はじめに

どうにかして株で儲けたい。そのためにはシステムトレードが有効らしい。しかし、プログラミングが壁になって二の足を踏んでしまう。そういうトレーダーは多いだろう。

そういう人達のために本書は書かれた。自分の手を動かし、トレードアイデアをプログラムで表現する喜びを味わうのが本書の目的だ。

世の中には、プログラミングを学んだけれども何を作ったらいいか分からない、という人もいる。それに対して、我々の作りたいものははっきりしている。トレーディングシステム検証プログラムだ。このように明確な動機づけがあることは、プログラミングを学ぶ上で大きな利点になる。

トレードで勝つためには、判断するときに極力感情を排除することが重要だ。そのために、明確なルールに従って機械的に売買する、システムトレードと呼ばれる方法が存在する。その準備としてコンピューターを使って売買ルールの有効性・収益性を検証するわけだが、それが完全に感情を排した作業かといえば、そうでもない。かなりの力仕事で、無味乾燥な単純作業も多いが、その推進力となるのはやはり感情だ。

儲けたいという情念が指先からほとばしり、キーボードを通じてコンピューターへと伝わっていく。それが筆者のプログラミングスタイルだ。トレードで負けた日の、プログラミングの進むこと！「感情」や「敗北」をこうして有効利用できる。素晴らしい世界だ。

こういう感じになるのは、もしかしたら Ruby という表現力豊かなプログラミング言語を使っているからかも知れない。ややこしいお膳立てをあまりすることなく、わりとストレートにやりたいことを書ける。そんな Ruby に出合わなかったら、筆者は今ごろシステムト

レードをやっていなかっただろう。

多くの人をこの素晴らしい世界にお誘いしたくて、「プログラミング言語 Ruby を学びながらトレーディングシステム検証ソフトを作る」という主題の元、本書を執筆した次第である。

なぜ自作か

システムトレードを支援するソフトもいろいろ出てきているのに、なぜ検証ソフトを自作するのか。

メリットはいろいろある。初期投資が抑えられるというのもそのひとつだが、それは大した問題ではない。

例えば、

- ●売買ルールに関して細かい制御ができる
- ●新しい指標を追加するのも思いのまま
- ●結果の統計数値も、自分の必要なものが追加できる
- ●プログラミング自体が面白い。熱中できる

といったことがある。もちろん、これらが可能になるためにはある程度の技術は必要だ。

プログラミングが楽しいかどうかは人それぞれではある。少なくとも筆者にとってはプログラミングが楽しいからこそシステムトレードをやっているという部分はある。

また、システムトレードとプログラミングには似た部分がある。

- ●論理的に考える
- ●少しずつ進んでいく
- ●知識の積み重ねができる

何か作業すれば、何かしら進む。それがたとえ失敗に終わっても、「これじゃダメなんだ」という知見が得られる。これが、システムトレードとプログラミングに共通した良さだ。筆者の見立てでは、システムトレードに向いている人はプログラミングにも向いている。

もちろんいいことばかりではない。自作の主なデメリットは、

- ●プログラミングを学ばなければならない
- ●時間がかかる

これらは、ちゃんと利益が出るようになれば実に些細なことだ。しかしそうなる前にこういった理由で挫折してしまう人が多いのもまた事実だ。

そこで Ruby が登場する。

なぜ Ruby か

本書では、Ruby というプログラミング言語を使ってトレーディングシステム検証ソフトを作っていく。

「Ruby とは」というようなことはもはや言いつくされており、ここで長々と説明することはしない。ごくかいつまんで言えば、まつもとゆきひろ氏という日本人が開発し、Ruby on Rails というアプリケーションの登場によって世界に爆発的に広まったオブジェクト指向プログラミング言語、だ。

