A STEP-BY-STEP GUIDE TO
QUANTITATIVE STRATEGIES

SUCCESSFUL ALGORITHMIC TRADING

Applying the scientific method
for profitable trading results.

By Michael L. Halls-Moore

Contenido

Límite de responsabilidad/Renuncia de garantía

Si bien el autor se ha esforzado al máximo en la preparación de este libro, no se responsabiliza ni garantiza con respecto a la precisión o integridad del contenido de este libro y renuncia específicamente a cualquier garantía implícita de comerciabilidad o idoneidad para un propósito particular. Se vende en el entendimiento de que el autor no se dedica a la prestación de servicios profesionales y el autor no será responsable de los daños derivados de la misma. Si se requiere asesoramiento profesional u otra asistencia de expertos, se deben buscar los servicios de un profesional competente.

yo

Parte I

Introducción al comercio algorítmico

Capítulo 1

Introducción al Libro

1.1 Introducción a QuantStart

QuantStart fue fundada por Michael Halls-Moore, en 2010, para ayudar a los analistas cuantitativos (QA) junior a encontrar trabajo en el difícil clima económico. Desde entonces, el sitio ha evolucionado hasta convertirse en un recurso sustancial para las finanzas cuantitativas. El sitio ahora se concentra en el comercio algorítmico, pero también analiza el desarrollo cuantitativo, tanto en Python como en C++.

Desde marzo de 2010, QuantStart ha ayudado a más de 200 000 visitantes a mejorar sus habilidades financieras cuantitativas. Siempre puede ponerse en contacto con QuantStart enviando un correo electrónico a mike@quantst art.com.

1.2 ¿Qué es este Libro?

*Comercio algorítmico exitoso*ha sido escrito para enseñar a comerciantes minoristas discrecionales y profesionales de comercio, con habilidades básicas de programación, cómo crear completamente automatizado*rentable*y*robusto* Sistemas de comercio algorítmicos que utilizan el lenguaje de programación Python. El libro describe la naturaleza de un sistema de comercio algorítmico, cómo obtener y organizar datos financieros, el concepto de backtesting y cómo implementar un sistema de ejecución. El libro está diseñado para ser extremadamente práctico, con ejemplos liberales de código Python a lo largo del libro para demostrar los principios y la práctica del comercio algorítmico.

1.3 ¿Para quién es este libro?

Este libro ha sido escrito tanto para comerciantes minoristas como para profesionales cuantitativos que tienen una exposición básica a la programación y desean aprender a aplicar lenguajes y bibliotecas modernos al comercio algorítmico. Está diseñado para aquellos que disfrutan del autoaprendizaje y pueden aprender con el ejemplo. El libro está dirigido a personas interesadas en la programación e implementación reales, ya que creo que el éxito real en el comercio algorítmico proviene de comprender completamente los detalles de implementación.

Los comerciantes cuantitativos profesionales también encontrarán útil el contenido. La exposición a nuevas bibliotecas y métodos de implementación puede conducir a una ejecución más óptima o a pruebas retrospectivas más precisas.

1.4 ¿Cuáles son los requisitos previos?

El libro es relativamente independiente, pero asume una familiaridad con los conceptos básicos del comercio en un entorno discrecional. El libro no requiere una gran experiencia en programación, pero se asume una familiaridad básica con un lenguaje de programación. Debe conocer los conceptos elementales de programación, como la declaración de variables, el control de flujo (if-else) y los bucles (for/while).

Algunas de las estrategias comerciales hacen uso de técnicas estadísticas de aprendizaje automático. Además, las secciones de optimización de cartera/estrategia hacen un uso extensivo de la búsqueda y optimización.

algoritmos Si bien una comprensión profunda de las matemáticas no es absolutamente necesaria, facilitará la comprensión de cómo funcionan estos algoritmos a nivel conceptual.

Si está oxidado con este material, o es nuevo para usted, eche un vistazo a la lectura de QuantStart lista.

1.5 Requisitos de software/hardware

Las aplicaciones comerciales cuantitativas en Python se pueden desarrollar en Windows, Mac OSX o Linux. Este libro es independiente del sistema operativo, por lo que es mejor usar cualquier sistema con el que se sienta cómodo. Sin embargo, recomiendo Mac OSX o Linux (uso Ubuntu), ya que he descubierto que la instalación y la gestión de datos son mucho más sencillas.

Para escribir programas en Python, simplemente necesita acceso a un editor de texto (preferiblemente con resaltado de sintaxis). En Windows suelo usar Notepad++. En Mac OSX utilizo SublimeText. En Ubuntu tiendo a usar emacs, pero por supuesto, puedes usar vim.

El código de este libro se ejecutará bajo la versión 2.7.x de Python (específicamente 2.7.6 en mi máquina Ubuntu 14.04) y Python 3.4.x (específicamente 3.4.0 en mi máquina Ubuntu 14.04).

En términos de hardware, probablemente querrá al menos 1 GB de RAM, pero más siempre es mejor. También querrá usar una CPU relativamente nueva y mucho espacio de almacenamiento en el disco duro para datos históricos, dependiendo de la frecuencia con la que pretenda operar. Un disco duro de 200 Gb debería ser suficiente para datos más pequeños, mientras que 1 TB es útil para un amplio universo de símbolos de datos de ticks.

1.6 Estructura del libro

El libro está diseñado para crear un conjunto de estrategias comerciales algorítmicas desde la idea hasta la ejecución automatizada. El proceso seguido se describe a continuación.

- ¿Por qué el comercio algorítmico? -Se discuten los beneficios de utilizar un enfoque sistemático/algorítmico para el comercio en lugar de una metodología discrecional. Además, se muestran los diferentes enfoques adoptados para el comercio algorítmico.
- Desarrollo del sistema comercial -Se cubre el proceso de desarrollo de un sistema de comercio algorítmico, desde la hipótesis hasta el comercio en vivo y la evaluación continua.
- Diseño del sistema de negociación -Los componentes reales que forman un sistema de comercio algorítmico están cubiertos. En particular, generación de señales, gestión de riesgos, medición del rendimiento, dimensionamiento/apalancamiento de posiciones, optimización y ejecución de carteras.
- Entorno del sistema comercial -Se lleva a cabo el procedimiento de instalación de todo el software Python y se obtienen, limpian y almacenan los datos históricos en un sistema de base de datos local.
- Análisis de series temporales -Se utilizan varios métodos de series de tiempo para la predicción, la reversión a la media, el impulso y la identificación de la volatilidad. Estos métodos estadísticos luego forman la base de las estrategias comerciales.
- Optimización -Se analizan los algoritmos de optimización/búsqueda y se consideran ejemplos de cómo se aplican a la optimización de estrategias.
- Medición del desempeño -Las implementaciones para varias medidas de riesgo/recompensa y otras métricas de rendimiento se describen en detalle.
- Gestión de riesgos -Se describen varias fuentes de riesgo que afectan a un sistema de comercio algorítmico y se proporcionan métodos para mitigar este riesgo.
- Implementación de la estrategia comercial -Se proporcionan ejemplos de estrategias comerciales basadas en medidas estadísticas e indicadores técnicos, junto con detalles sobre cómo optimizar una cartera de dichas estrategias.
- Ejecución -Se analizan la conexión a un corretaje, la creación de una infraestructura comercial automatizada basada en eventos y las herramientas de monitoreo/resiliencia.

1.7 Lo que el libro no cubre

Este no es un libro para principiantes sobre el comercio discrecional, ni un libro lleno de estrategias comerciales de "análisis técnico". Si no ha realizado ninguna negociación (discrecional o de otro tipo), le sugiero que lea algunos de los libros de la lista de lectura de QuantStart.

Tampoco es un libro tutorial de Python, aunque una vez más se puede consultar la lista de lectura de QuantStart. Si bien se ha hecho todo lo posible para presentar el código de Python como lo justifica cada ejemplo, una cierta familiaridad con Python será extremadamente útil.

1.8 Dónde obtener ayuda

El mejor lugar para buscar ayuda es la lista de artículos en QuantStart.com, que se encuentra en QuantStart.com/articles o poniéndose en contacto conmigo en mike@quantstart.com. He escrito más de 140 artículos sobre finanzas cuantitativas (y comercio algorítmico en particular), por lo que puede refrescarse leyendo algunos de estos.

También quiero agradecerles por comprar el libro y ayudarme a apoyarme mientras escribo más contenido; se lo agradezco mucho. ¡Buena suerte con tus estrategias algorítmicas! Ahora en algo de comercio...

Capitulo 2

¿Qué es el comercio algorítmico?

comercio algorítmico,tal como se define aquí, es el uso de un sistema automatizado para realizar transacciones, que se ejecutan de manera predeterminada a través de un algoritmo específicamente*sin ninguna intervención humana*. Este último énfasis es importante. Las estrategias algorítmicas se diseñan antes del comienzo de la negociación y se ejecutan sin*discrecional*aportes de comerciantes humanos.

En este libro, “comercio algorítmico” se refiere a la práctica minorista de comercio automatizado, sistemático y cuantitativo, que se tratarán como sinónimos a los efectos de este texto. En la industria financiera, el "comercio algorítmico" generalmente se refiere a una clase de algoritmos de ejecución (como el precio promedio ponderado por volumen, VWAP) que se utilizan para optimizar los costos de órdenes comerciales más grandes.

2.1 Resumen

El comercio algorítmico difiere sustancialmente del comercio discrecional. En esta sección se describen las ventajas y desventajas de un*sistemático*se esbozará el enfoque.

2.1.1 Ventajas

El comercio algorítmico posee numerosas ventajas sobre los métodos discrecionales.

Valoración Histórica

La ventaja más importante de crear una estrategia automatizada es que su rendimiento se puede determinar en los datos históricos del mercado, que (con suerte) son representativos de los datos futuros del mercado. Este proceso se conoce comobacktestingy será discutido en profundidad significativa dentro de este libro. El backtesting permite determinar las propiedades estadísticas (anteriores) de la estrategia, lo que proporciona una idea de si es probable que una estrategia sea rentable en el futuro.

Eficiencia

El comercio algorítmico es sustancialmente más eficiente que un enfoque discrecional. Con un sistema completamente automatizado, no hay necesidad de que un individuo o equipo esté constantemente monitoreando los mercados para la acción del precio o la entrada de noticias. Esto libera tiempo para que los desarrolladores de la estrategia comercial lleven a cabo más investigaciones y, por lo tanto, dependiendo de las limitaciones de capital, implementen más estrategias en una cartera.

Además, al automatizar el proceso de gestión de riesgos y dimensionamiento de posiciones, al considerar un conjunto de estrategias sistemáticas, es necesario ajustar automáticamente el apalancamiento y los factores de riesgo de forma dinámica, respondiendo directamente a la dinámica del mercado en tiempo real. Esto no es posible en un mundo discrecional, ya que un comerciante no puede calcular el riesgo de forma continua y debe tomar descansos ocasionales del control del mercado.

Sin entrada discrecional

Una de las principales ventajas de un sistema de comercio automatizado es que (teóricamente) no hay entrada discrecional posterior. Esto se refiere a la modificación de operaciones en el punto de ejecución o mientras se está en una posición. El miedo y la codicia pueden ser motivadores abrumadores cuando se lleva a negociación discrecional. cabo En el contexto del comercio sistemático, es raro que la entrada discrecional mejora el rendimiento de una estrategia.

Dicho esto, ciertamente es posible que las estrategias sistemáticas dejen de ser rentables debido a cambios de régimen u otros factores externos. En este caso, se requiere criterio para modificar los parámetros de la estrategia o para retirarla. Tenga en cuenta que este proceso aún no interfiere con las transacciones individuales.

Comparación

Las estrategias sistemáticas proporcionan información estadística sobre el desempeño histórico y actual. En particular, es posible determinar el crecimiento de la equidad, el riesgo (en varias formas), la frecuencia de negociación y una miríada de otras métricas. Esto permite una comparación de "manzanas con manzanas" entre varias estrategias, de modo que el capital se pueda asignar de manera óptima. Esto contrasta con el caso en el que solo se realiza un seguimiento de la información de pérdidas y ganancias (P&L) en un entorno discrecional, ya que enmascara el riesgo potencial de reducción.

Frecuencias más altas

Este es un corolario de la ventaja de eficiencia discutida anteriormente. Las estrategias que operan a frecuencias más altas en muchos mercados se vuelven posibles en un entorno automatizado. De hecho, algunas de las estrategias comerciales más rentables operan en el dominio de frecuencia ultra alta en los datos del libro de órdenes limitadas. Estas estrategias son simplemente imposibles de llevar a cabo para un ser humano.

2.1.2 Desventajas

Si bien las ventajas del comercio algorítmico son numerosas, existen algunas desventajas.

Requisitos de capital

El comercio algorítmico generalmente requiere una base de capital mucho mayor que la que se utilizaría para el comercio minorista discrecional, esto se debe simplemente al hecho de que hay pocos corredores que admiten la ejecución automatizada de operaciones que no requieren grandes mínimos de cuenta. La agencia de corretaje más prolífica en el espacio minorista automatizado es Interactive Brokers, que requiere un saldo de cuenta de $10,000. La situación está cambiando lentamente, especialmente porque otras casas de bolsa están permitiendo la conexión directa a través del protocolo FIX. Además, los requisitos de Pattern Day Trader definidos por la Comisión de Bolsa y Valores requieren que se mantenga un mínimo de $25,000 en capital de cuenta en todo momento, en ciertas situaciones de margen. Estos temas serán discutidos extensamente en la sección sobre Ejecución.

Además, obtener feeds de datos para estrategias cuantitativas intradía, especialmente si se utilizan contratos de futuros, no es barato para el comerciante minorista. Los feeds intradía minoristas comunes a menudo tienen un precio en el rango de $ 300- $ 500 por mes, con feeds comerciales un orden de magnitud más allá de eso. Dependiendo de sus necesidades de latencia, puede ser necesario ubicar un servidor en un intercambio, lo que aumenta los costos mensuales. Para el comerciante minorista interdiario esto no es necesariamente un problema, pero vale la pena considerarlo. También hay accesorios como una conexión a Internet más robusta y máquinas de escritorio potentes (y, por lo tanto, caras) para comprar.

Programación/Experiencia científica

Si bien existen ciertas plataformas comerciales sistemáticas, como Quantopian, QuantConnect y TradeStation, que alivian la mayor parte de la dificultad de programación, algunas aún (al momento de escribir este artículo) no admiten la ejecución en vivo. TradeStation es claramente una excepción en este caso. Por lo tanto, es un requisito que el comerciante algorítmico sea relativamente competente tanto en programación como en modelado científico.

He intentado demostrar una amplia variedad de estrategias, cuya base casi siempre se basa en una forma fácil de entender. Sin embargo, si posee habilidades de modelado numérico, es probable que le resulte más fácil utilizar los métodos estadísticos de series de tiempo presentes en la sección Modelado. La mayoría de las técnicas demostradas ya se han implementado en bibliotecas externas de Python, lo que nos ahorra una cantidad sustancial de trabajo de desarrollo. Por lo tanto, estamos "reducidos" a vincular nuestras bibliotecas de análisis y ejecución de datos para producir un sistema de comercio algorítmico.

2.2 Método científico

El diseño de estrategias comerciales dentro de este libro se basa únicamente en los principios de la *método científico*. El proceso del método científico comienza con la formulación de unpregunta,basado en observaciones. En el contexto de la negociación, un ejemplo sería "¿Existe una relación entre el ETF SPDR Gold Shares (GLD) y el ETF Market Vectors Gold Miners (GDX)?". Esto permite unhipótesisformarse que pueda explicar el comportamiento observado. En este caso, una hipótesis puede ser "¿Tiene el diferencial entre GLD y GDX un comportamiento de reversión a la media?". los*hipótesis nula*es que no hay un comportamiento de reversión a la media, es decir, el diferencial de precios es un*Caminata aleatoria*.

Después de la formulación de una hipótesis, depende del científico refutar la hipótesis nula y demostrar que, de hecho, existe un comportamiento de reversión a la media. Para llevar a cabo esto un predicción debe ser definido. Volviendo al ejemplo de GLD-GDX, la predicción es que la serie temporal que representa el diferencial de los dos ETF es*estacionario*. Para probar o refutar la hipótesis, la predicción está sujeta apruebas.En el caso de GLD-GDX, esto significa aplicar pruebas de estacionariedad estadística como las pruebas Dickey-Fuller aumentada, exponente de Hurst y relación de varianza (descritas en detalle en capítulos posteriores).

Los resultados del procedimiento de prueba proporcionarán una*estadístico*responder sobre si la hipótesis nula puede ser rechazada en un cierto nivel de confianza. Si no se puede rechazar la hipótesis nula, lo que implica que no hubo una relación perceptible entre los dos ETF, aún es posible que la hipótesis sea (parcialmente) cierta. También se puede probar un conjunto más grande de datos, la incorporación de información adicional (como un tercer ETF que afecta el precio). Este es el proceso de análisis.A menudo conduce al rechazo de la hipótesis nula, después del refinamiento.

La principal ventaja de usar el método científico para el diseño de la estrategia comercial es que si la estrategia "se rompe" después de un período anterior de rentabilidad, es posible revisar la hipótesis inicial y volver a evaluarla, lo que podría conducir a una nueva hipótesis que conduce a para recuperar la rentabilidad de una estrategia.

Esto está en contraste directo con el*procesamiento de datos*o*caja negra*enfoque en el que se aplica una gran cantidad de parámetros o "indicadores" a una serie temporal. Si tal "estrategia" es inicialmente rentable y luego el rendimiento se deteriora, es difícil (si no imposible) determinar por qué. A menudo conduce a la aplicación arbitraria de nueva información, indicadores o parámetros que pueden conducir temporalmente a la rentabilidad, pero que posteriormente conducen a una mayor degradación del rendimiento. En este caso, la estrategia generalmente se descarta y el proceso de "investigación" continúa nuevamente.

En este libro, todas las estrategias comerciales se desarrollarán con un enfoque de hipótesis de observación.

2.3 ¿Por qué Python?

Las secciones anteriores han descrito los beneficios del comercio algorítmico y el método científico. Ahora es el momento de centrar la atención en el lenguaje de implementación de nuestros sistemas comerciales. Para este libro he elegido Python. Python es un lenguaje de alto nivel diseñado para acelerar el desarrollo. Posee una amplia gama de bibliotecas para casi cualquier tarea computacional imaginable. También está ganando una adopción más amplia en las comunidades de gestión de activos y bancos de inversión.

Estas son las razones por las que he elegido Python como lenguaje para la investigación e implementación de sistemas comerciales:

- Aprendizaje -Python es extremadamente fácil de aprender en comparación con otros lenguajes como C++. Puede ser extremadamente productivo en Python después de solo unas pocas semanas (¡algunos dicen días!) de

uso.

- bibliotecas -La razón principal para usar Python es que viene con una asombrosa variedad de bibliotecas, lo que reduce significativamente el tiempo de implementación y la posibilidad de introducir errores en nuestro código. En particular, haremos uso de NumPy (operaciones vectorizadas), SciPy (algoritmos de optimización), pandas (análisis de series temporales), statsmodel (modelado estadístico), scikit-learn (aprendizaje estadístico/automático), IPython (desarrollo interactivo) y matplotlib. (visualización).

- Velocidad de desarrollo -Python sobresale en la velocidad de desarrollo hasta el punto de que algunos han comentado que es como escribir en “pseudocódigo”. La naturaleza interactiva de herramientas como IPython hace que la investigación de estrategias sea extremadamente rápida, sin sacrificar la solidez.

- Velocidad de ejecución -Si bien no es tan rápido como C ++, Python proporciona componentes de computación científica que están muy optimizados (a través de la vectorización). Si la velocidad de ejecución se convierte en un problema, se puede utilizar Cython y obtener velocidades de ejecución similares a C, para un pequeño aumento en la complejidad del código.

- Ejecución comercial -Existen complementos de Python para corredores más grandes, como Interactive Brokers (IBypy). Además, Python puede hacer uso fácilmente del protocolo FIX cuando sea necesario.

- Costo/Licencia -Python es gratuito, de código abierto y multiplataforma. Funcionará felizmente en Windows, Mac OSX o Linux.

Si bien Python es extremadamente aplicable a casi todas las formas de comercio algorítmico, no puede competir con C (o lenguajes de nivel inferior) en el ámbito del comercio de frecuencia ultra alta (UHFT). Sin embargo, ¡este tipo de estrategias están fuera del alcance de este libro!

2.4 ¿Pueden seguir compitiendo los comerciantes minoristas?

Es común, como un operador algorítmico principiante que practica a nivel minorista,pregunta si todavía es posible competircon los grandes fondos cuantitativos institucionales. En esta sección se argumentará que debido a la naturaleza del entorno normativo institucional, la estructura organizativa y la necesidad de mantener las relaciones con los inversores, los fondos sufren ciertas desventajas que no afectan a los operadores algorítmicos minoristas.

