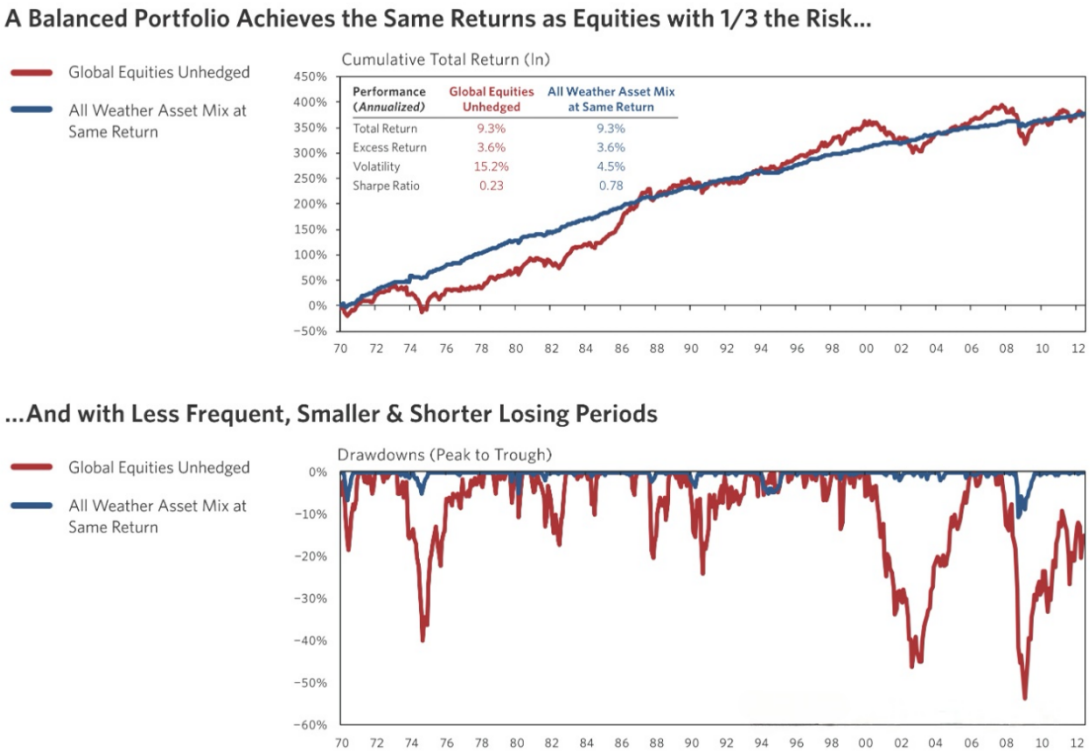
尊敬的 iQuant 用户、QMT 使用者以及广大量化爱好者们：

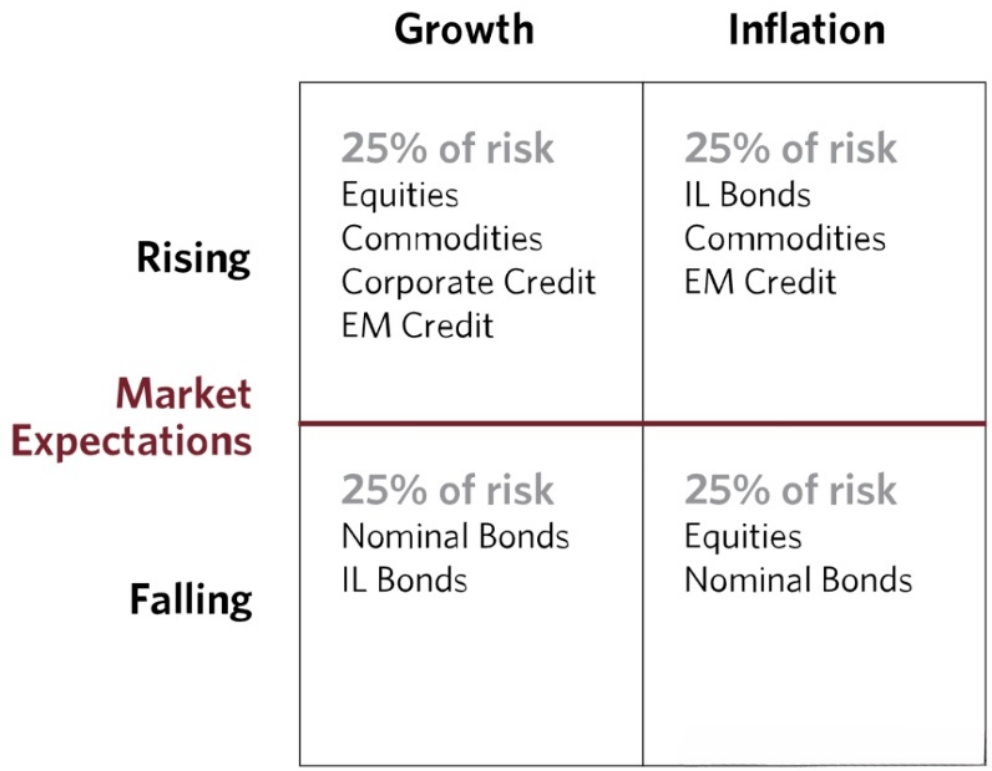
桥水基金，作为世界顶尖的量化对冲巨擘，它管理着超过 2350 亿美元的资金。这一创投传奇的缔造者，瑞•达利欧（Ray Dalio），不仅在金融界声名显赫，还以《原则》一书的作者身份，向世人分享他的智慧。桥水旗下的两大旗舰产品——自 1991 年便立下汗马功劳的绝对 Alpha 基金（Pure Alpha），以及 1996 年诞生、历经沧桑的全天候基金（All Weather）——都见证了达利欧非凡的投资眼光。



