

集合知の成功事例としての株価変動についての調査

プロジェクトマネジメントコース

ソフトウェア開発管理グループ

矢吹研究室

1242109

三宅琢己

目次

第 1 章	序論	4
1.1	研究背景	4
1.2	研究背景となった話	4
第 2 章	研究目的	8
第 3 章	集合知とは	9
3.1	集合知収集の 4 つの手順	9
第 4 章	ナレッジマネジメントとは	10
4.1	SECI モデル	10
4.2	具体的な手法	11
4.3	知識変換の「場」	13
第 5 章	株とは	15
5.1	株とは	15
5.2	株式会社とは	15
5.3	株価変動の要因	15
第 6 章	Ruby によるクローラー開発について	18
6.1	Ruby とは	18
6.2	Ruby の特性	18
6.3	Ruby のオブジェクト指向	19
6.4	クローラーとは	19
6.5	クローラーの目的	19
6.6	クローラーの構造	19
6.7	Ruby と Anemone	21
第 7 章	研究方法	23
7.1	Linux を使用	23
7.2	Oracle VM VirtualBox を使用	23
7.3	Oracle VM VirtualBox 用語説明	24
7.4	Ubuntu , VirtualBox をインストール	24

7.5	ゲストマシンを作成し，その中に Ubuntu をインストール	26
7.6	『Ruby によるクローラー開発技法の通りに株価を取得	29
7.7	株価取得ツール使用方法	36
7.8	背景にある調査の進め方	45
7.9	横浜マンション傾斜問題の調査	48
第 8 章	結果	50
8.1	旭化成	52
8.2	三井住友建設	54
8.3	日立ハイテクノロジーズ	56
第 9 章	考察	59
第 10 章	結論	60
参考文献	61

第 1 章

序論

1.1 研究背景

1986 年スペースシャトル・チャレンジャー号爆発事故が起きた。その直後，事故に関連していた 4 社の株価が急落した。そのうち 3 社の株価は持ち直したのだが，1 社の株価だけさらに下がり，持ち直すことはなかった。この事故の原因は明らかになっていないのにも関わらず，このような株価の変動が起きた。

その数か月後，その 1 社の部品が原因で爆発事故が起きたことを公表した [1]。

この結果だけ見ると株式市場は原因企業を特定していたのではないかと考えられる。

株式市場は賢く，原因企業を本当に特定していたのか，それとも偶然このようなことが起きたのか。

1.2 研究背景となった話

1986 年 1 月 28 日午前 11 時 38 分，スペースシャトルチャレンジャー号がフロリダ州ケープカナベラルから発射された。74 秒後，チャレンジャー号は地表から 16 キロ上空まで上昇し，そして爆発した。

その発射の様子はテレビ中継されていたので，事故のニュースは素早く伝わり，爆発から 8 分後，ダウジョーンズ・ニューズワイヤーズが最初の報道をした。

株式市場は最初の報道から数分もしないうちにチャレンジャー号発射にかかわった主要企業 4 社の株を投売り始めた。

- ロックウェルインターナショナル
シャトルとメインのエンジン担当
- ロッキード
地上支援担当
- マーティン・マリエッタ
シャトルの外部燃料タンク担当
- モートン・サイオコール
固体燃料ブースター担当

爆発から 21 分後、ロッキードの株価は 5 パーセント、マーティン・マリエッタの株価は 3 パーセント、ロックウェルの株価は 6 パーセント下落していた。

だが、一番下落幅が大きかったのはモートン・サイオコールの株価であった。

チャレンジャー号の賛辞をめぐる市場の反応については、ファイナンスの教授であるマイケル・T・マロニーと J・ハロルド・マルヘリンの 2 名の手になるすばらしい研究がある。

サイオコール株を売りたい投資家があまりに多かった一方で、買いたい投資家があまりに少なかったために、同株は瞬く間に取引停止に追い込まれた。爆発からほぼ 1 時間後に売買が再開されたとき、株価は 6 パーセント下落し、その日の終値では下落幅が約 12 パーセントだった。

対照的に、残りの 3 社の株価は持ち直して、下落幅は 2 パーセント程度にとどまった。これは、ほぼ瞬時に株式市場がチャレンジャー爆発の原因はモートン・サイオコールにあり、この惨事が同社のボトムラインに与える影響は深刻だと判断したことを示す確たる証拠である。

理論的に言えば株式市場は、企業が将来獲得するすべてのフリーキャッシュフローの現在価値を計算するメカニズムであるとされている（フリーキャッシュフローとは、企業が経費や税金を支払い、減価償却や投資をした後に残るお金である。自分がその企業のたった一人の株主なら、持ち帰って自分の口座に貯金できるお金とも言い換えられる）。

マロニーとマルヘリンが指摘するように、事故の当日、サイオコールに責任があるというコメントは 1 つとして公になっていなかった。

明るる朝、ニューヨークタイムズ紙に掲載された記事は当時出回っていた 2 つの噂に言及したが、いずれもサイオコールに責任があることをうかがわせる内容ではなかった。同氏は「事故の原因を特定する手がかりはない」と断言している。

だが、市場は正しかった。

爆発から 6 か月後、チャレンジャー号大統領調査委員会は低温のせいでサイオコール製のブースターロケットに取り付けられた O リングシールの弾力性が失われ、隙間ができたためにガス漏れが起きたこと明らかにした。O リングシールは燃焼ガスの漏れを防ぐための部品だが、チャレンジャーの事故では O リングシールから漏れた燃焼ガスがメインの燃料タンクを燃やして大爆発を引き起こした。それにより、サイオコールの責任が認められ、他社の嫌疑が晴れた。

市場がサイオコールを選び出したのは、単なる偶然だったのかもしれない。たまたま同社の事業が宇宙開発計画の中断を受けやすい、と思われただけかもしれない。あるいは取引停止という事態が、投資家に何かのメッセージを送ったのかもしれない。

こういう点にも注意すべきではあるが、この時の市場の動きは不思議だったとしか言いようがない。

一般的に株式市場はメディアの憶測やウォールストリートの狂乱などに歪められることが多く、個々の投資家の知恵を集約して集団の知恵を得るメカニズムとしては不安定である。それにもかかわらず、あの日の株式市場が純粋な計算機械としてきちんと機能したということが何とも不思議なのである。

市場はどうやって正しい意見にたどり着いたのだろうか。マロニーとマルヘリンをてこずらせたのがこの問いである。

自社に責任があると思っていたサイオコールの幹部が1月28日に株を投売りしたかもしれないと考え、インサイダー取引を示す証拠を探した。でもそんな証拠は見つからなかった。

Oリングについて知っていたライバル企業の幹部が、サイオコールの株を投売りして、残りの3社の株を買いあさった人もいなかった。

抜け目ないインサイダーのせいで、事故当日のサイオコールの株価が下落したのではない。比較的手持ちの情報の少ない投資家が、同社の株を買わなかっただけのことだった。

では、「どうして誰もサイオコールの株を買いたくなかったのだろうか」。

この問いに対してマロニーとマルヘリンは最後まで説得力のある答えを見つけられなかった。コーネル大学の経済学者であるモーリーン・オハラ含蓄のある言葉を引用した、「市場は実践的にはうまく機能しているように見えるが、実際にどのように機能しているか理論的には不明である」。

オハラの言葉ももっともだが、それはたぶん「理論」という言葉の定義にもよる。

あの日の出来事をごくごく単純化すると、次のようになる。

特に関係もない多数の個人の集まりが、「チャレンジャーが爆発してしまったいま、この4社の企業価値はどれほど下がったのだろうか」という、客観的に正しい答えが存在する問いを突き付けられた。ここに集団の平均的な予測値、この場合は金額加重の株価予測の確度が高くなる要件が隠されている。Oリングに関するインサイダー情報を知っている人がいた可能性は否定できない。だが、そんな人がいたかどうかに関係なく、その日一人ひとりのトレーダーの頭の中にあった事故に関する情報のかけらを足し集めたら、真相に近いものが生まれる可能性も十分ある。

サイオコールと事故の関係について確かなことを知っている個人は存在していなくても、確かなことを知っている集団は存在していたのだ。

この日、市場が賢い判断を下せたのは、賢い集団の4つの要件が満たされていたからだ。

- 意見の多様性

それが既知の事実のかなり突拍子もない解釈だとしても、各人が独自に私的情報を多少なりと持っている

- 独立性

他者の考えに左右されない

- 分散性

身近な情報に特化し、それを利用できる

- 集約性

個々人の判断を集約して集団として1つの判断に集約するメカニズムの存在

この4つの要件を満たした集団は、正確な判断が下しやすい。なぜか。多様で、自立した個人から構成される、ある程度の規模の集団に予測や推測をしてもらってその集団の回答を均すと、一人ひとりの個人が回答を出す過程で犯した間違いが相殺される。

言ってみれば、個人の回答には情報と間違いという2つの要素がある。算数のようなもので、間違いを引き算したら情報が残るというわけである。「集団の知恵」という言葉に実質の伴った価値を与え、私たちを驚かせてやまないのは、みんなの意見に含まれている情

報量の多さである。

チャレンジャーの事故のケースでは，集団が共有する集合的な「頭」の中に世界のほぼ完璧な絵が描かれている [1] 。

第 2 章

研究目的

背景にあることを調査するためには，銘柄・期間を指定し，さらに株価データの変動を可視化できるようにする必要がある．そのようなことを可能にするツールの開発を本研究での目的とする．

完成次第背景にある，株式市場は賢いのか，そうでもないのかを進めるものとする．

株価データの取得方法は Ruby によるクローラーで取得する．ネット上には膨大な情報量がある中で，自分から必要な情報を引っ張り出す必要がある．その勉強をするためにこのような株価データの取得方法をとる．

第 3 章

集合知とは

集合知というのはナレッジマネジメントの分野ではもともとの意味は複数人の智慧の集合と言う意味である。開発や課題解決に取り組む時、天才的な超優秀な一人より、それなりに優秀な何人かの集合が、複数の視点を上手に使うって取り組む方が高い成果を生むことができる場合もあるということ。

議論の中で様々な意見が飛び交う中でよりよいアイデアや意見が生まれる。

3.1 集合知収集の 4 つの手順

集合知を収集しやすくするためには以下の 4 つの手順通り進める必要がある。

1. 公開

情報を原則的には私物化せず、すべて公開する。

2. 連鎖

情報と情報を指標に基づき関連付けて、新たな意見、発想へと展開する。

3. 選別

必要な情報であるか、そうでないかを選別し、必要な情報のみを管理するようにする。

4. 評価

公開されている情報をコメントなどにより評価し、情報に優先順位をつける。

第 4 章

ナレッジマネジメントとは

企業経営における管理領域の一つ：生産管理，販売管理，財務管理，人的資源管理，情報管理に続く第 6 の管理領域：個人のもつ暗黙知を形式知に変換することにより，知識の共有化，明確化を図り，作業の効率化や新発見

を容易にしようとする企業マネジメント上の手法である [2] ．

4.1 SECI モデル

「個人の知識を組織的に共有し，より高次の知識を生み出す」ということを主眼に置いたナレッジマネジメントを実現する場合，そのフレームワークとして以下の 4 段階のプロセスが提示されている．

このプロセスは，各段階の英語名称の頭文字をとって SECI(セキ) プロセス，あるいは単に SECI(セキ) と呼ばれる．これは野中郁次郎 (一橋大学 名誉教授) と竹内弘高 (ハーバード大学ビジネススクール 教授一橋大学 名誉教授) が執筆した The Knowledge Creating Company(『知識創造企業』梅本勝博訳，東洋経済新報社) において，提唱された．知識とは「正当化された真なる信念 (Justified true belief)」であり，個人と個人の相互作用、あるいは組織と組織の相互作用により，ダイナミックに変化・深化・進化していくものであるという考えの下に構築されている．

SECI モデルは以下の頭文字をとったものである．

- 共同化 (Socialization) 暗黙知から暗黙知へ
組織内の個人，または小グループでの暗黙知共有，およびそれを基にした新たな暗黙知の創造である．
- 表出化 (Externalization) 暗黙知から形式知へ
各個人，小グループが有する暗黙知を形式知として洗い出すこと．
- 結合化 (Combination) 形式知から形式知へ
洗い出された形式知を組み合わせ，それを基に新たな知識を創造することである．
- 内面化 (Internalization) 形式知から暗黙知へ
新たに創造された知識を組織に広め，新たな暗黙知として習得することである [2] ．

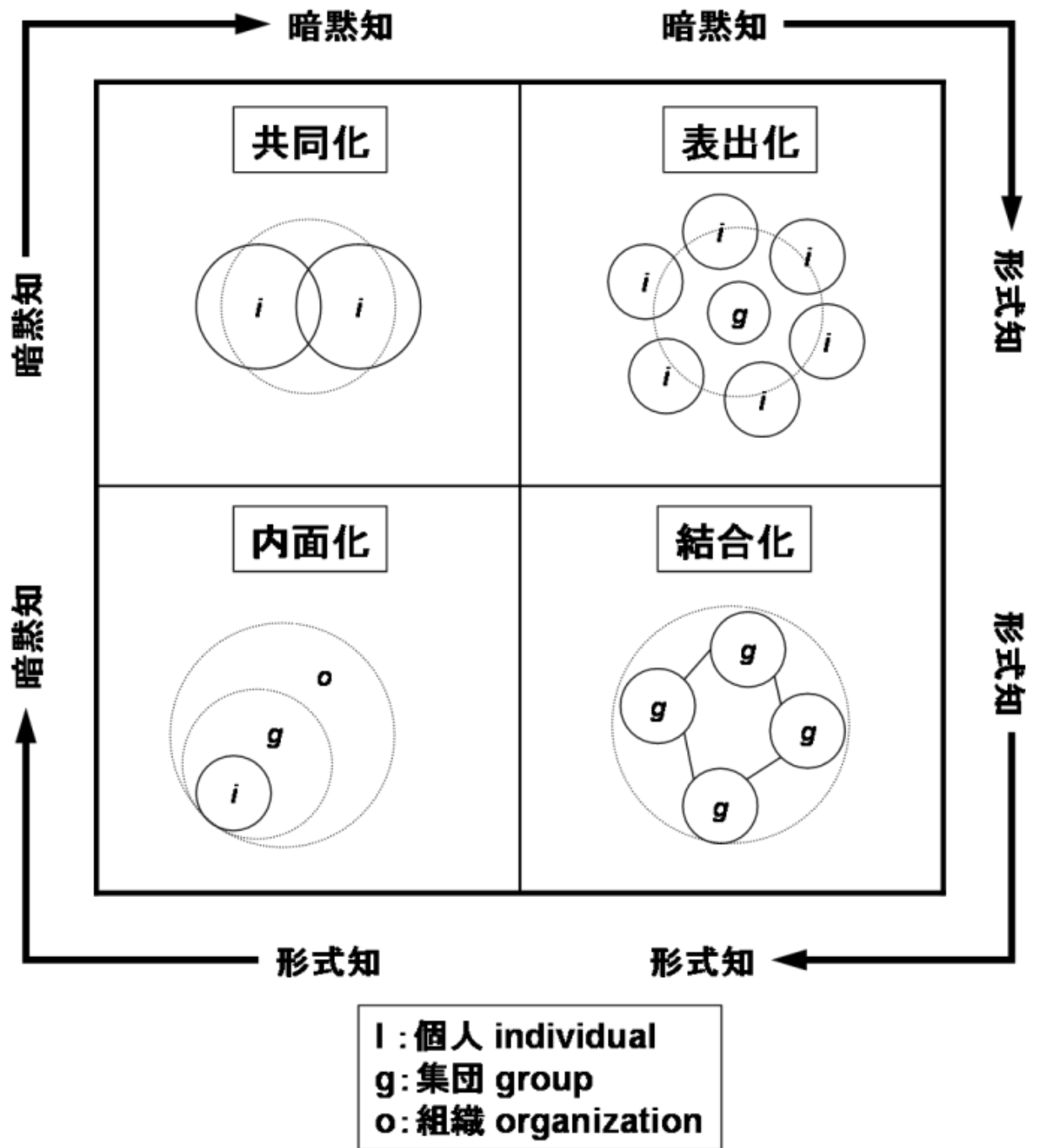


図 4.1 SECI プロセス

4.2 具体的な手法

主に以下の手法があるが、それぞれ独立したものではなく、相互依存的なものである [2]。

4.2.1 データマイニング

人工知能や統計学を利用してデータから知識を取り出そうとする試み．主に共起現象を探り，セールスに結びつけようとしている．

- 例1：スーパーでビデオとガムが共に売れる 両者を同じ場所に置く．
- 例2：本 A を買う人は，後に本 B を買うことが多い 購入者に本 B を薦めるダイレクトメールを送る．
- 従来の統計学と大差ないが，POS やオンラインショッピングによる大量の IT データの中から法則性を見つけ出すことに主眼が置かれている [2]．

4.2.2 データウェアハウス

データを多次元的に処理することにより，通常では察知しにくい傾向性を発見する技法．多次元データベースなど，幾つもの次元によって処理が可能なソフトウェアが開発されている．