Las restricciones regulatorias y de capital impuestas a los fondos conducen a ciertos comportamientos predecibles, que pueden ser explotados por un comerciante minorista. El "dinero grande" mueve los mercados y, como tal, se pueden idear muchas estrategias para aprovechar dichos movimientos. Algunas de estas estrategias se discutirán en capítulos posteriores. A continuación, se describirán las ventajas comparativas que disfruta el comerciante algorítmico sobre muchos fondos más grandes.

2.4.1 Ventajas comerciales

Hay muchas formas en las que un comerciante minorista de algo puede competir con un fondo solo en su proceso de negociación, pero también hay algunas desventajas:

- Capacidad -Un comerciante minorista tiene mayor libertad para jugar en mercados más pequeños. Pueden generar retornos significativos en estos espacios, incluso cuando los fondos institucionales no pueden.

- Abarrotando el comercio -Los fondos sufren de "transferencia de tecnología", ya que la rotación de personal puede ser alta. Los Acuerdos de no divulgación y los Acuerdos de no competencia mitigan el problema, pero aún conduce a que muchos fondos cuantitativos "persigan el mismo comercio". El sentimiento caprichoso de los inversores y la "próxima novedad" exacerban el problema. Los comerciantes minoristas no están obligados a seguir las mismas estrategias y, por lo tanto, pueden permanecer sin correlación con los fondos más grandes.

- Impacto en el mercado -Cuando se juega en mercados no extrabursátiles de gran liquidez, la baja base de capital de las cuentas minoristas reduce sustancialmente el impacto en el mercado.

- Aprovechar -Un comerciante minorista, dependiendo de su configuración legal, está limitado por las regulaciones de margen/apalancamiento. Los fondos de inversión privados no sufren la misma desventaja, aunque están igualmente limitados desde la perspectiva de la gestión de riesgos.

- Liquidez -Tener acceso a una agencia de corretaje principal está fuera del alcance del comerciante de algoritmos minorista promedio. Tienen que "arreglárselas" con una agencia de corretaje minorista como Interactive Brokers. Por lo tanto, hay un acceso reducido a la liquidez en ciertos instrumentos. El enrutamiento de órdenes comerciales también es menos claro y es una forma en que el rendimiento de la estrategia puede diferir de las pruebas retrospectivas.

- Flujo de noticias del cliente -Potencialmente, la desventaja más importante para el comerciante minorista es la falta de acceso al flujo de noticias de los clientes de su principal institución de corretaje o proveedora de crédito. Los comerciantes minoristas tienen que hacer uso de fuentes no tradicionales, como grupos de encuentro, blogs, foros y revistas financieras de acceso abierto.

2.4.2 Gestión de riesgos

Los comerciantes minoristas de algoritmos a menudo adoptan un enfoque diferente a la gestión de riesgos que los fondos cuantitativos más grandes. A menudo es ventajoso ser "pequeño y ágil" en el contexto del riesgo.

Fundamentalmente, no hay un presupuesto de gestión de riesgos impuesto al comerciante más allá del que ellos mismos se imponen, ni existe un departamento de cumplimiento o gestión de riesgos que haga cumplir la supervisión. Esto permite que el comerciante minorista implemente metodologías de modelado de riesgo personalizadas o preferidas, sin la necesidad de seguir los "estándares de la industria" (un requisito implícito del inversionista).

Sin embargo, el argumento alternativo es que esta flexibilidad puede llevar a los comerciantes minoristas a volverse "descuidados" con la gestión de riesgos. Las preocupaciones sobre el riesgo pueden incorporarse al backtest y al proceso de ejecución, sin que se le dé una consideración externa al riesgo de la cartera en su conjunto. Aunque el "pensamiento profundo" podría aplicarse al modelo alfa (estrategia), la gestión de riesgos podría no alcanzar un nivel similar de consideración.

2.4.3 Relaciones con inversores

Los inversores externos son la diferencia clave entre las tiendas minoristas y los grandes fondos. Esto impulsa todo tipo de incentivos para el fondo más grande, cuestiones con las que el comerciante minorista no debe preocuparse:

- Estructura de compensación -En el entorno minorista, el comerciante solo se preocupa por el rendimiento absoluto. No hay puntos altos que cumplir ni reglas de despliegue de capital que seguir. Los comerciantes minoristas también pueden sufrir curvas de acciones más volátiles, ya que nadie está observando su desempeño que podría ser capaz de redimir el capital de su fondo.

- Regulaciones e informes -Más allá de los impuestos, hay poco en el camino de las restricciones de informes regulatorios para el comerciante minorista. Además, no hay necesidad de proporcionar informes de rendimiento mensuales o "disfrazar" una cartera antes de que se envíe un boletín de noticias al cliente. Este es un gran ahorro de tiempo.

- Comparación de puntos de referencia -Los fondos no solo se comparan con sus pares, sino también con "puntos de referencia de la industria". Para un fondo de renta variable de EE. UU. solo largo, los inversores querrán ver rendimientos superiores al S&P500, por ejemplo. Los comerciantes minoristas no están obligados de la misma manera a comparar sus estrategias con un punto de referencia.

- Comisiones de rendimiento -La desventaja de administrar su propia cartera como comerciante minorista es la falta de tarifas de gestión y rendimiento que disfrutan los fondos cuantitativos exitosos. ¡No hay "2 y 20" disponibles a nivel minorista!

2.4.4 Tecnología

Un área en la que el comerciante minorista tiene una ventaja significativa es en la elección de la pila de tecnología para el sistema de comercio. El comerciante no solo puede elegir las "mejores herramientas para el trabajo" como mejor le parezca, sino que no hay preocupaciones sobre la integración de sistemas heredados o las políticas de TI de toda la empresa. Más nuevo

Los lenguajes como Python o R ahora poseen paquetes para construir un sistema integral de backtesting, ejecución, riesgo y gestión de cartera con muchas menos líneas de código (LOC) de las que pueden ser necesarias en un lenguaje más detallado como C++.

Sin embargo, esta flexibilidad tiene un precio. Uno tiene que construir la pila ellos mismos o subcontratar todo o parte de ella a los proveedores. Esto es costoso en términos de tiempo, capital o ambos. Además, un comerciante debe depurar todos los aspectos del sistema comercial, un proceso largo y potencialmente laborioso. Todas las máquinas de investigación de escritorio y cualquier servidor ubicado en el mismo lugar deben pagarse directamente con las ganancias comerciales, ya que no hay tarifas de administración para cubrir los gastos.

En conclusión, se puede ver que los comerciantes minoristas poseen ventajas comparativas significativas sobre los fondos cuantitativos más grandes. Potencialmente, hay muchas maneras en las que se pueden explotar estas ventajas. Los capítulos posteriores discutirán algunas estrategias que hacen uso de estas diferencias.

Parte II

Sistemas de comercio

Capítulo 3

Backtesting exitoso

El backtesting algorítmico requiere conocimiento de muchas áreas, incluidas la psicología, las matemáticas, las estadísticas, el desarrollo de software y la microestructura del mercado/intercambio. No podía esperar cubrir todos esos temas en un capítulo, así que los dividiré en dos o tres partes más pequeñas. ¿De qué hablaremos en esta sección? Comenzaré definiendo backtesting y luego describiré los conceptos básicos de cómo se lleva a cabo. Luego aclararé los sesgos que mencionamos en capítulos anteriores.

En los capítulos siguientes veremos los detalles de las implementaciones de estrategias que a menudo apenas se mencionan o se ignoran en otros lugares. También consideraremos cómo hacer que el proceso de backtesting sea más realista al incluir las idiosincrasias de un*intercambio comercial*. Luego, analizaremos los costos de transacción y cómo modelarlos correctamente en un entorno de prueba retrospectiva. Terminaremos con una discusión sobre la *actuación*de nuestros backtests y, finalmente, proporciona ejemplos detallados de estrategias cuantitativas comunes.

Comencemos discutiendo qué es el backtesting y por qué deberíamos llevarlo a cabo en nuestro trading algorítmico.

3.1 ¿Por qué estrategias de backtest?

El comercio algorítmico se distingue de otros tipos de clases de inversión porque podemos proporcionar expectativas sobre el rendimiento futuro de forma más fiable a partir del rendimiento pasado, como consecuencia de la abundante disponibilidad de datos. El proceso por el cual esto se lleva a cabo se conoce comobacktesting

En términos simples, el backtesting se lleva a cabo al exponer su algoritmo de estrategia particular a un flujo de datos financieros históricos, lo que conduce a un conjunto deseñales comerciales.Cada*comercio*(lo que aquí queremos decir que es un "viaje de ida y vuelta" de dos señales) tendrá una ganancia o pérdida asociada. La acumulación de esta ganancia/pérdida durante la prueba retrospectiva de su estrategia conducirá a la ganancia y pérdida total (también conocida como 'P & L' o 'PnL'). Esa es la esencia de la idea, aunque, por supuesto, ¡el “diablo siempre está en los detalles”!

¿Cuáles son las razones clave para realizar un backtesting de una estrategia algorítmica?

- Filtración -Si recuerda del capítulo anterior sobre Identificación de estrategias, nuestro objetivo en la etapa inicial de investigación era establecer un flujo de estrategias y luego filtrar cualquier estrategia que no cumpliera con ciertos criterios. El backtesting nos proporciona otro mecanismo de filtración, ya que podemos eliminar estrategias que no satisfacen nuestras necesidades de rendimiento.

- Modelado -El backtesting nos permite (¡de forma segura!) probar nuevos modelos de determinados fenómenos del mercado, como los costes de transacción, el enrutamiento de órdenes, la latencia, la liquidez u otros*microestructura del mercado* problemas.

- Optimización -A pesar de que*optimización de la estrategia*está plagado de sesgos, el backtesting nos permite aumentar el rendimiento de una estrategia modificando la cantidad o los valores de los parámetros asociados a esa estrategia y recalculando su rendimiento.

- Verificación -Nuestras estrategias a menudo se obtienen de forma externa, a través de nuestro *tubería de estrategia*. El backtesting de una estrategia asegura que no haya sido implementada incorrectamente. Aunque rara vez tendremos acceso a las señales generadas por estrategias externas, a menudo tendremos acceso a las métricas de rendimiento como el Sharpe Ratio y las características de Drawdown. Así podemos compararlos con nuestra propia implementación.

Backtesting proporciona una serie de ventajas para el comercio algorítmico. Sin embargo, no siempre es posible realizar una prueba retrospectiva directa de una estrategia. En general, a medida que aumenta la frecuencia de la estrategia, se vuelve más difícil modelar correctamente los efectos de microestructura del mercado y los intercambios. Esto conduce a pruebas retrospectivas menos confiables y, por lo tanto, a una evaluación más complicada de una estrategia elegida. Este es un problema particular donde el sistema de ejecución es la clave para el desempeño de la estrategia, como ocurre con los algoritmos de ultra alta frecuencia.

Desafortunadamente, el backtesting está plagado de sesgos de todo tipo y ahora los discutiremos en profundidad.

3.2 Sesgos de backtesting

Hay muchos sesgos que pueden afectar el rendimiento de una estrategia probada. Desafortunadamente, estos sesgos tienden a inflar el rendimiento en lugar de restarle valor. Por lo tanto, siempre debe considerar un backtest como un límite superior idealizado en el rendimiento real de la estrategia. Es casi imposible eliminar los sesgos del comercio algorítmico, por lo que es nuestro trabajo minimizarlos lo mejor que podamos para tomar decisiones informadas sobre nuestras estrategias algorítmicas.

Hay cuatro sesgos principales que deseo discutir: *Sesgo de optimización*, *Sesgo de anticipación*, *Sesgo de supervivencia* y *Sesgo cognitivo*.

3.2.1 Sesgo de optimización

Este es probablemente el más insidioso de todos los sesgos de backtesting. Implica ajustar o introducir parámetros comerciales adicionales hasta que el rendimiento de la estrategia en el conjunto de datos de backtest sea muy atractivo. Sin embargo, una vez en vivo, el rendimiento de la estrategia puede ser notablemente diferente. Otro nombre para este sesgo es "ajuste de curvas" o "sesgo de espionaje de datos".

El sesgo de optimización es difícil de eliminar ya que las estrategias algorítmicas a menudo involucran muchos parámetros. Los "parámetros" en este caso podrían ser los criterios de entrada/salida, los períodos retrospectivos, los períodos de promediación (es decir, el parámetro de suavizado del promedio móvil) o la frecuencia de medición de la volatilidad. El sesgo de optimización se puede minimizar manteniendo el número de parámetros al mínimo y aumentando la cantidad de puntos de datos en el conjunto de entrenamiento. De hecho, también hay que tener cuidado con estos últimos ya que los puntos de entrenamiento más antiguos pueden estar sujetos a un *régimen* (como un entorno regulatorio) y, por lo tanto, puede no ser relevante para su estrategia actual.

Un método para ayudar a mitigar este sesgo es realizar una *análisis de sensibilidad*. Esto significa variar los parámetros de forma incremental y trazar una "superficie" de rendimiento. Un razonamiento sólido y fundamental para la elección de parámetros debería, con todos los demás factores considerados, conducir a una superficie de parámetro más uniforme. Si tiene una superficie de rendimiento muy irregular, a menudo significa que un parámetro no refleja un fenómeno y es un artefacto de los datos de prueba. Existe una vasta literatura sobre algoritmos de optimización multidimensional y es un área de investigación muy activa. No me detendré en eso aquí, ¡pero manténgalo en mente cuando encuentre una estrategia con un backtest fantástico!

3.2.2 Sesgo de anticipación

El sesgo de anticipación se introduce en un sistema de backtesting cuando los datos futuros se incluyen accidentalmente en un punto de la simulación en el que esos datos no habrían estado realmente disponibles. Si estamos ejecutando el backtest cronológicamente y llegamos al punto de tiempo $norte$, entonces se produce un sesgo de anticipación si se incluyen datos para cualquier punto $norte+k$, dónde $k > 0$. Los errores de sesgo anticipado pueden ser increíblemente sutiles. Aquí hay tres ejemplos de cómo se puede introducir el sesgo de anticipación:

- Errores técnicos -Las matrices/vectores en el código a menudo tienen iteradores o variables de índice. Incorrecto *compensaciones*de estos índices puede dar lugar a un sesgo prospectivo al incorporar datos en*norte*+*k*para distinto de cero*k*.

- Cálculo de parámetros -Otro ejemplo común de sesgo de anticipación ocurre cuando se calculan parámetros de estrategia óptimos, como con regresiones lineales entre dos series de tiempo. Si se utiliza todo el conjunto de datos (incluidos los datos futuros) para calcular los coeficientes de regresión y, por lo tanto, se aplica retroactivamente a una estrategia comercial con fines de optimización, entonces se están incorporando datos futuros y existe un sesgo de anticipación.

- Máximos/Mínimos -Ciertas estrategias comerciales hacen uso de valores extremos en cualquier período de tiempo, como incorporar los precios altos o bajos en los datos de OHLC. Sin embargo, dado que estos valores máximos/mínimos solo se pueden calcular al final de un período de tiempo, se introduce un sesgo de anticipación si estos valores se utilizan durante el período actual. Siempre es necesario retrasar los valores altos/bajos al menos un período en cualquier estrategia comercial que los utilice.

Al igual que con el sesgo de optimización, se debe tener mucho cuidado para evitar su introducción. A menudo, es la razón principal por la que las estrategias comerciales tienen un rendimiento inferior al de sus pruebas retrospectivas significativamente en el "comercio en vivo".

3.2.3 Sesgo de supervivencia

El sesgo de supervivencia es un fenómeno particularmente peligroso y puede conducir a un rendimiento significativamente inflado para ciertos tipos de estrategia. Ocurre cuando las estrategias se prueban en conjuntos de datos que no incluyen el universo completo de activos anteriores que pueden haber sido elegidos en un momento determinado, sino que solo consideran aquellos que han "sobrevivido" hasta el momento actual.

Como ejemplo, considere probar una estrategia en una selección aleatoria de acciones antes y después de la caída del mercado de 2001. Algunas acciones tecnológicas quebraron, mientras que otras lograron mantenerse a flote e incluso prosperaron. Si hubiéramos restringido esta estrategia solo a las acciones que superaron el período de caída del mercado, estaríamos introduciendo un sesgo de supervivencia porque ya nos han demostrado su éxito. De hecho, este es solo otro caso específico de sesgo de anticipación, ya que la información futura se incorpora al análisis anterior.

Hay dos formas principales de mitigar el sesgo de supervivencia en las pruebas retrospectivas de su estrategia:

- Conjuntos de datos libres de sesgo de supervivencia -En el caso de los datos de renta variable, es posible comprar conjuntos de datos que incluyan entidades excluidas de la lista, aunque no son baratos y solo tienden a ser utilizados por empresas institucionales. En particular, los datos de Yahoo Finance NO están libres de sesgos de supervivencia, y esto es comúnmente utilizado por muchos comerciantes minoristas de algoritmos. También se puede negociar con clases de activos que no son propensas al sesgo de supervivencia, como ciertas materias primas (y sus futuros derivados).

- Usar datos más recientes -En el caso de las acciones, el uso de un conjunto de datos más reciente mitiga la posibilidad de que la selección de acciones elegida esté ponderada para los "supervivientes", simplemente porque hay menos probabilidad de que las acciones en general dejen de cotizar en períodos de tiempo más cortos. También se puede comenzar a construir un conjunto de datos personal libre de sesgo de supervivencia mediante la recopilación de datos desde el punto actual en adelante. Después de 3 o 4 años, tendrá un sólido conjunto de datos de renta variable libre de sesgo de supervivencia con el que podrá realizar pruebas retrospectivas de otras estrategias.

Ahora consideraremos ciertos fenómenos psicológicos que pueden influir en su rendimiento comercial.

3.2.4 Sesgo cognitivo

Este fenómeno particular no se discute a menudo en el contexto de*cuantitativo*comercio. Sin embargo, se discute extensamente con respecto a métodos de negociación más discrecionales. Al crear pruebas retrospectivas durante un período de 5 años o más, es fácil observar una curva de capital con tendencia ascendente, calcular el rendimiento anual compuesto, el índice de Sharpe e incluso las características de reducción y estar satisfecho con los resultados. Como ejemplo, la estrategia podría poseer un máximo relativo

disposición del 25% y una duración máxima de la disposición de 4 meses. Esto no sería atípico para una estrategia de impulso. Es sencillo convencerse de que es fácil tolerar esos períodos de pérdidas porque el panorama general es halagüeño. Sin embargo, en la práctica, ¡es mucho más difícil!

Si se producen reducciones históricas del 25% o más en las pruebas retrospectivas, entonces, con toda probabilidad, verá períodos de reducción similares en las operaciones en vivo. Estos períodos de reducción son psicológicamente difíciles de soportar. He observado de primera mano cómo puede ser una reducción prolongada, en un entorno institucional, y no es agradable, incluso si las pruebas retrospectivas sugieren que ocurrirán tales períodos. La razón por la que lo he denominado "sesgo" es que, a menudo, una estrategia que de otro modo sería exitosa deja de operar durante tiempos de reducción prolongada y, por lo tanto, conducirá a un bajo rendimiento significativo en comparación con un backtest. Por lo tanto, aunque la estrategia es de naturaleza algorítmica, los factores psicológicos aún pueden tener una gran influencia en la rentabilidad.

3.3 Problemas de intercambio

3.3.1 Tipos de órdenes

Una elección que debe hacer un comerciante algorítmico es cómo y cuándo hacer uso de los diferentes *órdenes de cambio*disponible. Esta elección por lo general cae en el ámbito de la*sistema de ejecución*, pero lo consideraremos aquí, ya que puede afectar en gran medida el rendimiento del backtest de la estrategia. Hay dos tipos de pedidos que se pueden realizar:órdenes de mercadoyórdenes limitadas.

Una orden de mercado ejecuta una operación de inmediato, independientemente de los precios disponibles. Por lo tanto, las grandes operaciones ejecutadas como órdenes de mercado a menudo obtendrán una mezcla de precios a medida que se completa cada orden de límite posterior en el lado opuesto. Se consideran órdenes de mercado*agresivo*ya que es casi seguro que se completarán, aunque con un costo potencialmente desconocido.

Las órdenes limitadas proporcionan un mecanismo para que la estrategia determine el peor precio al que se ejecutará la operación, con la advertencia de que es posible que la operación no se complete parcial o totalmente. Se consideran órdenes limitadas*pasivo*ya que a menudo no se completan, pero cuando lo son, el precio está garantizado. La colección de órdenes limitadas de un intercambio individual se conoce comolibro de órdenes limitadas,que es esencialmente una cola de órdenes de compra y venta a ciertos tamaños y precios.

Al realizar pruebas retrospectivas, es esencial modelar correctamente los efectos del uso de órdenes de mercado o limitadas. Para las estrategias de alta frecuencia en particular, las pruebas retrospectivas pueden superar significativamente el comercio en vivo si los efectos del impacto del mercado y el libro de órdenes límite no se modelan con precisión.

