A股宏观经济因子择时

引言

宏观因素是指影响企业生产经营的经济、政治、社会和自然环境等因素，它们反映了整个经济体系的基本面因素。宏观择时策略本质是市场对宏观层面趋势性信息的反应长期不足，使得基于经济趋势判断的择时能够显著获取超额收益。 尤其是对于较多采用量价指标的量化投资者，宏观择时指标是一个重要的辅助判断工具，可以有效提升策略的信息广度。

本文从宏观经济的五大角度（经济增长、利率、信贷、通货膨胀、汇率），九个细分指标对A股进行了择时策略的研究。本文主要通过布林带、均线策略等判断是否应该买入沪深300指数或者空仓，分析时间段为2006-01-01至2018-11-30。本文对所选择的九个细分指标分别进行了择时分析后将它们汇总进行了分析。从Calmar比率（年化收益率与最大回撤的比值）来看，大部分单个宏观经济因子的择时效果并不佳，最大回撤普遍偏高且年化收益收益较低。将所有选择的宏观因子汇总后对整体进行分析能得到一个相对漂亮的结论:年化收益和回撤都在15%左右。

	PMI	SHIBOR	期限利差	信用利差	M1与M2同比的剪刀差	社会融资规模	金融机构存款准备金率	即期汇率	通货膨胀	等权重汇总	信贷增强后的汇总
最大回撤	0.245	0.315	0.266	0.326	0.320	NaN	0.399	0.147	0.339	0.156	0.200
年化收益率	0.162	0.004	0.003	0.003	0.127	NaN	0.019	0.003	0.141	0.157	0.176
夏普比率	0.668	-0.753	-0.758	-0.847	0.432	NaN	-0.862	-1.111	0.526	0.743	0.819
Calmar比率	0.663	0.014	0.012	0.009	0.393	NaN	0.004	0.027	0.415	1.00	0.877

宏观因子介绍

1）PMI：采购经理指数

PMI指数的英文全称为Purchasing Managers' Index，中文含义为采购经理指数，是通过对采购经理的月度调查汇总出来的指数，反映了经济的变化趋势。它是一套月度发布的、综合性的经济指标检测体系。

如果PMI大于50，表示经济上升，反之则趋向下降。一般来说，汇总后的制造业综合指数高于50，表示整个制造业经济在增长，低于50表示制造业经济下降。PMI略大于50，说明经济在缓慢前进，PMI略小于50说明经济在慢慢走向衰退。国内通行观点认为，如果PMI指数超过55，经济就有过热的忧虑，如果低于40左右，则有萧条的忧虑。

每个月末会公布当月的PMI指数，可以据此对下个月的投资进行决策。

标的指数：沪深300
交易规则：PMI三月均线上行时买入指数，PMI三月均线下行时卖出指数
择时周期：月频

2）SHIBOR:上海银行间同业拆放利率

上海银行间同业拆放利率（Shanghai Interbank Offered Rate，简称Shibor），从2007年1月4日开始正式运行，是由信用等级较高的银行组成报价团自主报出的人民币同业拆出利率计算确定的算术平均利率。相比反应较慢的长期利率，Shibor:1M反映了中短端利率定价，又不会像隔夜或周度利率噪音较大，是一个合适的利率指标。当资金成本过高时，股票所需资金受到压制，市场资金供给降低，大概率下跌，反之上涨。

标的指数：沪深300
交易规则：当Shibor:1M向下突破自己的布林带下轨时买入指数，当Shibor:1M上穿自己的布林带中轨时卖出指数
择时周期：日频

3）期限利差

利率期限结构是指在某一时点上，不同期限资金的收益率与到期期限之间的关系。利率的期限结构反映了不同期限的资金供求关系。该指标与利率类似，但又不完全相同，期限利差指标反映的是相对利率，而利率指标反映的是绝对利率。

如果国债利率近似代表无风险收益率的话，那么不同期限的国债利率，其实就代表的是相应期限的实际收益率和预期的通胀率。理论上，期限长的国债可以看成是滚动持有多个短期国债，但实际上，持有长债并不完全等价于持有短债的组合，投资者的偏好和债券的供需都会带来影响，因而长债比短债收益率又多隐含了一个期限溢价。长端利率的上行放缓甚至回落反映了对未来经济增速预期的降低，是市场对于一系列经济数据综合判断的结果，自然也就和未来的衰退相关；另外，曲线趋于扁平意味着金融机构进行借短放长的收益会降低，使得银行在信贷上趋紧，抑制经济进一步扩张，从而形成了股市的利空信号。

在这里，我们用**国债10年-国债1个月到期收益率**作为期限利差的计算方法

标的指数：沪深300
交易规则：当期限利差向上突破自己的布林带上轨时买入指数，当期限利差下穿自己的布林带中轨时卖出指数
择时周期：日频

4）信用利差

信用利差是用以向投资者补偿基础资产违约风险的、高于无风险利率的利差。与期限利差类似，信用利差反映的也是相对利率。

信用利差反映了对经济前景的预期，信用利差是总体经济状况的一个函数。信用利差在经济扩张期会下降，而在经济收缩期增加。这是因为在经济收缩期，投资者信心不足，更愿投资于高信用等级债券以回避风险，而公司由于收入下降，现金流减少，为了吸引投资者购买公司债券，发行人必须提供较高的利率，因此会产生较高的信用利差。相反，在经济扩张时期，投资者对未来发展有信心，愿意投资于信用等级较低的证券以获得较高的收益，而公司收入增加，现金流充裕，不需要通过很高的成本来吸引来自外部资金，这样就导致较低的信用利差。鉴于此，可以将信用利差作为预测经济周期活动的一个指标。

在这里，我们用**一个月AAA企业债到期收益率-一个月国债到期收益率**作为期限利差的计算方法。值得一提的是，这种方法因为忽略了收益率曲线的coupon effect，简单来说就是票面和付息结构结构会严重影响收益率曲线。所以可能存在较大的偏误。

标的指数：沪深300
交易规则：当期限利差向上突破自己的布林带上轨时买入指数，当期限利差下穿自己的布林带中轨时卖出指数
择时周期：日频

5）M1与M2同比的剪刀差

M0：流通中现金，即在银行体系以外流通的现金；
M1：狭义货币供应量，即M0 + 企事业单位活期存款；
M2：广义货币供应量，即M1 + 企事业单位定期存款 + 居民储蓄存款。

在这三个层次中，M0与消费变动密切相关，是最活跃的货币；M1反映居民和企业资金松紧变化，是经济周期波动的先行指标，流动性仅次于M0；M2流动性偏弱，但反映的是社会总需求的变化和未来通货膨胀的压力状况，通常所说的货币供应量，主要指M2。

N1主要反映经济中的显示购买力，而M2不仅反映显示购买力，还反映潜在购买力。在一般情况下，M1和M2增速应当保持平衡。如果M1增速过快，则消费和终端市场活跃，就有可能从需求方角度导致商品和劳务市场的价格上涨。如果M2增速较快，则表明储蓄理财情绪较浓。当M1过高M2过低时，说明企业活期存款多，投资意愿强烈，经济扩张较快。此时企业愿意以更高的成本融资，储蓄存款之外的其他类型资产收益较高，人们往往将储蓄存款提出进行其他投资或者购买股票。

M1,M2一般在次月中上旬发布，例如2018年12月11日发布了2018年11月的数据，因此当月的择时需参考上上个月的指标。

标的指数：沪深300
交易规则：当M1-M2的三月均线上行时买入指数；当M1-M2的三月均线下行时卖出指数
择时周期：月频

6）社会融资规模

社会融资规模是指一定时期内（每月、每季或每年）实体经济从金融体系获得的全部资金总额，既包括银行体系的间接融资，又包括资本市场的债券、股票等市场的直接融资。社会融资规模是全面反映金融与经济关系，以及金融对实体经济资金支持的总量指标。社会融资总量与经济增长存在明显的相互作用、相互影响的关系。国内信用类宏观指标，具备明显的非平滑性与季节性。（原因是国内银行通常为了季度信贷考核，将平常季中的信贷匀至季度末发放；使得部分月份的信贷冲量数据不稳定。）因此，在信贷类数据上按三月平均进行处理。