- 例：時間，空間，取り扱い物によって販売量が明示される 時系列や地域、取り扱い物の傾向が分かる [2]．

4.2.3 知識共有化

電子掲示板やメーリングリスト，知識ベース，オンラインコラボレーションなどを使って，一部の人の資産であった知識の，集団全体への共有を図るもの．基本的には文字や印刷といったメディアの問題であるが，電子通信技術の一新によって，電子メール・電子掲示板に代表されるような新しい共有化のあり方が模索されている．具体的には，企業内ではグループウェアなどを使って知識共有の試みが行われることが多い．インターネット上でも，OKWave，はてなのように広範な分野を扱うサイトや，Apple Support Discussion のような特定者向けサイトによる知識共有化の試みが始まっている．近年，エンタープライズ 2.0 と呼ばれる大企業での情報共有が積極的に行われるようになってきた [2]．

4.2.4 可視化

人間における視覚の優位性を利用し，多次元・多要素で理解しにくい情報を，見える形で表現し，理解しやすくさせること．原理的にはグラフや図画であるが，ナレッジマネジメントでは CG を利用した立体的で動的な画像を使って表現するケースが多い．

様々な手法はあるものの，通常の技法と同じく，それを使いこなすのは熟練と才能が必要とされるため，電子メールや QA 知識ベースなどいくつかを除けば，実際に有効活用されている例は少ない．また暗黙知を明示化するには原理的に大きな困難が伴うため，共有化された知識はあまり役に立たない常識的なものがほとんどで，実際にほしい熟練した技

能や知恵は掘り出せないことが多いため、研究自体は尻すぼみになっている [2]。

4.2.5 エンタープライズサーチ

企業組織内の書類、人事、経営情報等を検索できるようにするためのシステム、またはそのコンセプトのこと [2]。

4.3 知識変換の「場」

組織として、知識の創造、共有、活用、蓄積を活発化させるために、個々のナレッジを共有したり、共同でナレッジを創造したりするための結節点が必要となる。この結節点を、「場」と言う。豊かな知識創造・知識経営が出来るかどうか、「場」のデザインにかかってくる。「場」は、SECI モデルの各フェーズに沿って、4つのパターンに分けることができる [3]。

4.3.1 創発場 (Originating Ba)

- 共同化に対応

経験、思い、信念、考え方などの暗黙知を共有する場である [3]。

4.3.2 対話場 (Dialoguing Ba)

- 表出化に対応

各自が対話 (ダイアログ) を通じて暗黙知を言語化・概念化して形式知に変換するための場である [3]。

4.3.3 システム場 (Systemizing Ba)

- 結合化に対応

形式知を相互に移転・共有・編集・構築し、新たな体系の形式知へと統合する場である [3]。

4.3.4 実践場 (Exercising Ba)

- 内面化に対応

形式知を個々人の暗黙知へと身体化するための場である。ここでは、単なる形式知の伝達ではなく、形式知に束ねる形で何らかの経験的要素や人間的要素を提供することで暗黙知としての移転・発展を促すことができる。サービス業などで特に重要な場である [3]。

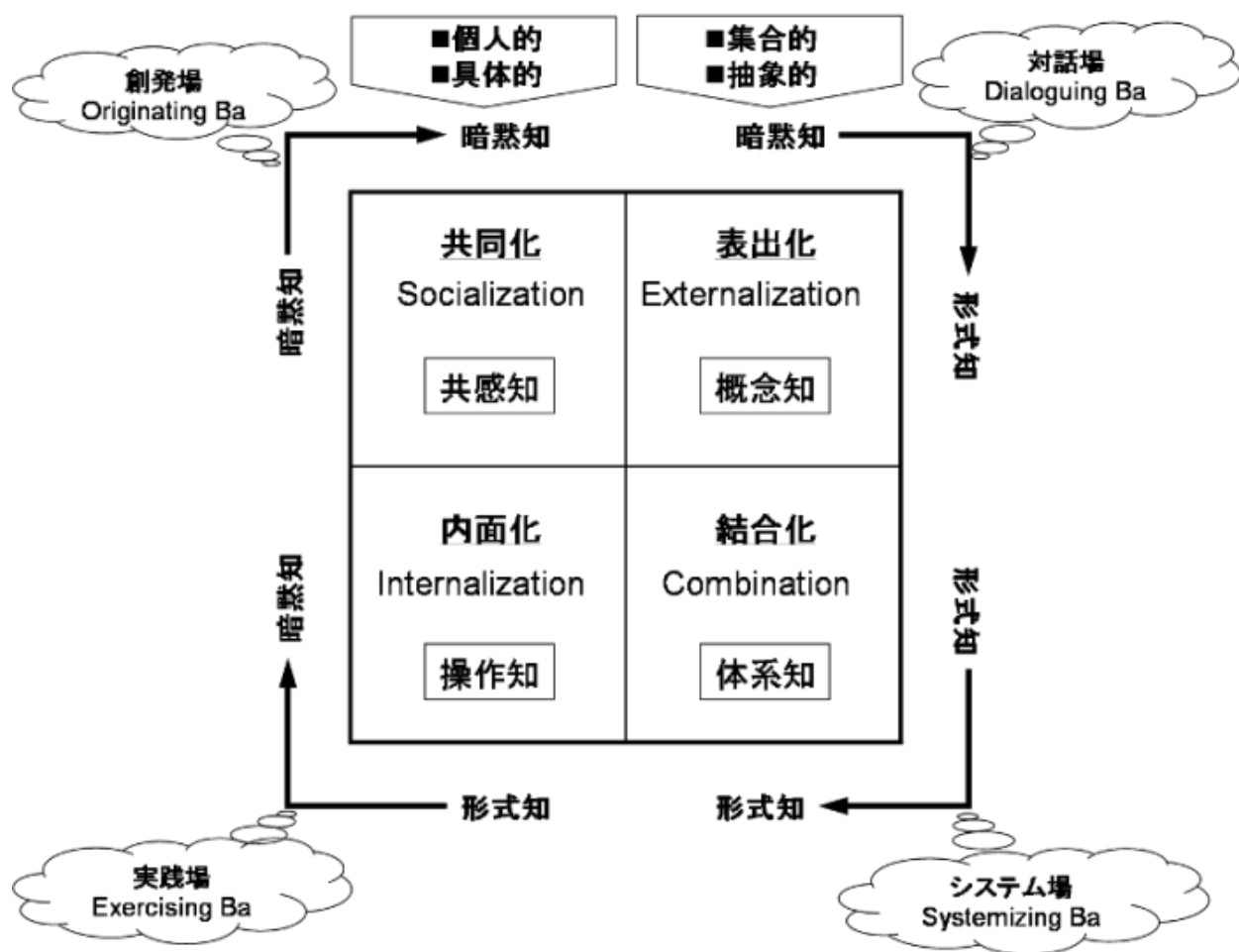


図 4.2 SECI プロセスの「場」について

第 5 章

株とは

5.1 株とは

企業が事業資金を調達するために、発行しているものである。企業は、投資家が株を買ってくれた資金等を使って事業を拡大する。「株を買う」ということは、株を発行している企業に出資を行い、事業資金を提供していることを意味する。

まとめると、株とは、「株を買う（株主になる）」＝「出資者になる」＝「会社のオーナーの一人になる」ことを意味する [4]。

また、株の大きな特徴として、買った株を第三者に転売することができる。株を転売すれば、会社のオーナー（株主）の権利等は、新しいオーナー（株主）に移ることになる [5]。

5.2 株式会社とは

細分化された社員権（株式）を有する株主から有限責任の下に資金を調達して株主から委任を受けた経営者が事業を行い、利益を株主に配当する、法人格を有する企業形態である。このような企業形態は各国で見られる。

5.3 株価変動の要因

株価変動の要因はさまざまである。
上昇の要因と下落の要因を説明する。

5.3.1 株価上昇の要因

基本的に株が上がって行く状態というのは、買い手が多い、つまりはその株は人気が高い状態にあるということである。

1. 業績が好調である

業績が好調であるということは、利益を生み出しやすい状態にあるということである。投資家から人気が高いのは当然である。

2. 業績の見通しを上方修正した

当初の会社の予想より、見通しを上方修正した時である。会社側が思っていたよりも利益が出そうという報告である。これも好感される。

3. 復配・増配をする

今期は配当金を出す、または前回より配当金を増やすと発表した時である。復配や増配は、業績が好調の証であるため、投資家に好感されることがある。

4. 新製品の発表・新しい工場の建設など

新しいことを始めるのは、リスクもちろんあるが、利益を増やすために行うものである。これにより利益が増えるという判断をされた場合には、株価が上がる。

5. 合併・買収

企業の合併・買収により、企業間の相乗効果が出て、企業価値が上がると判断された場合には、株価が上がる。

6. 割安株の修正

割安に放置されていた株が、とある出来事をきっかけに株価上昇が起きることがある。とある出来事というのは、上の1~5の理由もそうであるし、東証2部から東証1部へ移動することを発表し、たくさんの投資家の目に触れたときにも株価が上がる可能性がある[6]。

5.3.2 株価下落の要因

株価が下がるときは、株価が上がる時の要因の逆のことがおきたときである。基本的に株が下がって行く状態というのは、売り手が多い、つまりはその株は人気が低い状態にある。

1. 業績が不調である

業績が不調であるということは、利益を生みにくい状態にあるわけである。投資家から人気が落ちるのは当然のことである。

2. 業績の見通しを下方修正した

当初の会社予想より、見通しを下方修正した時である。会社側が思っていたよりも利益が少なそうという報告である。これも嫌気される。

3. 無配・減配にする

今まで出していたが今期は配当金をださない、または前期より配当金を減らすと発表した時である。無配や減配は、業績が不調の証であるから、投資家から嫌われる。

4. 問題が起きる

経営トップの不祥事、工場の環境汚染、法律違反、事故の原因など悪い材料が明るみに出た場合である。

5. 為替レート

円高・円安の影響を受ける企業がこれに当てはまる。特に輸出産業に関しては円高になると、製品の値段が実質的に上がってしまうため、利益が減るという意味で嫌気される。

6. 同業他社の不振・倒産

同業他社が倒産すると、その業界自体が冷え込んでいる可能性がある。そういった思惑売りが出る。逆にその会社だけに問題がある場合は、売り上げアップのチャンスになるため、好感されることもある [7]。

このように様々な理由で株価変動は起きるのである。

第 6 章

Ruby によるクローラー開発について

本章では『Ruby によるクローラー開発技法』

2014 年 8 月 28 日著者佐々木拓郎，るびきち

SB クリエイティブ株式会社により出版された本を参考にして書いていく．

6.1 Ruby とは

Ruby は 1993 年から開発されている国産オブジェクト指向スクリプト言語である．代表的な汎用スクリプト言語 Perl に可読性の高い構文とシンプルかつ強力なオブジェクト指向を加え，Lisp 風の味付けをしたものが Ruby である [8]．

6.2 Ruby の特性

クローラーを作るにあたって，Ruby の基礎の基礎さえ押さえておけば知識があるだけで作れてしまう．

複雑なクローラーを作るのには HTML パーサーやクローラーフレームワークなどといったものを使ったほうが良いが，簡単なものであればテキスト処理と難易度は変わらない．

ファイルを開く `open` メソッドの引数に URL が使えるように拡張することのできるライブラリが存在し，ファイル名の代わりに URL を指定すれば HTTP アクセスができる．その状態の `open` は HTTP クライアントであり，ファイルを開いて内容を読み取り，加工し，出力する，普通の Ruby スクリプトがクローラーになる．

また，取得した HTML から情報を抽出し，加工出力する処理はまさにテキスト処理そのものである．その為，文字列や配列やハッシュなどの基本的なオブジェクトの扱い方がわかればなんとかなるのが Ruby の特性である [8]．

6.3 Ruby のオブジェクト指向

Ruby はオブジェクト指向スクリプト言語である。だが、ちょっとした Ruby プログラミングをするのにクラスを定義する必要はない。なぜなら、豊富な組み込みクラスが用意されており、組み込みクラスを使うだけでもある程度のプログラミングができてしまう。

組み込みクラスを使うことでもメソッドの呼び出しが起きるため、クラスを定義しなくてもれっきとしたオブジェクト指向プログラミングである。

Ruby は全てのデータがオブジェクトであり、ユーザー定義クラスのオブジェクト以外にも、整数や文字列もオブジェクトである。さらにクラスでさえもオブジェクトとして使うことができる。

オブジェクト指向プログラミングにおいて重要なことは、内部構造をなるべく隠ぺいし、外に公開するのは最低限抑えることである。Ruby プログラミングは、正しいオブジェクト指向へ自然と誘導してくれるのである [8]。

6.4 クローラーとは

Web 上の文章や画像を自動的に取得するプログラムのことを指す。もともとは検索エンジンによって世界中の Web サイトをデータベース化、インデックス化、する目的で開発されたもので、Google のグーグルボットなどが有名である。クローラーは、クローラー以外にも様々な呼び名がある。「ボット (Bot)」、「スパイダー (spider)」、「ロボット (robot)」のような呼ばれ方もある。

また、Web 以外を対象に、ファイルサーバやデータベース内を巡回し、インデックス化する目的のプログラムもクローラーと呼ばれる [8]。

6.5 クローラーの目的

検索エンジンのクローラーは Web サイトを巡回してデータベース化することを目的とする。その成果として、検索エンジンを利用して様々な情報を収集することができる。しかしながら、検索エンジンを利用するだけでは自分の欲しい情報だけ抜き出し、定期的に取り得るといったことはできない。

そこで、自分のクローラーを作ることにより、取得元のサイトを絞り、必要とするデータのみを効率的に取得することができる [8]。

6.6 クローラーの構造

クローラーは対象の Web サイトに対して、コンテンツをダウンロードして保存をし、次の取得先を見つけていく。このような形で巡回をしていく。このクローラーの動きを構造にすると、3 つに分類することができる。

- コンテンツの取得

- データ解析
- データ保存

この三つについて説明をしていく [8] .

6.6.1 コンテンツの取得

コンテンツの取得は、サイトにアクセスをして Web ページなどをダウンロードする機能である .

クローラーは HTML をダウンロードして解析し、その中に含まれるリンク先を見つけてそのページをダウンロードするというプロセスの繰り返しである . その一連のプロセスを「クローリング」という .

クロール対象となるページは 3 つに分けられる .

- ステートレスなページ (状態を持たないページ)
URL を指定すると単純に HTML が返ってくるページである . ブログやニュースページなどの多くのサイトが、このステートレスなページにあたる .
- ステートフルなページ (状態を持つページ)
ログイン済みの状態でないと参照できないページや、POST など事前に送られた情報をサーバ側が保持し、その前提でないと参照できないパターンのことを言う . サーバサイドで動的に生成されるページの多くはこの形式である .
- JavaScript を元にクライアント側でページを組み立てるタイプ
JavaScript の指示のもとにデータの取得や処理をブラウザ内で行い、ページを組み立てて表示するタイプである . 目的のデータを取得するにはブラウザと同じように JavaScript を解釈して描写する必要がある . 普通のクローラーで対応するのは非常に困難である .

これに対応するには 2 つの方法がある . JavaScript の動作を人間が解釈して同様の動きをプログラム側に組み込む方法と、クローラーが JavaScript の解釈をできるようにする方法である . 前者は対象とするページの構造・複雑さによって難易度は格段に変わる . 後者は JavaScript を解釈させるためには、クローラーの内部に簡易ブラウザのエンジンを組み込んだり、クローラーがブラウザそのものを起動させるなどして、あたかも人間がブラウジングしているようにシミュレートする必要がある [8] .

本研究ではステートレスなページをクロールする .

6.6.2 データの解析

データの解析は、ダウンロードしたページを解析して特定のデータを引き抜く機能である . データを引き抜く部分は、「スクレイプ」と呼ばれる .

解析の実装方法は以下の 2 つがある .