3.3.2 Consolidación de Precios

Existen problemas particulares relacionados con las estrategias de backtesting cuando se utilizan datos diarios en forma de cifras Open-High-Low-Close (OHLC), especialmente para acciones. Tenga en cuenta que esta es precisamente la forma de datos proporcionada por Yahoo Finance, que es una fuente de datos muy común para los operadores algorítmicos minoristas.

Los conjuntos de datos baratos o gratuitos, si bien sufren el sesgo de supervivencia (que ya hemos discutido anteriormente), también suelen ser fuentes de precios compuestas de múltiples intercambios. Esto significa que los puntos extremos (es decir, la apertura, el cierre, el máximo y el mínimo) de los datos son muy susceptibles a valores "periféricos" debido a pedidos pequeños en bolsas regionales. Además, a veces es más probable que estos valores sean errores de marca que aún no se han eliminado del conjunto de datos.

Esto significa que si su estrategia comercial hace un uso extensivo de cualquiera de los puntos OHLC específicamente, el rendimiento del backtest puede diferir del rendimiento en vivo, ya que las órdenes pueden enrutarse a diferentes intercambios según su corredor y su acceso disponible a la liquidez. La única forma de resolver estos problemas es hacer uso de datos de mayor frecuencia u obtener datos directamente de un intercambio individual, en lugar de una fuente compuesta más barata.

3.3.3 Comercio de divisas y ECN

El backtesting de las estrategias de divisas es algo más complicado de implementar que el de las estrategias de renta variable. El comercio de divisas se produce en múltiples lugares y redes de comunicación electrónica (ECN). Los precios de oferta/demanda logrados en un lugar pueden diferir sustancialmente de los de otro lugar. Se debe tener mucho cuidado al utilizar la información de precios del lugar particular en el que operará en el backtest, a diferencia de una fuente consolidada de múltiples lugares, ya que esto será significativamente más indicativo de los precios que es probable que logre. delantero.

Otra idiosincrasia de los mercados de divisas es que los propios corredores no están obligados a compartir *comercio*precios/tamaños con cada participante comercial, ya que esta es su información de propiedad[6]. Por lo tanto, es más apropiado utilizar cotizaciones de oferta y demanda en sus pruebas retrospectivas y tener mucho cuidado con la variación de los costos de transacción entre corredores/lugares.

3.3.4 Restricciones de cortocircuito

Al realizar operaciones cortas en el backtest, es necesario tener en cuenta que algunas acciones pueden no haber estado disponibles (debido a la falta de disponibilidad en esas acciones para pedir prestado) o debido a una restricción del mercado, como la prohibición de la SEC de EE. UU. de las acciones financieras durante la crisis del mercado de 2008.

Esto puede inflar severamente los rendimientos de las pruebas retrospectivas, así que tenga cuidado de incluir tales restricciones de venta corta dentro de sus pruebas retrospectivas, o evite las posiciones cortas si cree que es probable que haya restricciones de liquidez en los instrumentos que opera.

3.4 Costos de transacción

uno de los maserrores frecuentes de los principiantesal implementar modelos comerciales es descuidar (o subestimar enormemente) los efectos de*costos de transacción*en una estrategia. Aunque a menudo se supone que los costos de transacción solo reflejan las comisiones de los corredores, de hecho, hay muchas otras formas en que los costos pueden acumularse en un modelo comercial. Los tres tipos principales de costos que deben considerarse incluyen:

3.4.1 Comisión

La forma más directa de costos de transacción en los que incurre una estrategia comercial algorítmica son las comisiones y tarifas. Todas las estrategias requieren alguna forma de acceso a un*intercambio*, ya sea directamente o a través de un intermediario de corretaje ("el corredor"). Estos servicios incurren en un costo incremental con cada comercio, conocido como*comisión*.

Los corredores generalmente brindan muchos servicios, aunque los algoritmos cuantitativos solo hacen uso de la infraestructura de intercambio. Por lo tanto, las comisiones de corretaje suelen ser pequeñas por operación. Los corredores también cobran*Tarifa*, que son los costos incurridos para compensar y liquidar operaciones. Más allá de esto están*impuestos*impuestos por los gobiernos regionales o nacionales. Por ejemplo, en el Reino Unido hay un *impuesto de timbre*para pagar en transacciones de acciones. Dado que las comisiones, tarifas e impuestos generalmente son fijos, son relativamente sencillos de implementar en un motor de backtest (ver más abajo).

3.4.2 Deslizamiento

El deslizamiento es la diferencia de precio lograda entre el momento en que un sistema comercial decide realizar transacciones y el momento en que se lleva a cabo realmente una transacción en un intercambio. El deslizamiento es un componente considerable de los costos de transacción y puede marcar la diferencia entre una estrategia muy rentable y una que funciona mal. El deslizamiento es una función de la volatilidad del activo subyacente, la latencia entre el sistema de comercio y el intercambio y el tipo de estrategia que se lleva a cabo.

Es más probable que un instrumento con mayor volatilidad se mueva, por lo que los precios entre la señal y la ejecución pueden diferir sustancialmente. La latencia se define como la diferencia de tiempo entre la generación de la señal y el punto de ejecución. Las estrategias de mayor frecuencia son más sensibles a la latencia

los problemas y las mejoras de milisegundos en esta latencia pueden marcar la diferencia hacia la rentabilidad. El tipo de estrategia también es importante. Los sistemas de impulso sufren más de deslizamiento en promedio porque están tratando de comprar instrumentos que ya se están moviendo en la dirección pronosticada. Lo contrario es cierto para las estrategias de reversión a la media, ya que estas estrategias se mueven en una dirección opuesta a la operación.

3.4.3 Impacto en el mercado

El impacto en el mercado es el costo en el que incurren los comerciantes debido a la dinámica de oferta/demanda del intercambio (y el activo) a través del cual intentan operar. Es probable que una orden grande en un activo relativamente ilíquido*mover el mercado*sustancialmente ya que el comercio necesitará acceder a un gran componente de la oferta actual. Para contrarrestar esto, las transacciones de grandes bloques se dividen en "trozos" más pequeños que se negocian periódicamente, a medida que llega nueva liquidez al intercambio. En el extremo opuesto, para instrumentos altamente líquidos como el contrato de futuros del índice S&P500 E-Mini, es poco probable que las transacciones de bajo volumen ajusten el "precio actual" en una gran cantidad.

Los activos más ilíquidos se caracterizan por una mayoruntado,cuál es la diferencia entre la oferta actual y los precios de venta en ellibro de órdenes limitadas.Este diferencial es un costo de transacción adicional asociado con cualquier operación. El margen es un componente muy importante del costo total de la transacción.
- como lo demuestra la miríada de empresas de apuestas a margen del Reino Unido cuyas campañas publicitarias expresan la "estrechez" de sus márgenes para instrumentos fuertemente negociados.

3.5 Backtesting vs Realidad

En resumen, hay una asombrosa variedad de factores que se pueden simular para generar un backtest realista. Los peligros del sobreajuste, la mala limpieza de datos, el manejo incorrecto de los costos de transacción, el cambio de régimen de mercado y las restricciones comerciales a menudo conducen a un rendimiento de prueba retrospectiva que difiere sustancialmente de la implementación de una estrategia en vivo.

Por lo tanto, uno debe ser muy consciente de que es muy poco probable que el rendimiento futuro coincida directamente con el rendimiento histórico. Discutiremos estos problemas con más detalle cuando lleguemos a implementar un motor de backtesting basado en eventos cerca del final del libro.

Capítulo 4

Ejecución Automatizada

La ejecución automatizada es el proceso de permitir que la estrategia genere automáticamente señales de ejecución que se envían al corredor sin intervención humana. Esta es la forma más pura de estrategia comercial algorítmica, ya que minimiza los problemas debido a la intervención humana. Es el tipo de sistema que consideraremos más a menudo durante este libro.

4.1 Plataformas de backtesting

El panorama del software para el backtesting de estrategias es enorme. Las soluciones van desde software sofisticado de grado institucional completamente integrado hasta lenguajes de programación como C ++, Python y R, donde casi todo debe escribirse desde cero (o obtener 'complementos' adecuados). Como comerciantes cuantitativos, estamos interesados en el equilibrio de poder "poseer" nuestra pila de tecnología comercial frente a la velocidad y confiabilidad de nuestra metodología de desarrollo. Estas son las consideraciones clave para la elección del software:

- Habilidad de programación -La elección del entorno dependerá en gran medida de su capacidad para programar software. Yo diría que tener el control de la pila total tendrá un mayor efecto en su PnL a largo plazo que subcontratar tanto como sea posible al software del proveedor. Esto se debe al riesgo negativo de tener errores externos o idiosincrasias que no puede corregir en el software del proveedor, que de otro modo se remediarían fácilmente si tuviera más control sobre su "pila tecnológica". También desea un entorno que logre el equilibrio adecuado entre productividad, disponibilidad de la biblioteca y velocidad de ejecución. Hago mi propia recomendación personal a continuación.
- Capacidad de ejecución/Interacción de intermediarios -Cierto software de backtesting, como Tradestation, se vincula directamente con un corredor. No soy partidario de este enfoque, ya que la reducción de los costos de transacción suele ser un componente importante para obtener un índice de Sharpe más alto. Si está atado a un corredor en particular (y Tradestation lo "obliga" a hacerlo), tendrá más dificultades para hacer la transición a un nuevo software (o un nuevo corredor) si surge la necesidad. Interactive Brokers proporciona una API que es robusta, aunque con una interfaz un poco obtusa.
- personalización -Un entorno como MATLAB o Python le brinda una gran flexibilidad al crear estrategias algorítmicas, ya que proporcionan bibliotecas fantásticas para casi cualquier operación matemática imaginable, pero también permiten una amplia personalización cuando sea necesario.
- Complejidad de la estrategia -Cierto software simplemente no está hecho para el procesamiento de números pesados o la complejidad matemática. Excel es una de esas piezas de software. Si bien es bueno para estrategias más simples, realmente no puede hacer frente a numerosos activos o algoritmos más complicados a gran velocidad.
- Minimización de sesgo -¿Una pieza de software o datos en particular se presta más a los sesgos comerciales? Debe asegurarse de que si desea crear toda la funcionalidad usted mismo,

que no introduzca errores que puedan conducir a sesgos. Un ejemplo aquí es el sesgo de anticipación, que Excel minimiza, mientras que un backtester de investigación vectorizado podría prestarse accidentalmente.

- Velocidad de desarrollo -Uno no debería tener que pasar meses y meses implementando un motor de backtest. La creación de prototipos solo debería llevar unas pocas semanas. Asegúrese de que su software no esté obstaculizando su progreso en gran medida, solo para obtener algunos puntos porcentuales adicionales de velocidad de ejecución.

- Velocidad de ejecución -Si su estrategia depende completamente de la puntualidad de la ejecución (como en HFT/UHFT), entonces será necesario un lenguaje como C o C++. Sin embargo, estará al borde de la optimización del kernel de Linux y el uso de FPGA para estos dominios, lo cual está fuera del alcance de este libro.

- Costo -Muchos de los entornos de software con los que puede programar estrategias comerciales algorítmicas son completamente gratuitos y de código abierto. De hecho, muchos fondos de cobertura utilizan software de código abierto para todas sus pilas de negociación de algoritmos. Además, Excel y MATLAB son relativamente baratos e incluso existen alternativas gratuitas para cada uno.

Diferentes estrategias requerirán diferentes paquetes de software. Las estrategias HFT y UHFT se escribirán en C/C++. En estos días, tales estrategias a menudo se llevan a cabo en GPU y FPGA. Por el contrario, las estrategias de acciones direccionales de baja frecuencia son fáciles de implementar en TradeStation, debido a la naturaleza "todo en uno" del software/corretaje.

4.1.1 Programación

El desarrollo personalizado de un lenguaje de backtesting dentro de un lenguaje de programación de primera clase proporciona la mayor flexibilidad al probar una estrategia. Por el contrario, una plataforma de backtesting integrada desarrollada por un proveedor siempre tendrá que hacer suposiciones sobre cómo se llevan a cabo los backtesting. La selección de lenguajes de programación disponibles es amplia y diversa. No es obvio antes del desarrollo qué idioma sería adecuado.

Una vez que una estrategia ha sido codificada en reglas sistemáticas, es necesario realizar una prueba retrospectiva de tal manera que el comerciante cuantitativo confíe en que su desempeño futuro reflejará su desempeño pasado. En general, existen dos formas de sistema de backtesting que se utilizan para probar esta hipótesis. En general, se clasifican como*probadores de espalda de investigación*y*probadores de espalda basados en eventos*.

4.1.2 Herramientas de investigación

La forma más simple de una herramienta de backtesting, la herramienta de investigación, generalmente se considera primero. La herramienta de investigación se utiliza para determinar rápidamente si es probable que una estrategia tenga algún rendimiento. Tales herramientas a menudo hacen suposiciones poco realistas sobre los costos de transacción, los precios de ejecución probables, las restricciones de venta al descubierto, la dependencia del lugar, la gestión de riesgos y una serie de otros problemas que se describieron en el capítulo anterior. Las herramientas comunes para la investigación incluyen MATLAB, R, Python y Excel.

La etapa de investigación es útil porque los paquetes de software proporcionan*vectorizado* capacidad, lo que conduce a una buena velocidad de ejecución y una implementación sencilla (menos líneas de código). Por lo tanto, es posible probar múltiples estrategias, combinaciones y variantes de manera rápida e iterativa.

Si bien estas herramientas se utilizan a menudo tanto para backtesting como para ejecución, dichos entornos de investigación generalmente no son adecuados para estrategias que abordan el comercio intradía a frecuencias más altas (subminutos). Esto se debe a que estos entornos a menudo no poseen las bibliotecas necesarias para conectarse a servidores de proveedores de datos de mercado en tiempo real o pueden interactuar con las API de corretaje de manera limpia.

A pesar de estas deficiencias de ejecución, los entornos de investigación se utilizan mucho en el entorno cuantitativo profesional. Son la "primera prueba" para todas las ideas de estrategia antes de promoverlas a una verificación más rigurosa dentro de un entorno de backtesting realista.

4.1.3 Backtesting basado en eventos

Una vez que una estrategia se ha considerado adecuada sobre la base de la investigación, debe probarse de una manera más realista. Tal realismo intenta explicar la mayoría (si no todos) de los problemas descritos en el capítulo anterior. La situación ideal es poder utilizar el mismo código de generación comercial para el backtesting histórico así como para la ejecución en vivo. Esto se puede lograr usando un*backtester basado en eventos*.

Los sistemas controlados por eventos se usan ampliamente en la ingeniería de software, comúnmente para manejar la entrada de la interfaz gráfica de usuario (GUI) dentro de los sistemas operativos basados en ventanas. También son ideales para el comercio algorítmico. Dichos sistemas suelen estar escritos en lenguajes de alto rendimiento como C++, C# y Java.

Considere una situación en la que una estrategia comercial automatizada está conectada a una fuente de mercado en tiempo real y un corredor (estos dos pueden ser el mismo). Se enviará nueva información de mercado al sistema, que desencadena y*evento*para generar una nueva señal comercial y por lo tanto una ejecución *evento*. Por lo tanto, dicho sistema está en un bucle continuo esperando recibir eventos y manejarlos de manera adecuada.

Es posible generar subcomponentes como un controlador de datos históricos y un simulador de corretaje, que pueden imitar a sus contrapartes en vivo. Esto permite realizar pruebas retrospectivas de las estrategias de una manera extremadamente similar a la ejecución en vivo.

La desventaja de tales sistemas es que son mucho más complicados de diseñar e implementar que una herramienta de investigación más simple. Por lo tanto, el "tiempo de comercialización" es más largo. Son más propensos a errores y requieren un conocimiento razonable de programación y, hasta cierto punto, metodología de desarrollo de software.

4.1.4 Latencia

En términos de ingeniería,*latencia*se define como el intervalo de tiempo entre una simulación y una respuesta. Para nuestros propósitos, generalmente se referirá a la demora de tiempo de ida y vuelta entre la generación de una señal de ejecución y la recepción de la información de llenado de un corredor que está llevando a cabo la ejecución.

Dicha latencia rara vez es un problema en las estrategias interdiarias de baja frecuencia, ya que el probable movimiento de precios durante el período de latencia no afectará en gran medida a la estrategia. Desafortunadamente, no ocurre lo mismo con las estrategias de mayor frecuencia. En estas frecuencias la latencia se vuelve importante. El objetivo final es reducir la latencia tanto como sea posible para minimizar*deslizamiento*, como se discutió en el capítulo anterior.

La disminución de la latencia implica minimizar la "distancia" entre el sistema de comercio algorítmico y el intercambio final en el que se ejecuta una orden. Esto puede implicar acortar la distancia geográfica entre los sistemas (y, por lo tanto, reducir los viajes por el cableado de la red), reducir el procesamiento realizado en el hardware de la red (importante en las estrategias HFT) o elegir un intermediario con una infraestructura más sofisticada.

La disminución de la latencia se vuelve exponencialmente más costosa en función de la "distancia de Internet" (es decir, la distancia de red entre dos servidores). Por lo tanto, para un comerciante de alta frecuencia, se debe llegar a un compromiso entre el gasto de reducción de latencia frente a la ganancia de minimizar el deslizamiento. Estos temas serán discutidos en la sección de*Colocación*abajo.

4.1.5 Opciones de idioma

Ya se han descrito algunos problemas que impulsan la elección del idioma. Ahora consideraremos las ventajas y desventajas de los lenguajes de programación individuales. En términos generales, he clasificado los lenguajes en alto rendimiento/desarrollo más difícil frente a bajo rendimiento/desarrollo más fácil. Estos son términos subjetivos y algunos no estarán de acuerdo dependiendo de sus antecedentes.

Uno de los aspectos más importantes de la programación de un entorno de backtesting personalizado es que el programador esté familiarizado con las herramientas que se utilizan. Ese es probablemente un criterio más importante que la velocidad de desarrollo. Sin embargo, para aquellos que son nuevos en el panorama de los lenguajes de programación, lo siguiente debería aclarar qué tiende a utilizarse dentro del comercio algorítmico.

C++, C# y Java

C++, C# y Java son todos ejemplos de*propósito general*Lenguajes de programación orientados a objetos. Es decir, se pueden usar sin un IDE correspondiente, son todos multiplataforma (se pueden ejecutar en Windows, Mac OSX o Linux), tienen una amplia gama de bibliotecas para casi cualquier tarea imaginable y poseen una velocidad de ejecución rápida.

Si se desea la máxima velocidad de ejecución, es probable que C++ (o C) sea la mejor opción. Ofrece la mayor flexibilidad para administrar la memoria y optimizar la velocidad de ejecución. Esta flexibilidad tiene un precio. C ++ es notoriamente difícil de aprender bien y, a menudo, puede generar errores sutiles. El tiempo de desarrollo puede llevar mucho más tiempo que en otros idiomas.

C# y Java son similares en el sentido de que ambos requieren que todos los componentes sean objetos, con la excepción de los tipos de datos primitivos, como los números flotantes y enteros. Se diferencian de C++ en que ambos realizan funciones automáticas.*recolección de basura*. Esto significa que la memoria no tiene que desasignarse manualmente tras la destrucción de un objeto. La recolección de basura agrega una sobrecarga de rendimiento, pero hace que el desarrollo sea sustancialmente más rápido. Estos lenguajes son buenas opciones para desarrollar un backtester, ya que tienen capacidades de GUI nativas, bibliotecas de análisis numérico y una velocidad de ejecución rápida.

Personalmente, utilizaría C++ para crear un backtester basado en eventos que necesite una velocidad de ejecución extremadamente rápida, como HFT. Esto es solo si siento que un sistema basado en eventos de Python se está convirtiendo en un cuello de botella, ya que este último lenguaje sería mi primera opción para dicho sistema.

MATLAB, R y Python

MATLAB es un entorno de desarrollo integrado (IDE) comercial para computación numérica. Ha ganado una amplia aceptación en los sectores académico, de ingeniería y financiero. Tiene una gran variedad de bibliotecas numéricas para muchas tareas de computación científica y posee una velocidad de ejecución rápida, suponiendo que cualquier algoritmo que se desarrolle esté sujeto a*vectorización*o *paralelización*. Es caro y esto a veces lo hace menos atractivo para los comerciantes minoristas con un presupuesto limitado. A veces, MATLAB se utiliza para la ejecución directa a través de una agencia de corretaje como Interactive Brokers.

R es menos un lenguaje de programación de propósito general y más un entorno de secuencias de comandos de estadísticas. Es gratuito, de código abierto, multiplataforma y contiene una gran variedad de paquetes estadísticos disponibles gratuitamente para realizar análisis extremadamente avanzados. R es muy utilizado en la industria de fondos de cobertura cuantitativos para la investigación estadística/estrategia. Si bien es posible conectar R a un corretaje, no se adapta bien a la tarea y debe considerarse más como una herramienta de investigación. Además, carece de velocidad de ejecución a menos que se vectoricen las operaciones.