特長を３つ挙げるとするならば、

- ●純粋なオブジェクト指向
- ●簡潔な文法

●スクリプト言語

といったところか。

「オブジェクト指向」とか「スクリプト言語」がなんなのかはおいおい説明する。何しろ一番に言いたいのは、Ruby は、

「読みやすくて書きやすい」

ということだ。

もちろんこれは主観的な問題で、書き方によっても大きく違ってくる。ただ、いろんなお膳立てをあまりすることなくやりたいことをパッと書けるというところは、これからプログラミングをはじめようという人にとってもいい感じなんじゃないだろうか。

個人のシステムトレーダーは孤独なものだ。一人でなんでもしなくちゃならない。検証も、発注も、専業トレーダーでない人は仕事もやらなくちゃいけない。そんな忙しいなか、プロの開発者でもない人が自分でプログラミングし、そのプログラムを何年にもわたってメンテナンスするためには、読みやすく書きやすいということは、一番大事なんじゃないかと筆者は考える。そういう目的のために、Ruby はベストの選択のひとつだと思う。

ただし、完璧な言語というものはない。もちろん Ruby にも弱点はある。その中でも最大のものは、「Ruby は遅い」ということだ。Ruby で書いたプログラムは、ほかのもっと速い言語のプログラムよりもだいぶ遅い。最近のバージョンはかなり速くなったとはいえ、それでもやっぱりほかに比べれば遅いだろう。

これは、たくさんの計算が必要なトレーディングシステムの検証には痛い。書きやすさを選ぶか、実行速度を選ぶか。これはなかなか難

しい問題だ。

結局、筆者は書きやすさを選んだ。筆者の技術では、ほかの言語を選んだならば儲けるところまでいく前にプログラムが破綻するんじゃないかと思ったからだ。過去の経験も踏まえての判断だ。結果として、この選択は成功した。

だからといって、Ruby が初心者向けのちゃちい言語というわけでは全然ない。機能はものすごく豊富で、ライブラリまで含めれば広大な世界だ。もちろんいきなり全部覚える必要はないが、使いこなせる範囲を少しずつ広げていくことで、気がついたらびっくりするほど作業が効率化している、ということがよくある。

もうひとつ、Ruby のありがたい点を挙げると、

●無料で使える

ということがある。無料で使えるプログラミング言語はほかにもたくさんあり、Ruby だけの特長ではないが、なんにせよありがたいことだ。「タダで使えるものを金儲けに使っちゃっていいのか？」と、多少の罪悪感を抱かないでもないが、もちろんいい、いいに決まっている。

……Ruby の宣伝をするのが本書の目的ではなかった。すでに得意なプログラミング言語がある方はそれを使えばいい。これからプログラミングをはじめようという方には、筆者なら Ruby をお勧めしたい、という話だ。

本書でやること、やらないこと

本書では、実際にトレーディングシステム検証プログラムを作りながらプログラミングを学んでいく。プログラミングの経験のない方に

も読んでいただけることを目指しているが、それと同時に、本当に「使える」プログラムを目指してもいる。これを両立するために、ある程度機能を絞り込んで、ややこしいところを詳しく解説するという形をとることにする。

◎やること

- CUI
- 株価データのダウンロード
- 日本株での売買ルール検証
- 1 銘柄ごと、1 売買単位でのシミュレーション
- 寄り付き、ザラ場、大引けでの仕掛け、手仕舞い
- 売買ルールの部品化、着脱
- 「移動平均乖離率システム」と「ブレイクアウトシステム」のプログラム化

◎やらないこと

- × GUI
- ×日本株以外（FX、先物など）での検証
- ×マネーマネジメント
- ×ポートフォリオの運用シミュレーション
- ×日々の仕込み
- ×発注
- ×システムトレーディングについての解説
- ×シミュレーション結果の評価
- ×勝てる売買ルールの提供

CUI というのは、文字によってコンピューターと入出力のやりとりをすることだ。Windows 付属のコマンドプロンプトから命令を入力するのが基本パターンになる。それが本書のプログラムのスタイルだ。