- 正規表現を利用して、パターンマッチングする

正規表現を利用する方法は、目的とするデータもしくはその周辺のデータの特徴をもとに、パターンマッチングする方法である。この方法は単一の要素を抜き出す場合は手軽で非常に簡単に実装できるケースが多い。構造化されていない HTML の場合にも有効である。一方で複数の要素を取得する場合は、ループや条件分岐を駆使した複雑なプログラムになりがちである。また、取得先のページのデザイン変更のたびに対応が必要な可能性がある。

- HTML や XML の文法を理解して、構文を解析する

構造解析は、取得した HTML や XML を構文解析ツールを利用し、CSS セレクタや XPath で要素を指定し、抜き出す。この方法は構文を解析したうえで順番をたどっていくために、解析の処理コストが大きくなる。しかし、取得先が構造化された HTML の場合であれば、簡潔に処理が記述できるうえに取得結果の正確性が高い [8]。

6.6.3 データの保存

データの保存は、取得したデータをメモリ内、もしくはファイルやデータベースなどに保存しておく機能である。メモリへの保存は、巡回・解析のための一時的なものであり、ファイルやデータベースへの保存は、データを永続化するために利用する。

永続化することにより、定期的なクロールでは訪問間隔や多重度の調整や取得済みのデータをスキップするなど効率的な巡回が可能になる。また、データを保存することにより巡回工程と解析工程の分離が可能になる。そして、それぞれの機能が疎結合で運用保守しやすいプログラムになる [8]。

6.7 Ruby と Anemone

本節では Anemone についてやその機能について書いていく。

6.7.1 Anemone とは

Anemone は 2009 年に Chris Kite によりクロール者のフレームワークとして開発された Ruby のライブラリである。クロール者が必要とするデータ取得、解析、保存のすべての機能に備えており、Ruby のクロール者ライブラリとしては最も完成度の高いものの 1 つである。

6.7.2 Anemone の機能

Anemone の主な機能について書いていく。

- 巡回機能

Web サイトの巡回と取得したページの処理に関する機能もここで実装されている。

- ページ解析機能

- ストレージ機能

Anemone は取得したページを保存したうえで処理する．ストレージは複数利用可能であり，目的に応じて使い分けることになる．ストレージを指定しない場合は，取得したデータはメモリ内に保存される．処理対象が多いほどメモリを利用し，プログラムを実行している PC にも影響を与える．

メモリ以外のストレージに保存したページは，クローラーのプログラムが終了した後も再利用が可能である．取得対象が多い場合や，定期的・継続的にクローラーを動かす場合は，ストレージを利用するのが良い．

第 7 章

研究方法

今回の研究での株価データ取得方法を書いていく。

7.1 Linux を使用

7.1.1 OS とは

OS というのはオペレーティング・システムの略であり，種類としては Windows，Mac，Linux などがある。

その中で今回は Linux を使う。

7.1.2 Linux を使用する理由

今回，OS は Linux を使用するのだがその理由は，開発環境を整えるのにコード 1 行で済むからである。だが，Windows だと開発環境を整えるのに様々なものをダウンロードしなくてはならない。よって今回は Linux を使用する。

7.2 Oracle VM VirtualBox を使用

7.2.1 Oracle VM VirtualBox とは

インストールした PC 上に仮想的な PC を作成し，別の OS を実行できるソフト。

仮想 PC 上で，本ソフトを実行している PC に接続された USB 機器を利用できるのが特長で，そのほかネットワークやサウンド機能も標準で利用できる。

仮想 PC の作成はウィザード形式で行え，各種 Windows や Linux，FreeBSD などインストールしたい OS を選択すると，仮想 PC に割り当てるメモリや HDD のサイズを自動設定してくれる。

ユーザーが直接サイズを指定することも可能だ。

そのほか本ソフト独自の機能として，LAN 上などの別 PC から，Windows 標準の「リモート デスクトップ接続」を利用して仮想 PC へ直接接続できる [9]。

7.3 Oracle VM VirtualBox 用語説明

本節では VirtualBox の用語について説明をする。

7.3.1 ホスト OS

仮想マシンで仮想的な OS を動かす OS である。

7.3.2 ゲスト OS

使っているコンピュータ上で別のコンピュータをエミュレートする仮想マシン上で動いている OS のことである。

7.3.3 仮想化ソフト

仮想化ソフトパソコンで、CD や DVD などの内容をハード-ディスクに収め、これを仮想的に使用できるようにするソフトウェアである。

7.3.4 仮想ディスク

ゲストマシンが使用する仮想のハードディスク

7.3.5 仮想マシン

Windows 上で別の OS を作動させられるソフトウェアのことである。

7.4 Ubuntu , VirtualBox をインストール

7.4.1 Ubuntu とは

Ubuntu(ウブントウ) とは、Linux ディストリビューションの一つである。Ubuntu というのは、元々はアフリカの言葉で、「他者への思いやり」とか「皆があつての私」といった意味があるそうで、Ubuntu を支援しているマーク・シャトルワース氏が、南アフリカ生まれである。

Ubuntu は使いやすさから、最近是最も人気のある Linux ディストリビューションの一つとなっている。Windows と比較しても遜色ない。それに、無償で提供されている OS で、Ubuntu コミュニティで開発されている。その Ubuntu コミュニティは、マーク・シャトルワース氏が創設した Canonical Ltd. という会社から資金提供を受けて開発している。Ubuntu 財団 (1,000 万米ドル) も創設されていて、もしもの時に備えている。

また将来に渡っても無償で提供が継続されるので、安心して利用できると言われている。ユーザーにとっては、OS が無償でずっと使えることは、大きな魅力である。もちろんビジネスでの利用も無償である。さらに「定期的にリリースする」(半年ごと)と宣言している

ので、常に最新版の OS に、無償でアップグレードできるのも魅力である。無償セキュリティアップデートが 3 年間 (デスクトップ) あるので、セキュリティー面でも安心である。

Ubuntu には、コードネームやバージョンがあり、サポート期限が決められている。サポート期限が期限が終了する前に、新しいバージョンの Ubuntu を入手してインストールすれば、継続して使い続けることができる [10]。

7.4.2 Ubuntu の特徴

Ubuntu を使って、Web を閲覧、メールを読み書きし、文書や表計算、画像を編集、その他さまざまなことができる。Ubuntu のデスクトップ CD には、早くて簡単なグラフィカルインストーラが搭載されている。一般的なコンピュータの場合、25 分以内でインストールが完了する。

それ以外にも以下のような特徴がある。

- シンプルなデスクトップ
初期状態、デスクトップにはアイコンがひとつもない。デフォルトのデスクトップテーマは、目にやさしいものを使用している。
- すぐに使える
面倒な設定はすべて終えてあり、Ubuntu を一度インストールすれば基本的なセットアップ作業は終わっているのですぐに使い始めることができます。
- オフィスアプリケーション搭載
オープンオフィスには、他のオフィススイートと似たユーザインタフェースと機能が備わっている。また、よく使われる主要なデスクトップアプリケーションが含まれている。
- さまざまな形式のファイルを編集可能
マイクロソフトオフィス、ワードパーフェクト、KOffice、StarOffice のファイルを開いて編集することができる。
- 簡単で手軽なアップデート
アップデートが利用できる時には、タスクバーのアップデートエリアに表示される。単純なセキュリティフィックスから、完全なバージョンアップグレードまで、このエリアで通知される。アップデート作業は簡単なのでマウスで何度かクリックするだけでシステムを最新の状態に保つことが可能である。
- 非常に充実したフリーソフトウェアライブラリ
もっと別のソフトウェアが必要ならば、カタログにある何千というソフトウェアパッケージから選ぶことができる。すべての利用可能なソフトウェアは、クリックしていきただけでダウンロードとインストールが可能である。もちろん、これらはすべて完全にフリーである。
- ヘルプとサポート
メニューから[システム][ヘルプとサポート]を選択するか、<https://help.ubuntu.com/> にアクセスすることで公式ドキュメントを参照することができる。もし、Ubuntu

の使い方について質問があるならば、誰かが既に質問していないか調べると良い。Ubuntu コミュニティでは、ドキュメントを整備しており、あなたの質問の答えを含んでいるかもしれない。あるいは、どこを参照したら良いか分かるかもしれない。また、Ubuntu コミュニティのチャットやメーリングリストにおいて、多くの言語でフリーサポートを受けることができる。あるいは、Canonical 社や各地のサービスプロバイダより、有償サポートを購入することもできる。

- 国際化とアクセシビリティ

Ubuntu は、できるだけ多くの人が利用できることを目標としていう。そのため、Ubuntu にはフリーソフトウェアコミュニティが提供できる最大限の国際化とアクセシビリティ機構が含まれている [11]。

7.4.3 Ubuntu をインストール

まず、<https://www.ubuntulinux.jp/download/ja-remix> から Ubuntu14.04 の ISO イメージをダウンロードする。

7.4.4 VirtualBox をインストール

<http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html> からインストーラをダウンロードし、指示に従い、VirtualBox をインストールする。