与M1，M2类似，社融数据一般也在次月中上旬发布。

标的指数：沪深300
交易规则：当社会融资规模：当月值的三月均线上行时买入指数；当社会融资规模：当月值的三月均线上行时卖出指数
择时周期：月频

7）金融机构存款准备金率

存款准备金，是指金融机构为保证客户提取存款和资金清算需要而准备的在中央银行的存款，中央银行要求的存款准备金占其存款总额的比例就是存款准备金率。存款准备金制度设置的最初目的是出于风险控制角度考虑，为了确保商业银行在遇到突然大量提取银行存款时，能有相当充足的清偿能力。后来，存款准备金制度还成为国家调节经济的重要手段。如果存款准备金率提高，就意味着吸纳的存款能够放贷出去的越少，间接性的减少了货币的供应量，反之，则意味着能够放贷出去的钱变多，间接增加货币供应量。

存款准备金率的变化与沪深300指数的走势的关系并非确定的。多数情况下，存款准备金率的走势与股市走势相反，例如2008年底存款准备金率下降后股市开启一番上涨，2010年底存款准备金率连番上升后，股市进入了两年多的慢熊市。而2006年底开始的存款准备金率上涨并没有阻碍2007年股市牛市的发生，2018年4月的降准也没有阻止2018年一年的不断下挫。通过单独的存款准备金率变化是难以对股市走势做出判断的，但总的来说降准对A股市场是一个积极的信号。

标的指数：沪深300
交易规则：当大型机构存款准备金率小于半年前时持有指数，否则空仓。
择时周期：日频

8）USDCNH:即期汇率

对于发达经济体而言，汇率与股市在多数情况下呈现出了负相关的关系。一般而言，汇率贬值反而容易刺激股市上涨。这背后的逻辑可能在于，发达市场大多数企业有大量的海外收入，汇率贬值有利于企业盈利改善，此外这些市场大多存在大量的海外投资者参与，汇率贬值意味着外币可以获取更多的本币资产，有利于吸引海外资金继续进入本国市场。

对于多数新兴经济体而言，汇率波动往往与国内资产价格呈现出正相关关系，也就是说汇率贬值对于国内股市而言并不是一件好事。对于海外投资者而言，新兴经济体股市往往是一种风险更高的资产，因此对其投资意愿在一定程度上取决于其国内经济与金融体系的稳健性，而汇率预期也体现出了市场对于经济体稳健性的预期。如果国内外经济条件发生变化，导致汇率贬值，一方面会影响市场风险偏好，另一方面可能引起货币政策的收紧，造成国内证券市场的下跌。

标的指数：沪深300
交易规则：当M1-M2的三月均线上行时买入指数；当M1-M2的三月均线下行时卖出指数并买入无风险资产（无风险收益年化3%）
择时周期：日频

9）PPI与CPI同比的剪刀差（通货膨胀）

CPI：消费者物价指数（Consumer Price Index，简称CPI）反映与居民生活有关的产品及劳务价格统计出来的物价变动指标，通常作为观察通货膨胀水平的重要指标。

PPI：生产者物价指数（Producer Price Index, PPI）是用来衡量生产者在生产过程中，所需采购品的物价状况，是CPI的先声。 通胀与经济的关系较为复杂，温和的通胀水平（通常在2%~3%）可以刺激经济的发展；过高的通胀水平（通常在5%以上），会使人们对货币的信心产生动摇，经济社会产生动荡；而过低的通胀甚至通缩又会抑制经济的发展。单独的通胀指数与股市走势并非一个线性的关系，对于预测股市而言，PPI与CPI同比的剪刀差是一个相对合适的指标。PPI与CPI同比剪刀差的含义是上游工业品与下游消费品的价格差。对于上游企业来说（上游工业品是上游企业的产出），PPI-CPI的扩大意味着利润增加，而对于下游企业来说（上游工业品是下游企业的原材料），PPI-CPI的扩大意味着利润减少，从这个角度来看，PPI-CPI对股市的影响是模糊的。根据郭于玮（兴业研究）的研究表明，PPI与CPI剪刀差收窄后可能触发货币政策调整，进而带来无风险利率的下行和风险偏好的改善，从而使得PPI与CPI剪刀差对利率和信用利差具有一定的领先型。

进一步的，无风险利率的下行和风险偏好的改善对于股市的复苏带来利好。因此，可以尝试基于PPI与CPI剪刀差的走势对A股的走势做出预测信号。

CPI、PPI指标在次月中上旬发布，因此当月的择时需参考上上个月的指标。

标的指数：沪深300
交易规则：当PPI-CPI的三月均线下行时买入指数；当PPI-CPI的三月均线上行时卖出指数（注：当CPI三月均线大于5时认为通胀较高，直接卖出指数，当CPI三月均线小于0时认为处于通缩，也直接卖出指数）
择时周期：月频

主要方法

1）布林带

布林线(Bollinger Band) 是根据统计学中的标准差原理设计出来的一种非常实用的技术指标。它由三条轨道线组成，其中上下两条线分别可以看成是价格的压力线和支撑线，在两条线之间是一条价格平均线，一般情况价格线在由上下轨道组成的带状区间游走，而且随价格的变化而自动调整轨道的位置。当波带变窄时，激烈的价格波动有可能随即产生；若高低点穿越带边线时，立刻又回到波带内，则会有回档产生。一般而言，股价的运动总是围绕某一价值中枢（如均线、成本线等）在一定的范围内变动，布林线指标指标正是在上述条件的基础上，引进了“股价通道”的概念，其认为股价通道的宽窄随着股价波动幅度的大小而变化，而且股价通道又具有变异性，它会随着股价的变化而自动调整。正是由于它具有灵活性、直观性和趋势性的特点，BOLL指标渐渐成为投资者广为应用的市场上热门指标。

主要功能：
1.布林线可以指示支撑和压力位置；
2.布林线可以显示超买、超卖；
3.布林线可以指示趋势；
4.布林线具备通道功能。

布林线的理论使用原则是：当股价穿越最外面的压力线（支撑线）时，表示卖点（买点）出现。当股价延着压力线（支撑线）上升（下降）运行，虽然股价并未穿越，但若回头突破第二条线即是卖点或买点。

BOLL指标的计算是建立在移动平均线与标准差的基础之上的，分为上轨线、中轨线和下轨线，在三线的基础上还可以计算出布林信道的宽度，具体计算的公式如下：

信道宽度：$width = \frac{UP-DN}{MB} \times 100\%$
中轨线：$MB = MA(N) = \frac{1}{N}\sum_{i=1}^{N} C_i$
上轨线：$UP = MB + \alpha STD(N)$
下轨线：$DN = MB - \alpha STD(N)$

C_i表示第i期的收盘价。
α表示置信区间。

Python中可以这样去计算上轨、中线以及下轨并将它们写入到DataFrame中:

In [ ]:

```
upperband,middleband,lowerband = (tl.BBANDS(df_interest[col].values, timeperiod=12, nbdevup=1.8,
nbdevdn=1.8))
df_interest['BBAND_upper']=upperband
df_interest['BBAND_middle']=middleband
df_interest['BBAND_lower']=lowerband
df_interest.head(5)
```

2）移动均线

移动平均线，Moving Average，简称MA，MA是用统计分析的方法，将一定时期内的证券价格（指数）加以平均，并把不同时间的平均值连接起来，形成一根MA，用以观察证券价格变动趋势的一种技术指标。

在python 2中，可以直接用rolling_mean去计算n期的移动平均值，在做均线比较时，可以用shift(m)去上下移动均线的值。

In [ ]:

```
pd.rolling_mean(df_boom[col],n).shift(1)>pd.rolling_mean(df_boom[col],n).shift(2)
```

3）买入和卖出

买入信号被记为1，卖出信号被记为0.所有的买入和卖出信号作为一个数列放在DataFrame中。买入和卖出信号被一个布尔式判断生成。比如，用三月均线生成信号：

In [ ]:

```
n=3
df_boom['position']=(pd.rolling_mean(df_boom[col],n).shift(1)>pd.rolling_mean(df_boom[col],n).sh
ift(2))*1.
```