He agrupado a Python bajo este encabezado, aunque se encuentra en algún lugar entre MATLAB, R y los lenguajes de propósito general antes mencionados. Es gratuito, de código abierto y multiplataforma. Está *interpretado*Opuesto a*compilado*, lo que lo hace más lento de forma nativa que C++. A pesar de esto, contiene una biblioteca para realizar casi cualquier tarea imaginable, desde computación científica hasta diseño de servidor web de bajo nivel. En particular, contiene NumPy, SciPy, pandas, matplotlib y scikit-learn, que brindan un sólido entorno de investigación numérica que, cuando se vectoriza, es comparable a la velocidad de ejecución del lenguaje compilado.

Además, tiene bibliotecas maduras para conectarse a corredores. Esto lo convierte en una "ventanilla única" para crear un entorno de ejecución en vivo y backtesting basado en eventos sin tener que pasar a otros idiomas. La velocidad de ejecución es más que suficiente para los operadores intradía que operan en la escala de tiempo de minutos. Finalmente, es muy sencillo de aprender y aprender, en comparación con lenguajes de nivel inferior como C++. Por estas razones hacemos un uso extensivo de Python en este libro.

4.1.6 Entornos de desarrollo integrado

El término IDE tiene múltiples significados dentro del comercio algorítmico. Los desarrolladores de software lo usan para referirse a una GUI que permite programar con funciones de resaltado de sintaxis, exploración de archivos, depuración y ejecución de código. Los comerciantes algorítmicos lo usan para referirse a un entorno de backtesting/comercio completamente integrado, que incluye descarga de datos históricos o en tiempo real, gráficos, evaluación estadística y

ejecución en vivo. Para nuestros propósitos, uso el término para referirme a cualquier entorno (a menudo basado en GUI) que es*no*un lenguaje de programación de propósito general, como C++ o Python. MATLAB se considera un IDE, por ejemplo.

Sobresalir

Si bien algunos cuantos más puros pueden menospreciar a Excel, he descubierto que es extremadamente útil para "verificar la cordura" de los resultados. El hecho de que todos los datos estén disponibles directamente, en lugar de estar escondidos detrás de objetos, hace que sea sencillo implementar estrategias muy básicas de señal/filtro. Los corredores, como Interactive Brokers, también permiten complementos DDE que permiten a Excel recibir datos de mercado en tiempo real y ejecutar órdenes comerciales.

A pesar de la facilidad de uso, Excel es extremadamente lento para cualquier escala razonable de datos o nivel de cálculo numérico. Solo lo uso para verificar errores cuando desarrollo contra otras estrategias y para asegurarme de haber evitado el sesgo de anticipación, que es fácil de ver en Excel debido a la naturaleza de hoja de cálculo del software.

Si no se siente cómodo con los lenguajes de programación y está llevando a cabo una estrategia entre días, entonces Excel puede ser la opción perfecta.

Software de backtesting comercial/minorista

El mercado de gráficos minoristas, "análisis técnicos" y software de backtesting es extremadamente competitivo. Las características que ofrece dicho software incluyen gráficos de precios en tiempo real, una gran cantidad de indicadores técnicos, lenguajes de backtesting personalizados y ejecución automatizada.

Algunos proveedores ofrecen una solución todo en uno, como TradeStation. TradeStation es una agencia de corretaje en línea que produce software comercial (también conocido como TradeStation) que proporciona ejecución de órdenes electrónicas en múltiples clases de activos. Actualmente, no creo que ofrezcan una API directa para la ejecución automática, sino que los pedidos deben realizarse a través del software. Esto contrasta con Interactive Brokers, que tienen una interfaz de negociación/gráficos más simplificada (Trader WorkStation), pero ofrecen sus API de ejecución de órdenes/mercado en tiempo real patentadas y una interfaz FIX.

Otra plataforma extremadamente popular es MetaTrader, que se utiliza en el comercio de divisas para crear 'Asesores expertos'. Estos son scripts personalizados escritos en un lenguaje propietario que se pueden usar para el comercio automatizado. No he tenido mucha experiencia ni con TradeStation ni con MetaTrader, así que no pasaré mucho tiempo discutiendo sus méritos.

Estas herramientas son útiles si no se siente cómodo con el desarrollo de software en profundidad y desea que se cuiden muchos de los detalles. Sin embargo, con tales sistemas se sacrifica mucha flexibilidad y, a menudo, está atado a un solo corredor.

Herramientas basadas en web

Los dos sistemas populares de backtesting basados en la web actuales son Quantopian (https://www.quantopian.com/) y QuantConnect (https://www.quantconnect.com/). El primero utiliza Python (y ZipLine, ver más abajo) mientras que el último utiliza C#. Ambos proporcionan una gran cantidad de datos históricos. Quantopian actualmente admite el comercio en vivo con Interactive Brokers, mientras que QuantConnect está trabajando para el comercio en vivo.

Software de backtesting de código abierto

Además de las ofertas comerciales, existen alternativas de código abierto para el software de backtesting.

Algo-Trader es una empresa con sede en Suiza que ofrece una licencia comercial y de código abierto para su sistema. Por lo que puedo deducir, la oferta parece bastante madura y tienen muchos clientes institucionales. El sistema permite backtesting histórico completo y*procesamiento de eventos complejos* y se relacionan con Interactive Brokers. La edición Enterprise ofrece muchas más funciones de alto rendimiento.

Marketcetera proporciona un sistema de backtesting que puede vincularse con muchos otros lenguajes, como Python y R, para aprovechar el código que quizás ya haya escrito. La estrategia

Studio' brinda la capacidad de escribir código de backtesting, así como algoritmos de ejecución optimizados y, posteriormente, hacer la transición de un backtest histórico a una negociación en papel en vivo.

ZipLine es la biblioteca de Python que impulsa el servicio Quantopian mencionado anteriormente. Es un entorno de backtest totalmente basado en eventos y actualmente admite acciones de EE. UU. en una base de barra por minuto. No he hecho un uso extensivo de ZipLine, pero conozco a otros que sienten que es una buena herramienta. Todavía quedan muchas áreas por mejorar, pero el equipo está trabajando constantemente en el proyecto, por lo que se mantiene muy activamente.

También hay algunos proyectos alojados en Github/Google Code que tal vez desee analizar. No he dedicado mucho tiempo a investigarlos. Dichos proyectos incluyen OpenQuant (http://code.google.com/p/openquant/), TradeLink (https://code.google.com/p/tradelink/) y PyAlgoTrade (http://gbeced.github.io /pyalgotrade/).

Software de Backtesting Institucional

Los sistemas de backtesting de grado institucional, como Deltix y QuantHouse, no suelen ser utilizados por los operadores algorítmicos minoristas. Las licencias de software generalmente están fuera del presupuesto de infraestructura. Dicho esto, dicho software es ampliamente utilizado por fondos cuantitativos, casas comerciales propietarias, oficinas familiares y similares.

Los beneficios de tales sistemas son claros. Proporcionan una solución todo en uno para la recopilación de datos, el desarrollo de estrategias, el backtesting histórico y la ejecución en vivo en instrumentos o carteras individuales, hasta el nivel de frecuencia ultra alta. Dichas plataformas han tenido pruebas exhaustivas y mucho uso "en el campo", por lo que se consideran sólidas.

Los sistemas se basan en eventos y, como tal, el entorno de backtesting a menudo puede simular bien el entorno en vivo. Los sistemas también admiten algoritmos de ejecución optimizados, que intentan minimizar los costos de transacción.

Debo admitir que no he tenido mucha experiencia con Deltix o QuantHouse más allá de algunos resúmenes superficiales. Dicho esto, el presupuesto por sí solo los pone fuera del alcance de la mayoría de los comerciantes minoristas, por lo que no me detendré en estos sistemas.

4.2 Colocación

El panorama del software para el comercio algorítmico ya ha sido examinado. Ahora es el momento de centrar la atención en la implementación del hardware que ejecutará nuestras estrategias.

Un comerciante minorista probablemente ejecutará su estrategia desde casa durante el horario de mercado, encenderá su PC, se conectará a la correduría, actualizará su software de mercado y luego permitirá que el algoritmo se ejecute automáticamente durante el día. Por el contrario, un fondo cuantitativo profesional con importantes*los activos bajo gestión*(AUM) contará con una infraestructura de servidor dedicada ubicada en el intercambio para reducir la latencia en la medida de lo posible para ejecutar sus estrategias de alta velocidad.

4.2.1 Escritorio de inicio

El enfoque más simple para la implementación de hardware es simplemente llevar a cabo una estrategia algorítmica con una computadora de escritorio doméstica conectada a la correduría a través de una conexión de banda ancha (o similar).

Si bien este enfoque es sencillo para comenzar, adolece de muchos inconvenientes. Principalmente, la máquina de escritorio está sujeta a fallas de energía, a menos que esté respaldada por un UPS. Además, una conexión a Internet en el hogar también está a merced del ISP. La pérdida de energía o la falla de la conectividad a Internet podrían ocurrir en un momento crucial en el comercio, dejando al comerciante algorítmico con posiciones abiertas que no se pueden cerrar.

En segundo lugar, una máquina de escritorio debe reiniciarse ocasionalmente, a menudo debido a la confiabilidad del sistema operativo. Esto significa que la estrategia adolece de un grado de intervención manual indirecta. Si esto ocurre fuera del horario comercial, el problema se mitiga. Sin embargo, si una computadora necesita reiniciarse durante el horario comercial, el problema es similar a una pérdida de energía. Las posiciones no cerradas aún pueden estar sujetas a riesgo.

La falla de los componentes también conduce al mismo conjunto de problemas de "tiempo de inactividad". Una falla en el disco duro, el monitor o la placa base a menudo ocurre precisamente en el momento equivocado. Por todas estas razones

Dudo en recomendar un enfoque de escritorio doméstico para el comercio algorítmico. Si decide seguir este enfoque, asegúrese de tener una computadora de respaldo Y una conexión a Internet de respaldo (por ejemplo, un dongle 3G) que pueda usar para cerrar posiciones en una situación de tiempo de inactividad.

4.2.2 SPV

El siguiente nivel desde un escritorio doméstico es hacer uso de unServidor Virtual Privado (VPS). Un VPS es un sistema de servidor remoto que a menudo se comercializa como un servicio de "nube". Son mucho más baratos que un servidor dedicado correspondiente, ya que un VPS es en realidad una partición de un servidor mucho más grande, con un entorno de sistema operativo aislado virtual únicamente disponible para usted. La carga de la CPU se comparte entre varios servidores y una parte de la RAM del sistema se asigna al VPS.

Los proveedores comunes de VPS incluyen Amazon EC2 y Rackspace Cloud. Proporcionan sistemas de nivel de entrada con poca RAM y uso básico de CPU, hasta servidores de alta RAM y alta CPU listos para la empresa. Para la mayoría de los comerciantes minoristas algorítmicos, los sistemas de nivel de entrada son suficientes para estrategias intradiarias o interdiarias de baja frecuencia y bases de datos de datos históricos más pequeñas.

Los beneficios de un sistema basado en VPS incluyen disponibilidad 24/7 (¡con cierto tiempo de inactividad realista!), capacidades de monitoreo más robustas, "complementos" fáciles para servicios adicionales, como almacenamiento de archivos o bases de datos administradas y una arquitectura flexible. Los inconvenientes incluyen los gastos a medida que el sistema crece, ya que el hardware dedicado se vuelve mucho más barato por rendimiento, asumiendo la colocación lejos de un intercambio, así como el manejo de escenarios de falla (es decir, creando un segundo VPS idéntico, por ejemplo).

Además, la latencia no siempre mejora al elegir un proveedor de VPS/nube. La ubicación de su hogar puede estar más cerca de un intercambio financiero en particular que los centros de datos de su proveedor de nube. Esto se mitiga de alguna manera al elegir una empresa que proporcione VPS orientados específicamente para el comercio algorítmico que se encuentran en o cerca de los intercambios, sin embargo, es probable que cuesten más que un proveedor de VPS "tradicional" como Amazon o Rackspace.

4.2.3 Intercambio

Para obtener la mejor minimización de latencia y sistemas más rápidos, es necesario ubicar un servidor dedicado (o un conjunto de servidores) directamente en el centro de datos de intercambio. Esta es una opción prohibitivamente costosa para casi todos los operadores algorítmicos minoristas (a menos que estén muy bien capitalizados). Es realmente el dominio del fondo o corretaje cuantitativo profesional.

Como mencioné anteriormente, una opción más realista es comprar un sistema VPS de un proveedor que se encuentra cerca de un intercambio. No me detendré demasiado en la colocación de intercambio, ya que el tema está un poco fuera del alcance del libro.

Capítulo 5

Ideas de estrategia de abastecimiento

En este capítulo quiero presentarles los métodos por los cuales yo mismo identifico estrategias comerciales algorítmicas rentables. Discutiremos cómo encontrar, evaluar y seleccionar tales sistemas. Explicaré cómo la identificación de estrategias tiene tanto que ver con las preferencias personales como con el rendimiento de la estrategia, cómo determinar el tipo y la cantidad de datos históricos para la prueba, cómo evaluar desapasionadamente una estrategia comercial y, finalmente, cómo proceder hacia la fase de backtesting y Implementación de la estrategia.

5.1 Identificación de sus propias preferencias personales para operar

Para ser un comerciante exitoso, ya sea discrecional o algorítmicamente, es necesario hacerse algunas preguntas honestas. El comercio le brinda la posibilidad de perder dinero a un ritmo alarmante, por lo que es necesario "conocerse a sí mismo" tanto como es necesario para comprender la estrategia elegida.

Yo diría que la consideración más importante en el comercio esser consciente de su propia personalidad.El comercio, y el comercio algorítmico en particular, requiere un grado significativo de disciplina, paciencia y desapego emocional. Dado que está permitiendo que un algoritmo realice sus transacciones por usted, es necesario resolverlo para no interferir con la estrategia cuando se está ejecutando. Esto puede ser extremadamente difícil, especialmente en períodos de reducción prolongada. Sin embargo, muchas estrategias que han demostrado ser altamente rentables en un backtest pueden arruinarse por una simple interferencia. ¡Comprenda que si desea ingresar al mundo del comercio algorítmico, será emocionalmente probado y que para tener éxito, es necesario superar estas dificultades!

La siguiente consideración es una detiempo.¿Tienes un trabajo de tiempo completo? ¿Trabajas medio tiempo? ¿Trabajas desde casa o tienes un viaje largo todos los días? Estas preguntas ayudarán a determinar la*frecuencia*de la estrategia que debes buscar. Para aquellos de ustedes que trabajan a tiempo completo, una estrategia de futuros intradía puede no ser apropiada (¡al menos hasta que esté completamente automatizada!). Sus limitaciones de tiempo también dictarán la metodología de la estrategia. Si su estrategia se negocia con frecuencia y depende de costosas fuentes de noticias (como una terminal de Bloomberg), ¡claramente tendrá que ser realista acerca de su capacidad para ejecutar esto con éxito mientras está en la oficina! Para aquellos de ustedes con mucho tiempo o las habilidades para automatizar su estrategia, es posible que deseen buscar una estrategia de negociación de alta frecuencia (HFT) más técnica.

Mi creencia es que es necesario llevar a caboinvestigación continuaen sus estrategias comerciales para mantener una cartera consistentemente rentable. Pocas estrategias permanecen "bajo el radar" para siempre. Por lo tanto, una parte significativa del tiempo asignado a la negociación se dedicará a la realización de investigaciones en curso. Pregúntese si está preparado para hacer esto, ya que puede ser la diferencia entre una fuerte rentabilidad o una lenta caída hacia las pérdidas.

También debe considerar sucapital comercial.La cantidad mínima ideal generalmente aceptada para una estrategia cuantitativa es de 50 000 USD (aproximadamente 35 000 £ para nosotros en el Reino Unido). Si tuviera que empezar de nuevo, empezaría con una cantidad mayor, probablemente más cercana a los 100 000 USD (aproximadamente 70 000 £). Esto se debe a que los costos de transacción pueden ser extremadamente costosos para estrategias de frecuencia media a alta y es necesario tener suficiente capital para absorberlos en tiempos.

de reducción. Si está considerando comenzar con menos de 10 000 USD, deberá limitarse a estrategias de baja frecuencia, comerciando con uno o dos activos, ya que los costos de transacción consumirán rápidamente sus ganancias.*Interactive Brokers, que es uno de los corredores más amigables para aquellos con habilidades de programación, debido a su API, tiene una cuenta minorista mínima de 10,000 USD.*

Habilidad de programaciónes un factor importante en la creación de una estrategia comercial algorítmica automatizada. Ser experto en un lenguaje de programación como C++, Java, C#, Python o R le permitirá crear usted mismo el almacenamiento de datos de extremo a extremo, el motor de backtest y el sistema de ejecución. Esto tiene una serie de ventajas, la principal de las cuales es la capacidad de estar completamente al tanto de todos los aspectos de la infraestructura comercial. También le permite explorar las estrategias de mayor frecuencia, ya que tendrá el control total de su "pila de tecnología". Si bien esto significa que puede probar su propio software y eliminar errores, también significa más tiempo dedicado a codificar la infraestructura y menos a implementar estrategias, al menos en la primera parte de su carrera comercial de algo. Es posible que se sienta cómodo operando con Excel o MATLAB y que pueda subcontratar el desarrollo de otros componentes. Sin embargo, no recomendaría esto, particularmente para aquellos que operan con alta frecuencia.

tienes que preguntartelo que esperas lograrpor negociación algorítmica. ¿Está interesado en un ingreso regular, por el cual espera obtener ganancias de su cuenta comercial? ¿O está interesado en una ganancia de capital a largo plazo y puede permitirse operar sin necesidad de retirar fondos? La dependencia de los ingresos dictará la frecuencia de su estrategia. Los retiros de ingresos más regulares requerirán una estrategia comercial de mayor frecuencia con menos volatilidad (es decir, una relación de Sharpe más alta). Los operadores a largo plazo pueden permitirse una frecuencia de negociación más tranquila.

Finalmente, ¡no se deje engañar por la idea de volverse extremadamente rico en un corto espacio de tiempo!El comercio de algoritmos NO es un esquema para hacerse rico rápidamente -en todo caso, puede ser un plan para empobrecerse rápidamente. Se necesita mucha disciplina, investigación, diligencia y paciencia para tener éxito en el comercio algorítmico. Puede llevar meses, si no años, generar una rentabilidad constante.

5.2 Abastecimiento de ideas comerciales algorítmicas

A pesar de las percepciones comunes de lo contrario, en realidad es bastante sencillo ubicar estrategias comerciales rentables en el dominio público. Nunca antes las ideas comerciales habían estado tan disponibles como en la actualidad. Las revistas académicas de finanzas, los servidores de preimpresión, los blogs comerciales, los foros comerciales, las revistas comerciales semanales y los textos especializados brindan miles de estrategias comerciales en las que basar sus ideas.

Nuestro objetivo como*investigadores comerciales cuantitativos*es establecer untubería de estrategia eso nos proporcionará un flujo de ideas comerciales en curso. Idealmente, queremos crear un enfoque metódico para obtener, evaluar e implementar las estrategias que encontramos. Los objetivos de la *tubería*son generar una cantidad consistente de nuevas ideas y proporcionarnos un marco para rechazar la mayoría de estas ideas con el mínimo de consideración emocional.

Debemos tener mucho cuidado de no dejar que los sesgos cognitivos influyan en nuestra metodología de toma de decisiones. Esto podría ser tan simple como tener una preferencia por una clase de activos sobre otra (me vienen a la mente el oro y otros metales preciosos) porque se perciben como más exóticos. Nuestro objetivo siempre debe ser encontrar estrategias consistentemente rentables, con expectativas positivas. La elección de la clase de activos debe basarse en otras consideraciones, como las restricciones de capital comercial, las tarifas de corretaje y las capacidades de apalancamiento.

5.2.1 Libros de texto

Si no está completamente familiarizado con el concepto de un*estrategia comercial*y los mercados financieros en general, entonces el primer lugar para buscar es con los libros de texto establecidos. Los textos clásicos proporcionan una amplia gama de ideas más sencillas y directas con las que familiarizarse con el comercio cuantitativo. Aquí hay una selección que recomiendo para aquellos que son nuevos en el comercio cuantitativo, que gradualmente se vuelve más sofisticado a medida que avanza en la lista.

Mercados Financieros y Participantes

La siguiente lista detalla los libros que describen cómo funcionan los mercados de capital y describen el comercio electrónico moderno.