7.5 ゲストマシンを作成し、その中に Ubuntu をインストール

以下の手順で行っていく。

1. インストールした VirtualBox を起動
2. ウィンドウ左上にある【新規 (N)】のボタンを押してゲストマシンを作成

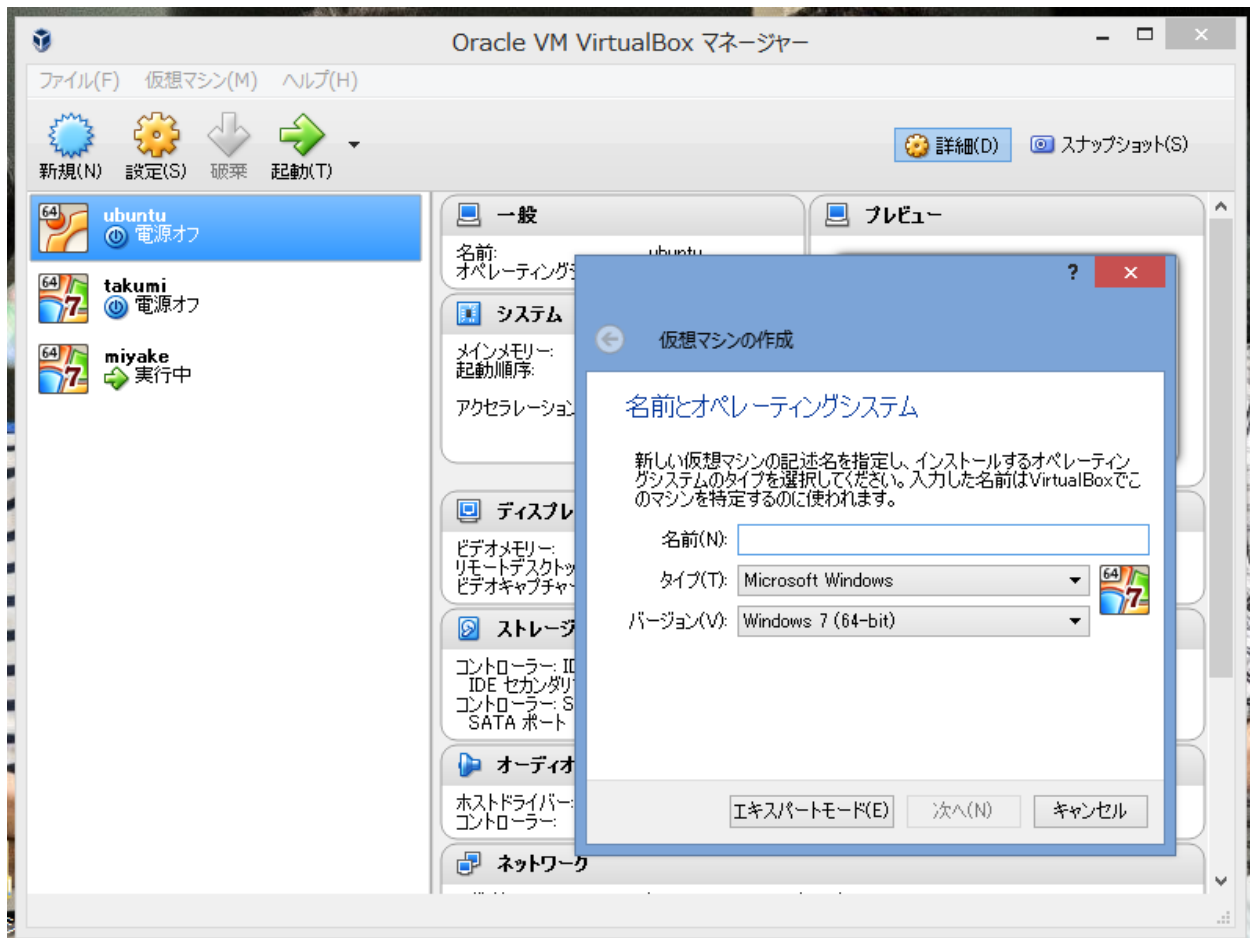


図 7.1 ゲストマシンを作成

ゲストマシンの名前，メモリサイズ，HDD の設定をする．

3. 【設定(S)】を開き，ストレージをクリックし，ストレージツリーのコントローラー：IDE 内の"空"を選択

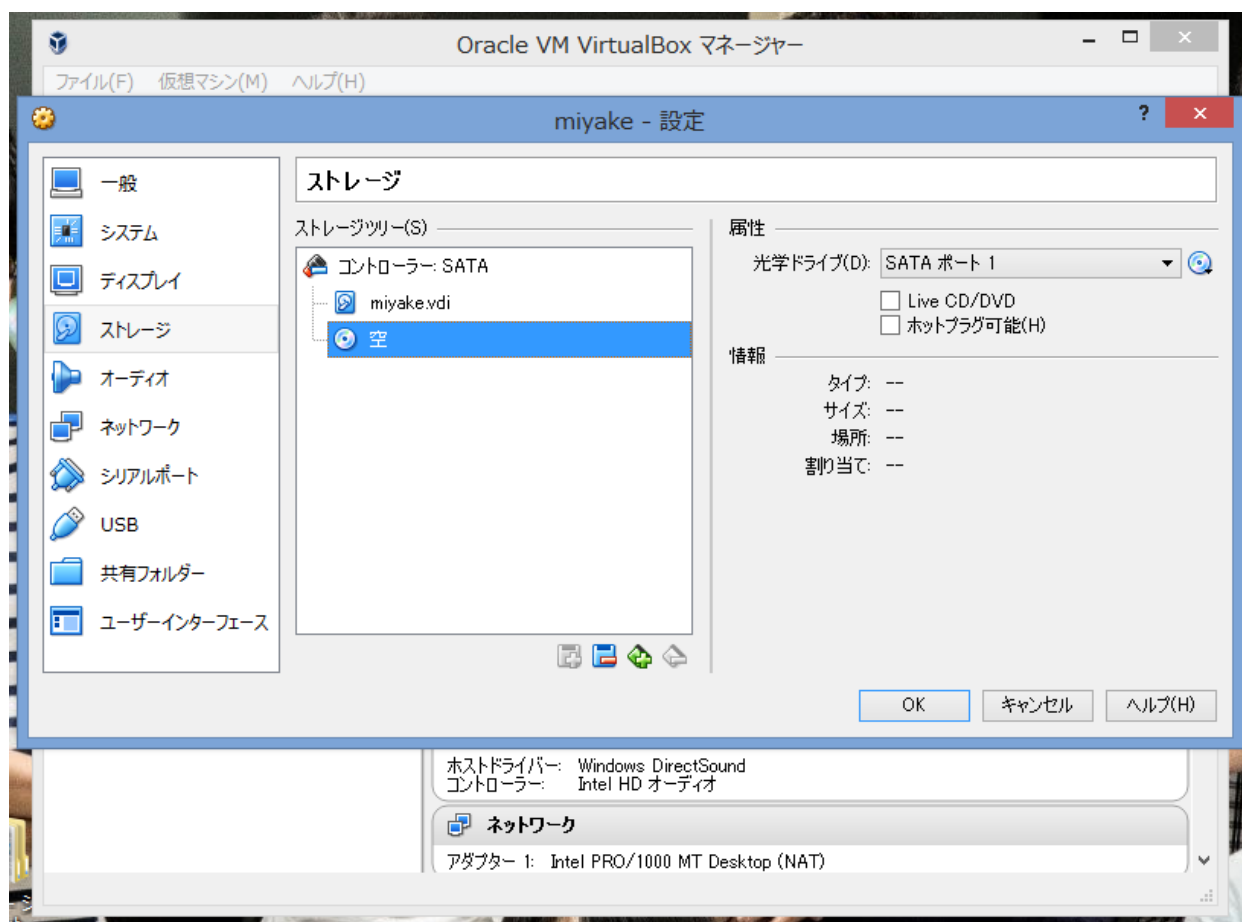


図 7.2 ストレージ設定画面

4. 属性内の一番右のディスクマークをクリックし,【仮想光学ディスクファイルを選択..】をクリックしてダウンロードした ISO イメージを選択

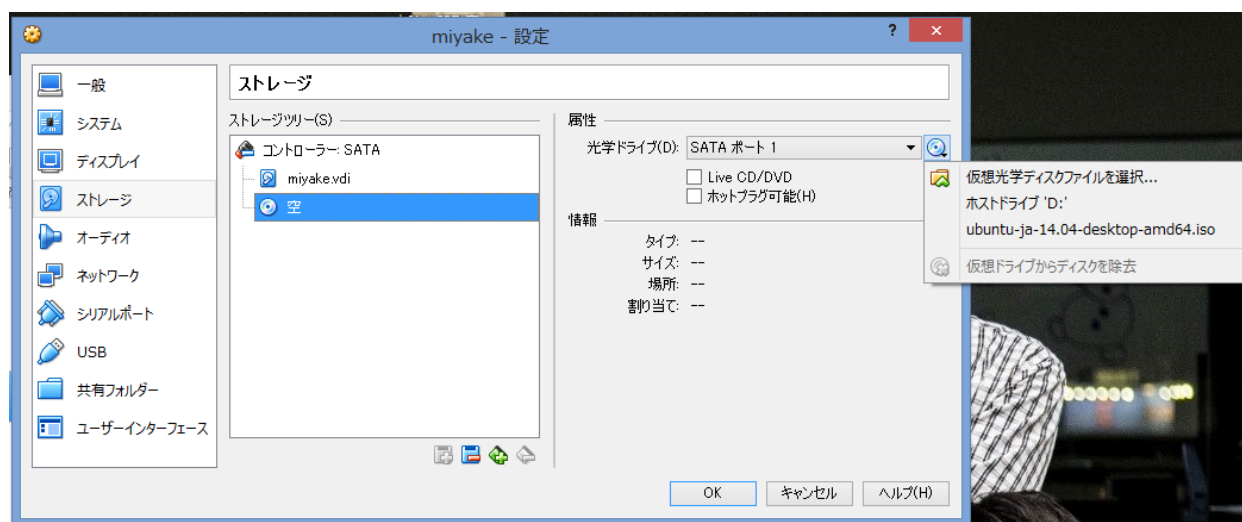


図 7.3 仮想光学ディスクファイルの選択

5. 【OK】をクリックし,【起動(T)】を押してゲストマシンを起動
6. Ubuntu をインストールと出るので, インストールをクリック

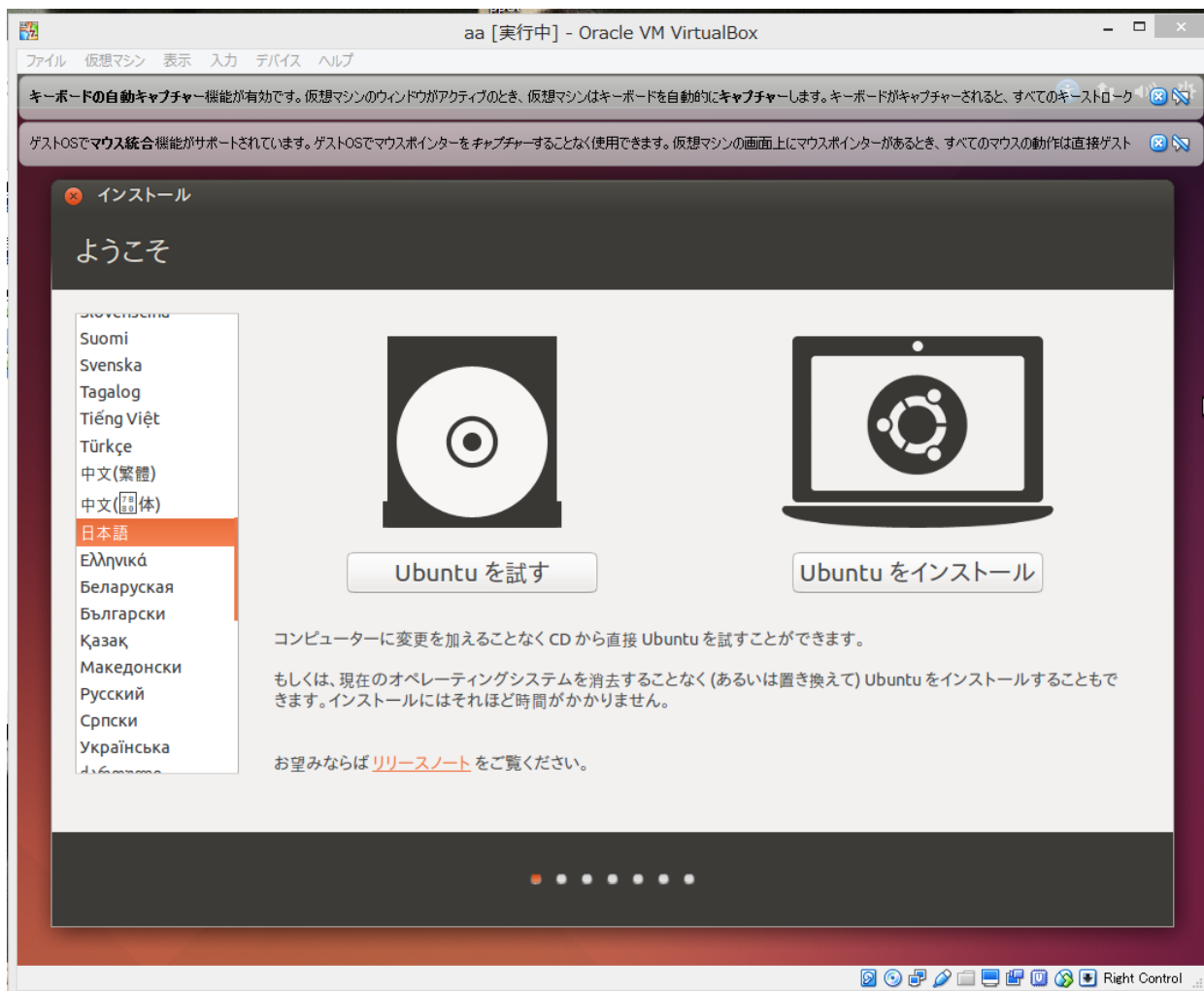


図 7.4 仮想光学ディスクファイルの選択

7. インストール完了したら,再起動と出るので再起動をクリック
8. 起動して,国を選んだ後に,名前,パスワードを決めて完了

7.6 『Ruby によるクローラー開発技法の通りに株価を取得

まずは,佐々木拓郎,るびきち『Ruby によるクローラー開発技法』(SB クリエイティブ, 2014)の5章14節の方法を参考にそのまま実行してみる。

7.6.1 端末を開く

画面の一番左上にある,マウスカーソルを乗せると「コンピュータとオンラインリソースを検索」と出るアイコンをクリック。

次に検索で「端末」と入力すると出てくるのでそれを使いコードを実行していく。



図 7.5 端末の開き方

7.6.2 Ruby と Anemone をインストールする

まず Ruby のインストールをする。

```
sudo apt-get -y install ruby-dev libxml2-dev zlib1g-dev
```

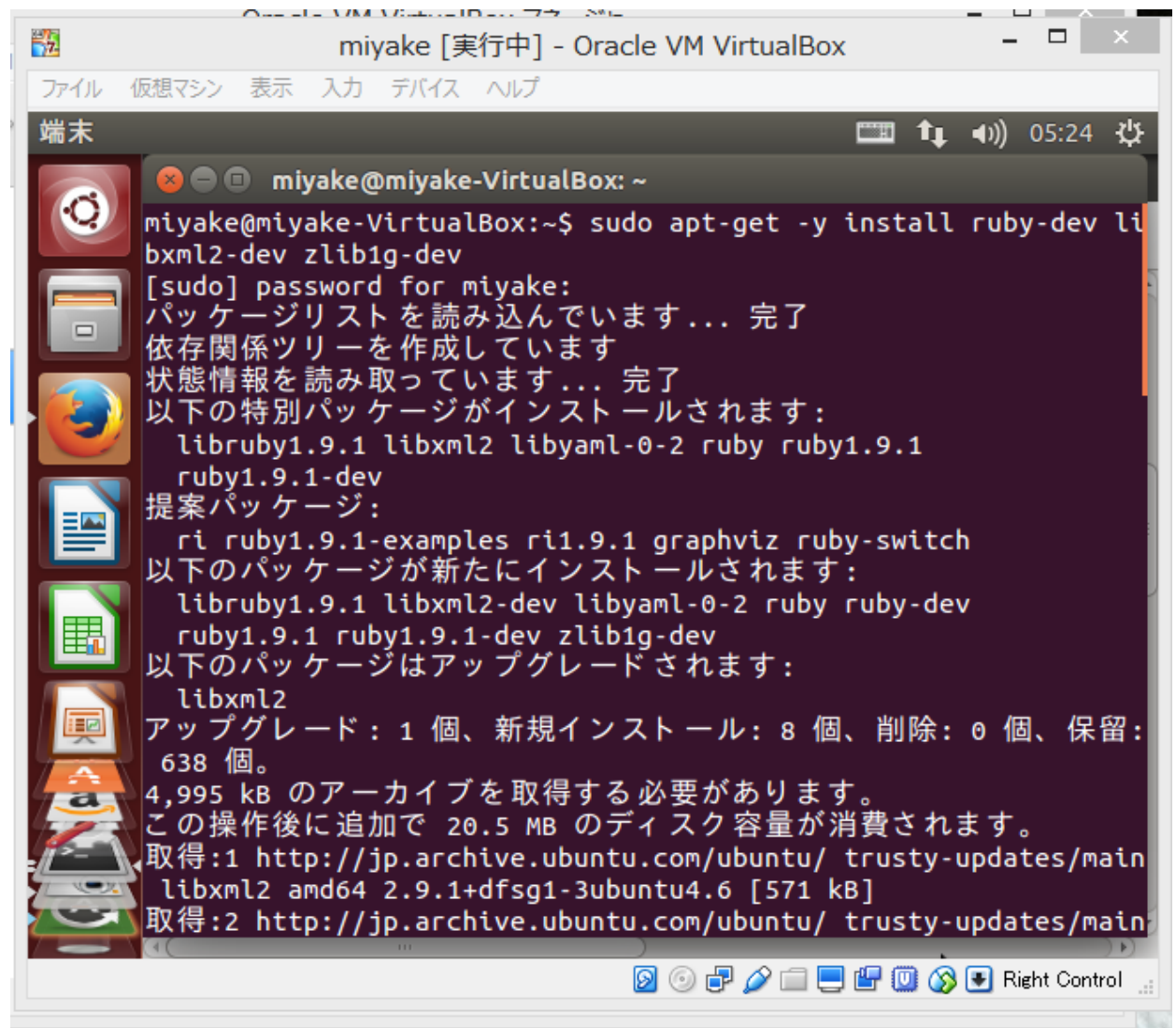


図 7.6 Ruby のインストール

次に、Anemone のインストールをする。sudo gem install anemone

Anemone とは Web サイトを自動でクローリングするための gem である。

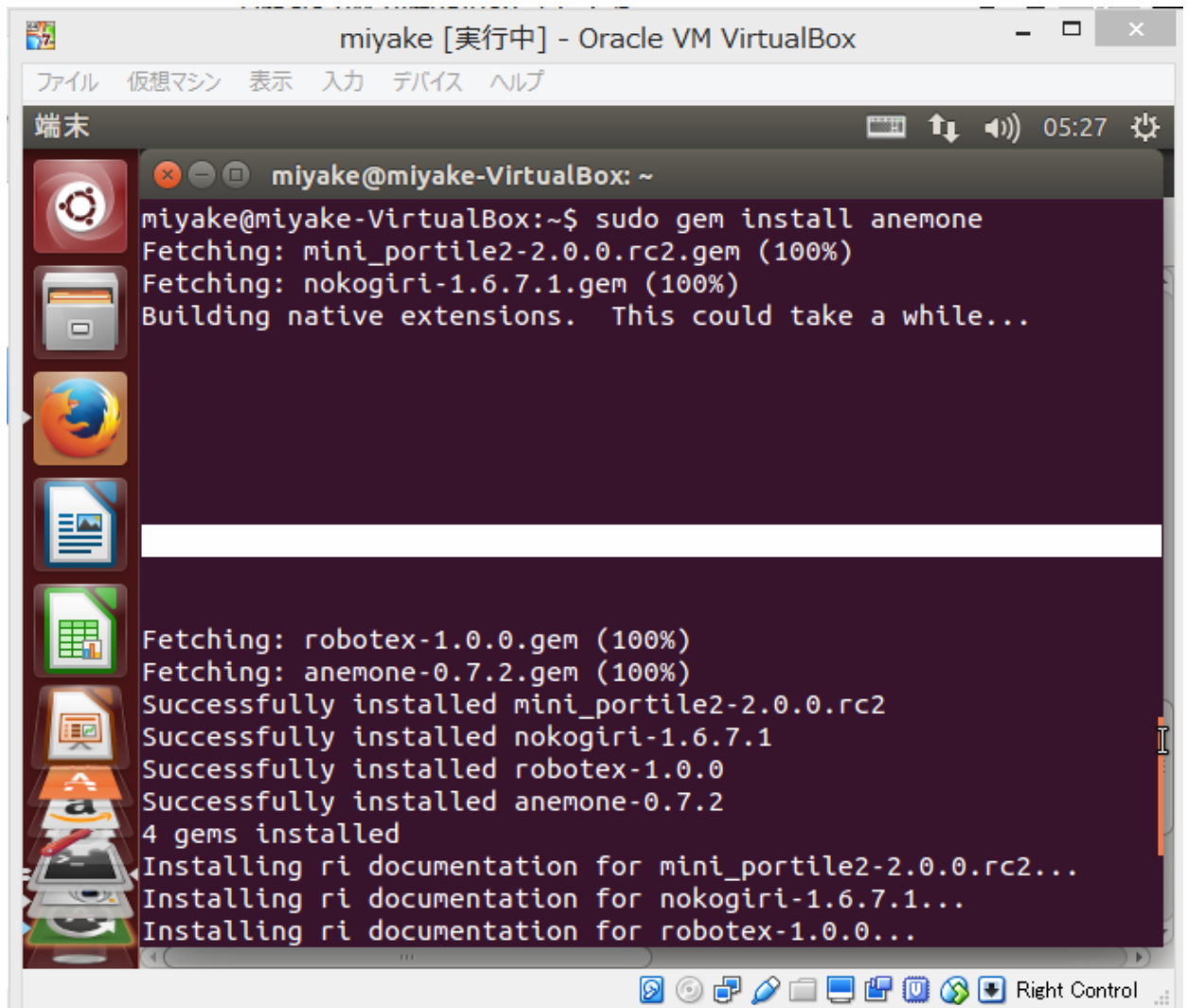


図 7.7 Anemone のインストール

7.6.3 作業ディレクトリを作り，移動する

以下のコードで作業ディレクトリを作り，そのディレクトリに移動する．

```
mkdir ruby
```

```
cd ruby
```

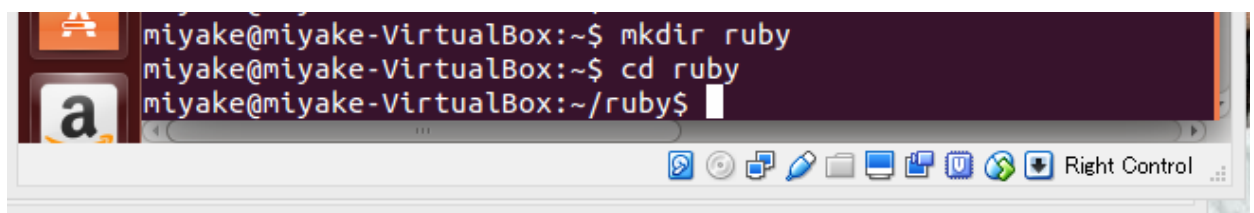


図 7.8 作業ディレクトリを作り，そのディレクトリに移動

7.6.4 サンプルスクリプトのダウンロードと使うファイルへの移動

<http://www.sbcr.jp/support/12019.html> からサンプルスクリプトをダウンロードしてから先に進む。

【『Ruby によるクローラー開発技法』サンプルスクリプト】をクリックするとダウンロードされる。

```
unzip /ダウンロード/RubyCrawlerSample.zip
```