其中，df_boom['position']是信号的时间序列集合。

再比如，用布林带生成信号序列：

In [ ]:

```
pre_position = 0
for date in df_interest.index:
    if df_interest.loc[date,col]>df_interest.loc[date,'BBAND_middle']:
        df_interest.loc[date,'position']=0
    elif df_interest.loc[date,col]<df_interest.loc[date,'BBAND_lower']:
        df_interest.loc[date,'position']=1.0
    else:
        df_interest.loc[date,'position']=pre_position
    pre_position=df_interest.loc[date,'position']
```

当Shibor:1M向下突破自己的布林带下轨时买入指数，当Shibor:1M上穿自己的布林带中轨时卖出指数，其余情况下，持仓情况维持不变(维持pre_position)。

已知信号序列后，可以用cumprod函数去计算收益率：

In [ ]:

```
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
```

整体收益率是df_pct['net_value_timing']。df_pct['pct']是沪深300指数在当月的收益率，如果买入指数(此时position=1)，那么当月收益率就是(df_pct['pct']+1),如果卖出指数，那么当月收益率就是(rate_riskfree+1)。

4）基准收益率的确定

直接从聚宽调用api获得沪深300的原始数据并初步处理即可。

In [ ]:

```
#调用沪深300初始收盘数据
prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')[
'close']
#将收盘数据转化为月度序列
prices_M = prices.resample('M',how = 'last')
#计算出每日百分比变化
df_pct['pct'] = prices_M.pct_change()
#计算出综合收益率
df_pct['net_value'] = (df_pct['pct']+1).cumprod()
```

5）PPI与CPI同比剪刀差交易

需求：当PPI-CPI的三月均线下行时买入指数；当PPI-CPI的三月均线上行时卖出指数（注：当CPI三月均线大于5时认为通胀较高，直接卖出指数，当CPI三月均线小于0时认为处于通缩，也直接卖出指数）

先定义一个good_cpi函数，再总体择时：

In [ ]:

```
def good_cpi(x):
    if x<0:
        y=0.
    elif x<5.:
        y=1.
    else:
        y=0
    return y
```

只有同时满足三月均线下行且三月均线在(0,5)范围内时期的position会被记为1:

In [ ]:

```
n=3

df_inflation['position']=(pd.rolling_mean(df_inflation[col_2],n).shift(2)<pd.rolling_mean(df_inf
lation[col_2],n).shift(3))*\
    (pd.rolling_mean(df_inflation[col_0],n).apply(good_cpi).shift(2))
```

6）宏观指标的汇总

In [ ]:

```
df_interest_M = df_interest['position'].reindex(date_list).fillna(method='ffill').reindex(prices
_M.index).shift(1)
#利率
df_termspread_M = df_termspread['position'].reindex(date_list).fillna(method='ffill').reindex(pr
ices_M.index).shift(1)
#期限利差
df_creditspread_M = df_creditspread['position'].reindex(date_list).fillna(method='ffill').reinde
x(prices_M.index).shift(1)
#信用利差
df_monetray = ((df_interest_M*1 + df_termspread_M*1 + df_creditspread_M*1)/3.)*1
#货币政策择时指标=利率+期限利差+信用利差
df_forex = (df_exchange_rate['position'].reindex(date_list).fillna(method='ffill').reindex(price
s_M.index).shift(1))
#汇率
df_credit_loan = ((df_credit_loan_1['position']*1+df_credit_loan_2['position']*1)/2.)*1
#信贷择时指标=M1,M2剪刀差+社融指标

df_month=pd.concat([df_monetray,df_forex,df_credit_loan,\
                    df_boom['position'],df_inflation['position']],axis=1)
factor_columns = ['monetary','forex','credit','boom','inflation']
df_month.columns = factor_columns
```

当大多数指标发出看多信号，即可进行买入操作，大部分指标发出看空信号，即可进行卖出操作。因此，我们进行了如下划分，当汇总得分小于0.45时仓位为0%，当汇总得分大于0.55时仓位为100%，中间状态的仓位为50%。

等比重为五个指标赋权时：

In [ ]:

```
weight_f=([1,1,1,1,1])
#每一期（月度）的满分（由于早期有些指标为NAN，因此满分不为1）
timing_count=((df_month[factor_columns]>=0)*weight_f).sum(axis=1)
#根据每一期得分占满分的比例做出仓位决策
df_month['tot_pos'] = ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.55)*0.5+\
    ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.45)*0.5
```

信贷比重提高到两倍时：

In [ ]:

```
weight_f=([1,1,2,1,1])
timing_count=((df_month[factor_columns]>=0)*weight_f).sum(axis=1)
df_month['tot_pos'] = ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.55)*0.5+\
    ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.45)*0.5
```

7）回测指标

可以将回测定义为一个类去实现最大回撤、夏普比率以及calmar比率的计算。其中，年化收益率的计算已在Talib中内置。

In [ ]:

```
class backtest_result():
    def __init__(self,data):
        self.data = data
        self.total_returns = data.iloc[-1]-1
        self.annualized_returns = data.iloc[-1]**(12./len(data))-1
        self.annualized_volatility = data.pct_change().std()*(12.**0.5)
    def Max_Drawback(self):
        net_value=self.data
        max_value=0
        df_tmp=pd.DataFrame(net_value)
        df_tmp.columns=['value']
        for j in range(0,len(net_value),1):
            max_value=max(max_value,df_tmp.ix[j,'value'])
            df_tmp.ix[j,'drawback']=1-df_tmp.ix[j,'value']/max_value
            drawback=df_tmp['drawback'].max()
        return drawback
    def Sharpe(self):
        net_value=self.data
        bench_pct=0.03
        df_tmp=pd.DataFrame(net_value)
        df_tmp.columns=['value']
        df_tmp['pct']=df_tmp['value'].pct_change()
        annual_pct = df_tmp.ix[-1,'value']**(12./len(df_tmp))-1
        sharpe = (annual_pct-bench_pct)/(df_tmp['pct'].std()*12**0.5)
        return sharpe
    def Calmar(self):
        Clamar = self.annualized_returns/self.Max_Drawback()
        return Calmar
```

需要调用回测时：

In [ ]:

```
print backtest_result(df_pct['net_value_timing']).Max_Drawback()#最大回撤
print backtest_result(df_pct['net_value_timing']).Sharpe()#夏普率
print backtest_result(df_pct['net_value_timing']).annualized_returns#年化收益
print backtest_result(df_pct['net_value_timing']).Calmar()#calmar比率
```

代码实现

1) 加载库

In [4]:

```
from jqdata import *
import numpy as np
import pandas as pd
import datetime as dt
from six import StringIO
from dateutil.parser import parse
import cPickle as pickle
import seaborn as sns
import matplotlib as mpl
import os
import statsmodels.api as sm
import scipy
import talib as tl


mpl.rcParams['font.family']='serif'
mpl.rcParams['axes.unicode_minus']=False # 处理负号

load_data={}

path = '/home/jquser/Macro'

class backtest_result():
    def __init__(self,data):
        self.data = data
        self.total_returns = data.iloc[-1]-1
        self.annualized_returns = data.iloc[-1]**(12./len(data))-1
        self.annualized_volatility = data.pct_change().std()*(12.**0.5)
    def Max_Drawback(self):
        net_value=self.data
        max_value=0
        df_tmp=pd.DataFrame(net_value)
        df_tmp.columns=['value']
        for j in range(0,len(net_value),1):
            max_value=max(max_value,df_tmp.ix[j,'value'])
            df_tmp.ix[j,'drawback']=1-df_tmp.ix[j,'value']/max_value
            drawback=df_tmp['drawback'].max()
        return drawback
    def Sharpe(self):
        net_value=self.data
        bench_pct=0.03
        df_tmp=pd.DataFrame(net_value)
        df_tmp.columns=['value']
        df_tmp['pct']=df_tmp['value'].pct_change()
        annual_pct = df_tmp.ix[-1,'value']**(12./len(df_tmp))-1
        sharpe = (annual_pct-bench_pct)/(df_tmp['pct'].std()*12**0.5)
        return sharpe
    def Calmar(self):
        clamar = self.annualized_returns/self.Max_Drawback()
        return clamar
```