全天候基金所采用的，正是名副其实的全天候策略。这一策略的核心理念，在于不去揣测未来经济周期的变迁，也不预设哪类资产会成为市场新宠。而是通过分散投资于各式金融资产，力求在任何经济气候中都能收获稳健的回报。根据官方数据，全天候策略在实现与股票同等收益率的同时，其波动性仅为股票市场的三分之一。这样的策略设计，确保了桥水基金能够穿越市场风云，稳步前行。

先来看看 Dalio 如何划分经济周期:

他根据经济增长(Growth)和通胀水平(Inflation)实际值和市场预期值(Market Expectation)之间的相对大小关系将经济周期划分为 4 种宏观状态：经济上行、经济下行、通胀上行、通胀下行。



再来看看不同宏观经济状态下哪种资产会带来更好的收益，一般而言，经济上行时股票类资产表现较好，经济下行或通胀下行时债券类资产表现出色，经济上行或通胀上行时大宗商品类资产表现更好。于是乎，Dalio 根据各类资产的历史表现和相关金融理论，为这 4 个经济周期分别安排了不同的适配资产。

经济上行期：股票(Equities)、商品(Commodities)、公司信用债(Corporate Credit)、新兴市场债(Emerging Market Credit)。

经济下行期：名义债券(Nominal Bonds)、通胀挂钩债券(Inflation-Linked Bonds)。

通胀上行期：通胀挂钩债券(Inflation-Linked Bonds)、商品(Commodities)、新兴市场债(Emerging Market Credit)。

通胀下行期：股票(Equities)、名义债券(Nominal Bonds)。

最后，Dalio 给每种可能出现的经济周期配置相同的风险敞口，也就是每个象限的子资产组合都是 25%，在每个子资产组合里面又把风险权重等量地分配给组合中的每种资产，进而获得每种资产风险权重和最终配置比例。

就这样，通过多元化的资产配置，并且利用各种资产之间相关性对抗风险，每个时期，都至少有一个象限的资产组合处在上涨周期，便可以在长周期里面获得较为稳定的收益，增强净值曲线的稳定性。

那具体怎么计算每种资产的风险权重和配置比例呢？这可以参照风险平价(Risk Parity)的计算流程。

第一步，选择底仓。

钱恩平博士在《Risk Parity Fundamentals》[2]（2016）中指出，资产背后的风险溢价分为三种，权益风险溢价、利率风险溢价和通胀风险溢价。从理论上看，选择相关性较低、资产属性差别较大、流动性较好的资产，有助于完善风险平价的配置图谱。对低波动资产加杠杆，以及通过定期调仓进行再平衡，构建风险平价组合。