でダウンロードしたものを読み込む。

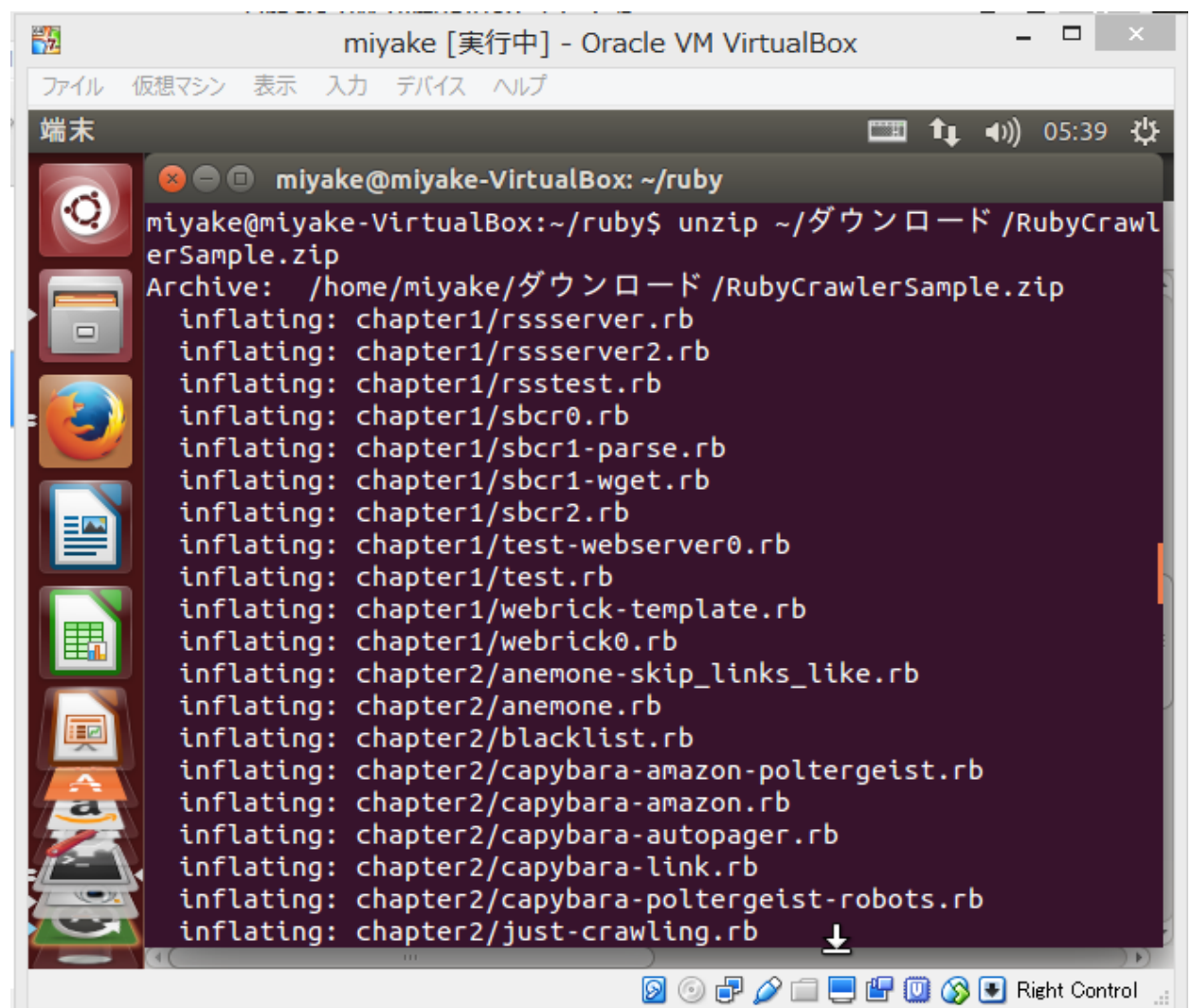


図 7.9 ダウンロードファイルの読み込み

```
cd chapter5
```

このコードで今回使うコードが入っている chapter5 に移動する。

7.6.5 株価データを収集する

ruby nokogiri-stock-history.rb

で株価を取得する。

これは本通りに進めているため、ヤフーの株価を期間問わずに取得している。

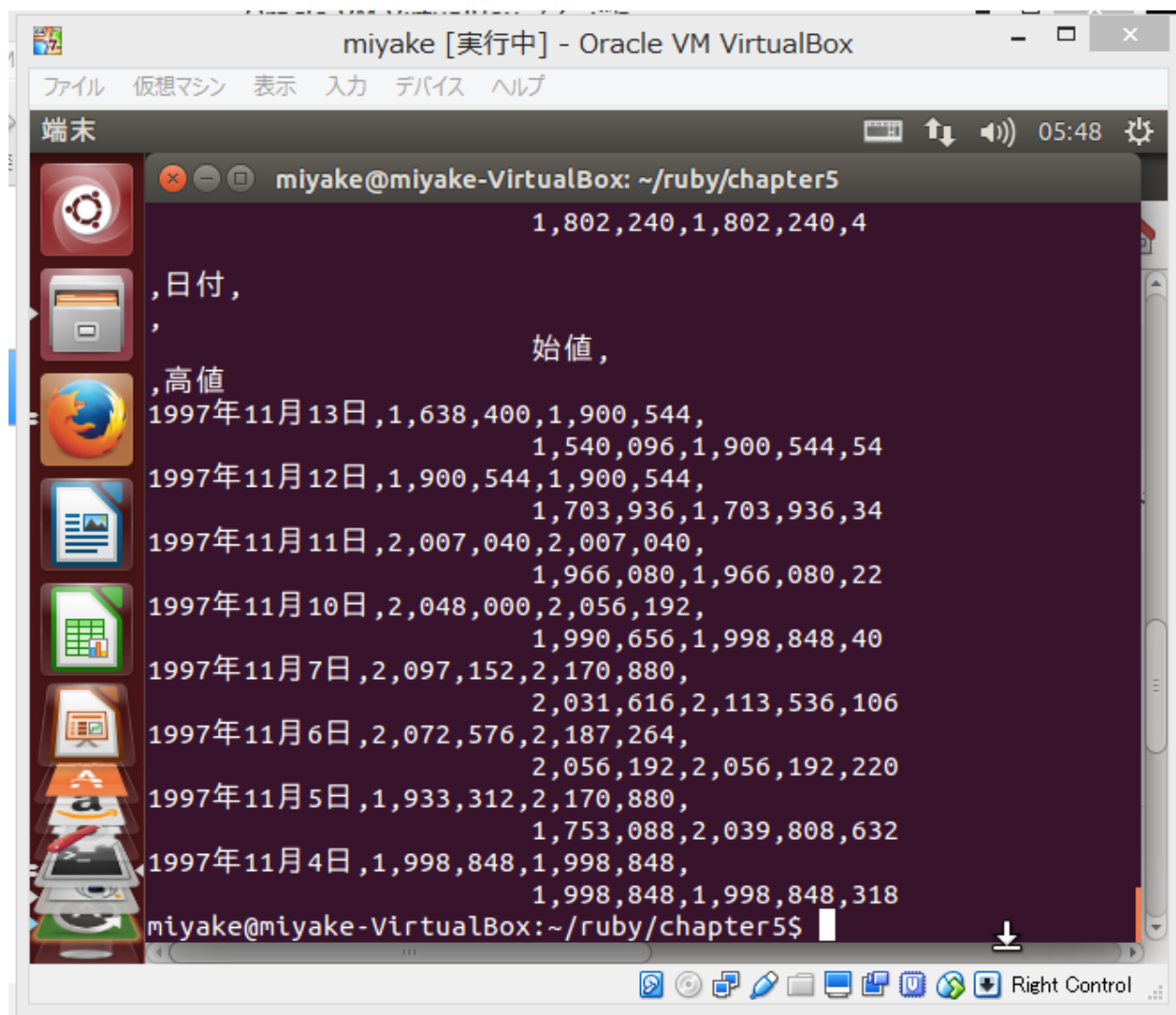


図 7.10 ダウンロードファイルの読み込み

左から、日付、始値、高値、安値、終値、出来高を表示している。

7.6.6 株価取得コード

ruby nokogiri-stock-history.rb

に書いてあるコードにより、ヤフーファイナンスから株価を取得するのだが、そのコード(ヤフーファイナンスから1900年からこのコードを実行した日(最新のものの)までを取得するもの)は以下に記載する。

```

1  # -*- coding: utf-8 -*-
2  require 'nokogiri'
3  require 'open-uri'
4
5  def get_nokogiri_doc(url)
6      begin
7          html = open(url)
8          rescue OpenURI::HTTPError
9              return
10         end
11         Nokogiri::HTML(html.read, nil, 'utf-8')
12     end
13
14     def has_next_page?(doc)
15         doc.xpath("//*[@id='main']/ul/a").each {|element|
16             return true if element.text == "次へ"
17         }
18         return false
19     end
20
21     def get_daily_data(doc)
22         doc.xpath("//table[@class='boardFin_yjSt_marB6']/tr").each {|element|
23             # 日付行および株式分割告知を回避
24             if element.children[0].text != "日
                付" && element.children[1][:class] != "through"
25
26                 # 日付
27                 day = element.children[0].text
28
29                 # 始値
30                 open_price = element.children[1].text
31
32                 # 高値
33                 hight_price = element.children[2].text
34
35                 # 安値
36                 low_price = element.children[3].text
37
38                 # 終値
39                 closing_price = element.children[4].text
40
41                 # 出来高
42                 volume = element.children[5].text.gsub(/,/,'')
43
44                 puts "#{day},{open_price},{hight_price},
45                 _____#{low_price},{closing_price},{volume}"
46             end
47         }
48     end
49
50     # 証券コード
51     code="4689"
52
53     # 検索日
54     day=Time.now
55     ey=day.year
56     em=day.month

```

```
57 ed=day.day
58 start_url="http://info.finance.yahoo.co.jp/history/?
59 sy=1900&sm=1&sd=1&ey=#{ey}&em=#{em}&ed=#{ed}&tm=d&code=#{
   code}"
60 num=1
61 puts "日付始値高値安値終値出来高,,,,,"
62 loop {
63     url = "#{start_url}&p=#{num}"
64     doc = get_nokogiri_doc(url)
65     get_daily_data(doc)
66     break if !has_next_page?(doc)
67     num = num+1
68 }
```

7.7 株価取得ツール使用方法

本節では研究で開発した、株価取得の方法を書いていく。

7.7.1 Ruby と Anemone のインストール

第 8 章にある 8.6.2 と同じように進める。

もうすでに両方ともインストールしてあるため、入力は省略する。

7.7.2 作業ディレクトリの作成と移動

第 8 章にある 8.6.3 と同じように進めるが今回は名前を「kabuka」にして進める。

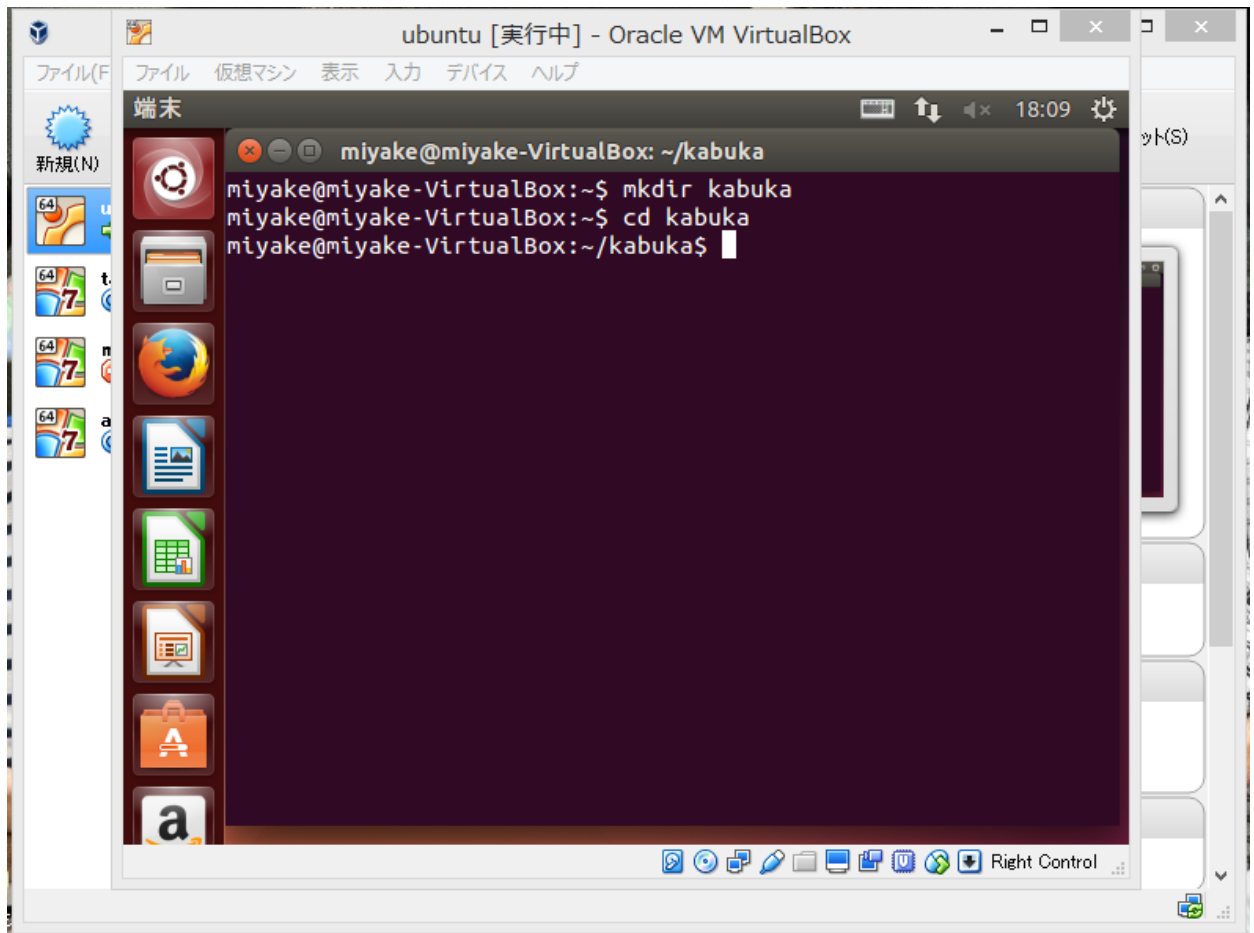


図 7.11 作業ディレクトリの作成と移動 2

7.7.3 ディレクトリにコードファイルを入れる

この 3 つのファイルを作成した「kabuka」のディレクトリに入れておく。

- history.pl
- plot.sh
- nokogiri-stock-history2.rb

それぞれのコードとその役割を説明する。

「history.pl」は取得した株価データをどのようなグラフの描き方にするのかを決め、そのグラフを描くものである。

```
1 set datafile separator ","
2 set xdata time
3 set timefmt "%Y/%m/%d"
4 set format x "%Y/%m"
5 set xl "month"
6 set yl "price"
7 set terminal png
8 set out "stock.png"
```

```
9 plot 'stock.csv' using 1:2 with l linetype -1 title ""
```

「plot.sh」は取得した株価データを CSV 形式に保存したり、PNG ファイルにするものである。

```
1 rm -f stock.csv
2 cp $1.csv stock.csv
3
4 gnuplot history.pl
5
6 cp stock.png $1.png
7 rm -f stock.png
```

「nokogiri-stock-history2.rb」は銘柄・期間を指定し、株価をヤフーファイナンスから取得してくるものである。

```
1 # -*- coding: utf-8 -*-
2 require 'nokogiri'
3 require 'open-uri'
4
5 def get_nokogiri_doc(url)
6   begin
7     html = open(url)
8     rescue OpenURI::HTTPError
9     return
10  end
11  Nokogiri::HTML(html.read, nil, 'utf-8')
12 end
13
14 def has_next_page?(doc)
15   doc.xpath("//*[@id='main']/ul/a").each {|element|
16     return true if element.text == "次へ"
17   }
18   return false
19 end
20
21 def get_daily_data(doc)
22   doc.xpath("//table[@class='boardFin_yjSt_marB6']/tr").each {|element|
23     # 日付行および株式分割告知を回避
24     if element.children[0].node_name == "td" && element.children[1][:class] !=
25       "through"
26
27       # 日付
28       day = element.children[0].text.gsub年(/,/).gsub月(/,/).gsub日(/,/,'')
29
30       # 始値
31       open_price = element.children[1].text.gsub(/,/,'')
32
33       # 高値
34       hight_price = element.children[2].text.gsub(/,/,'')
35
36       # 安値
37       low_price = element.children[3].text.gsub(/,/,'')
38
39       # 終値
40       closing_price = element.children[4].text.gsub(/,/,'')
```

```

41             # 出来高
42             volume = element.children[5].text.gsub(/./, '')
43
44             puts "#{day},#{open_price},#{hight_price},#{low_price},#{
               closing_price},#{volume}"
45         end
46     }
47 end
48
49
50 # 証券コード ( コマンドライン引数 )
51 code=ARGV[0]
52
53 # 検索日
54 sy=ARGV[1]
55 sm=ARGV[2]
56 sd=ARGV[3]
57 ey=ARGV[4]
58 em=ARGV[5]
59 ed=ARGV[6]
60 start_url="http://info.finance.yahoo.co.jp/history/?sy=#{sy}&sm=#{sm}&sd=#{
               sd}&ey=#{ey}&em=#{em}&ed=#{ed}&tm=d&code=#{code}"
61 num=1
62 puts "日付始値高値安値終値出来高,,,,,"
63 loop {
64     url = "#{start_url}&p=#{num}"
65     doc = get_nokogiri_doc(url)
66     get_daily_data(doc)
67     break if !has_next_page?(doc)
68     num = num+1
69 }

```

これらのコードファイルが今回の研究成果となる，
株価データの可視化に使われるものである．

\section{ディレクトリのコードファイルを読み込む}

ls を実行すると以下のように読み取ることができる．

```

\begin{figure}[H]
\centering
\includegraphics[width=15cm]{yomikomi.PNG}
\caption{ファイル読み込み}\label{サンプル図}
\end{figure}