1) PMI：采购经理指数

In [6]:

```
body=read_file(path+'/PMI组合.xls')
df_boom=pd.read_excel(StringIO(body))
print df_boom.columns
col =u'PMI'
df_boom=df_boom.set_index(u'日期')
df_boom.plot(figsize=(15,6),title='PMI')
```

```
Index([u'日期', u'PMI', u'PMI:生产', u'PMI:新订单', u'PMI:新出口订单', u'PMI:进口'], dtype='object')
```

Out[6]:

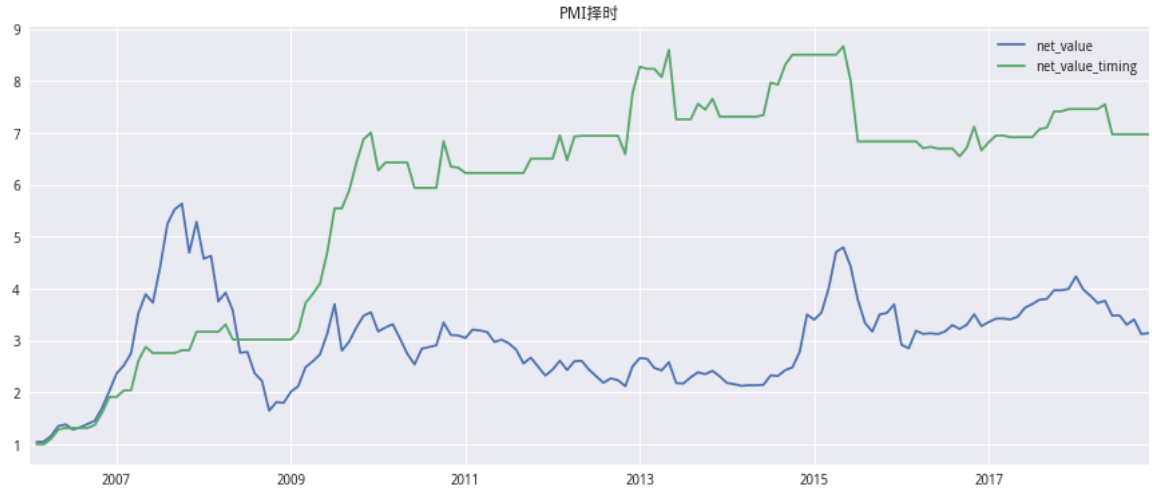
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd7603a10>
```

In [24]:

```
n=3
df_boom['position']=(pd.rolling_mean(df_boom[col],n).shift(1)>pd.rolling_mean(df_boom[col],n).sh
ift(2))*1.
prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')[
'close']
prices_M = prices.resample('M',how='last')
rate_riskfree = 0
df_pct=pd.DataFrame()
df_pct['pct']=prices_M.pct_change()
df_pct['position']=df_boom['position']
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()

df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='PMI择时')
```

Out[24]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd18b8910>



In [25]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.244583943116
夏普率 = 0.668818372855
年化收益 = 0.16222769407
Calmar比率 = 0.663280230106
```

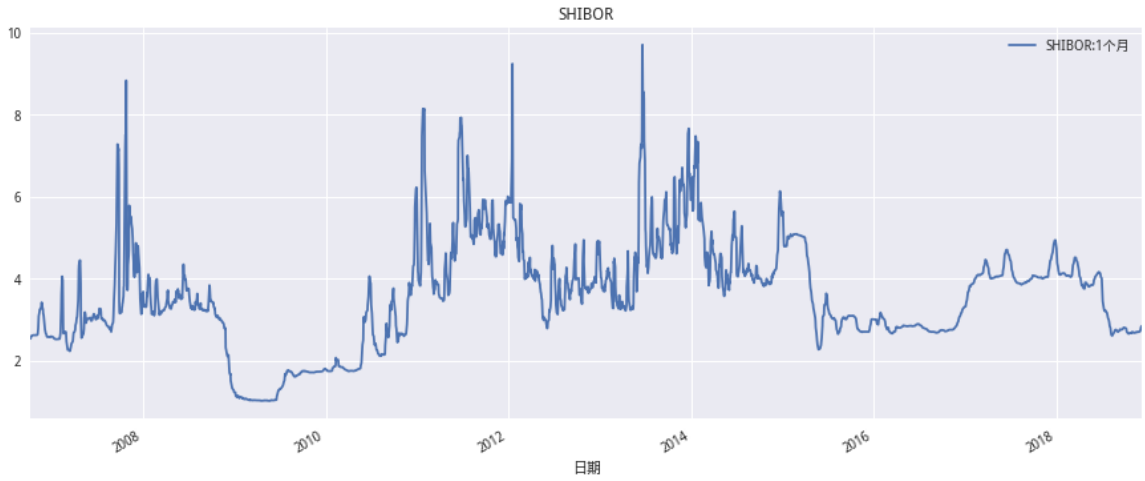
2）SHIBOR:上海银行间同业拆放利率

In [26]:

```
body=read_file(path+'/SHIBOR数据.xls')
df_interest=pd.read_excel(StringIO(body))
col = u'SHIBOR:1个月'
df_interest=df_interest.set_index(u'日期')
df_interest.iloc[:,1:2].plot(figsize=(15,6),title='SHIBOR')
```

Out[26]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd17806d0>

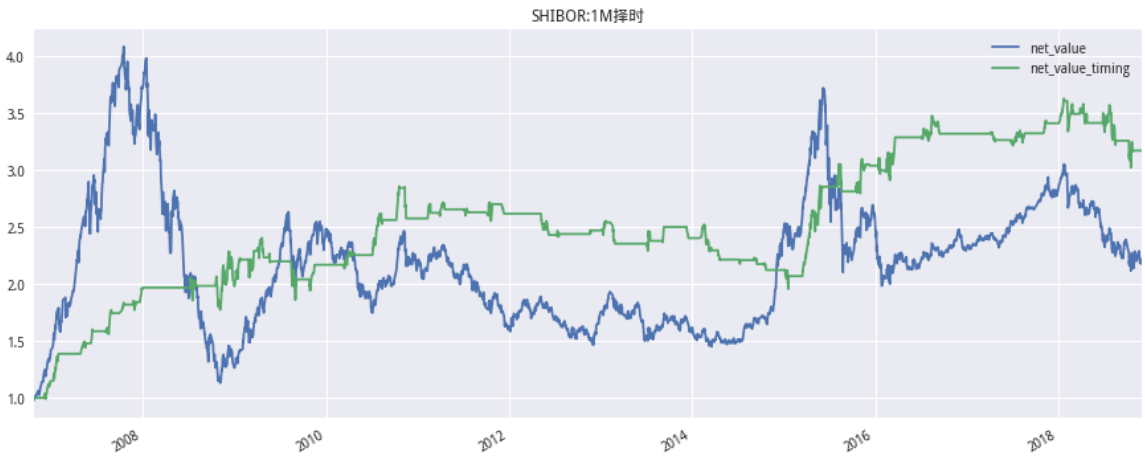


In [27]:

```
df_interest=df_interest[[col]]
upperband,middleband,lowerband = (tl.BBANDS(df_interest[col].values, timeperiod=12, nbdevup=1.8,
nbdevdn=1.8))
# print df_1

df_interest['BBAND_upper']=upperband
df_interest['BBAND_middle']=middleband
df_interest['BBAND_lower']=lowerband

pre_position = 0
for date in df_interest.index:
    if df_interest.loc[date,col]>df_interest.loc[date,'BBAND_middle']:
        df_interest.loc[date,'position']=0
    elif df_interest.loc[date,col]<df_interest.loc[date,'BBAND_lower']:
        df_interest.loc[date,'position']=1.0
    else:
        df_interest.loc[date,'position']=pre_position
    pre_position=df_interest.loc[date,'position']
df_interest['position']=df_interest['position'].shift(1)

df_pct=pd.DataFrame()
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-11-30',fields='close')[
'close']
df_pct['pct']=prices.pct_change()

rate_riskfree = 0
df_pct = pd.concat([df_pct,df_interest],axis=1)['2006-01-01':'2018-11-30'].dropna()
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='SHIBOR:1M择时')
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd1780150>



In [28]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.315660344668
夏普率 = -0.753461742652
年化收益 = 0.00470353746367
Calmar比率 = 0.0149006282959
```