- Guía del Financial Times sobre los mercados financierospor Glen Arnold[1] - Este libro está diseñado para principiantes en los mercados financieros y es extremadamente útil para obtener información sobre todos los participantes del mercado. Para nuestros propósitos, nos proporciona una lista de mercados en los que más adelante podríamos formar estrategias comerciales algorítmicas.
- Comercio e intercambios: microestructura de mercado para profesionalespor Larry Harris[7] - Este es un libro extremadamente informativo sobre los participantes de los mercados financieros y cómo operan. Proporciona detalles significativos sobre cómo se realizan las transacciones, los diversos motivos de los jugadores y cómo se regulan los mercados. Si bien algunos pueden considerarlo "lectura en seco", creo que es absolutamente esencial comprender estos conceptos para ser un buen comerciante algorítmico.
- Comercio algorítmico y DMA: una introducción a las estrategias comerciales de acceso directopor Barry Johnson[10] - El libro de Johnson está más orientado hacia el lado tecnológico de los mercados. Analiza los tipos de órdenes, los algoritmos de ejecución óptimos, los tipos de intercambios que aceptan el comercio algorítmico, así como estrategias más sofisticadas. Al igual que el libro de Harris anterior, explica en detalle cómo funcionan los mercados de comercio electrónico, cuyo conocimiento también creo que es un requisito previo esencial para llevar a cabo estrategias sistemáticas.

Comercio cuantitativo

El siguiente conjunto de libros trata directamente sobre el comercio algorítmico/cuantitativo. Describen algunos de los conceptos básicos y describen estrategias particulares que se pueden implementar.

- Comercio cuantitativo: cómo construir su propio negocio de comercio algorítmico por Ernest Chan[5] - El primer libro de Ernest Chan es una "guía para principiantes" sobre estrategias comerciales cuantitativas. Si bien no contiene muchas ideas estratégicas, presenta un marco sobre cómo configurar un negocio comercial, con ideas de gestión de riesgos y herramientas de implementación. Este es un gran libro si es completamente nuevo en el comercio algorítmico. El libro hace uso de MATLAB.
- Comercio algorítmico: estrategias ganadoras y su justificaciónpor Ernest Chan[6] - El segundo libro de Chan es muy pesado en ideas de estrategia. Esencialmente comienza donde lo dejó el libro anterior y se actualiza para reflejar las "condiciones actuales del mercado". El libro analiza las estrategias basadas en la reversión a la media y el impulso en las frecuencias interdiarias e intradiarias. también aborda brevemente el comercio de alta frecuencia. Al igual que con el libro anterior, hace un uso extensivo del código MATLAB.
- Dentro de la caja negra: La simple verdad sobre el comercio cuantitativo y de alta frecuencia, 2.ª ed.por Rishi Narang[12] - El libro de Narang brinda una descripción general de los componentes de un sistema comercial empleado por un fondo de cobertura cuantitativo, incluidos los generadores alfa, la gestión de riesgos, la optimización de la cartera y los costos de transacción. La segunda edición entra en detalles significativos sobre las técnicas de negociación de alta frecuencia.
- Comercio de volatilidadpor Euan Sinclair[16] - El libro de Sinclair se concentra únicamente en modelos/pronósticos de volatilidad y estrategias de opciones diseñadas para aprovechar estos modelos. Si planea negociar opciones de forma cuantitativa, este libro le proporcionará muchas ideas de investigación.

5.2.2 Internet

Después de obtener una base en el proceso de comercio algorítmico a través de los textos clásicos, se pueden obtener ideas de estrategia adicionales de Internet. Los blogs de finanzas cuantitativas, los agregadores de enlaces y los foros comerciales proporcionan ricas fuentes de ideas para probar.

Sin embargo, una nota de precaución: muchos recursos comerciales en Internet se basan en el concepto de análisis técnico.El análisis técnico implica utilizar análisis de señales básicas*indicadores*y*psicología del comportamiento*para determinar tendencias o patrones de inversión en los precios de los activos.

A pesar de ser extremadamente popular en el espacio comercial general, el análisis técnico se considera algo controvertido en la comunidad financiera cuantitativa. ¡Algunos han sugerido que no es mejor que leer un horóscopo o estudiar las hojas de té en términos de su poder predictivo! En realidad, hay personas exitosas que hacen un amplio uso del análisis técnico en sus operaciones.

Como quants con una caja de herramientas matemática y estadística más sofisticada a nuestra disposición, podemos evaluar fácilmente la efectividad de tales estrategias "basadas en TA". Esto nos permite tomar decisiones impulsadas por el análisis de datos y la prueba de hipótesis, en lugar de basar dichas decisiones en consideraciones emocionales o ideas preconcebidas.

Blogs cuánticos

Recomiendo los siguientes blogs cuantitativos para obtener buenas ideas comerciales y conceptos sobre el comercio algorítmico en general:

- Comercio de MATLAB -http://matlab-trading.blogspot.co.uk/
- Comercio cuantitativo (Ernest Chan) -http://epchan.blogspot.com
- cantidad -http://quantivity.wordpress.com
- Cuantópico -http://blog.quantopian.com
- Cuantpedia -http://quantpedia.com

Agregadores

Se ha puesto de moda en los últimos años que los enlaces temáticos se agreguen y luego se discutan. Leí los siguientes agregadores:

- Cuantocracia -http://www.quantocracy.com
- Noticias cuánticas -http://www.quantnews.com
- Algo Trading Sub-Reddit -http://www.reddit.com/r/algotrading

Foros

El siguiente lugar para encontrar ideas estratégicas adicionales es en los foros comerciales. No se deje intimidar por estrategias más orientadas al "análisis técnico". Estas estrategias a menudo brindan buenas ideas que pueden probarse estadísticamente:

- Foros de Elite Trader -http://www.elitetrader.com
- Financiamiento nuclear -http://www.nuclearphynance.com
- QuantNet -http://www.quantnet.com
- laboratorio de riqueza -http://www.wealth-lab.com/Foro
- Foros de Wilmott -http://www.wilmott.com

5.2.3 Literatura de revistas

Una vez que haya tenido algo de experiencia en la evaluación de estrategias más simples, es hora de mirar las ofertas académicas más sofisticadas. Algunas revistas académicas serán de difícil acceso, sin altas suscripciones o costos únicos. Si es miembro o ex alumno de una universidad, debería poder obtener acceso a algunas de estas revistas financieras. De lo contrario, puede mirar*servidores de preimpresión*, que son repositorios de Internet de los últimos borradores de trabajos académicos que se someten a revisión por pares. Dado que solo estamos interesados en estrategias que podamos replicar con éxito, realizar pruebas retrospectivas y obtener rentabilidad, una revisión por pares tiene menos importancia para nosotros.

La principal desventaja de las estrategias académicas es que a menudo pueden estar desactualizadas, requieren datos históricos oscuros y costosos, comercian con clases de activos ilíquidos o no tienen en cuenta las tarifas, el deslizamiento o el diferencial. También puede no estar claro si la estrategia comercial se llevará a cabo con órdenes de mercado, órdenes limitadas o si contiene stop loss, etc. Por lo tanto, es absolutamente esencial replicar la estrategia usted mismo lo mejor que pueda, probarla y agregar transacciones realistas. costos que incluyen tantos aspectos de las clases de activos que desea negociar.

Aquí hay una lista de los servidores de preimpresión y revistas financieras más populares de los que puede obtener ideas:

- arXiv -http://arxiv.org/archive/q-fin
- SSRN -http://www.ssrn.com
- Revista de Estrategias de Inversión -http://www.risk.net/type/journal/source/journalof-investment-strategies
- Revista de Finanzas Computacionales -http://www.risk.net/type/journal/source/journalof-computational-finance
- Finanzas Matemáticas -http://onlinelibrary.wiley.com/journal/10.1111/%28ISSN%291467-9965

5.2.4 Investigación independiente

¿Qué hay de formar sus propias estrategias cuantitativas? Esto generalmente requiere (pero no se limita a) experiencia en una o más de las siguientes categorías:

- Microestructura del mercado -Para estrategias de mayor frecuencia en particular, uno puede hacer uso de *microestructura del mercado*, es decir, la comprensión de la*dinámica de la cartera de pedidos*con el fin de generar rentabilidad. Los diferentes mercados tendrán varias limitaciones tecnológicas, regulaciones, participantes del mercado y restricciones que están todas abiertas a la explotación a través de estrategias específicas. Esta es un área muy sofisticada y a los profesionales minoristas les resultará difícil ser competitivos en este espacio, particularmente porque la competencia incluye grandes fondos de cobertura cuantitativos bien capitalizados con sólidas capacidades tecnológicas.
- Estructura del fondo -Los fondos de inversión mancomunados, como los fondos de pensión, las sociedades de inversión privada (fondos de cobertura), los asesores comerciales de productos básicos y los fondos mutuos están limitados tanto por una regulación estricta como por sus grandes reservas de capital. Así, ciertos comportamientos consistentes pueden ser explotados con aquellos que son más ágiles. Por ejemplo, los grandes fondos están sujetos a*restricciones de capacidad*debido a su tamaño. Por lo tanto, si necesitan descargar (vender) rápidamente una cantidad de valores, tendrán que escalonarla para evitar "mover el mercado". Los algoritmos sofisticados pueden aprovechar esta y otras idiosincrasias, en un proceso general conocido como*arbitraje de estructura de fondos*.
- Aprendizaje automático/inteligencia artificial -Los algoritmos de aprendizaje automático se han vuelto más frecuentes en los últimos años en los mercados financieros. Los clasificadores (como Naive-Bayes, et al.), los emparejadores de funciones no lineales (redes neuronales) y las rutinas de optimización (algoritmos genéticos) se han utilizado para predecir rutas de activos u optimizar estrategias comerciales. Si tiene experiencia en esta área, puede tener una idea de cómo se pueden aplicar algoritmos particulares a ciertos mercados.

Al continuar monitoreando las fuentes anteriores semanalmente, o incluso diariamente, se está preparando para recibir una lista consistente de estrategias de una amplia gama de fuentes. El siguiente paso es determinar cómo rechazar un gran subconjunto de estas estrategias para minimizar la pérdida de tiempo y los recursos de prueba retrospectiva en estrategias que probablemente no sean rentables.

5.3 Evaluación de estrategias comerciales

La primera consideración, y posiblemente la más obvia, es si realmenteentender la estrategia.¿Sería capaz de explicar la estrategia de manera concisa o requiere una serie de advertencias y listas interminables de parámetros? Además, ¿tiene la estrategia una base buena y sólida en la realidad? Por ejemplo, ¿podría señalar alguna lógica de comportamiento o restricción de la estructura de fondos que podría estar causando los patrones que está tratando de explotar? ¿Resistiría esta restricción un cambio de régimen, como una alteración drástica del entorno regulatorio? ¿La estrategia se basa en reglas estadísticas o matemáticas complejas? ¿Se aplica a cualquier serie temporal financiera o es específico de la clase de activos en la que se afirma que es rentable? Debe pensar constantemente en estos factores al evaluar nuevos métodos comerciales, de lo contrario, puede perder una cantidad significativa de tiempo intentando realizar pruebas retrospectivas y optimizar estrategias no rentables.

Una vez que haya determinado que comprende los principios básicos de la estrategia, debe decidir si se ajusta a su perfil de personalidad mencionado anteriormente. ¡Esta no es una consideración tan vaga como parece! Las estrategias diferirán sustancialmente en sus características de desempeño. Hay ciertos tipos de personalidad que pueden manejar períodos de reducción más significativos o están dispuestos a aceptar un mayor riesgo para obtener una mayor rentabilidad. A pesar de que nosotros, como quants, tratamos de eliminar tantos sesgos cognitivos como sea posible y deberíamos poder evaluar una estrategia desapasionadamente, los sesgos siempre aparecerán. Por lo tanto, necesitamos un medio consistente y sin emociones a través del cual evaluar el desempeño de las estrategias. . Aquí está la lista de criterios por los que juzgo una nueva estrategia potencial:

- Metodología -¿La estrategia se basa en el impulso, revierte la media, es neutral en el mercado y direccional? ¿La estrategia se basa en técnicas estadísticas o de aprendizaje automático sofisticadas (o complejas) que son difíciles de entender y requieren un doctorado en estadística para comprenderlas? ¿Estas técnicas introducen una cantidad significativa de parámetros que podrían conducir a un sesgo de optimización? ¿Es probable que la estrategia resista un*Cambio de regimén*(es decir, una posible nueva regulación de los mercados financieros)?

- Relación de Sharpe -La relación de Sharpe caracteriza heurísticamente la relación recompensa/riesgo de la estrategia. Cuantifica cuánto retorno puede lograr para el nivel de volatilidad soportado por la curva de acciones. Naturalmente, necesitamos determinar el período y la frecuencia en que se miden estos rendimientos y volatilidad (es decir, la desviación estándar). Una estrategia de mayor frecuencia requerirá una mayor tasa de muestreo de la desviación estándar, pero un período de tiempo total de medición más corto, por ejemplo.

- Aprovechar -¿La estrategia requiere un apalancamiento significativo para ser rentable? ¿La estrategia requiere el uso de contratos de derivados apalancados (futuros, opciones, swaps) para obtener un rendimiento? Estos contratos apalancados pueden tener características de gran volatilidad y, por lo tanto, pueden conducir fácilmente a*llamadas de margen*. ¿Tiene el capital comercial y el temperamento para tal volatilidad?

- Frecuencia -La frecuencia de la estrategia está íntimamente relacionada con su pila de tecnología (y, por lo tanto, la experiencia tecnológica), el índice de Sharpe y el nivel general de los costos de transacción. Todos los demás problemas considerados, las estrategias de mayor frecuencia requieren más capital, son más sofisticadas y más difíciles de implementar. Sin embargo, suponiendo que su motor de backtesting sea sofisticado y esté libre de errores, a menudo tendrán índices de Sharpe mucho más altos.

- Volatilidad -La volatilidad está fuertemente relacionada con el "riesgo" de la estrategia. La relación de Sharpe caracteriza esto. Una mayor volatilidad de las clases de activos subyacentes, si no están cubiertas, a menudo conduce a una mayor volatilidad en la curva de acciones y, por lo tanto, a índices de Sharpe más bajos. Por supuesto, asumo que la volatilidad positiva es aproximadamente igual a la volatilidad negativa. Algunas estrategias pueden tener una mayor volatilidad a la baja. Tienes que ser consciente de estos atributos.

- Ganancia/pérdida, ganancia/pérdida promedio -Las estrategias diferirán en sus características de ganancia/pérdida y ganancia/pérdida promedio. Uno puede tener una estrategia muy rentable, incluso si el número de operaciones perdedoras supera el número de operaciones ganadoras. Las estrategias de impulso tienden a tener este patrón, ya que se basan en una pequeña cantidad de "grandes éxitos" para ser rentables. Las estrategias de reversión a la media tienden a tener perfiles opuestos donde la mayoría de las operaciones son "ganadoras", pero las operaciones perdedoras pueden ser bastante graves.

- Reducción máxima -La reducción máxima es la mayor caída porcentual general de pico a valle en la curva de capital de la estrategia. Es bien sabido que las estrategias de impulso sufren períodos de caídas prolongadas (debido a una serie de muchas operaciones perdedoras incrementales). Muchos comerciantes se darán por vencidos en períodos de reducción prolongada, incluso si las pruebas históricas han sugerido que esto es "negocios como de costumbre" para la estrategia. Deberá determinar qué porcentaje de reducción (y durante qué período de tiempo) puede aceptar antes de dejar de operar con su estrategia. Esta es una decisión muy personal y por lo tanto debe ser considerada cuidadosamente.

- Capacidad/Liquidez -A nivel minorista, a menos que esté operando con un instrumento altamente ilíquido (como una acción de pequeña capitalización), no tendrá que preocuparse demasiado por la estrategia. *capacidad*. La capacidad determina la escalabilidad de la estrategia para aumentar el capital. Muchos de los fondos de cobertura más grandes sufren importantes problemas de capacidad a medida que sus estrategias aumentan la asignación de capital.

- Parámetros -Ciertas estrategias (especialmente las que se encuentran en la comunidad de aprendizaje automático) requieren una gran cantidad de parámetros. Cada parámetro adicional que requiere una estrategia la hace más vulnerable al sesgo de optimización (también conocido como "ajuste de curvas"). Debe probar y orientar estrategias con la menor cantidad de parámetros posible o asegurarse de tener suficientes cantidades de datos con los que probar sus estrategias.

- punto de referencia -Casi todas las estrategias (a menos que se caractericen como "rendimiento absoluto") se miden con respecto a algún punto de referencia de rendimiento. El punto de referencia suele ser un *índice* que caracteriza una gran muestra de la clase de activo subyacente en la que se negocia la estrategia. Si la estrategia negocia acciones estadounidenses de gran capitalización, entonces el S&P500 sería un punto de referencia natural para medir su estrategia. Escuchará los términos "alfa" y "beta", aplicados a estrategias de este tipo.

Tenga en cuenta que no hemos discutido el verdadero *devoluciones* de la estrategia ¿Por qué es esto? De forma aislada, los rendimientos en realidad nos brindan información limitada sobre la efectividad de la estrategia. No le dan una idea del apalancamiento, la volatilidad, los puntos de referencia o los requisitos de capital. Por lo tanto, las estrategias rara vez se juzgan solo por sus ganancias. Considere siempre los atributos de riesgo de una estrategia antes de mirar los rendimientos.

En esta etapa, muchas de las estrategias encontradas en su pipeline serán rechazadas de inmediato, ya que no cumplirán con sus requisitos de capital, restricciones de apalancamiento, tolerancia máxima a la reducción o preferencias de volatilidad. Las estrategias que quedan ahora pueden ser consideradas para *backtesting*. Sin embargo, antes de que esto sea posible, es necesario considerar un último criterio de rechazo: el de los datos históricos disponibles sobre los cuales probar estas estrategias.

5.4 Obtención de datos históricos

Hoy en día, la amplitud de los requisitos técnicos en todas las clases de activos para el almacenamiento de datos históricos es sustancial. Para seguir siendo competitivos, tanto el lado comprador (fondos, prop-desks) como el lado vendedor (corredores/agentes) invierten mucho en su infraestructura técnica. Es imperativo considerar su importancia. En particular, estamos interesados en la puntualidad, la precisión y los requisitos de almacenamiento. Discutiremos el almacenamiento de datos en capítulos posteriores del libro.

En la sección anterior, habíamos configurado un flujo de estrategias que nos permitía rechazar ciertas estrategias en función de nuestros propios criterios de rechazo personales. En esta sección filtraremos más estrategias en base a nuestras propias preferencias para la obtención de datos históricos. Las principales consideraciones (especialmente a nivel de minorista) son los costos de los datos, los requisitos de almacenamiento y su nivel de

conocimientos técnicos. También necesitamos discutir los diferentes tipos de datos disponibles y las diferentes consideraciones que cada tipo de datos nos impondrá.

Comencemos discutiendo los tipos de datos disponibles y los temas clave en los que tendremos que pensar, con el entendimiento de que exploraremos estos temas con mucha profundidad en el resto del libro:

- Datos Fundamentales -Esto incluye datos sobre tendencias macroeconómicas, como tasas de interés, cifras de inflación, acciones corporativas (dividendos, división de acciones), presentaciones ante la SEC, cuentas corporativas, cifras de ganancias, informes de cosechas, datos meteorológicos, etc. Estos datos se utilizan a menudo para valorar empresas o otros activos en un*fundamental*base, es decir, a través de algún medio de los flujos de efectivo futuros esperados. No incluye series de precios de acciones. Algunos datos fundamentales están disponibles gratuitamente en los sitios web gubernamentales. Otros datos fundamentales históricos a largo plazo pueden ser extremadamente costosos. Los requisitos de almacenamiento a menudo no son particularmente grandes, a menos que se estudien miles de empresas a la vez.

- Datos de noticias -Los datos de noticias suelen ser de naturaleza cualitativa. Se compone de artículos, publicaciones de blog, publicaciones de microblog ("tweets") y editoriales. Técnicas de aprendizaje automático como*clasificadores* a menudo se utilizan para interpretar*sentimiento*. Estos datos también suelen estar disponibles de forma gratuita o barata, a través de la suscripción a los medios de comunicación. Las bases de datos de almacenamiento de documentos "NoSQL" más recientes están diseñadas para almacenar este tipo de datos cualitativos no estructurados.

- Datos de precios de activos -Este es el dominio de datos tradicional del quant. Consiste en series temporales de precios de activos. Las acciones (acciones), los productos de renta fija (bonos), las materias primas y los precios de divisas se encuentran dentro de esta clase. Los datos históricos diarios suelen ser fáciles de obtener para las clases de activos más simples, como las acciones. Sin embargo, una vez que se incluyen la precisión y la limpieza y se eliminan los sesgos estadísticos, los datos pueden volverse costosos. Además, los datos de series de tiempo a menudo poseen requisitos de almacenamiento significativos, especialmente cuando se consideran los datos intradía.

- Instrumentos financieros -Las acciones, los bonos, los futuros y las opciones de derivados más exóticas tienen características y parámetros muy diferentes. Por lo tanto, no existe una estructura de base de datos de "talla única" que pueda acomodarlos. Se debe prestar especial atención al diseño y la implementación de estructuras de bases de datos para varios instrumentos financieros.