```

\section{銘柄の選択}

今回はヤフーの株価を取得してみる．

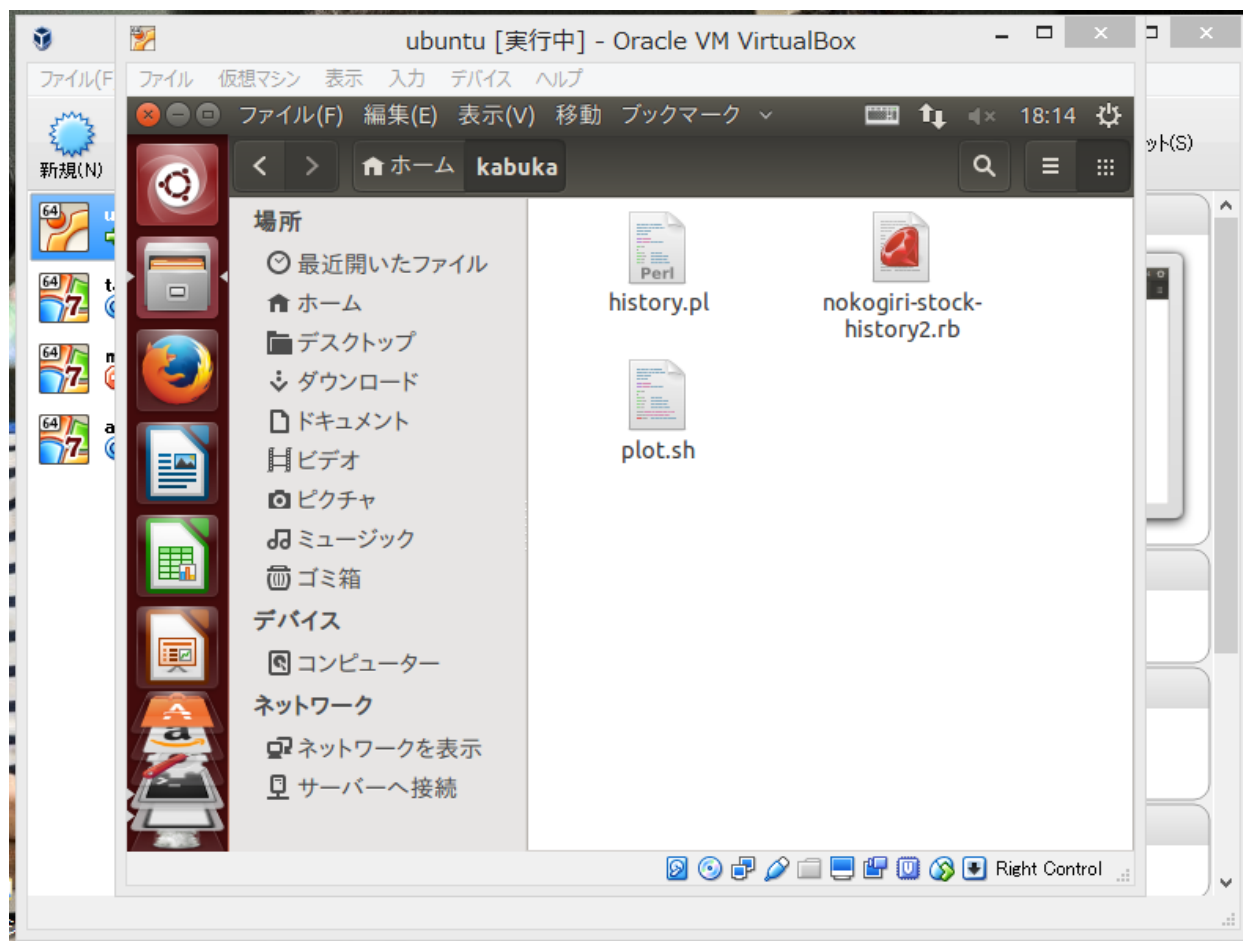


図 7.12 ファイルを入れる

銘柄コードは 4689 である .

```
\begin{figure}[H]
\centering
\includegraphics[width=15cm]{code.PNG}
\caption{銘柄選択}\label{サンプル図}
\end{figure}
```

code=

イコールのあとに取得したい銘柄を入力するとその銘柄の株価を取得することができる .

```
\section{日付選択をし株価の取得}
```

```
\begin{figure}[H]
\centering
\includegraphics[width=15cm]{hiduke.PNG}
\caption{日付選択をし株価の取得}\label{サンプル図}
\end{figure}
```



```
{\small  
\begin{verbatim}  
ruby nokogiri-stock-history2.rb ${code} 2015 12 1 2015 12 31 > ${code}.csv
```

の `${code} 2015 12 1 2015 12 31 > ${code}` の中の数字を変えれば
日付を指定して株価を取得することができる。

今回はヤフーの 2015 年 12 月 1 日から 2015 年 12 月 31 日までの株価を取得する。

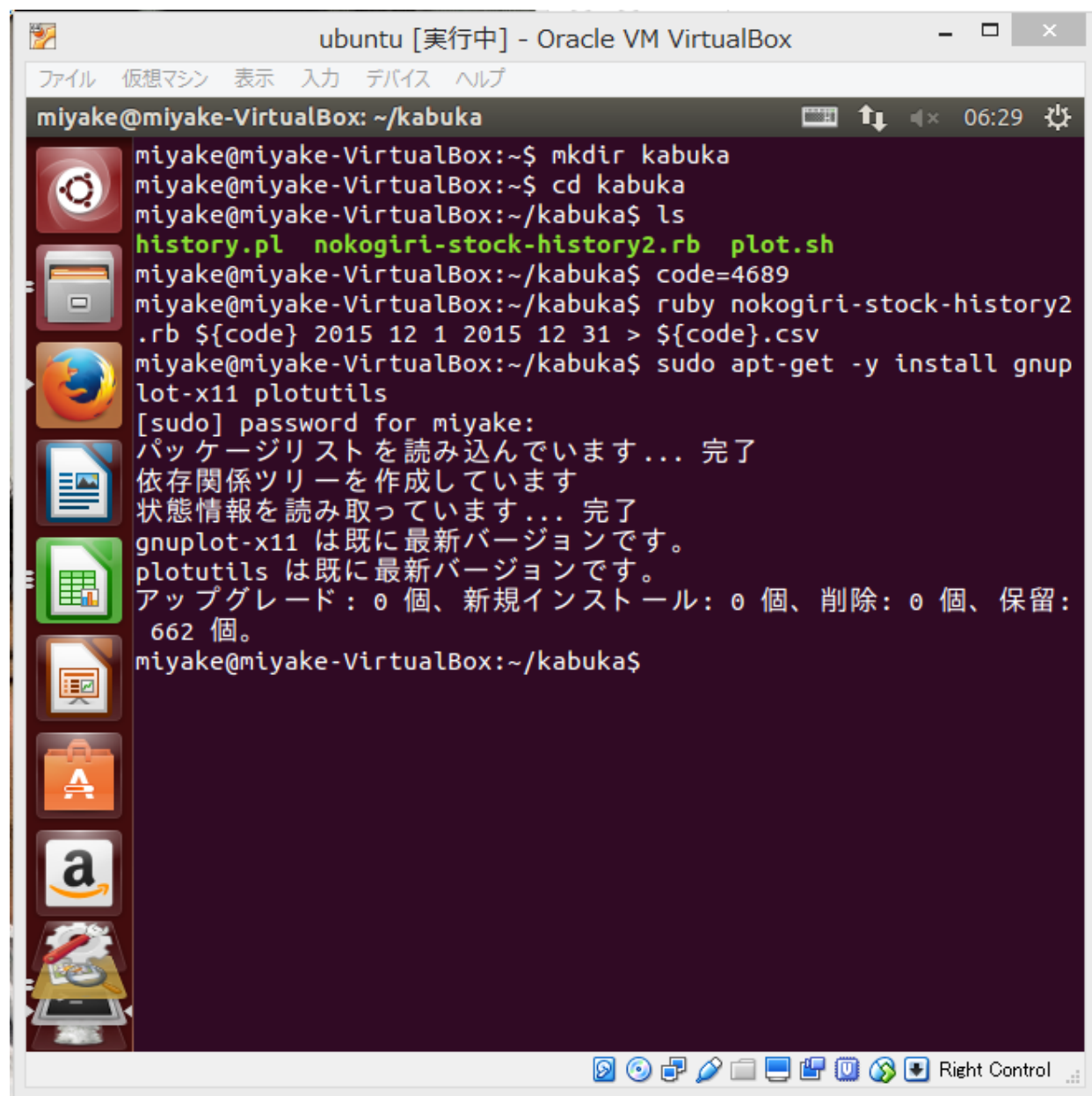


図 7.13 Gnuplot のインストール

```
sudo apt-get -y install gnuplot-x11 plotutils
```

このコードで「Gnuplot」をインストールすることができる。

7.7.5 株価データの可視化

株価データの可視化は取得した株価変動データを png にして、最初に作った「kabuka」のファイルに入れる。

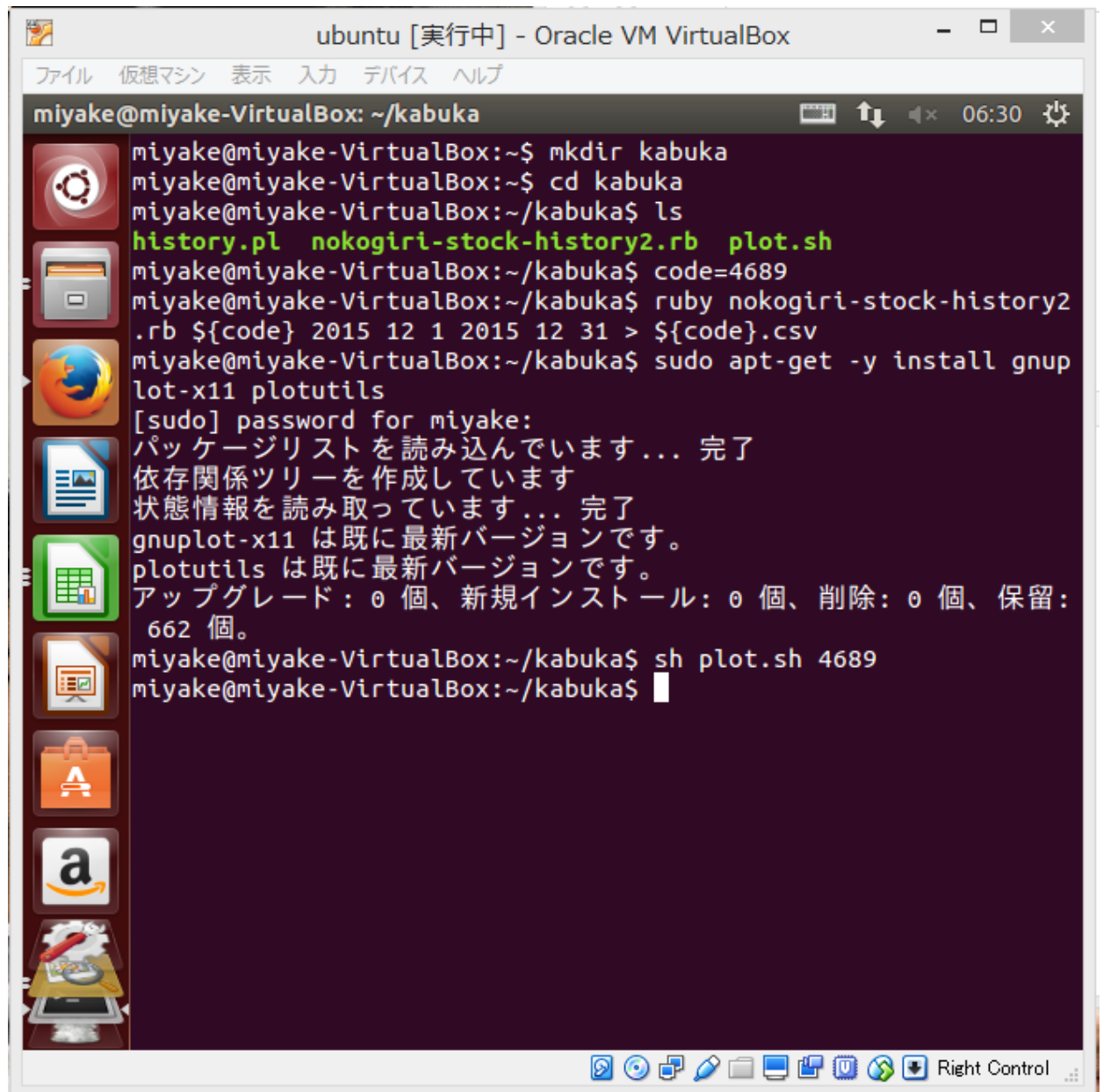


図 7.14 株価変動データを png にする

sh plot.sh 4689

を入力すると「4689.png」と「4689.csv」ができています。

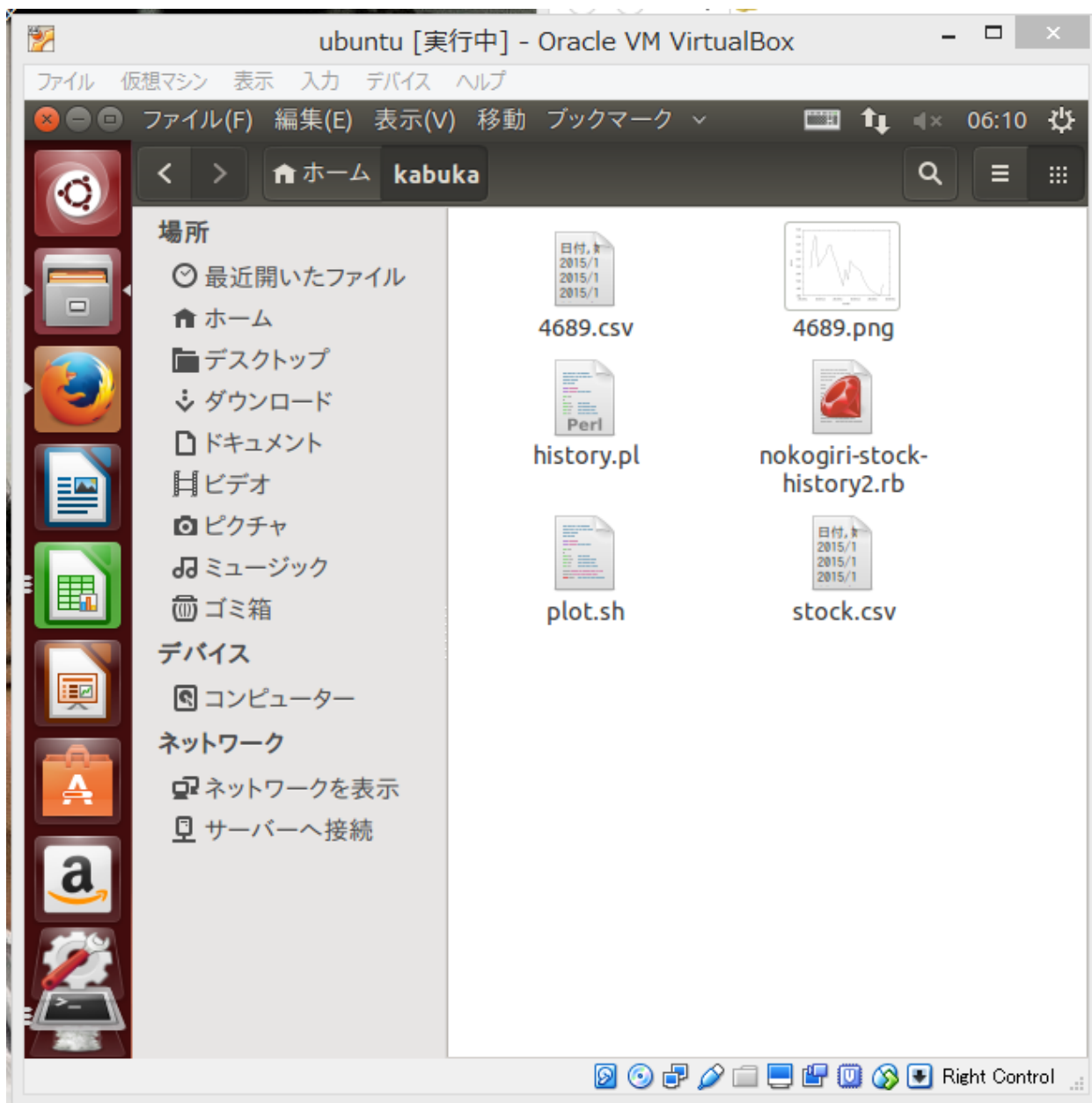


図 7.15 4689.png 作成

7.7.6 描写した株価変動データ

ヤフーの 2015 年 12 月 1 日から 2015 年 12 月 31 日までの株価変動データである．以下のように株価を取得し，可視化をすることができる．



図 7.16 ヤフーの株価変動

日付と株価については「4689.csv」を開くと、以下のように取得することができる。

	A	B	C	D	E	F	G	H	I
1	日付	始値	高値	安値	終値	出来高			
2	2015-12-30	498	501	494	494	4606500			
3	2015-12-29	491	496	490	495	5751700			
4	2015-12-28	491	495	486	493	5610100			
5	2015-12-25	494	494	487	489	4949700			
6	2015-12-24	496	497	490	491	7312000			
7	2015-12-22	502	504	500	500	7752600			
8	2015-12-21	507	511	499	506	10157100			
9	2015-12-18	511	520	503	504	11349800			
10	2015-12-17	514	514	507	508	10120200			
11	2015-12-16	501	511	492	508	19273500			
12	2015-12-15	508	513	496	499	9869700			
13	2015-12-14	497	503	492	502	10334600			
14	2015-12-11	514	525	510	512	11854800			
15	2015-12-10	518	523	511	522	15550800			
16	2015-12-09	525	531	518	526	11031200			
17	2015-12-08	524	528	519	526	10190000			
18	2015-12-07	518	528	518	522	5928300			
19	2015-12-04	510	518	507	515	8110100			
20	2015-12-03	531	535	512	517	19234400			
21	2015-12-02	526	537	525	533	16321800			
22	2015-12-01	509	525	507	520	13556700			

図 7.17 4689.csv

7.8 背景にある調査の進め方

背景にある調査はどんな事故や事例でも進められるものではなく、チャレンジャー号墜落事故の事例と同じような条件の事故を調べる。

7.8.1 調べる事故の条件

- 複数の企業が関わっていること
- 原因企業が判明するまでに時間がかかっていること
- 原因企業が株式会社であること

7.8.2 条件に当てはまる事故

ウェブ検索で見つかった事故は以下の 4 件である．

- スペースシャトル・チャレンジャー号爆発事故

1986 年 1 月 28 日，アメリカ合衆国のスペース・シャトルチャレンジャー号が打ち上げから 73 秒後に分解し，7 名の乗組員が死亡した事故である．同オービタは北米東部標準時午前 11 時 39 分 (16:39UTC , 1 月 29 日 1:39JST) にアメリカ合衆国フロリダ州中部沖の大西洋上で空中分解した．