3）期限利差

In [29]:

```
body=read_file(path+'/国债到期收益率.xls')
df_tbonds=pd.read_excel(StringIO(body))
df_tbonds.set_index(u'日期',inplace=True)
df_tbonds=df_tbonds.fillna(method='ffill')
term_spread_tbonds = df_tbonds[u'中债国债到期收益率:10年']-df_tbonds[u'中债国债到期收益率:1个月']
term_spread_tbonds_diff = term_spread_tbonds.diff(21)
term_spread_tbonds=pd.rolling_mean(term_spread_tbonds,1)
term_spread_tbonds.plot(figsize=(15,6),title='10年-1年国债期限利差')
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd154a650>
```

In [30]:

```
df_termspread=pd.DataFrame()
col='termspread'
df_termspread=term_spread_tbonds.to_frame('termspread')
upperband,middleband,lowerband = (tl.BBANDS(df_termspread[col].values, timeperiod=25, nbdevup=1.
8, nbdevdn=1.8))

df_termspread['BBAND_upper']=upperband
df_termspread['BBAND_middle']=middleband
df_termspread['BBAND_lower']=lowerband

df_termspread.head()
pre_position = 0
for date in df_termspread.index:
    if df_termspread.loc[date,col]<df_termspread.loc[date,'BBAND_middle']:
        df_termspread.loc[date,'position']=0
    elif df_termspread.loc[date,col]>df_termspread.loc[date,'BBAND_upper']:
        df_termspread.loc[date,'position']=1.0
    else:
        df_termspread.loc[date,'position']=pre_position
    pre_position=df_termspread.loc[date,'position']
df_termspread['position']=df_termspread['position'].shift(1)
df_termspread.head().append(df_termspread.tail())

df_pct=pd.DataFrame()
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-11-30',fields='close')[
'close']
df_pct['pct']=prices.pct_change()

rate_riskfree = 0
df_pct = pd.concat([df_pct,df_termspread],axis=1)['2007-01-01':'2018-11-30'].dropna()
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='国债期限利差择时')
```

Out[30]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd144ec50>

In [31]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.265692286739
夏普率 = -0.758548178305
年化收益 = 0.00328037459482
Calmar比率 = 0.012346517978
```

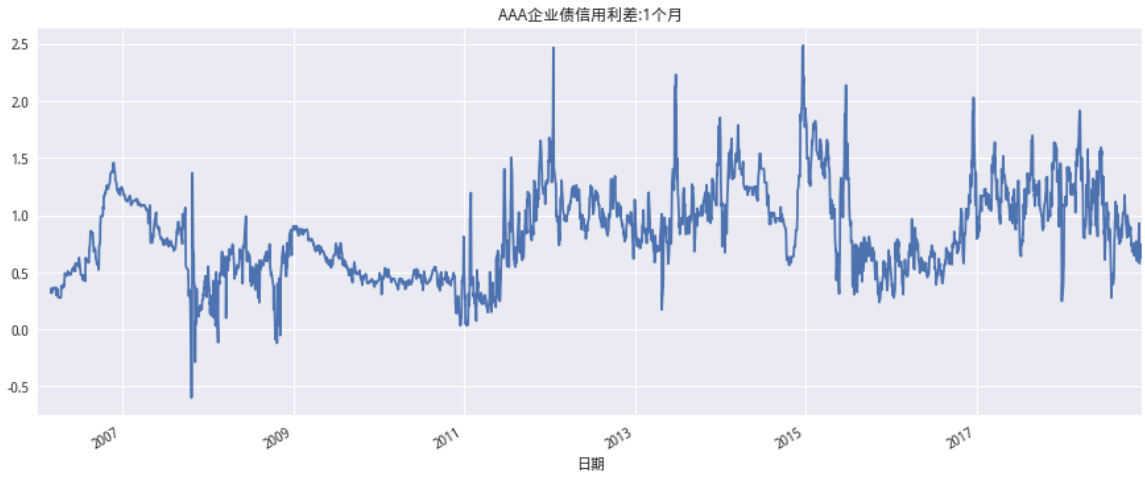
4）信用利差

In [32]:

```
body=read_file(path+'/企业债到期收益率(AAA).xls')
df_cbonds=pd.read_excel(StringIO(body))
df_cbonds.set_index(u'日期',inplace=True)
df_cbonds=df_cbonds.fillna(method='ffill')
credit_spread=df_cbonds[u'中债企业债到期收益率(AAA):1个月']-df_tbonds[u'中债国债到期收益率:1个月']
credit_spread=pd.rolling_mean(credit_spread,1)
credit_spread['2006-01-01':].plot(figsize=(15,6),title='AAA企业债信用利差:1个月')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd137c0d0>
```



In [34]:

```
df_creditspread=pd.DataFrame()
col='creditspread'
df_creditspread=credit_spread.to_frame('creditspread')
upperband,middleband,lowerband = (tl.BBANDS(df_creditspread[col].values, timeperiod=25, nbdevup=
1.8, nbdevdn=1.8))

df_creditspread['BBAND_upper']=upperband
df_creditspread['BBAND_middle']=middleband
df_creditspread['BBAND_lower']=lowerband
pre_position = 0
for date in df_creditspread.index:
    if df_creditspread.loc[date,col]>df_creditspread.loc[date,'BBAND_middle']:
        df_creditspread.loc[date,'position']=0
    elif df_creditspread.loc[date,col]<df_creditspread.loc[date,'BBAND_lower']:
        df_creditspread.loc[date,'position']=1.0
    else:
        df_creditspread.loc[date,'position']=pre_position
    pre_position=df_creditspread.loc[date,'position']
df_creditspread['position']=df_creditspread['position'].shift(1)

df_pct=pd.DataFrame()
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-11-30',fields='close')[
'close']
df_pct['pct']=prices.pct_change()

rate_riskfree =0
df_pct = pd.concat([df_pct,df_creditspread],axis=1)['2007-01-01':'2018-11-30'].dropna()
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='信用利差择时')
```

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd11d8c90>

In [35]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

最大回撤 = 0.326340290274
夏普率 = -0.846550264062
年化收益 = 0.00325843901344
Calmar比率 = 0.00998478922326

5）M1与M2同比的剪刀差

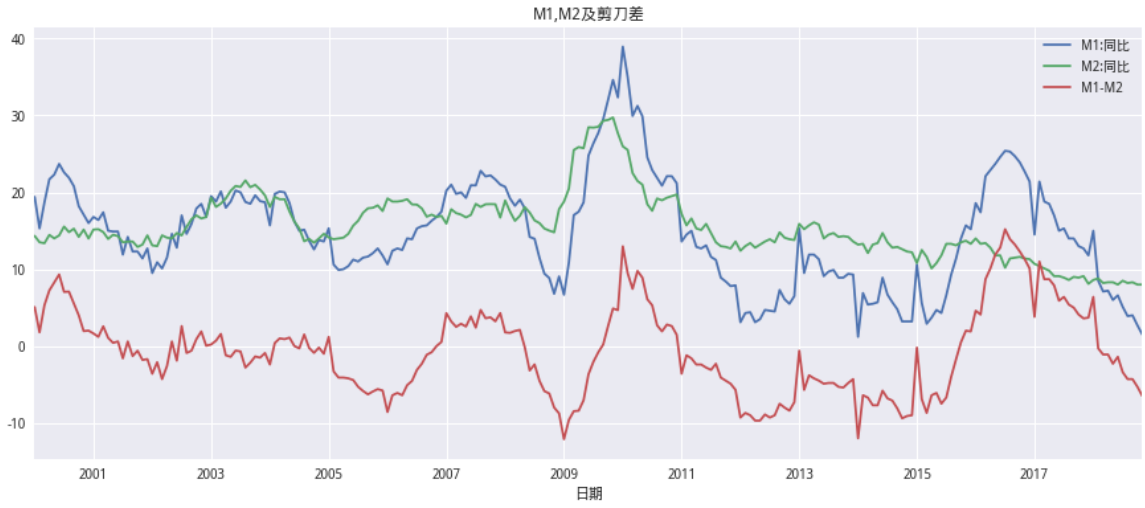
In [36]:

```
#M1,M2剪刀差择时
body=read_file(path+'/信贷.xls')
df_credit_loan_1=pd.read_excel(StringIO(body))
df_credit_loan_1=df_credit_loan_1.set_index(u'日期')
print df_credit_loan_1.columns
col = u'社会融资规模:当月值'
col_1 = u'M1:同比'
col_2 = u'M2:同比'
# col = col_1
col = u'M1-M2'
df_credit_loan_1[u'M1-M2'] = df_credit_loan_1[u'M1:同比']-df_credit_loan_1[u'M2:同比']
n=3
df_credit_loan_1.iloc[:,1:].plot(figsize=(15,6),title='M1,M2及剪刀差')
```

Index([u'社会融资规模:当月值', u'M1:同比', u'M2:同比'], dtype='object')

Out[36]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd10f8750>

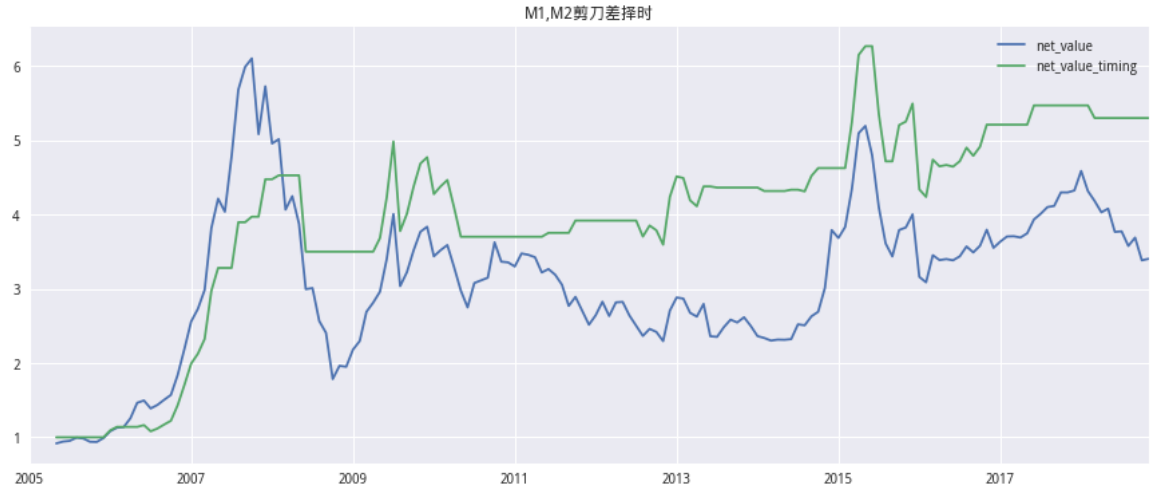


In [37]:

```
df_credit_loan_1['position']=pd.rolling_mean(df_credit_loan_1[col],n).shift(2)>pd.rolling_mean(df_credit_loan_1[col],n).shift(3)
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-11-30',fields='close')['close']
prices_M = prices.resample('M',how='last')

rate_riskfree = 0

df_pct=pd.DataFrame()
df_pct['pct']=prices_M.pct_change()
df_pct['position']=df_credit_loan_1['position']
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['position'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='M1,M2剪刀差择时')
```

Out[37]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0f62e10>



In [38]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

最大回撤 = 0.323846148622
夏普率 = 0.432180022147
年化收益 = 0.127283740601
Calmar比率 = 0.393037685155

6）社会融资规模

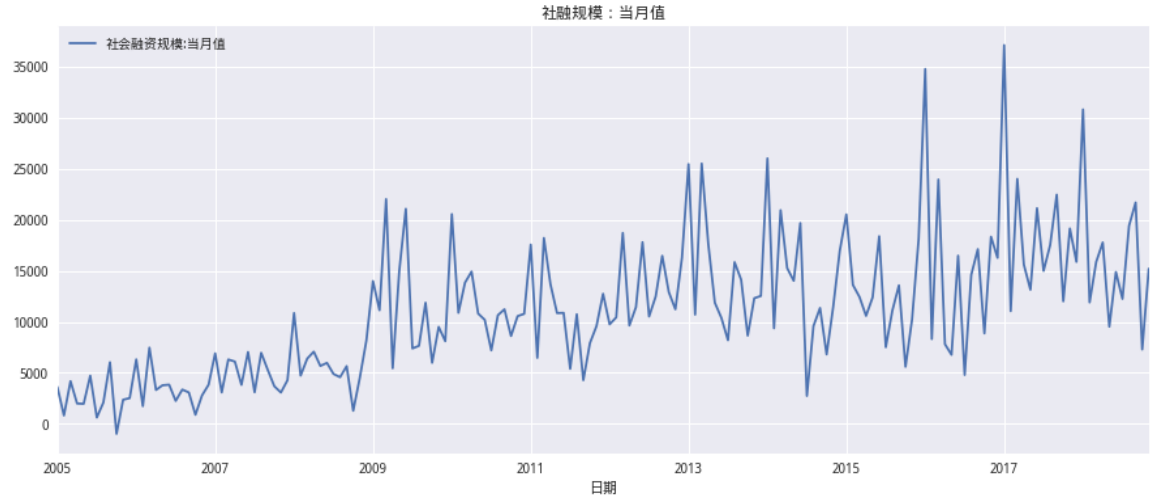
In [39]:

```
body=read_file(path+'/信贷.xls')
df_credit_loan_2=pd.read_excel(StringIO(body))
df_credit_loan_2=df_credit_loan_2.set_index(u'日期')
print df_credit_loan_2.columns
col = u'社会融资规模:当月值'
col_1 = u'M1:同比'
col_2 = u'M2:同比'
n=3
pd.rolling_mean(df_credit_loan_2.iloc[:,:1],1)['2005-01-01':].plot(figsize=(15,6),title='社融规模：当月值')
```

```
Index([u'社会融资规模:当月值', u'M1:同比', u'M2:同比'], dtype='object')
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0e8ddd0>
```



In [42]:

```
df_credit_loan_2['position']=pd.rolling_mean(df_credit_loan_2[col],n).shift(2)>pd.rolling_mean(df_credit_loan_2[col],n).shift(3)
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-12-07',fields='close')['close']
prices_M = prices.resample('M',how='last')

rate_riskfree = 0

df_pct=pd.DataFrame()
df_pct['pct']=prices_M.pct_change()
df_pct['position']=df_credit_loan_2['position']
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['position'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='社融择时')
```

Out[42]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd5354dd0>

In [46]:

```
#print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
#print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
#print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
#print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

7）金融机构存款准备金率

In [47]:

```
body=read_file(path+'/存款准备金率-大型存款类机构.xls')
df_reserves=pd.read_excel(StringIO(body))
df_reserves = df_reserves.set_index(u'日期')
df_reserves
col=u'人民币存款准备金率:大型存款类金融机构'
df_pct=pd.DataFrame()
prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')['close']
prices_M = prices.resample('M',how='last')
df_pct['pct']=prices.pct_change()
df_pct
rate_riskfree=0

df_reserves=df_reserves.reindex(prices.index).fillna(method='ffill')
fig = plt.figure(figsize=(15,6))
ax1=fig.add_subplot(111)
ax1.plot(df_reserves[col],'y-',linewidth=2,label='大型机构存款准备金率')
ax1.legend(loc=2,fontsize=12)
ax2=ax1.twinx()
ax2.plot(prices,'b-',linewidth=2,label='沪深300指数')
ax2.legend(loc=1,fontsize=12)
ax2.grid(False)
plt.show()
```