- Frecuencia -Cuanto mayor sea la frecuencia de los datos, mayores serán los costos y los requisitos de almacenamiento. Para estrategias de baja frecuencia, los datos diarios suelen ser suficientes. Para estrategias de alta frecuencia, podría ser necesario obtener datos de nivel de tick e incluso copias históricas de un intercambio comercial en particular.*libro de pedidos*datos. La implementación de un motor de almacenamiento para este tipo de datos requiere mucha tecnología y solo es adecuada para aquellos con una sólida formación en programación/técnica.

- Puntos de referencia -Las estrategias descritas anteriormente a menudo se compararán con un*punto de referencia*. Esto generalmente se manifiesta como una serie de tiempo financiera adicional. En el caso de las acciones, suele ser un índice de referencia bursátil nacional, como el índice S&P500 (EE. UU.) o el FTSE100 (Reino Unido). Para un fondo de renta fija, es útil compararlo con una canasta de bonos o productos de renta fija. La "tasa libre de riesgo" (es decir, la tasa de interés adecuada) también es otro punto de referencia ampliamente aceptado. Todas las categorías de clases de activos poseen un punto de referencia favorito, por lo que será necesario investigar esto en función de su estrategia particular, si desea obtener interés en su estrategia externamente.

- Tecnología -Las pilas de tecnología detrás de un centro de almacenamiento de datos financieros son complejas. Sin embargo, generalmente se centra en un motor de clúster de base de datos, como un Sistema de gestión de bases de datos relacionales (RDBMS), como MySQL, SQL Server, Oracle o un Motor de almacenamiento de documentos (es decir, "NoSQL"). Se accede a esto a través de un código de aplicación de "lógica empresarial" que consulta la base de datos y proporciona acceso a herramientas externas, como MATLAB, R o Excel. A menudo, esta lógica empresarial está escrita en C++, Java o Python. También deberá alojar estos datos en algún lugar, ya sea en su propia computadora personal o de forma remota a través de servidores de Internet. Productos como Amazon Web Services han simplificado y abaratado esto en

últimos años, pero aún requerirá una gran experiencia técnica para lograrlo de manera sólida.

Como puede verse, una vez que se ha identificado una estrategia a través del pipeline, será necesario evaluar la disponibilidad, los costos, la complejidad y los detalles de implementación de un conjunto particular de datos históricos. Es posible que descubra que es necesario rechazar una estrategia basada únicamente en consideraciones de datos históricos. Esta es un área grande y los equipos de doctores trabajan en grandes fondos para asegurarse de que los precios sean precisos y oportunos. ¡No subestime las dificultades de crear un centro de datos robusto para sus propósitos de backtesting!

Sin embargo, quiero decir que muchas plataformas de backtesting pueden proporcionarle estos datos automáticamente, a un costo. Por lo tanto, le quitará gran parte del dolor de la implementación y podrá concentrarse únicamente en la implementación y optimización de la estrategia. Herramientas como TradeStation poseen esta capacidad. Sin embargo, mi opinión personal es implementar tanto como sea posible internamente y evitar subcontratar partes de la pila a proveedores de software. Prefiero estrategias de mayor frecuencia debido a sus relaciones de Sharpe más atractivas, pero a menudo están estrechamente vinculadas a la pila de tecnología, donde la optimización avanzada es fundamental.

Parte III

Desarrollo de plataforma de datos

Capítulo 6

Instalación de software

Este capítulo discutirá en detalle cómo instalar un entorno de comercio algorítmico. La elección del sistema operativo se considera un primer paso necesario, con las tres opciones principales descritas. Posteriormente, se elige Linux como el sistema de elección (Ubuntu en particular) y se instala Python con todas las bibliotecas necesarias.

La instalación de paquetes/bibliotecas a menudo se pasa por alto en libros adicionales, pero personalmente creo que puede ser un obstáculo para muchos, por lo que le dediqué un capítulo completo. Desafortunadamente, la realidad es que el capítulo quedará fechado en el momento en que se publique. Surgen nuevas versiones de sistemas operativos y los paquetes se actualizan constantemente. Por lo tanto, es probable que haya detalles de implementación específicos.

Si tiene problemas para instalar o trabajar con estos paquetes, asegúrese de verificar las versiones instaladas y actualice si es necesario. Si aún tiene problemas, no dude en enviarme un correo electrónico a mike@quantstart.com e intentaré ayudarlo.

6.1 Elección del sistema operativo

La primera elección importante a la hora de decidirse por una plataforma de negociación algorítmica es la del sistema operativo. En cierto sentido, esto será dictado por el lenguaje de programación principal o los medios para conectarse a la correduría. En la actualidad, la mayoría del software, especialmente el de código abierto, es multiplataforma, por lo que la elección es menos restringida.

6.1.1 Microsoft Windows

Windows es probablemente la opción "predeterminada" de muchos comerciantes algorítmicos. Es extremadamente familiar y, a pesar de las críticas en sentido contrario, en ciertas formas es bastante robusto. Windows 8 no ha sido muy bien recibido, pero la versión anterior, Windows 7, se considera un sistema operativo sólido.

Ciertas herramientas en el espacio comercial algorítmico solo funcionarán en Windows, en particular el servidor IQFeed, necesario para descargar datos de ticks de DTN IQFeed. Además, Windows es la plataforma nativa de Microsoft .NET Framework, en la que se escribe una gran cantidad de software financiero, utilizando C++ y C#.

Si no desea utilizar Windows, a veces es posible ejecutar software basado en Windows en un sistema basado en UNIX utilizando el emulador WINE (http://www.winehq.org/).

6.1.2 MacOSX

Mac OSX combina la facilidad gráfica de Windows (¡algunos dicen que lo mejora sustancialmente!) con la robustez de un sistema basado en UNIX (FreeBSD). Si bien uso una MacBook Air para todo mi trabajo "día a día", como la web/correo electrónico y el desarrollo del sitio QuantStart, me resultó extremadamente doloroso instalar una pila de investigación algorítmica completa, basada en Python, en Mac. OS X.

El panorama de paquetes de Mac OSX está significativamente fragmentado, siendo Homebrew y Mac-Ports los principales contendientes. La instalación desde la fuente es complicada debido a la propiedad

proceso de compilación (usando XCode). ¡Todavía no he instalado con éxito NumPy, SciPy y pandas en mi MacBook al momento de escribir este artículo!

Sin embargo, si puede navegar por el campo minado que es la instalación de Python en Mac OSX, puede proporcionar un excelente entorno para la investigación algorítmica. Dado que Interactive Brokers Trader Workstation está basada en Java, no tiene problemas para ejecutarse en Mac OSX.

6.1.3 Linux

"Linux" se refiere a un conjunto de distribuciones UNIX gratuitas como Cent OS, Debian y Ubuntu. No deseo entrar en detalles sobre los beneficios/inconvenientes de cada distribución, sino que me concentraré en la distribución basada en Debian. En particular, consideraré Ubuntu Desktop como el entorno comercial algorítmico.

La administración de paquetes de aptitude facilita la instalación de las bibliotecas subyacentes necesarias. Además, es sencillo crear un*ambiente virtual*para Python que puede aislar su código comercial de algo de otras aplicaciones de Python. Nunca he tenido ningún problema (importante) para instalar un entorno de Python en un sistema Ubuntu moderno y, como tal, lo he elegido como el entorno principal desde el que realizar mis operaciones.

Si desea probar Ubuntu antes de comprometerse por completo, por ejemplo, mediante un arranque dual, entonces es posible usar VirtualBox (https://www.virtualbox.org/) para instalarlo. Tengo una guía detallada sobre QuantStart (http://www.quantstart.com/articles/Installing-a-Desktop-Algorithmic-Trading-Research-Environment-using-Ubuntu-Linux-and-Python), que describe el proceso.

6.2 Instalación de un entorno de Python en Ubuntu Linux

En esta sección, discutiremos cómo configurar un entorno de desarrollo robusto, eficiente e interactivo para la investigación de estrategias comerciales algorítmicas utilizando Ubuntu Desktop Linux y el lenguaje de programación Python. Utilizaremos este entorno para todas las implementaciones comerciales algorítmicas posteriores.

Para crear el entorno de investigación, instalaremos las siguientes herramientas de software, todas ellas de código abierto y de descarga gratuita:

- Ubuntu Escritorio Linux -El sistema operativo
- Pitón -El entorno de programación central
- NumPy/SciPy -Para un cálculo de matriz/arreglo vectorizado rápido y eficiente
- IPython -Para el desarrollo interactivo visual con Python
- matplotlib -Para visualización gráfica de datos
- pandas -Para "disputas" de datos y análisis de series temporales
- scikit-aprender -Para algoritmos de aprendizaje automático e inteligencia artificial
- IbPy -Para realizar transacciones con la API de Interactive Brokers

Estas herramientas, junto con una base de datos maestra de valores MySQL adecuada, nos permitirán crear un entorno interactivo rápido de investigación de estrategias y backtesting. Pandas está diseñado para la "disputa de datos" y puede importar y limpiar datos de series temporales de manera muy eficiente. NumPy/SciPy ejecutándose debajo mantiene el sistema extremadamente bien optimizado. IPython/matplotlib (y la qtconsole que se describe a continuación) permiten la visualización interactiva de resultados y una iteración rápida. scikit-learn nos permite aplicar técnicas de aprendizaje automático a nuestras estrategias para mejorar aún más el rendimiento.

6.2.1 Pitón

Las últimas versiones de Ubuntu, que en el momento de escribir este artículo es la 13.10, todavía utilizan la familia de versiones Python 2.7.x. Si bien se está realizando una transición a 3.3.x, la mayoría de las bibliotecas son totalmente compatibles con la rama 2.7.x. Por lo tanto, he elegido usar esto para el comercio algorítmico. Sin embargo, es probable que las cosas evolucionen rápidamente, por lo que en un par de años 3.3.x puede ser la rama predominante. Ahora comenzaremos con la instalación del entorno Python.

Lo primero que debe hacer en cualquier nuevo sistema Ubuntu Linux es*actualizary mejora* los paquetes. El primero le informa a Ubuntu sobre los nuevos paquetes que están disponibles, mientras que el segundo realiza el proceso de reemplazar los paquetes más antiguos con versiones más nuevas. Ejecute los siguientes comandos en una sesión de terminal y se le solicitarán sus contraseñas:

```
sudo apt-get -y actualizar sudo
apt-get -y actualizar
```

Tenga en cuenta que el prefijo -y le dice a Ubuntu que desea aceptar 'sí' a todas las preguntas de sí/no. "sudo" es un comando de Ubuntu/Debian Linux que permite ejecutar otros comandos con privilegios de administrador. Dado que estamos instalando nuestros paquetes en todo el sitio, necesitamos 'acceso raíz' a la máquina y, por lo tanto, debemos usar 'sudo'.

Una vez que ambos comandos de actualización se hayan ejecutado con éxito, debemos instalar los paquetes de desarrollo de Python y los compiladores necesarios para compilar todo el software. Observe que estamos instalandoconstruir-esencialque contiene los compiladores GCC y la biblioteca de álgebra lineal LAPACK, así comopepitaque es el sistema de gestión de paquetes de Python:

```
sudo apt-get install python-pip python-dev python2.7-dev \ build-essential
liblapack-dev libblas-dev
```

La siguiente etapa es instalar las bibliotecas de análisis numérico y de datos de Python.

6.2.2 NumPy, SciPy y Pandas

Una vez que los paquetes necesarios están instalados arriba, podemos continuar e instalar NumPy a través de pip, el administrador de paquetes de Python. Pip descargará un archivo zip del paquete y luego lo compilará desde el código fuente para nosotros. Tenga en cuenta que la compilación llevará algún tiempo, posiblemente 10 minutos o más, dependiendo de su CPU:

```
sudo pip instalar numpy
```

Una vez que se haya instalado NumPy, debemos verificar que funcione antes de continuar. Si miras en la terminal, verás tu nombre de usuario seguido del nombre de tu computadora. en mi caso es mhallsmoore@algobox ,que es seguido por el aviso. En el tipo de solicitudpitóny luego intente importar NumPy. Probaremos que funciona calculando el promedio medio de una lista:

```
mhallsmoore@algobox :~$ python
Python 2.7.4 (predeterminado, 26 de septiembre de 2013,
03:20:26) [GCC 4.7.3] en linux2
Escriba "ayuda", "derechos de autor", "créditos"o"licencia"pormás información.
> > > importarentumecido
> > > deentumecidoimportarsignificar
> > > media([1,2,3])
2.0
> > > salir()
```

Ahora que NumPy se ha instalado correctamente, queremos instalar la biblioteca Python Scientific conocida como SciPy. Tiene algunas dependencias de paquetes propias, incluida la biblioteca ATLAS y el compilador GNU Fortran, que debe instalarse primero:

```
sudo apt-get install libatlas-base-dev gfortran
```

Estamos listos para instalar SciPy ahora, con pip. La compilación llevará bastante tiempo, tal vez entre 10 y 20 minutos, dependiendo de la velocidad de la CPU:

```
sudo pip instalar scipy
```

SciPy ahora se ha instalado. Lo probaremos de manera similar a NumPy al calcular la desviación estándar de una lista de enteros:

```
mhallsmoore@algobox :~$ python
Python 2.7.4 (predeterminado, 26 de septiembre de 2013,
03:20:26) [GCC 4.7.3] en linux2
Escriba "ayuda", "derechos de autor", "créditos"o"licencia"pormás información.
> > > importarespía
> > > deespíaimportarestándar
> > > estándar([1,2,3])
0.81649658092772603
> > > salir()
```

La tarea final de esta sección es instalar la biblioteca de análisis de datos de pandas. No necesitamos dependencias adicionales en esta etapa, ya que están cubiertas por NumPy y SciPy:

```
sudo pip instalar pandas
```

Ahora podemos probar la instalación de pandas, como antes:

```
> > > depandasimportarMarco de datos
> > > pd = trama de datos()
> > > PD
Marco de datos vacío
Columnas: []
Índice: []
> > > salir()
```

Ahora que se han instalado las bibliotecas numéricas y científicas básicas, instalaremos las bibliotecas estadísticas y de aprendizaje automático, statsmodels y scikit-learn.

6.2.3 Modelos estadísticos y Scikit-Learn

La instalación continúa como antes, haciendo uso de pip para instalar los paquetes:

```
sudo pip install statsmodels sudo pip
install scikit-learn
```

Ambas bibliotecas se pueden probar:

```
mhallsmoore@algobox :~$ python
Python 2.7.4 (predeterminado, 26 de septiembre de 2013,
03:20:26) [GCC 4.7.3] en linux2
Escriba "ayuda", "derechos de autor", "créditos"o"licencia"pormás información.
> > > deaprenderimportarconjuntos de datos
> > > iris = conjuntos de datos.load_iris()
> > > iris
. .
. .
'ancho del pétalo (cm)']}
> > >
```

Ahora que las dos bibliotecas estadísticas están instaladas, podemos instalar las herramientas de visualización y desarrollo, IPython y matplotlib.

6.2.4 PyQt, IPython y Matplotlib

La primera tarea es instalar los paquetes de dependencia para matplotlib, la biblioteca gráfica de Python. Dado que matplotlib es un paquete de Python, no podemos usar pip para instalar las bibliotecas subyacentes para trabajar con PNG, JPEG y fuentes de tipo libre, por lo que necesitamos que Ubuntu las instale por nosotros:

```
sudo apt-get install libpng-dev libjpeg8-dev libfreetype6-dev
```

Ahora podemos instalar matplotlib:

```
sudo pip instalar matplotlib
```

La última tarea de esta sección es instalar IPython. Este es un intérprete de Python interactivo que proporciona un flujo de trabajo significativamente más optimizado en comparación con el uso de la consola estándar de Python. En capítulos posteriores, enfatizaremos toda la utilidad de IPython para el desarrollo del comercio algorítmico:

```
sudo pip instalar ipython
```

Si bien IPython es lo suficientemente útil por sí solo, puede hacerse aún más poderoso al incluir el*qtconsola*, que proporciona la capacidad de visualizar visualizaciones de matplotlib en línea. Sin embargo, se necesita un poco más de trabajo para poner esto en marcha.

Primero, necesitamos instalar elBiblioteca qt:

```
sudo apt-get install libqt4-core libqt4-gui libqt4-dev
```

qtconsole tiene algunos paquetes de dependencia adicionales, a saber, las bibliotecas ZMQ y Pygments:

```
sudo apt-get install libzmq-dev sudo pip
install pyzmq
sudo pip instalar pigmentos
```

Es sencillo probar IPython escribiendo el siguiente comando:

```
ipython qtconsole --pylab=en línea
```

Para probar IPython, se puede generar un gráfico simple escribiendo los siguientes comandos. Tenga en cuenta que he incluido la entrada/salida numerada de IPython que no necesita escribir:

```
En [1]: x=np.matriz([1,2,3])

En [2]: parcela(x)
Salida[2]: [<matplotlib.lines.Line2D en 0x392a1d0>]
```

Esto debería mostrar un gráfico matplotlib en línea. Cerrar IPython nos permite continuar con la instalación.

6.2.5 IbPy y Trader Workstation

Interactive Brokers es una de las principales casas de bolsa utilizadas por los comerciantes algorítmicos minoristas debido a sus requisitos mínimos de saldo de cuenta relativamente bajos (10,000 USD) y API (relativamente) sencilla. En esta sección instalaremos IbPy y Trader Workstation, que luego utilizaremos para llevar a cabo la ejecución automatizada de operaciones.

¡Quiero enfatizar que no vamos a intercambiar capital vivo con esta descarga! Simplemente vamos a instalar un software que nos permitirá probar una "cuenta de demostración", que proporciona un simulador de mercado con datos desactualizados en "tiempo real".

Divulgación: no tengo afiliación con Interactive Brokers. Los he usado antes en un contexto de fondos profesionales y, como tal, estoy familiarizado con su software.

IbPy es un contenedor de Python escrito en torno a la API de Interactive Brokers basada en Java. Hace que el desarrollo de sistemas comerciales algorítmicos en Python sea algo menos problemático. Se utilizará como base para todas las comunicaciones posteriores con Interactive Brokers. Una alternativa es usar el protocolo FIX, pero no consideraremos ese método en este libro.

Dado que IBPy se mantiene en el sitio web de control de versiones del código fuente de GitHub, como repositorio de git, necesitaremos instalar git. Esto es manejado por:

```
sudo apt-get install git-core
```

Una vez instalado git es necesario crear un subdirectorio para almacenar IBPy. Simplemente se puede colocar debajo del directorio de inicio:

```
mkdir ~/ibapi
```

El siguiente paso es descargar IBPy a través del comando 'git clone':

```
cd ~/ibapi
clon de git https://github.com/blampe/IbPy
```

El paso final es ingresar al directorio IbPy e instalar usando las herramientas de configuración de Python:

```
cd ~/ibapi/IbPy
configuración de python.py.enInstalar en pc
```

Eso completa la instalación de IBPy. El siguiente paso es instalar Trader Workstation. Al momento de escribir este artículo, era necesario seguir este enlace (IB), que lo lleva directamente a la página de descarga de Trader Workstation en Interactive Brokers. Seleccione la plataforma que desea utilizar. En este caso, he elegido la descarga de UNIX, que se puede encontrar aquí (Descarga de IB Unix).

En ese enlace se describirá el resto del proceso, pero lo replicaré aquí para completarlo. El archivo descargado se llamará *unixmacosx_latest.jar*. Abre el archivo:

```
jar xf unixmacosx_latest.jar
```

Luego cambie al directorio IBJts y cargue TWS:

```
discos compactos
java -cp jts.jar:total.2013.jar -Xmx512M -XX:MaxPermSize=128M jclient.LoginFr ame
```

Esto le presentará la pantalla de inicio de sesión de Trader Workstation. Si elige el nombre de usuario "edemo" y la contraseña "usuario de demostración", iniciará sesión en el sistema.

Esto completa la instalación de un entorno comercial algorítmico completo bajo Python y Ubuntu. La siguiente etapa es comenzar a recopilar y almacenar datos de precios históricos para nuestras estrategias.

Capítulo 7

Almacenamiento de datos financieros

En el comercio algorítmico, el centro de atención suele centrarse en el*modelo alfa*componente del sistema de comercio completo. Este componente genera las señales de negociación, previo filtrado por un sistema de gestión de riesgos y construcción de carteras. Como tal, los comerciantes de algoritmos a menudo dedican una parte significativa de su tiempo de investigación a refinar el modelo alfa para optimizar una o más métricas antes de implementar la estrategia en producción.