第二步，计算资产对组合的风险贡献。

定义资产对组合的边际风险贡献为组合波动率对权重的偏导数，资产对总风险的贡献为该资产权重与其边际风险贡献的乘积：

$$TRC_i = w_i \frac{\partial \sigma_p}{\partial w_i}$$

其中，w_i表示资产分配权重，θ表示资产的协方差矩阵，$\sigma_w = \sqrt{w^T \theta w}$ 衡量组合风险。

第三步，优化组合风险贡献，计算资产权重。

对于风险平价组合，资产对总风险的贡献水平相等，即：

$$TRC_i = TRC_j$$

权重求解本质上是二次优化问题，求资产对组合风险贡献差值的最小值，本文采用 Python 中的 minimize 函数求解：

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} (TRC_i - TRC_j)^2$$

$$s.t. \sum w_i = 1, 0 \le w_i \le 1$$

这里补充说明一下全天候策略和风险平价之间的关系，经常看到“全天候策略是参照风险平价理论构建的”等类似的说法，这样是不严谨的，因为全天候基金在 1996 年就成立了，而风险平价是磐安基金(Panagora)的钱恩平(Edward Qian)博士在 2005 年才提出的，属于是先有实践后有理论总结，只不过是这理论总结的是太好了，太符合全天候策略的本质内核了。

风险平价的中心思想就是希望在资产组合当中，每个资产的风险敞口都是一样的，也就是每个资产都是面对相同的风险。也就说，放弃对资产组合收益的预测，把重心转移到资产组合风险预算的规划上，资产权重直接反映风险属性。如果你觉得某些资产的风险太低的话，可以通过放杠杆的方式调节，只不过这时需要考虑资金成本的问题。

要实现全天候策略，在计算风险平价确定各个资产配置权重这一步，对量化萌新比较难，不过也不用担心，Dalio 很贴心地给大伙儿准备了一个简易版或者说是躺平版的全天候策略。