<事故の概略>

機体全体の分解は，右側固体燃料補助ロケット (Solid Rocket Booster, SRB) の密閉用 O リングが発進時に破損したことから始まった．O リングの破損によってそれが密閉していた SRB 接続部から漏洩が生じ，固体ロケットエンジンが発生する高温・高圧の燃焼ガスが噴き出して隣接する SRB 接続部材と外部燃料タンク (External Tank, ET) に悪影響を与えた．この結果，右側 SRB の尾部接続部分が分離すると共に外部燃料タンクの構造破壊が生じた．空気力学的な負荷により軌道船は一瞬の内に破壊された [12] ．



図 7.18 チャレンジャー号爆発」

- トルコ航空 D C-10 パリ墜落事故

1974 年にフランスで発生したトルコ航空 981 便の DC-10-10(マクドネルダグラス社製, 機体記号 TC-JAV) が墜落した航空事故である (別名: トルコ航空 981 便墜落事故)。

<事故の概略>

事故機は離陸 10 分後, パリから北へ 15km 離れたサン=パトゥス村 (Saint-Pathus) 上空、高度 11500 フィート (3500 メートル) まで上昇したときに, ロックが不完全だった左側後部貨物室ドアが客室の与圧により吹き飛ばされて急減圧が起こった。この時にパイロットが持っていたマニュアルが壁にたたきつけられる音がコックピットボイスレコーダーに収録されている。貨物室の減圧に伴い客室の床が破壊され, 乗客 6 名が座席ごと空中に放り出された。また, 床下を通るコントロールラインが切断されて, 方向舵, 昇降舵, 尾部エンジンの制御が不可能になり, 操縦不能のままドアの脱落から 1 分 17 秒後に 430 ノット (約 796km/h) の速さで墜落に至った [13]。



図 7.19 トルコ航空 DC-10」

- 日本航空 123 便墜落事故

1985 年 (昭和 60 年) 8 月 12 日月曜日 18 時 56 分に, 東京 (羽田) 発大阪 (伊丹) 行同社定期 123 便ボーイング 747SR-100(ジャンボジェット, 機体記号 JA8119, 製造番号 20783) が, 群馬県多野郡上野村の高天原山の尾根 (通称「御巣鷹の尾根」) に墜落した航空事故である。

<事故の概略>

事故機の後部圧力隔壁が損壊し, その損壊部分から客室内の空気が機体後部に流出したことによって, 機体尾部と垂直尾翼の破壊が起こった。さらに, 4 系統ある油圧パイプがすべて破壊されたことで作動油が流出し, 操縦機能の喪失が起こった [14]。



図 7.20 日本航空 123 便」

- 東京航空交通管制システム障害

2003 年 3 月 1 日 7 時ごろ、飛行計画情報処理システム (FDP) のシステムトラブルは、航空会社の欠航便 205 便、30 分以上の遅延便は 1462 便、また 2 日間にわたってダイヤが乱れるという大障害となった。

<障害の概略>

2002 年 9 月に変更した FDP に NEC (日本電気 (株)) によるプログラム・ミスがあった。2003 年 3 月 1 日、「防衛庁システム対応プログラム」を変更したことと、当日 7 時、オンライン情報処理プログラムが起動したことにより、これまで隠れていたこのミスが表に現れ、システムがダウンした [15]。

7.8.3 各事故に関連する企業情報の入手

事故の原因企業以外にも事故に関連する企業情報を取得することができないと、比較対象がないため、調査を進めることができない。

本研究ではその技術の開発を目的とはしていないため、これより先は進めることができない。

7.8.4 事故関連企業の株価変動データの比較・解析

事故関連企業の株価変動データの比較・解析を行う。

この流れを事件事例を増やして行っていく。

7.9 横浜マンション傾斜問題の調査

開発したツールを使い株価変動データをいくつか比較できるかテストする。

それと同時に背景にあることを調査するための条件に当てはまる事例が「横浜マンション傾斜問題」であり、関連する企業も公になっているためこの事例について調査する。

7.9.1 横浜マンション傾斜問題とは

この問題は 2015 年 10 月 14 日新聞報道があって話題になった。

三井不動産レジデンスが横浜市に、2006（平成 18）年から販売が開始されたマンション、「パークシティ LaLa 横浜」のタイルを積んだ際にできる継ぎ目である「目地」が上下で最大 2.4cm ずれていたという問題である。

7.9.2 横浜マンション傾斜理由

マンション傾斜の理由は 10 月 14 日、新聞報道があった後に下請けであった「旭化成建材」が杭打ちのデータ改ざんを認めた。本来使うはずであった杭より短いものを使っていたため、マンションが傾いてしまった。

だが、短い杭を意図的に発注したのは「三井住友建設」である。

7.9.3 横浜マンション建設の流れ

「パークシティ LaLa 横浜」の建設の流れは以下のとおりである。

1. 三井不動産レジデンシャルが工事発注をする
2. 元請けとして三井住友建設が入る
3. 1 次下請けとして日立ハイテクノロジーズが入る
4. 2 次下請けとして旭化成建材が入る

このような流れからこのマンションは建てられた。

7.9.4 次々と発覚する事実

この問題は次々と事実が発覚していった。

2015 年 10 月 14 日に三井不動産がマンションが傾いていることを公表した。

公表当日には旭化成建材が杭打ちのデータ改ざんを認めた。

2015 年 10 月 24 日三井住友建設には、施工主の三井住友建設が、杭の長さが足りないとが判明した場所の強固な地盤「支持層」が実際には深さ 16 メートル付近にあるのに、設計段階で 2 メートル浅い約 14 メートルと見込んでくいを発注していたことが取材により発覚した。

さらには、2015 年 10 月 27 日に日立ハイテクノロジーズが問題があった場合の責任は 2 次下請けが負うとの契約を結んでいたという。

「技術的な知見はなく、旭化成建材の担当者が改ざんした工事データの「信憑性は判断できなかった」と旭化成建材に丸投げをした疑惑が残った。

7.9.5 原因企業

前節の各企業で主犯であったのは元請けである三井住友建設であると考えられる。

なぜなら 2 メートル短い杭を発注し、下請けである旭化成建材にそのまま建設をさせたからである。

そのことを前提としてこの事例に関連する 4 社の株価を取得していく。

第 8 章

結果

「横浜マンション傾斜問題」の関連企業の株価を取得した結果を書いていく .
section 株価の可視化

8.0.6 三井不動産

株価変動データは以下のものであった .

ubuntu [実行中] - Oracle VM VirtualBox

ファイル 仮想マシン 表示 入力 デバイス ヘルプ

8801.csv - LibreOffice Calc

06:03

TakaoPGothic 10

G4 $f(x)$ Σ =

	A	B	C	D	E	F	G	H	I
1	日付	始値	高値	安値	終値	出来高			
2	2015-10-30	3296	3356	3239	3310	5223000			
3	2015-10-29	3343	3348	3241	3270	4014000			
4	2015-10-28	3318	3344	3298	3325	2469000			
5	2015-10-27	3315	3379	3293	3306	3198000			
6	2015-10-26	3382	3388	3328	3339	2973000			
7	2015-10-23	3342	3372	3310	3337	5276000			
8	2015-10-22	3259	3283	3223	3248	3634000			
9	2015-10-21	3202	3296	3198	3272	6596000			
10	2015-10-20	3388	3390	3238	3262	5522000			
11	2015-10-19	3337	3363	3318	3350	4293000			
12	2015-10-16	3369	3480	3369	3400	5135000			
13	2015-10-15	3294	3367	3281	3343	4037000			
14	2015-10-14	3393	3398	3321	3325	4742000			
15	2015-10-13	3493	3504	3421	3423	3780000			
16	2015-10-09	3483	3524	3459	3524	3877000			
17	2015-10-08	3480	3499	3430	3457	3906000			
18	2015-10-07	3445	3491	3415	3486	5034000			
19	2015-10-06	3499	3536	3447	3460	4565000			
20	2015-10-05	3464	3465	3395	3432	3007000			
21	2015-10-02	3401	3426	3358	3402	2658000			
22	2015-10-01	3320	3459	3312	3430	6549000			