Sin embargo, un modelo alfa es tan bueno como los datos que se introducen en él. Este concepto está muy bien caracterizado por el viejo adagio informático de*"basura dentro basura fuera."*Es absolutamente crucial que se utilicen datos precisos y oportunos para alimentar el modelo alfa. De lo contrario, los resultados serán, en el mejor de los casos, deficientes o, en el peor de los casos, completamente incorrectos. Esto conducirá a un bajo rendimiento significativo cuando el sistema se implemente en vivo.

En este capítulo, analizaremos los problemas relacionados con la adquisición y el suministro de datos oportunos y precisos para un sistema de backtesting de estrategia algorítmica y, en última instancia, un motor de ejecución comercial. En particular, estudiaremos cómo obtener datos financieros y cómo almacenarlos. Los capítulos siguientes discutirán cómo limpiarlo y cómo exportarlo. En la industria financiera, este tipo de servicio de datos se conoce comobase de datos maestra de valores.

7.1 Bases de datos maestras de valores

Un maestro de valores es una base de datos de toda la organización que almacenafundamental, fijación de preciosy transaccionaldatos para una variedad de instrumentos financieros a través de clases de activos. Proporciona acceso a esta información de manera coherente para que la utilicen otros departamentos, como gestión de riesgos, compensación/liquidación y negociación por cuenta propia.

En organizaciones grandes, se almacenará una variedad de instrumentos y datos. Estos son algunos de los instrumentos que pueden ser de interés para una empresa:

- Acciones
- Opciones de acciones
- Índices
- Divisas
- Tasas de interés
- Futuros
- Productos básicos
- Bonos - Gobierno y Corporativo
- Derivados - Caps, Floors, Swaps

Las bases de datos maestras de valores a menudo tienen equipos de desarrolladores y especialistas en datos que garantizan *alta disponibilidad*dentro de una entidad financiera. Si bien esto es necesario en las grandes empresas, a nivel minorista o en un fondo pequeño, un maestro de valores puede ser mucho más simple. De hecho, mientras que los grandes maestros de valores hacen uso de costosos sistemas de análisis y bases de datos empresariales, es posible utilizar software de código abierto básico para proporcionar el mismo nivel de funcionalidad, suponiendo un sistema bien optimizado.

7.2 Conjuntos de datos financieros

Para el comerciante minorista algorítmico o el pequeño fondo cuantitativo, los conjuntos de datos más comunes son los precios históricos al final del día y dentro del día para acciones, índices, futuros (principalmente materias primas o renta fija) y divisas (forex). Para simplificar esta discusión, nos concentraremos únicamente en los datos al final del día (EOD) para acciones, ETF e índices de acciones. Las secciones posteriores discutirán la adición de datos de mayor frecuencia, clases de activos adicionales y datos de derivados, que tienen requisitos más avanzados.

Los datos EOD para acciones son fáciles de obtener. Hay una serie de servicios que brindan acceso de forma gratuita a través de API disponibles en la web:

- Yahoo Finanzas -http://finanzas.yahoo.com
- Finanzas de Google -https://www.google.com/finanzas
- Cotización cuantitativa -https://www.quantquote.com (solo datos EOD de S&P500)
- Datos EOD -http://eoddata.com (requiere registro)

Es sencillo descargar manualmente datos históricos para valores individuales, pero lleva mucho tiempo si es necesario descargar muchas acciones diariamente. Por lo tanto, un componente importante de nuestro maestro de valores será la actualización automática del conjunto de datos.

Otro problema es*período de revisión*. ¿Qué tan lejos en el pasado tenemos que ir con nuestros datos? Esto será específico para los requisitos de su estrategia comercial, pero existen ciertos problemas que abarcan todas las estrategias. El más común esCambio de regimén,que a menudo se caracteriza por un nuevo entorno regulatorio, períodos de mayor/menor volatilidad o mercados con tendencias a más largo plazo. Por ejemplo, una estrategia de impulso/seguimiento de tendencia a corto plazo a largo plazo probablemente funcionaría muy bien entre 2000 y 2003 o entre 2007 y 2009. Sin embargo, habría tenido un momento difícil desde 2003-2007 o 2009 hasta el presente.

Mi regla general es obtener la mayor cantidad de datos posible, especialmente para datos EOD donde el almacenamiento es barato. El hecho de que los datos existan en su maestro de seguridad no significa que deban utilizarse. Hay advertencias sobre el rendimiento, ya que las tablas de base de datos más grandes significan tiempos de consulta más prolongados (consulte a continuación), pero los beneficios de tener más puntos de muestra generalmente superan cualquier problema de rendimiento.

Al igual que con todos los datos financieros, es imperativo estar al tanto de los errores, como precios altos/bajos incorrectos osesgo de supervivencia,que he discutido extensamente en capítulos anteriores.

7.3 Formatos de almacenamiento

Hay tres formas principales de almacenar datos financieros. Todos ellos poseen diversos grados de acceso, rendimiento y capacidades estructurales. Consideraremos cada uno a su vez.

7.3.1 Almacenamiento de archivos planos

El almacén de datos más simple para datos financieros, y la forma en que es probable que reciba los datos de cualquier proveedor de datos, es el formato de archivo plano. Los archivos sin formato suelen utilizar el formato de variables separadas por comas (CSV), que almacenan una matriz bidimensional de datos como una serie de filas, con los datos de las columnas separados mediante un delimitador (a menudo una coma, pero puede ser un espacio en blanco, como como un espacio o pestaña). Para los datos de precios de EOD, cada fila representa un día de negociación a través del paradigma OHLC (es decir, los precios de apertura, máximo, mínimo y cierre del período de negociación).

La ventaja de los archivos planos es su simplicidad y la capacidad de comprimirse mucho para archivarlos o descargarlos. Las principales desventajas radican en su falta de capacidad de consulta y bajo rendimiento para la iteración en grandes conjuntos de datos.SQLiteySobresalirmitigar algunos de estos problemas proporcionando ciertas capacidades de consulta.

7.3.2 Almacenes de documentos/NoSQL

Los almacenes de documentos/bases de datos NoSQL, aunque ciertamente no son un concepto nuevo, han ganado una importancia significativa en los últimos años debido a su uso en empresas de "escala web" como Google, Facebook y Twitter. Se diferencian sustancialmente de los sistemas RDBMS en que no existe el concepto de esquemas de tablas. En cambio, hay*colecciones*y*documentos*, que son las analogías más cercanas a tablas y registros, respectivamente. Existe una amplia taxonomía de almacenes de documentos, ¡cuya discusión está fuera de este capítulo! Sin embargo, algunas de las tiendas más populares incluyenMongoDB, CassandrayCouchDB.

Los almacenes de documentos, en aplicaciones financieras, se adaptan principalmente a metadatos o datos fundamentales. Los datos fundamentales para los activos financieros vienen en muchas formas, como acciones corporativas, declaraciones de ganancias, presentaciones de la SEC, etc. Por lo tanto, la naturaleza sin esquema de las bases de datos NoSQL es adecuada. Sin embargo, las bases de datos NoSQL no están bien diseñadas para series temporales, como datos de precios de alta resolución, por lo que no las consideraremos más en este capítulo.

7.3.3 Sistemas de gestión de bases de datos relacionales

A*sistema de gestión de bases de datos relacionales*(RDBMS) hace uso de la*modelo relacional*para almacenar datos. Estas bases de datos son especialmente adecuadas para datos financieros porque diferentes "objetos" (como intercambios, fuentes de datos, precios) se pueden separar en tablas con relaciones definidas entre ellos.

RDBMS hace uso deLenguaje de consulta estructurado (SQL) para realizar consultas de datos complejas sobre datos financieros. Los ejemplos de RDBMS incluyenOracle, MySQL, SQLServer y PostgresSQL.

Las principales ventajas de RDBMS son su simplicidad de instalación, independencia de la plataforma, facilidad de consulta, facilidad de integración con los principales software de backtest y capacidades de alto rendimiento a gran escala (¡aunque algunos argumentarían que este último no es el caso!). Sus desventajas a menudo se deben a la complejidad de la personalización y las dificultades para lograr dicho rendimiento sin un conocimiento subyacente de cómo se almacenan los datos RDBMS. Además, poseen esquemas semirrígidos y, por lo tanto, a menudo los datos deben modificarse para adaptarse a dichos diseños. Esto es diferente a los almacenes de datos NoSQL, donde no hay esquema.

Para todo el futuro código histórico de implementación de precios del libro, utilizaremos MySQL RDBMS. Es gratuito, de código abierto, multiplataforma, muy robusto y su comportamiento a escala está bien documentado, lo que lo convierte en una opción sensata para el trabajo cuantitativo.

7.4 Estructura de datos históricos

Existe un importante cuerpo de teoría e investigación académica llevada a cabo en el ámbito de la informática para el diseño óptimo de los almacenes de datos. Sin embargo, no entraremos en demasiados detalles ya que es fácil perderse en minucias. En su lugar, presentaré un patrón común para la construcción de un maestro de seguridad de acciones, que puede modificar como mejor le parezca para sus propias aplicaciones.

La primera tarea es definir nuestra*entidades*, que son elementos de los datos financieros que finalmente se asignarán a tablas en la base de datos. Para una base de datos maestra de acciones, preveo las siguientes entidades:

- Intercambios -¿Cuál es la última fuente original de los datos?
- Vendedor -¿De dónde se obtiene un punto de datos en particular?
- Instrumento/Ticker -El ticker/símbolo de la acción o el índice, junto con la información corporativa de la empresa o el fondo subyacente.

- Precio -El precio real de un valor en particular en un día en particular.

- Acciones corporativas -La lista de todas las divisiones de acciones o ajustes de dividendos (esto puede conducir a una o más tablas), necesaria para ajustar los datos de precios.

- Días festivos nacionales -Para evitar clasificar erróneamente los feriados comerciales como errores de datos faltantes, puede ser útil almacenar los feriados nacionales y las referencias cruzadas.

Existen problemas significativos con respecto al almacenamiento de tickers canónicos. ¡Puedo dar fe de esto por experiencia de primera mano en un fondo de cobertura que se ocupa de este problema exacto! Diferentes proveedores usan diferentes métodos para resolver tickers y, por lo tanto, combinan múltiples fuentes para lograr precisión. Además, las empresas quiebran, están expuestas a actividades de fusiones y adquisiciones (es decir, son adquiridas y cambian de nombre/símbolos) y pueden tener varias clases de acciones que cotizan en bolsa. Muchos de ustedes no tendrán que preocuparse por esto porque su universo de tickers se limitará a los componentes de índices más grandes (como el S&P500 o el FTSE350).

7.5 Evaluación de la precisión de los datos

Los datos históricos de precios de los proveedores son propensos a muchas formas de error:

- Acciones corporativas -Manejo incorrecto de splits de acciones y ajustes de dividendos. Uno debe estar absolutamente seguro de que las fórmulas se han implementado correctamente.

- Picos -Puntos de precios que superan con creces ciertos niveles históricos de volatilidad. Uno debe tener cuidado aquí, ya que estos picos ocurren: vea May Flash Crash para ver un ejemplo aterrador. Los picos también pueden ser causados por no tener en cuenta las divisiones de acciones cuando ocurren.*filtro de picos* Los scripts se utilizan para notificar a los comerciantes de tales situaciones.

- Agregación OHLC -Los datos gratuitos de OHLC, como los de Yahoo/Google, son particularmente propensos a situaciones de "agregación de ticks incorrecta" en las que los intercambios más pequeños procesan operaciones pequeñas muy por encima de los precios de intercambio "principales" del día, lo que lleva a máximos/mínimos sobreinflados una vez agregados. Esto es menos un "error" como tal, pero más un problema del que hay que tener cuidado.

- Datos perdidos -La falta de datos puede deberse a la falta de transacciones en un período de tiempo particular (común en los datos de resolución de segundo/minuto de empresas de pequeña capitalización sin liquidez), por vacaciones comerciales o simplemente un error en el sistema de intercambio. Los datos faltantes pueden ser*acolchado*(es decir, lleno con el valor anterior),*interpolado*(linealmente o de otra manera) o ignorados, dependiendo del sistema comercial.

Muchos de estos errores se basan en el juicio manual para decidir cómo proceder. Es posible automatizar la notificación de dichos errores, pero es mucho más difícil automatizar su solución. Por ejemplo, uno debe elegir el umbral para que le informen sobre los picos: ¿cuántas desviaciones estándar usar y durante qué período retrospectivo? Un stdev demasiado alto perderá algunos picos, pero demasiado bajo y muchos anuncios de noticias inusuales darán lugar a falsos positivos. Todos estos problemas requieren un juicio avanzado por parte del operador cuantitativo.

También es necesario decidir cómo corregir los errores. ¿Deberían corregirse los errores tan pronto como se conozcan y, de ser así, debería realizarse un seguimiento de auditoría? Esto requerirá una tabla adicional en la base de datos. Esto nos lleva al tema del backfilling, que es un tema particularmente insidioso para el backtesting. Se trata de la corrección automática de datos incorrectos aguas arriba. Si su proveedor de datos corrige un error histórico, pero se está produciendo una estrategia de negociación comprobada basada en la investigación de sus datos anteriores incorrectos, entonces se deben tomar decisiones con respecto a la efectividad de la estrategia. Esto puede mitigarse en cierta medida al ser plenamente consciente de las métricas de rendimiento de su estrategia (en particular, la variación en sus características de ganancia/pérdida para cada operación).

7.6 Automatización

El beneficio de escribir secuencias de comandos de software para realizar la descarga, el almacenamiento y la limpieza de los datos es que las secuencias de comandos se pueden automatizar a través de herramientas proporcionadas por el sistema operativo. En sistemas basados en UNIX (como Mac OSX o Linux), se puede hacer uso de lacrontab,que es un proceso de ejecución continua que permite que se ejecuten scripts específicos en momentos definidos de forma personalizada o en períodos regulares. Existe un proceso equivalente en MS Windows conocido como Programador de tareas.

Un proceso de producción, por ejemplo, podría automatizar la descarga de todos los precios de fin de día del S&P500 tan pronto como se publiquen a través de un proveedor de datos. A continuación, ejecutará automáticamente los datos faltantes mencionados anteriormente y los scripts de filtración de picos, alertando al comerciante por correo electrónico, SMS o alguna otra forma de notificación. En este punto, cualquier herramienta de backtesting tendrá acceso automáticamente a los datos recientes, ¡sin que el comerciante tenga que mover un dedo! Dependiendo de si su sistema comercial está ubicado en una computadora de escritorio o en un servidor remoto, puede optar por tener un proceso semiautomático o totalmente automático para estas tareas.

7.7 Disponibilidad de datos

Una vez que los datos se actualizan automáticamente y residen en el RDBMS, es necesario ingresarlos al software de backtesting. Este proceso dependerá en gran medida de cómo esté instalada su base de datos y de si su sistema comercial es local (es decir, en una computadora de escritorio) o remoto (como con un servidor de intercambio ubicado en el mismo lugar).

Una de las consideraciones más importantes es minimizar la Entrada/Salida (E/S) excesiva, ya que esto puede ser extremadamente costoso tanto en términos de tiempo como de dinero, asumiendo conexiones remotas donde el ancho de banda es costoso. La mejor manera de abordar este problema es mover solo los datos a través de una conexión de red que necesite (a través de consultas selectivas) o exportar y comprimir los datos.

Soporte para muchos RDBMS*replicación*tecnología que permite clonar una base de datos en otro sistema remoto, generalmente con cierto grado de latencia. Dependiendo de su configuración y cantidad de datos, esto puede ser solo del orden de minutos o segundos. Un enfoque simple es replicar una base de datos remota en un escritorio local. Sin embargo, tenga en cuenta que los problemas de sincronización son comunes y requieren mucho tiempo para solucionarse.

7.8 MySQL para maestros de valores

Ahora que hemos discutido la idea detrás de una base de datos maestra de seguridad, es hora de construir una. Para ello haremos uso de dos tecnologías de código abierto: lamysqlbase de datos y la Pitónlenguaje de programación. Al final de este capítulo, tendrá un maestro de seguridad de acciones completo con el que podrá realizar más análisis de datos para su investigación comercial cuantitativa.

7.8.1 Instalación de MySQL

La instalación de MySQL dentro de Ubuntu es sencilla. Simplemente abra una terminal y escriba lo siguiente:

```
sudo apt-get install mysql-server
```

Eventualmente, se le pedirá una contraseña de root. Esta es su contraseña de administración principal, ¡así que no la olvide! Ingrese la contraseña y la instalación continuará y finalizará.

7.8.2 Configuración de MySQL

Ahora que MySQL está instalado en su sistema, podemos crear un nuevo*base de datos*y un*usuario*para interactuar con él. Se le pedirá una contraseña de root en la instalación. Para iniciar sesión en MySQL desde la línea de comandos, use la siguiente línea y luego ingrese su contraseña:

```
mysql -u raíz -p
```

Una vez que haya iniciado sesión en MySQL, puede crear una nueva base de datos llamadamaestro_de_valores y luego seleccionarlo:

```
mysql> CREAR BASE DE DATOS valores_maestro;
mysql> USE valores_maestro;
```

Una vez que crea una base de datos, es necesario agregar una nueva*usuario*para interactuar con la base de datos. Si bien puedes usar elraízusuario, se considera una mala práctica desde el punto de vista de la seguridad, ya que otorga demasiados permisos y puede llevar a un sistema comprometido. En una máquina local, esto es irrelevante, pero en un entorno de producción remoto, seguramente necesitará crear un usuario con permisos reducidos. En este caso nuestro usuario será llamadoseg_user.Recuerda reemplazarclavecon una contraseña segura:

```
mysql> CREAR USUARIO 'sec_user'@'localhost' IDENTIFICADO POR 'contraseña'; mysql> OTORGAR TODOS LOS PRIVILEGIOS EN security_master.* A 'sec_user'@'localhost'; mysql> PRIVILEGIOS DE DESCARGA;
```

Las tres líneas anteriores crean y autorizan al usuario a usarmaestro_de_valoresy aplicar esos privilegios. A partir de ahora cualquier interacción que se produzca con la base de datos hará uso de laseg_userusuario.

7.8.3 Diseño de esquema para acciones EOD

Ahora hemos instalado MySQL y hemos configurado un usuario con el que interactuar con nuestra base de datos. En esta etapa estamos listos para construir las tablas necesarias para almacenar nuestros datos financieros. Para un maestro de acciones simple y directo, crearemos cuatro tablas:

- Intercambio -La tabla de intercambio enumera los intercambios de los que deseamos obtener información sobre precios de acciones. En este caso, será casi exclusivamente la Bolsa de Valores de Nueva York (NYSE) y la Asociación Nacional de Cotizaciones Automatizadas de Distribuidores de Valores (NASDAQ).
- Proveedor de datos -Esta tabla muestra información sobre proveedores de datos de precios históricos. Usaremos Yahoo Finance para obtener nuestros datos al final del día (EOD). Al presentar esta tabla, simplificamos la adición de más proveedores si es necesario, como Google Finance.
- Símbolo -La tabla de símbolos almacena la lista de símbolos de teletipo y la información de la empresa. En este momento, evitaremos problemas como las diferentes clases de acciones y los múltiples nombres de símbolos.
- Precio diario -Esta tabla almacena la información de precios diarios para cada valor. Puede llegar a ser muy grande si se agregan muchos valores. Por lo tanto, es necesario optimizarlo para el rendimiento.

MySQL es una base de datos extremadamente flexible que le permite personalizar cómo se almacenan los datos en un servidor subyacente.*motor de almacenamiento*. Los dos contendientes principales en MySQL sonMiISAM y InnoDB.Aunque no entraré en los detalles de los motores de almacenamiento (¡de los cuales hay muchos!), diré que MyISAM es más útil para la lectura rápida (como consultar grandes cantidades de información de precios), pero no admite transacciones (necesarias para*Retroceder*una operación de varios pasos que falla a mitad de camino). InnoDB, aunque es seguro para las transacciones, es más lento para las lecturas.

InnoDB también permite el bloqueo a nivel de fila al realizar escrituras, mientras que MyISAM bloquea toda la tabla al escribir en ella. Esto puede tener problemas de rendimiento cuando se escribe mucha información en puntos arbitrarios de la tabla (como con las instrucciones UPDATE). Este es un tema profundo, ¡así que dejaré la discusión para otro día!

Vamos a utilizar InnoDB, ya que es seguro para transacciones de forma nativa y proporciona bloqueo a nivel de fila. Si encontramos que una tabla tarda en leerse, podemos crear*índices*como primer paso y luego cambie el motor de almacenamiento subyacente si el rendimiento sigue siendo un problema. Todas nuestras tablas utilizarán el conjunto de caracteres UTF-8, ya que deseamos admitir intercambios internacionales.