在托尼•罗宾斯(Tony Robbins)的著作《Money: Master the Game》里面，Dalio 在接受作者采

访时，给出了一组全天候策略资产配置比例的经验数值，根据这组数值定期进行再平衡就可以了。

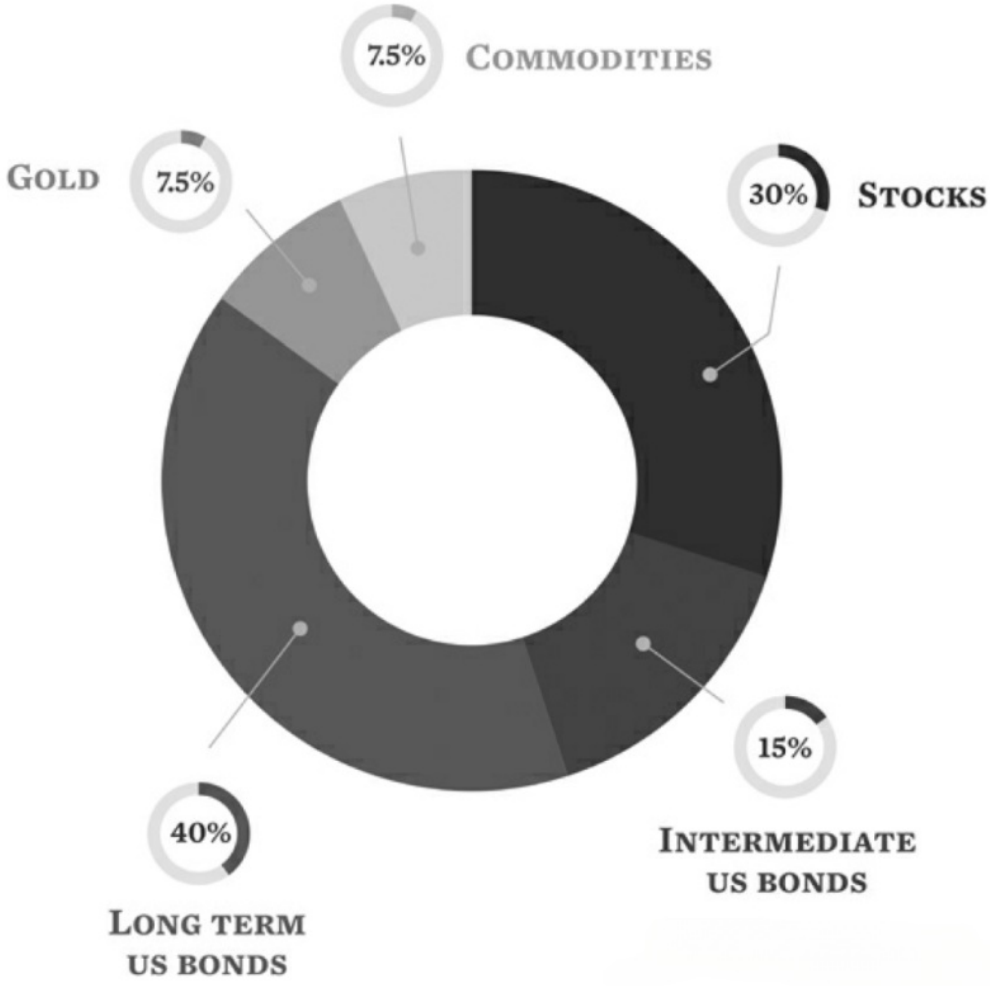
30%投资于股票(stocks)：股价波动大弹性好，在经济上行期收益更支棱。

15%投资于中期债券(intermediate term bonds)：能提供稳定的利息收益，经济下行或通胀下行时表现更出色。

40%投资于长期债券(longterm bonds)：配置原理同中期债券，但相较于中期债券而言，对利率的变动更为敏感。

7.5%投资于黄金(gold)：带有明显的避险属性，在通胀上行和世界动荡时表现出色。

7.5%投资于大宗商品(commodities)：在经济上行或通胀上行时大宗商品类资产往往表现优异。



为了让这个全天候策略在我国 A 股市场也具有投资意义，我打算用对应不同资产类别的 ETF 完成资产组合的构建，秉承着最适配和尽量分散的原则，组合清单如下。

股票类(30%)：沪深 300ETF(510300.SH)，中证 500ETF(510500.SH)，标普 500ETF(513500.SH)，纳指 ETF(513300.SH)；
中期债券(15%)：10 年期国债 ETF(511260.SH)；
长期债券(40%)：30 年期国债 ETF(511090.SH)；
黄金(7.5%)：黄金 ETF(518880.SH)；
大宗商品(7.5%)：大宗商品 ETF(510170.SH)。

下面是如何在 Iquant 中实现全天候策略：
首先介绍 order_target_percent-（比例仓位管理函数）指定目标比例交易

用法： order_target_percent(stockcode, tar_percent[, style, price], ContextInfo[, accId])
释义： 指定目标比例交易，买入/卖出证券以自动调整该证券的仓位到占有一个指定的投资组合的目标百分比。投资组合价值等于所有已有仓位的价值和剩余现金的总和。买 / 卖单会被下舍入一手股数（A 股是 100 的倍数）的倍数。目标百分比应该是一个小数，并且最大值应该小于等于 1，比如 0.5 表示 50%，

每周一进行再平衡，开平仓费率各万分之三，回测一下近 10 年的情况，如下。

在 Iquant 中回测结果如下 ，为了分析和评价你提供的策略性能，我们可以根据所给数据点来理解这个全天候策略在特定时期内的表现。以下是一些关键指标的解释和分析：

1. **单位净值**：1.410 表示策略自开始以来的累计收益率为 41.0%。
2. **基准净值**：0.957 表明在同一时期内，基准的累计收益率为-4.3%，即基准有所下跌。
3. **开仓/平仓次数**：策略执行了 908 次开仓和 985 次平仓操作，表示策略相当活跃。
4. **胜率**：0.780 或 78%的胜率表示策略在所有交易中有 78%是盈利的。
5. **年化收益**：策略的年化收益率为 23.6%，而基准的年化收益率为-8.0%，说明策略大幅超越了基准。
6. **贝塔系数**：0.408 的贝塔值表明策略相对于基准的市场风险较低。
7. **阿尔法系数**：31（通常以百分比形式表示）意味着策略在调整了市场风险后产生了显著的超额回报。
8. **波动率**：策略的波动率为 5.1%，表示策略的回报波动程度较低。
9. **夏普比率**：4.026 是一个非常高的夏普比率，意味着每承担一单位总风险，策略产生了高额的超额回报。
10. **下方差**：0.398 表明策略的下行风险较小。
11. **索提诺比率**：0.177，与夏普比率相似，但仅考虑下行波动，该值也表明策略具有良好的风险调整后的回报。
12. **跟踪误差**：0.021 表示策略相对于基准的偏离程度非常小。
13. **信息比率**：0.052 表示策略相对于基准的超额回报与跟踪误差的比例，正值表明策略的超额回报是积极的。
14. **最大回撤率**：0.069 或 6.9%的最大回撤表明策略潜在的最大损失较小。