それに対して GUI というのは普通の Windows プログラムのようにメニューがあり、ボタンなどがあって、マウスでぽちぽちやって動かすソフトだ。

Ruby の基本は CUI だ。GUI やブラウザで表示する Web プログラミングもできるが、もともとは CUI だ。別途ライブラリをインストールすることなく標準ライブラリの範囲でできることをする、という方針のもと、本書では CUI スタイルをとることにした。

チャートを表示したりという華々しい機能はないが、システムトレードの検証ではそれで十分と考えている。もし結果の折れ線グラフなどを表示したければ、Excel などほかのソフトを使ってできる。

また、単純化のため、シミュレーションは 1 銘柄ごと、1 売買単位で行うこととし、資産総額や相場のボラティリティなどによって建玉を変えるマネーマネジメントルールは取り入れない。ポートフォリオを組んで資産の増減を検証するのではなく、1 銘柄ごとに別々に検証していく。また売買手数料も、1 日定額制などややこしいものもあることだし、思い切って考慮しないことにした。信用取引の諸経費なども省略した。

検証の対象は株とする。銘柄が多くてかなり大変だが、コンピューターパワーを存分に活用できる分野でもある。株で検証できるようになれば、FX や先物に移行するのは簡単なはずだ。

日々のシグナル出しや発注機能は備えていない。筆者自身は、そういうものは証券会社の売買ツールを使って行っている。

ザラ場での仕掛けや手仕舞いは本書の自慢だ。寄り付きや大引けの売買のみで検証する人も多いようだが、ザラ場での取引を可能にすることでやれることが広がる。ただ、そのぶんプログラムが多少ややこしくなる。

売買ルールの部品化こそが、本書の核となるところだ。ルールの組み替えが容易で、検証作業がグンと楽になる。設定ファイルをちょっと書き換えるだけでルールを「着脱」できることを目指す。

なお、本書では株やシステムトレードについての解説はしない。これらについては読者がある程度知っているものと仮定し、ここまでがすでにそうであったように、投資用語の解説なども特にしない。疑問点はご自分でお調べいただきたい。拙著『**パチンコトレーダー**』（パンローリング）にはひととおり用語解説が掲載されている。

本書では「移動平均乖離率」と「ブレイクアウト」のふたつの売買システムをプログラムにする。これらの用語が何を意味するか、くらいはやはりあらかじめ知っておいていただきたい。

構成は、大きな流れとして乖離率システムを作ることを本線とし、最終章で総復習としてブレイクアウトシステムを作る、という形になる。

ただし、これらのシステムが利益を生むことを保証するものではない。本書に出てくるルールはあくまでプログラミング学習のための例であって、検証上も実践上も収支がプラスになることは意図していない。勝てるシステムは、ぜひともトレーダーである読者自身が発見していただきたい。

Ruby のバージョン

本書で使用する Ruby のバージョンは、1.9.3 だ。

その前に、Ruby というのは「プログラムを動かすためのプログラム」であって、Ruby 自体が日々改良を加えられ、バージョンアップを繰り返している、ということを知っておいていただきたい。同じ Ruby 言語で書かれたプログラムでも、それを動かすための Ruby 自体のバージョンが違えば、動作しないこともある。特に大きなバージョンアップのときには注意が必要だ。

執筆時点での最新版は、Ruby 2.1.1 だ。内部事情をぶちまければ、ずっと 1.9 系で本書のプログラムを開発していたのだが、原稿の完成が延びに延びている間に Ruby がどんどんバージョンアップしていき、ついに 2.0 がリリースされてしまった。これは大きなバージョンアップだ。最新版で動かなかったらどうしようかと思ったが、幸いなことに手直しなしで動かすことができた。Ruby 2.0 の互換性に救われた。その後リリースされた 2.1 でも同様に動いた。よって、本書のプログラムは Ruby 2.0 や 2.1 でも動かすことができる。

ただし、2.0 以降の新機能は使っていない。エラーの際の出力なども 1.9 のものを掲載している。従って、本書の正式な対応バージョンは Ruby 1.9 とする。1.9 系の中でも、執筆時点での最新版である 1.9.3 を使う。