Comencemos con el esquema yCREAR MESAcódigo SQL para elintercambiomesa. Almacena la abreviatura y el nombre de la bolsa (es decir, NYSE - Bolsa de valores de Nueva York), así como

la ubicación geográfica También admite una moneda y un desplazamiento de zona horaria de UTC. También almacenamos una fecha de creación y última actualización para nuestros propios fines internos. Finalmente, configuramos la clave de índice principal para que sea una ID de número entero de incremento automático (que es suficiente para manejar2^{32}registros):

```
CREAR TABLA 'intercambio' (
    'id' int NOT NULL AUTO_INCREMENT,
    'abbrev' varchar(32) NOT NULL, 'name'
    varchar(255) NOT NULL, 'city' varchar(255)
    NULL,
    'país' varchar(255) NULL, 'moneda' varchar(64)
    NULL, 'timezone_offset' time NULL, 'created_date'
    datetime NOT NULL, 'last_updated_date' datetime
    NOT NULL, PRIMARY KEY ('id')


) MOTOR = InnoDB AUTO_INCREMENT = 1 JUEGO DE CARACTERES POR DEFECTO = utf8;
```

Aquí está el esquema y`CREAR MESA`código SQL para el`vendedor_de_datos`mesa. Almacena el nombre, el sitio web y el correo electrónico de soporte. Con el tiempo, podemos agregar más información útil para el proveedor, como una URL de punto final de API:

```
CREAR TABLA 'proveedor_datos' (
    'id' int NOT NULL AUTO_INCREMENT, 'name'
    varchar(64) NOT NULL, 'website_url' varchar(255)
    NULL, 'support_email' varchar(255) NULL,
    'created_date' datetime NOT NULL,
    'last_updated_date' datetime NOT NULL, CLAVE
    PRINCIPAL ('id')

) MOTOR = InnoDB AUTO_INCREMENT = 1 JUEGO DE CARACTERES POR DEFECTO = utf8;
```

Aquí está el esquema y`CREAR MESA`código SQL para el`símbolo`mesa. Contiene un enlace de clave externa a un intercambio (por el momento, solo admitiremos instrumentos cotizados en bolsa), un símbolo de cotización (por ejemplo, GOOG), un tipo de instrumento ('acción' o 'índice'), el nombre del índice bursátil o bursátil, un sector de acciones y una moneda.

```
CREAR TABLA 'símbolo' (
    'id' int NOT NULL AUTO_INCREMENT,
    'exchange_id' int NULL,
    'ticker' varchar(32) NO NULO, 'instrumento'
    varchar(64) NO NULO, 'nombre' varchar(255)
    NULO,
    'sector' varchar(255) NULL, 'currency' varchar(32)
    NULL, 'created_date' datetime NOT NULL,
    'last_updated_date' datetime NOT NULL, PRIMARY
    KEY ('id'),

    CLAVE 'index_exchange_id' ('exchange_id')
) MOTOR = InnoDB AUTO_INCREMENT = 1 JUEGO DE CARACTERES POR DEFECTO = utf8;
```

Aquí está el esquema y`CREAR MESA`código SQL para el`Precio diario`mesa. Esta tabla es donde se almacenan realmente los datos históricos de precios. Hemos prefijado el nombre de la tabla con `diariamente_`ya que es posible que deseemos crear datos de resolución de minutos o segundos en tablas separadas en una fecha posterior para estrategias de mayor frecuencia. La tabla contiene dos claves foráneas: una para el proveedor de datos y otra para un símbolo. Esto identifica de manera única el punto de datos y nos permite almacenar los mismos datos de precios para múltiples proveedores en la misma tabla. También almacenamos una fecha de precio (es decir, el período diario durante el cual los datos de OHLC son válidos) y las fechas de creación y última actualización para nuestros propios fines.

Los campos restantes almacenan los precios de apertura-alto-bajo-cierre y cierre ajustado. Yahoo Finance nos proporciona dividendos y divisiones de acciones, cuyo precio termina en el`adj_close_price`

columna. Observe que el tipo de datos esdecimal(19,4).Cuando se trata de datos financieros, es absolutamente necesario ser preciso. Si hubiéramos usado elflotartipo de datos terminaríamos con errores de redondeo debido a la naturaleza de cómoflotarlos datos se almacenan internamente. El último campo almacena el volumen de negociación del día. Esto usa elEmpezandotipo de datos para que no trunquemos accidentalmente días de volumen extremadamente alto.

```
CREAR TABLA 'precio_diario' (
    'id' int NOT NULL AUTO_INCREMENT,
    'data_vendor_id' int NOT NULL, 'symbol_id'
    int NOT NULL,
    'price_date' datetime NOT NULL, 'created_date'
    datetime NOT NULL, 'last_updated_date' datetime
    NOT NULL, 'open_price' decimal(19,4) NULL,
    'high_price' decimal(19,4) NULL, 'low_price'
    decimal(19, 4) NULL, 'close_price' decimal (19,4)
    NULL, 'adj_close_price' decimal (19,4) NULL,
    'volumen' bigint NULL,


    CLAVE PRIMARIA ('id'),
    KEY 'index_data_vendor_id' ('data_vendor_id'), KEY
    'index_symbol_id' ('symbol_id')
) MOTOR = InnoDB AUTO_INCREMENT = 1 JUEGO DE CARACTERES POR DEFECTO = utf8;
```

Al ingresar todos los comandos SQL anteriores en la línea de comandos de MySQL, se crearán las cuatro tablas necesarias.

7.8.4 Conexión a la base de datos

Antes de que podamos usar MySQL con Python, necesitamos instalar elcliente mysqlbiblioteca.cliente mysql es en realidad una bifurcación de otra biblioteca, conocida como Python-MySQL. Desafortunadamente, la última biblioteca no es compatible con Python3, por lo que debemos usarcliente mysql.En las máquinas tipo Mac OSX/UNIX necesitamos ejecutar los siguientes comandos:

```
sudo apt-get install libmysqlclient-dev pip install
mysqlclient
```

Ahora estamos listos para comenzar a interactuar con nuestra base de datos MySQL a través de Python y pandas.

7.8.5 Uso de un asignador relacional de objetos

Para aquellos de ustedes con experiencia en administración y desarrollo de bases de datos, es posible que se pregunten si es más sensato utilizar unAsignador relacional de objetos (ORM). Un ORM permite que los objetos dentro de un lenguaje de programación se asignen directamente a las tablas en las bases de datos, de modo que el código del programa desconozca por completo el motor de almacenamiento subyacente. No están exentos de problemas, pero pueden ahorrar una gran cantidad de tiempo. Sin embargo, el ahorro de tiempo generalmente se produce a expensas del rendimiento.

Un ORM popular para Python esAlquimia SQL.Le permite especificar el esquema de la base de datos dentro de Python mismo y así generar automáticamente elCREAR MESAcódigo. Dado que hemos elegido específicamente MySQL y nos preocupa el rendimiento, he optado por no utilizar un ORM para este capítulo.

Recuperación de símbolos

Comencemos por obtener todos los símbolos de cotización asociados con la lista de Standard & Poor's de 500 acciones de gran capitalización, es decir, el S&P500. Por supuesto, esto es simplemente un ejemplo. Si opera desde el Reino Unido y desea utilizar índices nacionales del Reino Unido, también podría obtener la lista de empresas FTSE100 que cotizan en la Bolsa de Valores de Londres (LSE).

Wikipedia enumera convenientemente los componentes del S&P500. ¡Tenga en cuenta que en realidad hay 502 componentes en el S&P500! Lo haremos*raspar*el sitio web usando Pythonpeticionesy HermosaSopa bibliotecas y luego agregue el contenido directamente a MySQL. En primer lugar, asegúrese de que las bibliotecas estén instaladas:

```
solicitudes de instalación de pip
pip instalar beautifulsoup4
```

El siguiente código usará las solicitudes y las bibliotecas BeautifulSoup para agregar los símbolos directamente a la base de datos MySQL que creamos anteriormente. Recuerde reemplazar 'contraseña' con su contraseña elegida como se creó anteriormente:

```
#!/usr/bin/python
# - * - codificación: utf-8 -*-

# insertar_simbolos.py

de__futuro__importarimprimir_funcion

importarfecha y hora
deMatemáticasimportarhacer techo

importarbs4
importarMySQLdb como mdb
importarpeticiones


definitivamenteobtener_parse_wiki_snp500():
    """
    Descargue y analice la lista de Wikipedia de los componentes
    del S&P500 mediante solicitudes y BeautifulSoup.

    Devuelve una lista de tuplas para agregar a MySQL. """

    # Almacena la hora actual, para el registro created_at ahora =
    fechahora.fechahora.utcnow()

    # Use las solicitudes y BeautifulSoup para descargar el
    # lista de empresas S&P500 y obtener la tabla de símbolos respuesta
    = solicitudes.obtener(
        "http://en.wikipedia.org/wiki/List_of_S%26P_500_companies"
    )
    sopa = bs4.HermosaSopa(respuesta.texto)

    # Esto selecciona la primera tabla, usando la sintaxis del Selector de CSS
    # y luego ignora la fila del encabezado ([1:]) lista de simbolos =
    sopa.select('tabla')[0].select('tr')[1:]

    # Obtenga la información del símbolo para cada
    # fila en la tabla de componentes del S&P500
    símbolos = []
    poryo, símboloenenumerar (lista de símbolos):
        tds = símbolo.select('td')
        símbolos.append(
            (
                tds[0].select('a')[0].texto, 'acción',      # teletipo

                tds[1].select('a')[0].texto,                # Nombre
```

```
                        tds[3].texto,#Sector
                        'USD', ahora, ahora
                    )
                )
        devolversimbolos


definitivamenteinsert_snp500_symbols(símbolos):
        """
        Inserte los símbolos S&P500 en la base de datos MySQL. """

        # Conéctese a la instancia de MySQL
        db_host = 'localhost'
        usuario_bd = 'usuario_sec'
        db_pass = 'contraseña'
        db_name = 'securities_master' con =
        mdb.connect(
                host=db_host, usuario=db_usuario, passwd=db_pass, db=nombre_db
        )

        # Crear las cadenas de inserción
        columna_str = """ticker, instrumento, nombre, sector,
                         moneda, created_date, last_updated_date """

        insert_str = ("%s, " * 7)[:-2]
        final_str = "INSERTAR EN el símbolo (%s) VALORES (%s)" % \
                (cadena_columna, cadena_inserto)

        # Usando la conexión MySQL, lleve a cabo
        # un INSERT INTO para cada símbolo con
        estafa:
                cur = con.cursor()
                cur.executemany(final_str, símbolos)


si__nombre__ == "__principal__":
        símbolos = obtener_parse_wiki_snp500()
        insert_snp500_symbols(símbolos)
        impresión("%s símbolos se agregaron con éxito." % len(símbolos))
```

En esta etapa, tendremos los 502 componentes de símbolos actuales del índice S&P500 en la base de datos. Nuestra próxima tarea es obtener los datos de precios históricos de fuentes separadas y unirlos con los símbolos.

Recuperación de precios

Para obtener los datos históricos de los componentes actuales del S&P500, primero debemos consultar la base de datos para obtener la lista de todos los símbolos.

Una vez que se haya devuelto la lista de símbolos, junto con las identificaciones de los símbolos, es posible llamar a la API de finanzas de Yahoo y descargar los datos de precios históricos para cada símbolo.

Una vez que tenemos cada símbolo, podemos insertar los datos en la base de datos por turnos. Aquí está el código de Python para llevar a cabo esto:

```
#!/usr/bin/python
# - * - codificación: utf-8 -*-

# precio_recuperación.py
```

```
de__futuro__importarimprimir_funcion

importarfecha y hora
importaradvertencias

importarMySQLdb como mdb
importarpeticiones


# Obtenga una conexión de base de datos a la instancia de MySQL
db_host = 'localhost'
usuario_bd = 'usuario_sec'
db_pass = 'contraseña'
db_name = 'seguros_maestro'
con = mdb.connect(db_host, db_user, db_pass, db_name)


definitivamenteobtener_lista_de_tickers_bd():
    """
    Obtiene una lista de los símbolos de cotización en la base de datos. """

    con estafa:
        cur = con.cursor()
        cur.execute("SELECCIONE id, ticker DESDE símbolo") data =
        cur.fetchall()
        devolver[(d[0], d[1])pordendatos]


definitivamenteget_daily_historic_data_yahoo(
        ticker, start_date=(2000,1,1),
        end_date=datetime.date.today().timetuple()[0:3]
    ) :
    """
    Obtiene datos de las devoluciones de Yahoo Finance y una lista de tuplas.

    ticker: símbolo de Yahoo Finance, p. ej., "GOOG" para Google, Inc. start_date: fecha
    de inicio en formato (AAAA, M, D)
    end_date: fecha de finalización en formato (AAAA, M, D) """

    # Construya la URL de Yahoo con los parámetros de consulta enteros correctos
    # para las fechas de inicio y finalización. Tenga en cuenta que algunos parámetros están basados   en
    cero. ticker_tup = (
        teletipo, fecha_inicio[1]-1, fecha_inicio[2], fecha_inicio[0],
        fecha_finalización[1]-1, fecha_finalización[2],
        fecha_finalización[0]
    )
    yahoo_url = "http://ichart.finance.yahoo.com/table.csv" yahoo_url += "?
    s=%s&a=%s&b=%s&c=%s&d=%s&e=%s&f=%s" yahoo_url = yahoo_url %
    ticker_tup

    # Intenta conectarte a Yahoo Finance y obtener los datos
    # En caso de falla, imprime un mensaje de error.
    probar:
        yf_data = solicitudes.get(yahoo_url).text.split("\n")[1:-1] precios = []
```

```
            poryenyf_datos:
                p = y.strip().split(',')
                precios.append(
                    (fechahora.fechahora.strptime(p[0], '%Y-%m-%d'), p[1], p[2],
                    p[3], p[4], p[5], p [6])
                )
        exceptoExcepción como e:
            impresión("No se pudieron descargar los datos de Yahoo: %s" %e)
        devolverprecios


definitivamenteinsert_daily_data_into_db(
            data_vendor_id, symbol_id, daily_data
        ) :
        """
        Toma una lista de tuplas de datos diarios y la agrega a la base de datos MySQL. Agrega la
        identificación del proveedor y la identificación del símbolo a los datos.

        daily_data: Lista de tuplas de los datos OHLC (con adj_close y
        volumen)
        """
        # Crea el tiempo ahora
        ahora = fechahora.fechahora.utcnow()

        # Modifique los datos para incluir la identificación del proveedor y la identificación del
        símbolo datos_diarios = [
            (id_proveedor_datos, id_símbolo, d[0], ahora, ahora, d[1],
            d[2], d[3], d[4], d[5], d[6])
            pordendatos_diarios
        ]

        # Crear las cadenas de inserción
        columna_str = """data_vendor_id, symbol_id, price_date, created_date,
                        last_updated_date, open_price, high_price, low_price, close_price,
                        volumen, adj_close_price"""
        insert_str = ("%s, " * 11)[:-2]
        final_str = "INSERTAR EN VALORES de precio_diario (%s) (%s)" % \
            (cadena_columna, cadena_inserto)

        # Usando la conexión MySQL, realice un INSERT INTO para cada símbolo con estafa:

            cur = con.cursor()
            cur.executemany(final_str, daily_data)


si__nombre__ == "__principal__":
        # Esto ignora las advertencias sobre el truncamiento de datos
        # de los tipos de datos de precisión de Yahoo a Decimal (19,4)
        advertencias.filterwarnings('ignorar')

        # Recorra los tickers e inserte el histórico diario
        # datos en la base de datos
        teletipos = obtener_lista_de_tickers_bd() lentickers
        = len(tickers)
        poresoenenumerar (tickers):
            impresión(
```

```
                "Agregando datos para %s: %s de %s" % (t[1],
                i+1, lentickers)
            )
            yf_data = get_daily_historic_data_yahoo(t[1])
            insert_daily_data_into_db('1', t[0], yf_data)
    impresión("Se agregaron correctamente los datos de precios de Yahoo Finance a la base de datos").
```

Tenga en cuenta que ciertamente hay formas en que podemos optimizar este procedimiento. Si hacemos uso de PythonScraPybiblioteca, por ejemplo, obtendríamos una alta concurrencia de las descargas, ya que ScraPy se basa en el eventoRetorcidoestructura. De momento cada descarga se realizará de forma secuencial.

7.9 Recuperación de datos del maestro de valores

Ahora que hemos descargado los precios históricos de todos los constituyentes actuales del S&P500, queremos poder acceder a ellos desde Python. lospandaslibrary hace que esto sea extremadamente sencillo. Aquí hay una secuencia de comandos que obtiene los datos Open-High-Low-Close (OHLC) para las acciones de Google durante un cierto período de tiempo de nuestra base de datos maestra de valores y genera el*cola* del conjunto de datos:

```
#!/usr/bin/python
# - * - codificación: utf-8 -*-

# recuperando_datos.py

de__futuro__importarimprimir_funcion

importarpandas como pd
importarMySQLdb como mdb


si__nombre__ == "__principal__":
    # Conéctese a la instancia de MySQL
    db_host = 'localhost'
    usuario_bd = 'usuario_sec'
    db_pass = 'contraseña'
    db_name = 'seguros_maestro'
    con = mdb.connect(db_host, db_user, db_pass, db_name)

    # Seleccione todos los datos históricos de cierre ajustados por Google
    sql = """SELECCIONE dp.price_date, dp.adj_close_price
                DESDE símbolo COMO sym INTERNO
                ÚNETE daily_price AS dp ON
                dp.symbol_id = sym.id WHERE
                sym.ticker = 'GOOG' ORDENAR POR
                dp.price_date ASC;"""

    # Crear un marco de datos de pandas a partir de la consulta SQL
    goog = pd.read_sql_query(sql, con=con, index_col='price_date')

    # Salida de la cola del marco de
    datos impresión(goog.tail())
```

La salida del script sigue:

```
                adj_close_price
precio_fecha
2015-06-09              526.69
```

2015-06-10	536.69
2015-06-11	534.61
2015-06-12	532.33
2015-06-15	527.20

Obviamente, este es solo un script simple, pero muestra cuán poderoso puede ser tener un maestro de valores almacenado localmente. Es posible realizar pruebas retrospectivas de ciertas estrategias extremadamente rápido con este enfoque, ya que la velocidad de entrada/salida (E/S) de la base de datos será significativamente más rápida que la de una conexión a Internet.

Capítulo 8

Procesamiento de datos financieros

En el capítulo anterior, describimos cómo construir una base de datos maestra de valores basada en acciones. Este capítulo tratará un tema que a menudo no se considera en gran medida en la mayoría de los libros de comercio, el procesamiento de datos del mercado financiero antes de su uso en una prueba de estrategia.

La discusión comenzará con una descripción general de los diferentes tipos de datos que serán de interés para los comerciantes algorítmicos. A continuación, se considerará la frecuencia de los datos, desde los datos trimestrales (como los informes de la SEC) hasta los datos del libro de pedidos y ticks en la escala de milisegundos. Las fuentes de dichos datos (tanto gratuitos como comerciales) se describirán junto con el código para obtener los datos. Finalmente, se discutirá la limpieza y preparación de los datos para su uso en estrategias.

8.1 Clasificación de mercado e instrumentos

Como comerciantes algorítmicos, a menudo nos interesa una amplia gama de datos de mercados financieros. Esto puede abarcar desde precios de series temporales de instrumentos subyacentes y derivados, datos basados en texto no estructurados (como artículos de noticias) hasta información de ganancias corporativas. Este libro analizará predominantemente los datos de series temporales financieras.

8.1.1 Mercados

Las acciones estadounidenses e internacionales, el tipo de cambio, las materias primas y la renta fija son las principales fuentes de datos de mercado que serán de interés para un comerciante algorítmico. En el mercado de valores, sigue siendo extremadamente común comprar el activo subyacente directamente, mientras que en los últimos tres mercados, los instrumentos derivados de gran liquidez (futuros, opciones o instrumentos más exóticos) se utilizan con mayor frecuencia para negociar.

Esta categorización amplia esencialmente hace que sea relativamente sencillo negociar en los mercados de valores, aunque con problemas relacionados con el manejo de datos de acciones corporativas (ver más abajo). Por lo tanto, una gran parte del panorama comercial algorítmico minorista se basará en acciones, como acciones corporativas directas o fondos cotizados en bolsa (ETF). Los mercados de divisas ("forex") también son muy populares ya que los corredores permitirán *margen de las operaciones* en *porcentaje en punto* (PIP) movimientos. A *pepita* es una unidad del cuarto punto decimal en una tasa de cambio. Para las monedas denominadas en dólares estadounidenses, esto equivale a 1/100 de un centavo.

Los mercados de materias primas y de renta fija son más difíciles de negociar directamente en el subyacente. ¡Un comerciante algorítmico minorista a menudo no está interesado en entregar barriles de petróleo a un depósito de petróleo! En cambio, los contratos de futuros sobre el activo subyacente se utilizan con fines especulativos. Una vez más, se emplea el comercio de margen que permite un amplio apalancamiento en dichos contratos.

8.1.2 Instrumentos

Una amplia gama de instrumentos subyacentes y derivados están disponibles para el comerciante algorítmico. La siguiente tabla describe los casos de uso común de interés.