总体上，这个策略显示出相对较低的风险和高收益，特别是在夏普比率和年化收益上表现出色，远远超越了基准。

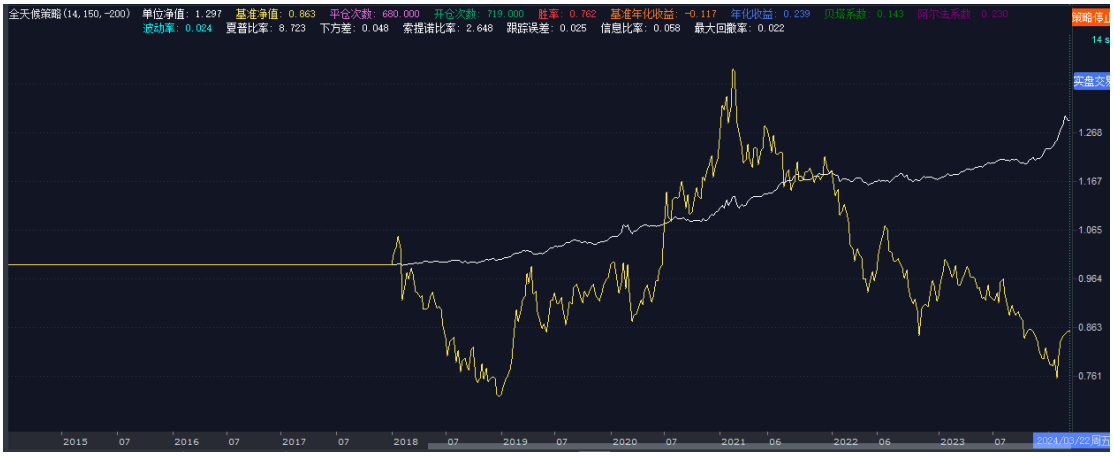
需要注意的是，没有策略能保证未来的表现，且过去的表现不代表未来的结果。投资者应在充分了解策略风险的基础上进行投资决策。

下面是交易收益汇总：

	市场	代码	名称	品种类型	行业	多空	累计盈亏	累计交易量	累计交易费用	累计交易额	效益权重	累计持仓天数	盈利排名	亏损排名	累计交易额排名
1	SH	510170	大宗商品ETF	ETF	其它	-	59461.671	771700	0.000	401871.159	30.94%	2905	1	8	2
2	SH	510300	沪深300ETF	ETF	其它	-	21328.209	68700	0.000	312880.570	11.10%	2905	4	5	4
3	SH	510500	中证500ETF	ETF	其它	-	-1764.804	56000	0.000	315897.179	-0.92%	2905	8	1	3
4	SH	511090	30年国债ETF	ETF	其它	-	304.663	5200	0.000	521639.900	0.16%	182	7	2	1
5	SH	511260	十年国债ETF	ETF	其它	-	1024.635	1000	0.000	102106.400	0.95%	2303	6	3	8
6	SH	513300	纳斯达克ETF	ETF	其它	-	14226.732	176600	0.000	188088.900	7.40%	1134	5	4	7
7	SH	513500	标普500ETF	ETF	其它	-	57886.815	336600	0.000	282415.713	30.12%	2905	2	7	5
8	SH	518880	黄金ETF	ETF	其它	-	35359.865	78500	0.000	221909.700	18.40%	2905	3	6	6

下面介绍基于风险平价模型的全天候策略：

首先是收益曲线(2018-2024)



比起固定比例的简单全天候策略，策略的曲线更加平滑。回撤更加小，投资者的投资体验更佳！

	市场	代码	名称	品种类型	行业	多空	累计盈亏	累计交易量	累计交易费用	累计交易额	效益权重	累计持仓天数	盈利排名	亏损排名	累计交易额排名
1	SH	510170	大宗商品ETF	ETF	其它	-	19704.645	265700	0.000	138775.230	13.08%	2265	4	5	8
2	SH	510300	沪深300ETF	ETF	其它	-	3215.099	55700	0.000	210325.326	2.13%	2265	8	1	6
3	SH	510500	中证500ETF	ETF	其它	-	5804.618	29200	0.000	158405.864	3.85%	2265	6	3	7
4	SH	511090	30年国债ETF	ETF	其它	-	5758.500	4900	0.000	500525.200	3.82%	193	7	2	3
5	SH	511260	十年国债ETF	ETF	其它	-	32916.189	5800	0.000	604408.000	21.84%	2265	3	6	1
6	SH	513300	纳斯达克ETF	ETF	其它	-	35135.565	389400	0.000	401566.700	23.32%	1229	1	8	4
7	SH	513500	标普500ETF	ETF	其它	-	14970.347	210600	0.000	224518.168	9.93%	2265	5	4	5
8	SH	518880	黄金ETF	ETF	其它	-	33186.739	151200	0.000	526818.000	22.02%	2265	2	7	2