In [48]:

```
pre_position=0
delay_days=120
for i in range(delay_days,len(df_reserves)):
    pre_index = df_reserves.index[i-delay_days]
    index = df_reserves.index[i]
#     print df_reserves.loc[index,col]
    if df_reserves.loc[index,col]<df_reserves.loc[pre_index,col]:
        df_reserves.loc[index,'position']=1
    elif df_reserves.loc[index,col]>df_reserves.loc[pre_index,col]:
        df_reserves.loc[index,'position']=0.
    else:
        df_reserves.loc[index,'position']=0.
    pre_position = df_reserves.loc[index,'position']
df_pct['position'] = df_reserves['position']
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='存款准备金择时')
```

Out[48]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd09f9c10>

In [49]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.399381223366
夏普率 = -0.861921530246
年化收益 = 0.00195914549677
Calmar比率 = 0.0049054521899
```

8）USDCNH:即期汇率

In [50]:

```
body=read_file(path+'/离岸汇率数据.xls')
df_exchange_rate=pd.read_excel(StringIO(body))
col = u'USDCNH:即期汇率'
df_exchange_rate=df_exchange_rate.set_index(u'日期')
df_exchange_rate.plot(figsize=(15,6),title='USDCNH:即期汇率')
```

Out[50]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd1278850>

In [51]:

```
upperband,middleband,lowerband = (tl.BBANDS(df_exchange_rate[col].values, timeperiod=25, nbdevup
=1.8, nbdevdn=1.8))
df_exchange_rate['BBAND_upper']=upperband
df_exchange_rate['BBAND_middle']=middleband
df_exchange_rate['BBAND_lower']=lowerband
df_exchange_rate.head()
pre_position = 0
for date in df_exchange_rate.index:
    if df_exchange_rate.loc[date,col]>df_exchange_rate.loc[date,'BBAND_middle']:
        df_exchange_rate.loc[date,'position']=0
    elif df_exchange_rate.loc[date,col]<df_exchange_rate.loc[date,'BBAND_lower']:
        df_exchange_rate.loc[date,'position']=1.0
    else:
        df_exchange_rate.loc[date,'position']=pre_position
    pre_position=df_exchange_rate.loc[date,'position']
df_exchange_rate['position']=df_exchange_rate['position'].shift(1)
df_exchange_rate.head().append(df_exchange_rate.tail())

df_pct=pd.DataFrame()
prices = get_price('000300.XSHG',start_date='2005-01-01',end_date='2018-12-07',fields='close')[
'close']
df_pct['pct']=prices.pct_change()
df_pct = pd.concat([df_pct,df_exchange_rate],axis=1)['2007-01-01':'2018-11-30'].dropna()
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='USDCNH:即期汇率择时')
```

Out[51]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd09e6bd0>

In [52]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.146660846067
夏普率 = -1.11192583531
年化收益 = 0.00399159825772
Calmar比率 = 0.0272165227788
```

9）PPI与CPI同比的剪刀差（通货膨胀）

In [53]:

```
body=read_file(path+'/CPI与PPI.xls')
df_inflation=pd.read_excel(StringIO(body))
print df_inflation.columns
col_0 =u'CPI:当月同比'
col_1 =u'PPI:全部工业品:当月同比'
df_inflation=df_inflation.set_index(u'日期')['2005-01-31':]
col_2 = u'PPI-CPI'
df_inflation[col_2] = -df_inflation[u'CPI:当月同比']+df_inflation[u'PPI:全部工业品:当月同比']
df_inflation[[col_0,col_1,col_2]].plot(figsize=(15,6))
```

```
Index([u'日期', u'CPI:当月同比', u'预测平均值:CPI:当月同比', u'PPI:全部工业品:当月同比',
       u'预测平均值:PPI:当月同比'],
      dtype='object')
```

Out[53]:

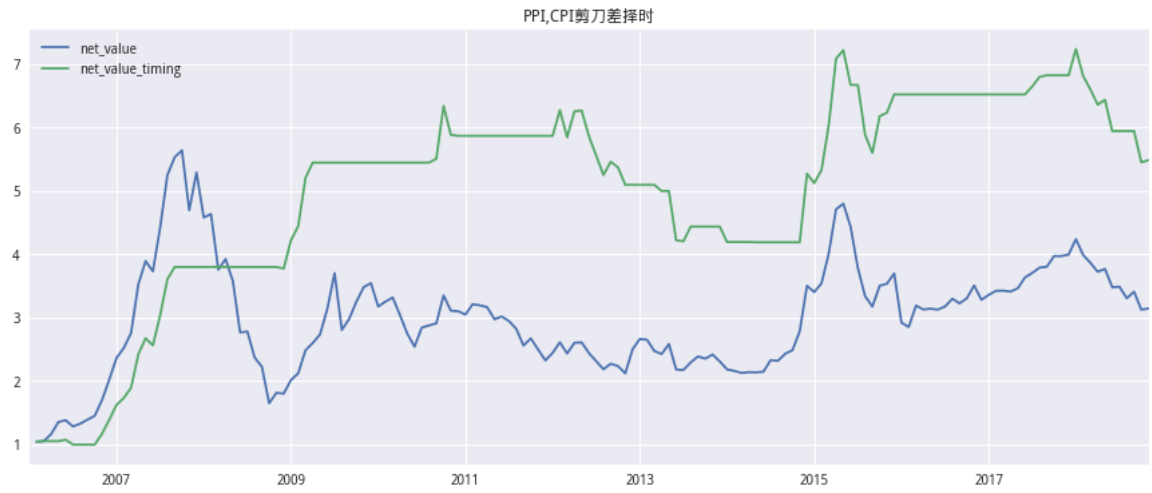
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd083eed0>
```

In [54]:

```
def good_cpi(x):
    if x<0:
        y=0.
    elif x<5.:
        y=1.
    else:
        y=0
    return y

n=3
df_inflation['position']=(pd.rolling_mean(df_inflation[col_2],n).shift(2)<pd.rolling_mean(df_inf
lation[col_2],n).shift(3))*\
    (pd.rolling_mean(df_inflation[col_0],n).apply(good_cpi).shift(2))

prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')[
'close']
prices_M = prices.resample('M',how='last')
rate_riskfree = 0#(1+1.25e-4)**20.0-1
df_pct=pd.DataFrame()
df_pct['pct']=prices_M.pct_change()
df_pct['position']=df_inflation['position']
df_pct['net_value'] =(df_pct['pct']+1).cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1).cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='PPI,CPI剪刀差择时')
```

Out[54]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0655b90>



In [55]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

最大回撤 = 0.339225056683
夏普率 = 0.525666162045
年化收益 = 0.140756420622
Calmar比率 = 0.414935211444

10）汇总分析

In [59]:

```
def dateRange(beginDate, endDate):
    dates = []
    dt=beginDate
    date = beginDate
    while date <= endDate:
        dates.append(date)
        dt = dt + datetime.timedelta(1)
        date = dt
    return dates
date_list = dateRange(prices_M.index[0],prices_M.index[-1])
```

In [58]:

```
df_interest_M = df_interest['position'].reindex(date_list).fillna(method='ffill').reindex(prices_M.index).shift(1)
df_termspread_M = df_termspread['position'].reindex(date_list).fillna(method='ffill').reindex(prices_M.index).shift(1)
df_creditspread_M = df_creditspread['position'].reindex(date_list).fillna(method='ffill').reindex(prices_M.index).shift(1)
df_monetray = ((df_interest_M*1 + df_termspread_M*1 + df_creditspread_M*1)/3.)*1
df_forex = (df_exchange_rate['position'].reindex(date_list).fillna(method='ffill').reindex(prices_M.index).shift(1))
df_credit_loan = ((df_credit_loan_1['position']*1+df_credit_loan_2['position']*1)/2.)*1

df_month=pd.concat([df_monetray,df_forex,df_credit_loan,\
                    df_boom['position'],df_inflation['position']],axis=1)
factor_columns = ['monetary','forex','credit','boom','inflation']
df_month.columns = factor_columns
```

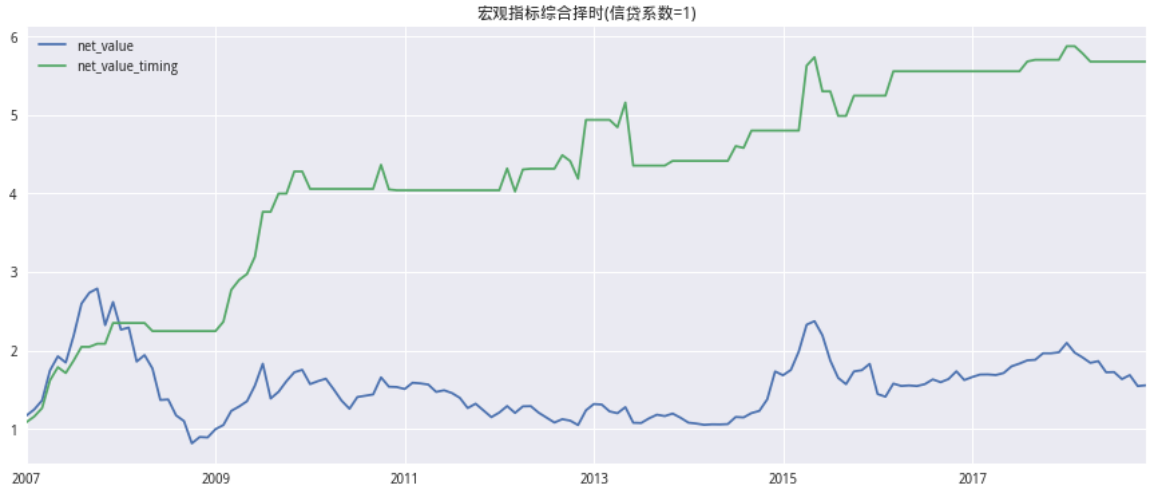
将五大类宏观经济因子静态等权重分配时：

In [66]:

```
weight_f=([1,1,1,1,1])
timing_count=((df_month[factor_columns]>=0)*weight_f).sum(axis=1)

df_month['tot_pos'] = ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.55)*0.5+\
    ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.45)*0.5
prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')[
'close']
prices_M = prices.resample('M',how='last')
rate_riskfree = 0
df_pct=pd.DataFrame()
start_date='2007-01-01'
df_pct['pct']=prices_M.pct_change()[start_date:]
df_pct['position']=df_month['tot_pos']
df_pct['net_value'] =(df_pct['pct']+1)[start_date:].cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1)[start_date:].cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='宏观指标综合择时(信贷系数=
1)')
```

Out[66]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd00e8bd0>



In [67]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

最大回撤 = 0.155688048403
夏普率 = 0.743063697024
年化收益 = 0.156875441294
Calmar比率 = 1.00762674401

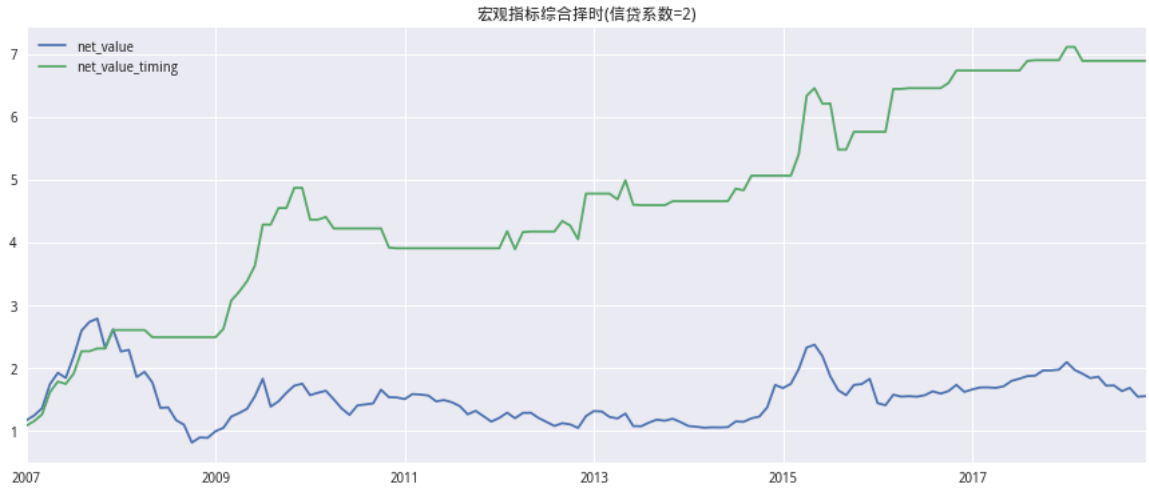
将信贷系数设置为2倍时:

In [68]:

```
weight_f=([1,1,2,1,1])
timing_count=((df_month[factor_columns]>=0)*weight_f).sum(axis=1)

df_month['tot_pos'] = ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.55)*0.5+\
    ((df_month[factor_columns]*weight_f).sum(axis=1)/timing_count>0.45)*0.5
prices = get_price('000300.XSHG',start_date='2006-01-01',end_date='2018-11-30',fields='close')[
'close']
prices_M = prices.resample('M',how='last')
rate_riskfree = 0
df_pct=pd.DataFrame()
start_date='2007-01-01'
df_pct['pct']=prices_M.pct_change()[start_date:]
df_pct['position']=df_month['tot_pos']
df_pct['net_value'] =(df_pct['pct']+1)[start_date:].cumprod()
df_pct['net_value_timing'] = (df_pct['pct']*df_pct['position']+rate_riskfree*(1-df_pct['positio
n'])+1)[start_date:].cumprod()
df_pct[['net_value','net_value_timing']].plot(figsize=(15,6),title='宏观指标综合择时(信贷系数=
2)')
```

Out[68]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1dcbf9bd50>



In [69]:

```
print '最大回撤 = '+ str(backtest_result(df_pct['net_value_timing']).Max_Drawback())
print '夏普率 = '+ str(backtest_result(df_pct['net_value_timing']).Sharpe())
print '年化收益 = '+ str(backtest_result(df_pct['net_value_timing']).annualized_returns)
print 'Calmar比率 = '+ str(backtest_result(df_pct['net_value_timing']).Calmar())
```

```
最大回撤 = 0.200431171329
夏普率 = 0.818573049115
年化收益 = 0.175762390985
Calmar比率 = 0.876921438015
```

结论与局限性

本文分别分析并回测了九个宏观经济因子以及他们汇总后在A股的表现。单个宏观因子尽管有着很强的经济学含义，但却不能很好地体现在大盘上，而这些因子汇总后的整体表现良好。

本文在数据的选择上存在诸多问题，而这些问题很大程度上来源于宏观因子的局限性：

1.一些宏观因子数据数年都不变，或者时间尺度过大（季度数据），而不能提供必要的信息；

2.一些宏观因子（特别是来自于银行的因子），会因为追求业绩而不够准确；

3.一些宏观因子存在事后修正，所以存在使用未来函数的风险。

除此之外，基于周期的宏观因子分析还存在过拟合、数据窥探和幸存者偏差的风险。（Bender,2018）所以，本文的主要目的还是在于定性地去分析宏观经济因子择时在A股中的表现。

未来的研究方向

好的策略应该是能穿越牛熊市的，本文的汇总策略虽然规避了2008年全球性的金融危机，却没能穿越2015年的股灾。然而，独立因子中，SHIBOR和即期汇率能够成功做到这一点。值得进一步探讨的是，究竟哪些宏观经济因子以及哪些宏观因子的组合能成功穿越这次股灾。除此之外，回顾十多年来我国经济的发展，可以发现国内经济显著受信贷周期驱动，另外信贷体量（M1,M2,社融等指标）也是国内特殊的数量型货币政策色彩背景下非常重要的中介目标变量，对未来经济发展与货币政策制定有着重要参考意义。本文仅单纯地将信贷系数翻倍做了研究，就少量提高了年化收益和夏普比率。如何在避免过拟合的情况下确定信贷因子的最佳系数以及信贷这个因子在大盘波动背后扮演的角色值得进一步的研究。