loss, risk_per_trade=0.01):
    risk_amount = account_balance * risk_per_trade
    position_size = risk_amount / (entry_price - stop_loss)
    # ATR 14 calculation
    atr = calculate_atr(14)  # Average True Range
    volatility_multiplier = atr * 1.4  # Historical average ATR
    adjusted_position_size = position_size / max(volatility_multiplier, 1.0)
    return adjusted_position_size
```

3. Maximum Intraday Drawdown:

python

```python
# Python code for max intraday drawdown
```

Advantage: Avoids "dead" operations prematurely • Disadvantage: Can cut winners prematurely

Specific Risk Management by Intraday Strategy: Breakout Trading:

python

```python
class IntradayRiskManager:
    def __init__(self, daily_loss=0.03, max_consecutive_losses=3):
        self.max_daily_loss = daily_loss
        self.max_consecutive_losses = max_consecutive_losses
        self.consecutive_losses = 0
        self.trading_allowed = True

    def update_pnl(self, trade_result):
        self.daily_pnl += trade_result
        if trade_result < 0:
            self.consecutive_losses += 1
        else:
```

```
            self.consecutive_losses = 0

        # Circuit breakers
        if abs(self.daily_pnl) > self.max_daily_loss:
            print("Daily loss limit reached. Trading suspended.")
            self.trading_allowed = False

        if self.consecutive_losses >= self.max_consecutive_losses:
            self.trading_allowed = False
            print("Too many consecutive losses. Take a break!")

    def can_trade(self):
        return self.trading_allowed
```

Breakout Trading:

• Risk: 2x the breakout range high volatility

• Position Size: Reduce 50% on high volatility days •

Time Stop: 30 minutes without progress

Mean Reversion:

• Risk: Distance to moving average +1 ATR

• Scaling: Initial position 50%, add 25% if improves

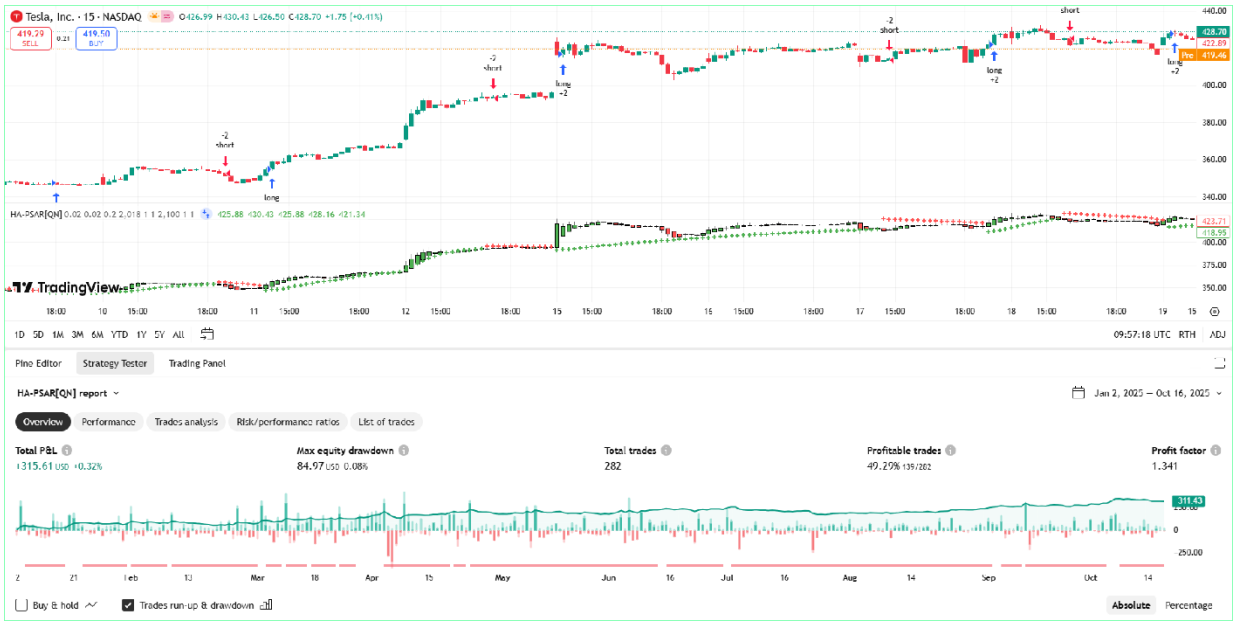
• Max Trades: 5 per day on same instrument

News Trading:

• Risk: 3x the normal spread of the instrument

• Position Size: 25% of normal size

• Time Limit: Close in 15 minutes regardless



Risk Management for Swing Trading

Swing Risk Characteristics:

• Overnight Risk: Exposure to gaps and news

• Correlation Risk: Multiple correlated positions

• Time Decay: Holding losing positions too long

• Opportunity Cost: Capital immobilized in slow trades

Advanced Risk Management Framework:

1. Portfolio Correlation:

python