投资的收益也更加稳健。

以下是核心代码解释：

```python
1 个用法 (1 个动态)
def calculate_risk_parity_weights(self, returns):
    # 计算协方差矩阵
    cov_matrix = returns.cov()

    # 目标函数：最小化组合的风险贡献度差异
    def risk_parity_objective(x):
        portfolio_volatility = np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))
        risk_contributions = x * np.dot(cov_matrix, x) / portfolio_volatility
        return np.std(risk_contributions)  # 返回风险贡献的标准差

    # 初始权重
    x_initial = np.array([1/len(returns.columns)] * len(returns.columns))

    # 约束条件：所有权重之和为1
    constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1.0})

    # 权重范围：在0和1之间，无杠杆和空头
    bounds = tuple((0, 1) for _ in range(len(returns.columns)))

    # 使用优化器寻找风险平价权重
    optimal_weights = minimize(
        risk_parity_objective, x_initial, method='SLSQP',
        bounds=bounds, constraints=constraints
    )

    return optimal_weights.x
```

这段代码定义了一个名为 calculate_risk_parity_weights 的方法，它用于计算一个投资组合中不同资产的风险平价权重。这个方法采用历史收益率数据作为输入，并通过优化算法找到一组权重，使得每个资产对投资组合整体风险的贡献尽可能相等。

1. cov_matrix = returns.cov(): 这行代码计算了资产收益率的协方差矩阵。协方差矩阵描述了资产收益率之间的相互关系，是风险分析的关键组成部分。

2. def risk_parity_objective(x): 这是一个内部定义的函数，称为目标函数。它接受一个参数 x，表示资产的权重数组。目标函数的目的是要最小化的量，这里是投资组合的风险贡献度之间的标准差。

portfolio_volatility: 计算整个投资组合的波动性（标准差）。

risk_contributions: 计算每个资产的风险贡献，即每个资产的权重乘以它们在协方差矩阵中的条目，再除以投资组合的总波动性。

np.std(risk_contributions): 计算风险贡献度之间的标准差，我们希望通过优化过程将这个值最小化。

3. x_initial: 设置初始权重，这里简单地给每个资产分配了相同的初始权重。

4. constraints: 定义了优化过程中的约束条件。在这种情况下，约束条件是所有权重的和必须等于 1（资产权重完全分配且无现金持有）。

5. bounds: 设置了权重的界限。在这个示例中，每个资产的权重被限制在 0 和 1 之间，意味着不允许使用杠杆（借贷资金投资）也不允许做空（卖出未持有的资产）。

6. optimal_weights = minimize(...): 使用 scipy.optimize.minimize 函数来执行优化。它试图找到一组权重，这组权重能够最小化目标函数 risk_parity_objective。使用的优化算法是 SLSQP（Sequential Least Squares Programming）。

7. return optimal_weights.x: 一旦找到最优解，就返回这组权重。

这个方法返回的权重可以用来构建一个风险平价投资组合，其中每个资产的风险贡献都大致相等。这样的投资组合被认为在风险管理方面具有良好的多元化特性。

全天候策略是一种优雅的投资方法。它以其稳健的构架，有效地分散了风险，提供了平稳的回报。低波动和低回撤是它的标志性特征，使得投资者能够在各种市场环境中保持从容。随着时间推移，全天候策略坚持不懈地追求长期增长，这就像是一个持续向上的曲线，为投资者带来了可靠的财富积累之路。简而言之，全天候策略是追求稳定和长期增长的投资者的明智选择。