Sheet1

Sheet 1 / 1 標準

合計=0 100%

Right Control

図 8.1 三井不動産株価変動

グラフにより可視化したのは以下のものである。

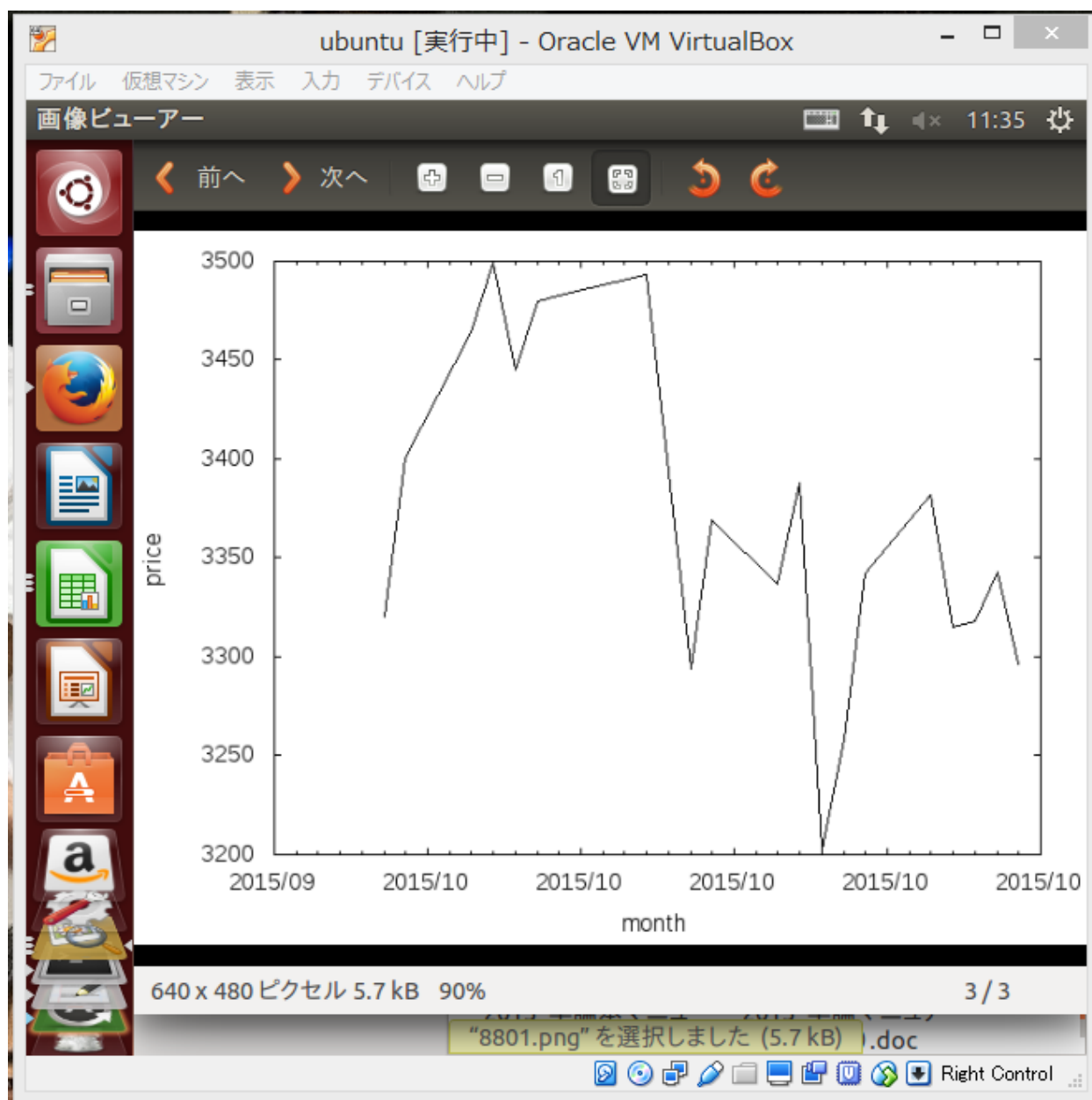


図 8.2 三井不動産株価変動グラフ

8.1 旭化成

株価変動データは以下のようであった。

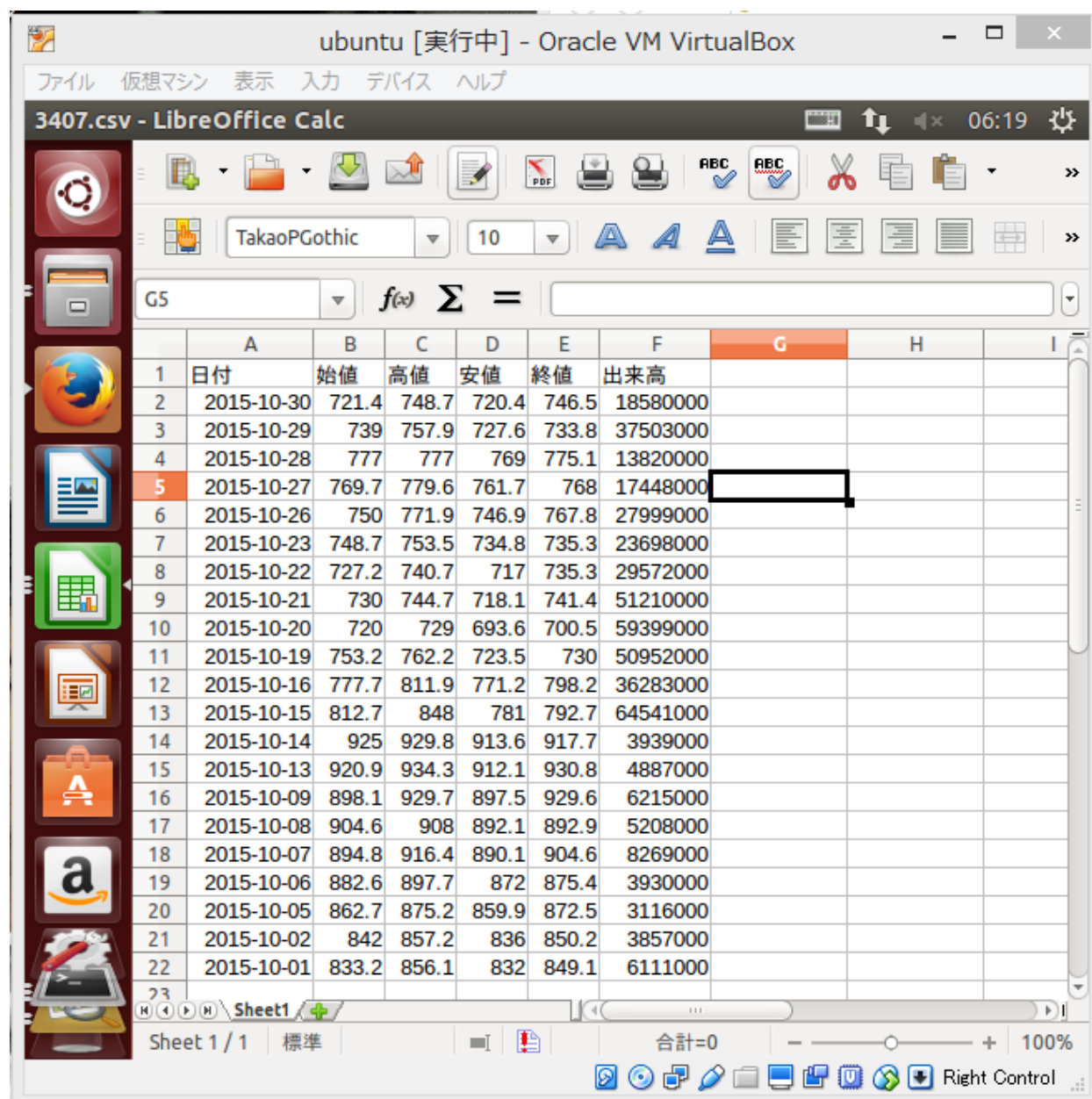


図 8.3 旭化成株価変動

グラフにより可視化したのは以下のものである。

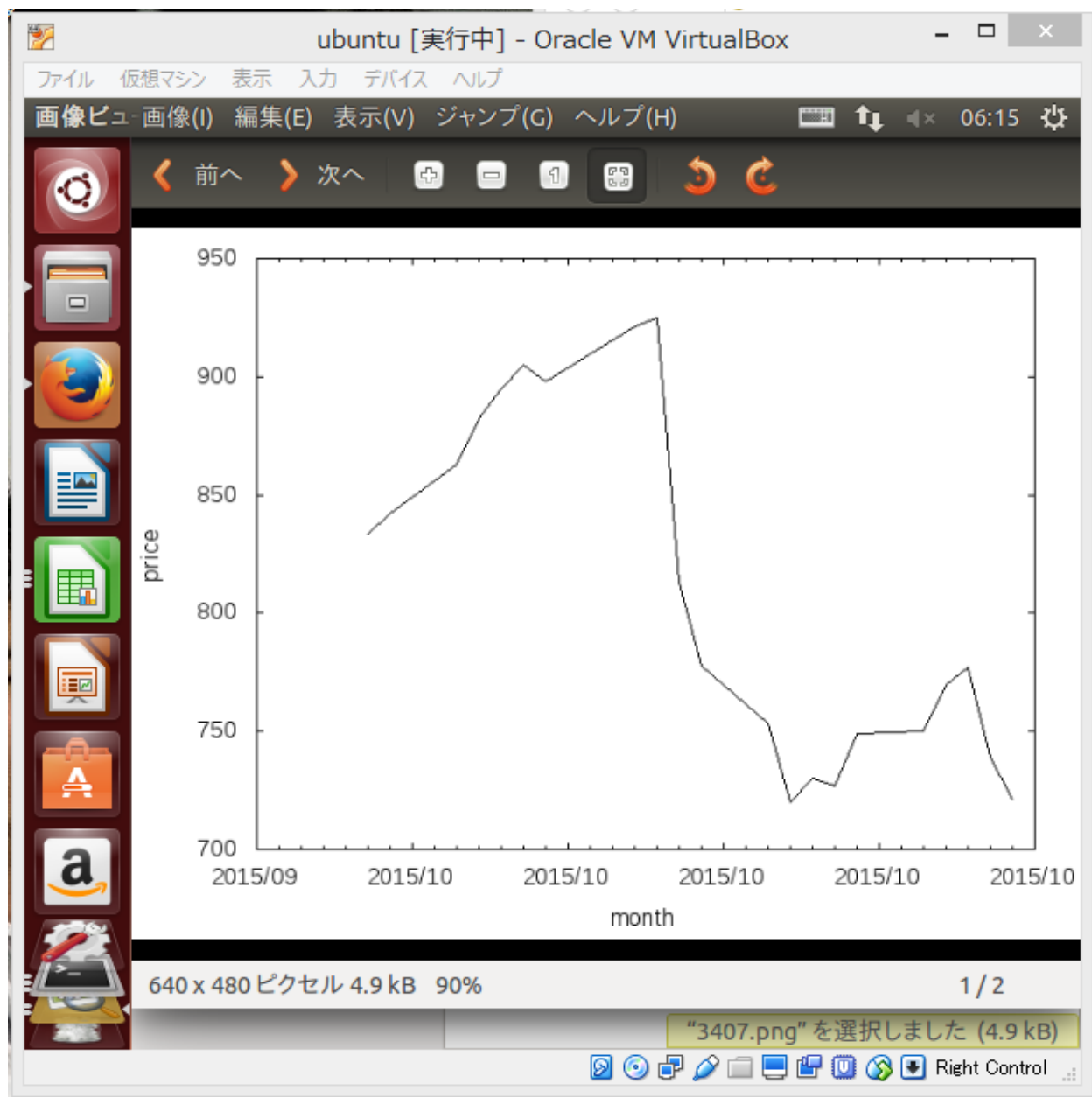


図 8.4 旭化成株価変動グラフ

8.2 三井住友建設

株価変動データは以下のようであった。

1821.csv - LibreOffice Calc

ファイル 仮想マシン 表示 入力 デバイス ヘルプ

04:51

関数記号 $f(x)$ Σ =

	A	B	C	D	E	F	G	H	I
1	日付	始値	高値	安値	終値	出来高			
2	2015-10-30	116	118	115	118	26496300			
3	2015-10-29	117	118	115	116	31026400			
4	2015-10-28	117	118	115	116	29949300			
5	2015-10-27	120	120	116	116	44082800			
6	2015-10-26	119	120	117	120	37480700			
7	2015-10-23	122	124	120	123	31304400			
8	2015-10-22	119	122	118	119	44721100			
9	2015-10-21	118	120	115	118	68875400			
10	2015-10-20	121	122	112	115	81252400			
11	2015-10-19	122	124	119	122	86225500			
12	2015-10-16	131	133	128	131	69832600			
13	2015-10-15	133	137	131	135	231782600			
14	2015-10-14	142	146	109	109	338788700			
15	2015-10-13	160	162	158	159	21035100			
16	2015-10-09	156	161	154	160	27770600			
17	2015-10-08	156	157	154	156	10701300			
18	2015-10-07	154	156	152	155	8714200			
19	2015-10-06	157	158	153	155	13017300			
20	2015-10-05	147	155	147	155	28026400			
21	2015-10-02	144	149	144	146	8465000			
22	2015-10-01	146	147	143	146	5468500			

Sheet1

Sheet 1 / 1 標準

合計=0 100%

Right Control

図 8.5 三井住友建設株価変動

グラフにより可視化したのは以下のものである。

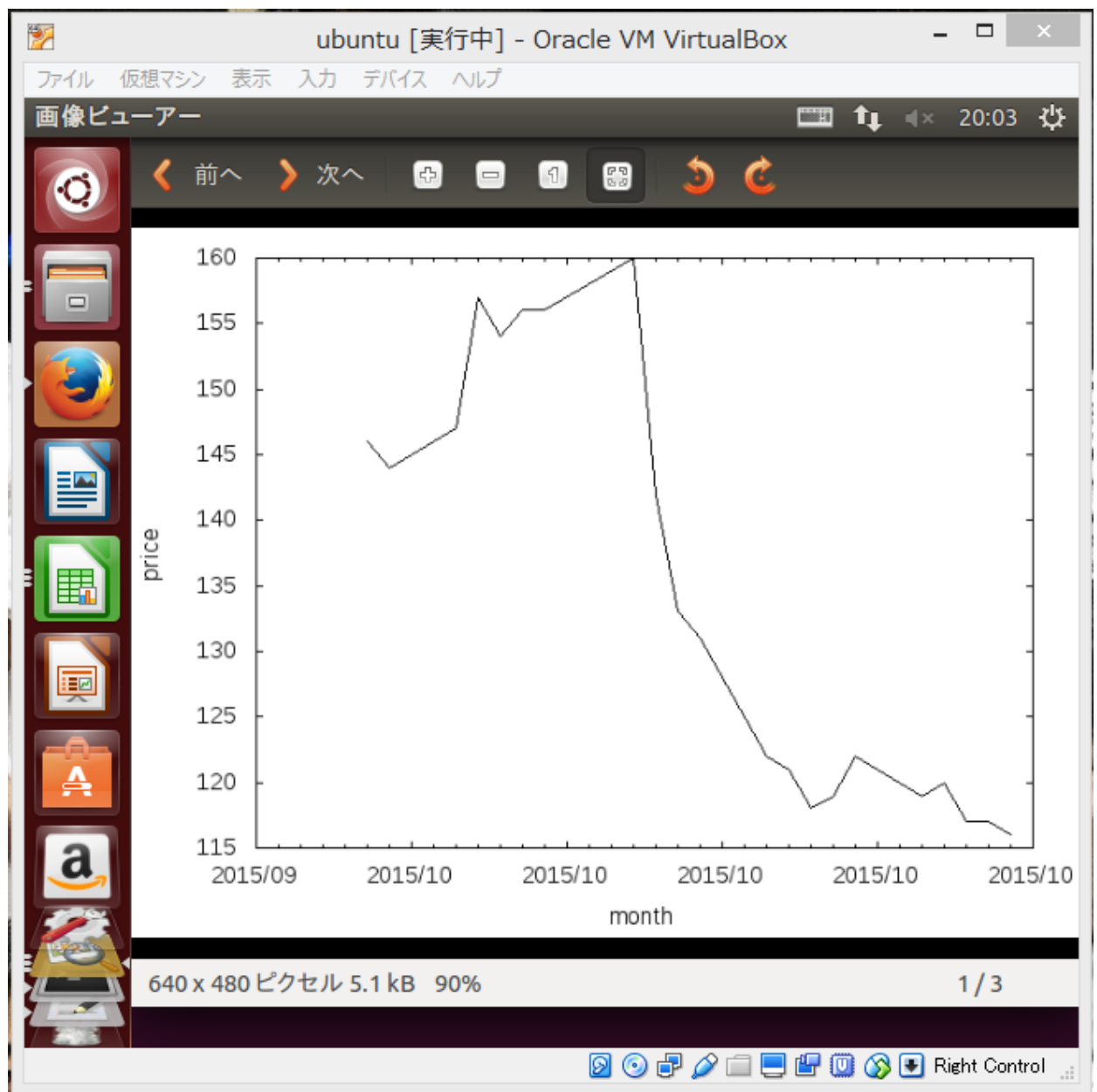
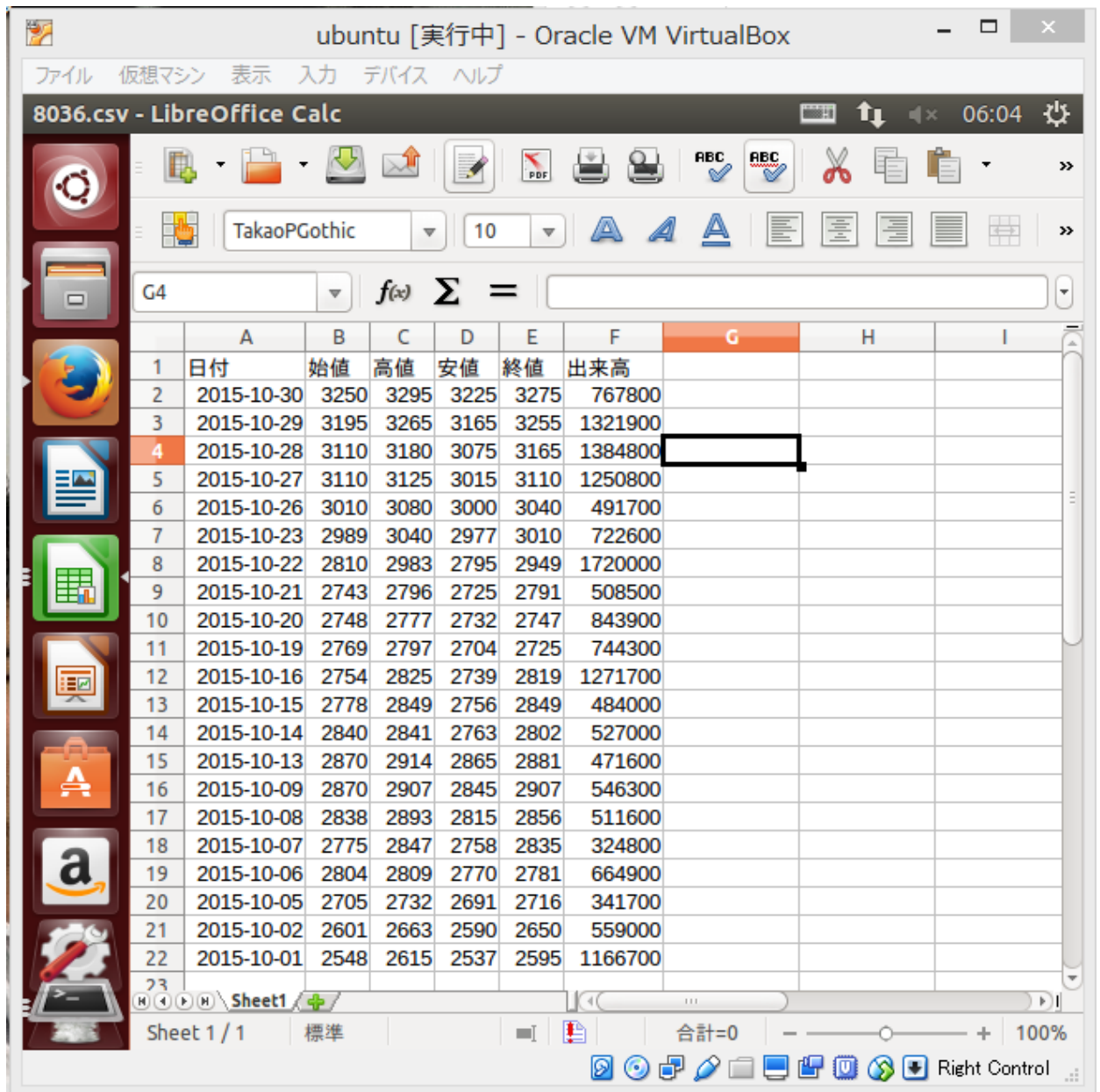


図 8.6 三井住友建設株価変動グラフ

8.3 日立ハイテクノロジーズ

株価変動データは以下のようであった。



	A	B	C	D	E	F	G	H	I
1	日付	始値	高値	安値	終値	出来高			
2	2015-10-30	3250	3295	3225	3275	767800			
3	2015-10-29	3195	3265	3165	3255	1321900			
4	2015-10-28	3110	3180	3075	3165	1384800			
5	2015-10-27	3110	3125	3015	3110	1250800			
6	2015-10-26	3010	3080	3000	3040	491700			
7	2015-10-23	2989	3040	2977	3010	722600			
8	2015-10-22	2810	2983	2795	2949	1720000			
9	2015-10-21	2743	2796	2725	2791	508500			
10	2015-10-20	2748	2777	2732	2747	843900			
11	2015-10-19	2769	2797	2704	2725	744300			
12	2015-10-16	2754	2825	2739	2819	1271700			
13	2015-10-15	2778	2849	2756	2849	484000			
14	2015-10-14	2840	2841	2763	2802	527000			
15	2015-10-13	2870	2914	2865	2881	471600			
16	2015-10-09	2870	2907	2845	2907	546300			
17	2015-10-08	2838	2893	2815	2856	511600			
18	2015-10-07	2775	2847	2758	2835	324800			
19	2015-10-06	2804	2809	2770	2781	664900			
20	2015-10-05	2705	2732	2691	2716	341700			
21	2015-10-02	2601	2663	2590	2650	559000			
22	2015-10-01	2548	2615	2537	2595	1166700			

図 8.7 日立ハイテクノロジーズ株価変動

グラフにより可視化したのは以下のものである。



図 8.8 日立ハイテクノロジーズ株価変動グラフ

section 株価変動の結果

関連企業の株価変動結果は以下の通りであった。

- 三井不動産は 13 日から下がり始め、14 日は 2 パーセント下落
- 旭化成は 10 月 14 日まで変動はなく、15 日の始値は 14 日の終値から 11.4 パーセント下落
- 三井住友建設は 13 日終値から 14 日始値は 12 パーセント下落し、14 日始値と終値で 30 パーセント下落、その後持ち直さなかった
- 日立ハイテクノロジーズはトレンドを乱すことなく、特に変化なし

第 9 章

考察

株価に変動があったのは日立ハイテクノロジーズ以外の 3 社である。

旭化成は 14 日に杭打ちデータ改ざんを認めたため株価が下がるのは当たり前であるとする。

だが三井住友建設は 10 月 24 日に短い杭を 8 流していたことを認めたにもかかわらず、14 日に大幅にトレンドが乱れ下落している。この結果を見る限り、株式市場は三井住友建設が主犯であると言っているように感じられる。

さらにこの調査を進めるためには、背景にある事故条件とマッチする事故・事例を増やし、その事故・事例に関連する企業の情報を取得する技術を確立していけないといけないと考える。

Ruby によるクローラー開発技法により、今回であれば取得したい銘柄の株価を期間を指定して取得することができた。それができたということは、何か調べ物をしたりするときにはクローラー開発ができれば、ネット上にある膨大な情報の中から自分の取得したい情報のみが取得できると考える。

第 10 章

結論

本研究の目的である株価取得とその可視化は成功し、いくつかの企業の株価変動データの比較もできるようになった。

背景の調査に関してはマンション傾斜の事例だけを見れば株式市場は三井住友建設が今回の主犯であったとマスメディアより早く特定ができていたと言えるが、事例が 1 つしかないため断言はできない。開発したツールを使い、事例を増やし研究していけば背景にある調査はさらに進められる。

参考文献

- [1] ジェームズ・スロウィッキー. 「みんなの意見」は案外正しい. 角川文庫, 2009.
- [2] ナレッジマネジメント wikipedia <https://ja.wikipedia.org/wiki/>
- [3] 知識交換の場 osamu hasegawa films <http://www.osamuhasegawa.com/sei>
- [4] 株式会社 wikipedia <https://ja.wikipedia.org/wiki/>
- [5] 株とは 楽天証券 <https://www.rakuten-sec.co.jp/web/domestic/stock/beginner/about.html>.
- [6] 株価上がる理由 やさしい株のはじめ方 <http://kabukiso.com/apply/change/upkabu.html>.
- [7] 株価下がる理由 やさしい株のはじめ方 <http://kabukiso.com/apply/change/down.html>.
- [8] 佐々木拓郎. Ruby によるクローラー開発技術. SB クリエイティブ, 2014.
- [9] Oracle vm virtualbox <http://www.forest.impress.co.jp/library/software/virtualbox/>.
- [10] Ubuntu とは <http://linux.pgtop.net/category/7315601-1.html>.
- [11] Ubuntu の特徴 <https://www.ubuntulinux.jp/ubuntu/features>.
- [12] チャレンジャー号爆発事故 wikipedia <https://ja.wikipedia.org/wiki/>
- [13] トルコ航空 dc-10 パリ墜落事故 wikipedia <https://ja.wikipedia.org/wiki/>
- [14] 日本航空 123 便墜落事故 wikipedia <https://ja.wikipedia.org/wiki/>
- [15] 東京航空交通管制システム障害 wikipedia <https://ja.wikipedia.org/wiki/>