```
import numpy as np
from scipy import stats

def calculate_portfolio_risk(positions, correlation_matrix, individual_volatilities):
    """Calculate total portfolio risk considering correlations"""
    weights = np.array([pos['weight'] for pos in positions])
    volatilities = np.array([individual_volatilities[pos['symbol']] for pos in positions])
    # Portfolio variance = w^T * Σ * w
    portfolio_variance = np.dot(weights, np.dot(correlation_matrix, weights))
```

```
    portfolio_volatility = np.sqrt(portfolio_variance)
    # Diversification ratio
    weighted_avg_vol = np.sum(weights * volatilities)
    diversification_ratio = weighted_avg_vol / portfolio_volatility
    return {
        'portfolio_volatility': portfolio_volatility,
        'diversification_ratio': diversification_ratio,
        'concentration_risk': max(weights)  # Largest single position
    }

def rebalance_for_risk(current_positions, target_portfolio_vol=0.15):
    """Rebalance positions to maintain volatility objective"""
    current_risk = calculate_portfolio_risk(current_positions)
    if current_risk['portfolio_volatility'] > target_portfolio_vol:
        # Reduce positions proportionally
        scale_factor = target_portfolio_vol / current_risk['portfolio_volatility']
        for position in current_positions:
            position['shares'] *= scale_factor
        return current_positions
```

2. Advanced Trailing Stops:

Chandelier Exit:

python

```
def chandelier_exit(prices, atr_period=22, multiplier=3.0):
    """Stop loss that follows the trend using ATR"""
    highs = prices['High'].rolling(window=atr_period, min_periods=1).max()
    lows = prices['Low'].rolling(window=atr_period, min_periods=1).min()
    atr = calculate_atr(prices, atr_period)
    long_stop = highs - (multiplier * atr)
    short_stop = lows + (multiplier * atr)
    return long_stop, short_stop

def parabolic_sar_stop(prices, start=0.02, increment=0.02, max_af=0.20):
    """Parabolic SAR for trailing stops"""
    # Implementation of Parabolic SAR
    # Complete Parabolic SAR logic
    sar = [start]
    trend = 1  # 1 for uptrend, -1 for downtrend
    return sar
```

3. Multi-Factor Position Sizing:

```
python
def advanced_position_sizing(symbol, account_balance, confidence_level,
market_regime):
    """Position sizing that considers multiple factors: confidence, market level"""
    base_risk = 0.02  # 2% base risk per trade
    # Confidence factor (based on setup quality)
    confidence_multipliers = {
        'high': 1.5,
        'medium': 1.0,
        'low': 0.5
    }
    [confidence_level]
    # Market regime factor
    regime_multipliers = {
        'trending': 1.2,
        'choppy': 0.8,
        'volatile': 0.6
    } [market_regime]
    # Factor for correlation with existing portfolio
    correlation_penalty = calculate_correlation_penalty(symbol)
    adjusted_risk = base_risk * confidence_multipliers[confidence_level] *
regime_multipliers[market_regime] * correlation_penalty
    return min(adjusted_risk, 0.05)  # Cap at 5%
```

Risk Management for Position Trading

Long-Term Risk Characteristics:

• Fundamental Risk: Changes in investment thesis

• Macro Risk: Economic cycles and monetary policies

• Liquidity Risk: Difficulty exiting in crises

• Concentration Risk: Few positions, high individual impact

Long-Term Risk Management Framework:

1. Risk Budgeting:

```
classPortfolioRiskBudget:
def__init__(self, total_capital):
self.total_capital = total_capital
self.risk_buckets ={'growth_stocks':{'allocation':0.40,'max_vol':0.25},
'value_stocks':{'allocation':0.30,'max_vol':0.20},'
defensive_stocks':{'allocation':0.20,'max_vol':0.15},
'alternatives':{'allocation':0.10,'max_vol':0.30}
}
defallocate_capital(self):
allocations ={}
for bucket, params in self.risk_buckets.items():
capital_allocation = self.total_capital * params['allocation']
max_portfolio_contribution = params['max_vol']* params['allocation']
allocations[bucket]={
'capital': capital_allocation,
'max_vol_contribution': max_portfolio_contribution
}
return allocations
defmonitor_risk_budget(self, actual_volatilities):
"""
Monitor whether each bucket is within its risk budget
"""
violations =[]
for bucket, actual_vol in actual_volatilities.items():
max_allowed = self.risk_buckets[bucket]['max_vol']

if actual_vol > max_allowed:
violations.append({
'bucket': bucket,
'actual': actual_vol,
'limit': max_allowed,
'excess': actual_vol - max_allowed})
return violations
```

2. Stress Testing:

```
        defportfolio_stress_test(portfolio, scenarios):
        """
        Test the portfolio against various stress scenarios
        """
        stress_results ={}
        for scenario_name, scenario in scenarios.items():
        portfolio_impact =0
        for position in portfolio:
        symbol = position['symbol']
        weight = position['weight']

        # Apply scenario-specific shock

        if symbol in scenario['asset_shocks']:
        shock = scenario['asset_shocks'][symbol]
        impact = weight * shock
        portfolio_impact += impact
        stress_results[scenario_name]= portfolio_impact
        return stress_results

        # Define stress scenarios
        stress_scenarios ={
        'market_crash':{
        'description':'2008-style market crash',
        'asset_shocks':{
        'SPY':-0.40,'QQQ':-0.45,'IWM':-0.50,
        'VTI':-0.38,'GOOGL':-0.35,'AAPL':-0.30
        }
        },

'inflation_spike':{'description':'CPI jumps to
8%+','asset_shocks':{'XLE':0.15,'GLD':0.08,'TIP':0.05,'XRT':-0.12,'XLK':-0.15,'BOND':-
0.10

                }

        }

}
```

3. Dynamic Hedging:

```
defdynamic_portfolio_hedge
(portfolio, market_conditions):
"""
Adjust hedges based on market conditions
"""
portfolio_beta = calculate_portfolio_beta(portfolio)
portfolio_value =sum(pos['value']for pos in portfolio)
hedge_recommendations =[]

# General market hedge
if market_conditions['vix']>25:# High volatility
spy_hedge_ratio =min
(portfolio_beta *0.5,0.3)# Hedge up to 30% hedge_recommendations.append({
'instrument':'SPY_PUT',
'notional': portfolio_value * spy_hedge_ratio,
'reason':'High VIX protection'
})
# Interest rate hedge
if portfolio['duration_risk']>5.0:
tlt_hedge_ratio =min(portfolio['duration_risk']*0.1,0.15)
hedge_recommendations.append({
'instrument':'TLT_PUT',
'notional': portfolio_value * tlt_hedge_ratio,
'reason':'Duration risk protection'
})
# Sector hedge sector_concentrations =
calculate_sector_concentration(portfolio)
for sector, concentration in sector_concentrations.items():
if concentration >0.25:# More than 25% in single sector
hedge_recommendations.append({
'instrument':f'{sector}_PUT',
'notional': portfolio_value *(concentration -0.20),
'reason':f'Sector concentration in {sector}'
})

return hedge_recommendations
```

versopropfirm.com