文后提供完整的测试代码，有兴趣可以试试。大家对量化什么话题感兴趣，欢迎评论区给我们留言。学量化请关注“国信 iQuant 量化实验室”官方账号！

下面给出全部代码 ：

```
# encoding:gbk
import pandas as pd
import numpy as np
from scipy.optimize import minimize
import datetime
def init(ContextInfo):

    ContextInfo.etfs = {
        "510300.SH": 0,   # 沪深 300ETF
        "510500.SH": 1,   # 中证 500ETF
        "513500.SH": 2,   # 标普 500ETF
        "513300.SH": 3,   # 纳指 ETF
        "511260.SH": 4,   # 10 年期国债 ETF
        "511090.SH": 5,   # 30 年期国债 ETF
        "518880.SH": 6,   # 黄金 ETF
```

```
            "510170.SH": 7    # 大宗商品 ETF
        }



        ContextInfo.trade_code_list = list(ContextInfo.etfs.keys())
        # # 深交所用.SZ 结尾  上交所用.SH 结尾

        ContextInfo.set_universe(ContextInfo.trade_code_list)
        # ContextInfo.accID = str(account)
        ContextInfo.accID = 'tests'
        # # 回测时使用 test 账号

        ##创建对象
        global created_future
        created_future = []
        global futures
        futures = {}
        for item in ContextInfo.trade_code_list:
            # 使用字符串的倒数前三个字符作为对象名
            obj_name = item[:-3]
            print('\n', item)
            # 创建对象并存储为变量名
            if item not in created_future:
                vars()[obj_name] = Futures(ContextInfo)
                created_future.append(item)
                futures[obj_name] = vars()[obj_name]

    class Futures(ContextInfo):
        def __init__(self, ContextInfo):
            self.longhands = 0
            self.canuse = 0
            self.last_close = 0

        def treat_data(self, ContextInfo, code):
            self.num = ContextInfo.etfs[code]
            data1 = ContextInfo.get_market_data(['close'], [code], '20150101', '', True, '1w',
'none')['close'][-100:]
            self.last_close = data1[-1]
            self.close_array=data1



        def calculate_risk_parity_weights(self, returns):
```

```
        # 计算协方差矩阵
        cov_matrix = returns.cov()

        # 目标函数：最小化组合的风险贡献度差异
        def risk_parity_objective(x):
            portfolio_volatility = np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))
            risk_contributions = x * np.dot(cov_matrix, x) / portfolio_volatility
            return np.std(risk_contributions)   # 返回风险贡献的标准差

        # 初始权重
        x_initial = np.array([1/len(returns.columns)] * len(returns.columns))

        # 约束条件：所有权重之和为 1
        constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1.0})

        # 权重范围：在 0 和 1 之间，无杠杆和空头
        bounds = tuple((0, 1) for _ in range(len(returns.columns)))

        # 使用优化器寻找风险平价权重
        optimal_weights = minimize(
            risk_parity_objective, x_initial, method='SLSQP',
            bounds=bounds, constraints=constraints
        )

        return optimal_weights.x


def handlebar(ContextInfo):
    #if not ContextInfo.is_last_bar():
        #return

    ###创建对象 future
    global created_future
    global futures
    all_returns = []
    for code in ContextInfo.trade_code_list:
        # 使用字符串的倒数前三个字符作为对象名
        obj_name = code[:-3]
        print('\n', code)
        # 创建对象并存储为变量名

        ###调用对象
        future_bullet = futures[obj_name]
```

```
            future_bullet.treat_data(ContextInfo, code)


            all_returns.append(future_bullet.close_array.pct_change().dropna())

            # 将所有资产的收益率数据合并到一个 DataFrame 中
        returns_df = pd.concat(all_returns, axis=1)

        # 计算风险平价权重 只使用第一个对象计算权重向量
        future = futures[ContextInfo.trade_code_list[0][:-3]]
        risk_parity_weights = future.calculate_risk_parity_weights(returns_df)
        print(risk_parity_weights,sum(risk_parity_weights))
        for code in ContextInfo.trade_code_list:
            # 使用字符串的倒数前三个字符作为对象名
            obj_name = code[:-3]
            print('\n', code)
            # 创建对象并存储为变量名

            ###调用对象
            future_bullet = futures[obj_name]
            future_bullet.pos_percent = risk_parity_weights[future_bullet.num]

            order_target_percent(code, future_bullet.pos_percent, 'COMPETE', ContextInfo,
ContextInfo.accID)




    def current_bartime(ContextInfo):
        index = ContextInfo.barpos
        timetag = ContextInfo.get_bar_timetag(index)
        print(timetag_to_datetime(timetag, '%Y%m%d %H:%M:%S'))
        print(timetag_to_datetime(timetag, '%H:%M:%S'))
        current = timetag_to_datetime(timetag, '%H:%M:%S')

        return current
```