以前からの Ruby ユーザーの中には、いまだに 1.8 系を使っている人もいるかも知れない。しかし残念ながら、本書のプログラムは 1.8 では動かない。Ruby1.8 と 1.9 の間にはかなりの差があり、互換性を保つのが難しいからだ。

1.8 系に比べて、1.9 系はだいぶ速い。ものにもよるが、筆者が以前に書いたコードで調べた限りでは 1.8 の 2 ～ 3 倍程度までスピードアップするものもあった。十分乗り換える価値はある。

1.9 では機能も大幅にアップされており、よりエレガントなコードを書くことができるようになった。コーディングを 1.9 スタイルにしたことが、1.8 では動作しない一因になっている。

1.8 から 1.9 への乗り換えはなかなか頭の痛い問題で、筆者も 1.8 時代に作ったシミュレーションソフトを、本書執筆にあたって 1.9 対応にしようと試みたのだが難航し、ほぼ全面的に書き直すことになった。しかしその結果、前よりいいものができたので苦労は報われた。

開発環境

まずOSに関してだが、本書はWindows環境を前提にしている。本書用のプログラムは、WindowsXP上で書かれたものである。

Rubyのプログラムは、テキストエディタがあれば書ける。テキストエディタというのは、簡単に言えばパソコンで文字を書くためのソフトだ。Windowsについているメモ帳もテキストエディタだ。

ただし、メモ帳でプログラムを書くのはちょっと心もとない。せめてプログラムが見やすくなるよう文字の色分けをしてくれるようなものを使いたい。

秀丸エディタというのが有名だ。筆者がよく使っているのはEmEditorというやつだ。これらは有料だが、無料のソフトでもいいものはたくさんある。自分が使いやすいものを探そう。

筆者が実際にプログラムを書くのに使っているのは、Linuxの世界の超有名エディタEmacsのWindows版であるMeadowだ。これは機能がとんでもなく強力ではあるが、手になじむのに時間がかかるため、誰にでもお勧めするというわけにはいかない。

また、Rubyのプログラムを動かすにはWindows付属のコマンドプロンプトを使う。Rubyをインストールすると、コマンドプロンプトをRuby実行用に設定したRubyコマンドプロンプトというものが一緒にインストールされる。本書ではそれを使うことにする。これについてはのちほど詳しく説明する。

本書の読み方

プログラミングは脳内だけで学ぶことはできない。手を動かして実際に書いてみる以上の上達法はない。本書に掲載されているプログラムは、できればすべて読者自身の手でエディタを使って入力してみていただきたい。

ふだん日本語のローマ字入力しかしない方は最初は手がつりそうになるかもしれない。しかしその苦しみを乗り越えた先には、黄金が待っている。保証はできないが、待っていると信じてやり抜いていただきたい。

もちろんただ入力するだけでなく、１行１行内容を理解していただきたい。筆者自身もたくさん経験したが、プログラミングには随所に「壁」がある。しかし、モチベーションさえ保てれば必ず突破できる。幸い、われわれには「金持ちになる」という素晴らしい動機がある。

とはいえ、どうしてもキツいときには疑問を残しつつ先に進んでもよいだろう。こっちの部品がないとあっちが作れない、というような作業の都合を優先したため、必ずしも「やさしい順」にプログラムが掲載されてはいない。前に分からなかったことが、あとになって分かることもあるだろう。最終的な目標は、読者自身が、試したいテクニカル指標や売買ルールをプログラム化できることだ。それさえ見失わなければよい。

本書だけでRubyのすべてをカバーできるわけではない。参考文献に掲げるような書籍は、本書を読む途中はもちろん、のちのちまでも役に立つものなので、どれかはお手元に置いておかれることを強くお勧めする。

書いたプログラムが動くかどうか確かめるには、動かしてみるしかない。書いては動かすことを繰り返すことで学習が深まっていく。学習だけでなく、一般にソフトウエア開発はそうやって行われている。書いて、動かし、修正する。この繰り返しだ。

それでは、宝の山を目指して、大いなる一歩を踏み出そう！！