VBAの高速化について

常三島技術部門
情報システムグループ　　片岡　由樹 (KATAOKA Yoshiki)

１．はじめに

とある部署で使用するファイルを作成した。ファイルはエクセルでVBAを使うxlsmファイルである。同じフォルダにデータソースとなるCSVファイルを配置し，それを読み込み，必要なデータ処理をして表示するプログラムである。仕様変更もたびたびあったので簡単に説明しながら，使用したVBAコードのポイントをここに記述していく。

可能な限り一般性を持たせた記述で表現して，具体的に記述しないようにしている。どのようなデータかは各自で想像して補完して読み進めていただきたい。

２．構成

当初のデータソースとなるCSVファイルについては以下の要素で構成されている。

- 個人を特定するIDとそれに対応する文字列（氏名）
- 日付データ
- 日付データに関連する時刻データ二つ
- 個人に関連付けされたID（管理ID）とそれに対応する文字列

上の要素が一列にカンマ区切りで記述された文字列(ShiftJIS文字コード)の集合である。CSVファイル名は(個人を特定するID）＋"_YYYYMM.csv"という名称である。ここでYYYYMMは201912など西暦と月から作成される。つまり一つのファイルに個人に関連する一か月分の日付時刻データが格納されている。時刻データ二つから期間が得られる。ある基準からその期間がどのくらい違うのか差分も得る。基準からの差分は時分表示（表示形式：[hh]:mm）と分表示で表示する。

２．１　高速化が必要な理由

後半には仕様変更で約3000人分のデータで,一か月分のCSVファイルに変更になった。前半の段階では一人，もしくは数十人規模で小分け（管理IDごと）にしての使用を想定していたのだが,すべて処理することになった。対象とする個人が約3000人に変更になった為に同じプログラムでは処理時間がかかり，実用にならなかった。その為に高速化するVBAコードに変更する必要があった為，高速化手法について調査した。[1,2]

２．２　バージョン１

バージョン１は７つのシートで構成されている。「readme」，「フォルダ一覧」，「月別」，「個人別」，「詳細」，「list」，「list履歴」の７つである。

- 「readme」シートはエクセル文書の説明用シートである。
- 「フォルダ一覧」シートはエクセル文書と同じフォルダに存在するCSVファイルを一覧表示する。
- 「月別」シートは表示する年月を選択するセルに従って対象のCSVファイルを制限し，月平均や月最大を表示する。
- 「個人別」シートは表示する個人を選択するセルに従って対象のCSVファイルを制限し，月平均や月最大を表示する。
- 「詳細」シートは表示する個人・年月を選択するセルに従って対象のCSVファイルを一つに絞り込み，時刻データ二つから期間データに変換する。ある基準との比較やその差分の累計もある。
- 「list」シートは個人のIDと氏名をリストアップしている。このシートの内容が様々なシートで表示する個人を決める。
- 「list履歴」シートは「list」シートで使用されなくなったデータなど履歴として残せるように用意されている。VBAでは一切参照しない。

２．２．１　「フォルダー覧」シート

CSVファイル名は一定の規則に従った名前の為，対象外は「対象外である」と表示をする。また「list」シートにリストアップしている個人IDとは異なるCSVファイル名のファイルは対象外であるのでリストを確認するかファイルを削除するように注意書きを表示した。

個人IDだけだとだれのデータなのかわかりにくいので「list」シートにリストアップしている個人氏名をVLOOKUP関数で表示させた。シートには二つのボタンを配置した。一つは「データ更新」ボタンである。このボタンに登録している関数はエクセル文書を開いた段階で実行するようにしている。もう一つのボタンはリストにないデータ（CSVファイル）を削除するボタンである。CSVファイル自体を完全に削除するので注意が必要だが，システムの性質上，存在しているはずのないファイルである為，Kill関数を使いファイル削除機能を用意した。

２．２．２　「月別」シート

表示する年月を選択するセルに従って対象のCSVファイルを制限し，月平均や月最大・累積を表示する。存在する個人のIDの数だけ「詳細」シートと同じことをしている。それぞれの演算結果から個人の指定する年月の最大・平均，累計最大などを分表示・時分表示している。

管理IDが管理するデータをlistにリストアップし，CSVファイルを対象の管理データにしておくと管理している個人のリストが月ごとに見ることができるようになる。

２．２．３　「個人別」シート

表示する個人を選択するセルに従って対象のCSVファイルを制限し，月平均や月最大を表示する。存在する年月の数だけ「詳細」シートと同じことをしている。

それぞれの演算結果から指定する個人の年月ごとの最大・平均，累計最大などを分表示・時分表示している。これにより項目の月ごとの推移がわかる。

２．２．４　「詳細」シート

表示する個人・年月を選択するセルに従って対象のCSVファイルを一つに絞り込み，時刻データ二つから様々なデータに変換する。

二つの時刻差から期間が得られる。ある基準からその期間がどのくらい違うのか差分も得て，時分表示と分表示をした。指定の年月における基準からの差異を累計し，時分表示・分表示した。これにより指定月における日ごとの項目推移（期間や時間，累計）がわかる。

２．２．５　「list」シート

個人のIDと氏名をリストアップしている。このシートの内容が様々なシートで表示する個人を決める。データは空行を挟まないで詰めてリストアップする。

２．３　バージョン２

CSVファイルの仕様が変更になり，先頭1行はヘッダとして使うようになった。それ以外でも項目が追加された。主に「詳細」シートに項目が増えた。

CSVファイルはエクセル文書と同じフォルダに存在していることが前提であったが，エクセルからCSVファイルを入手してほしいという要望があった。CSVファイル自体は学内限定サーバーにてシボレス認証により入手できる。これについては検討としてテストを実施した。

２．３．１　エクセルという環境

エクセルでWebからのデータのダウンロードだが，一般的にWebスクレイビングといわれる技術を用いる。Webスクレイビングをする環境としては エクセル(VBA)は難易度が高く推奨される環境ではない。Pythonでする人が多い。スクレイビング自体はHTMLやJavaScriptを一通りできる人なら，Document Object Model (DOM) について理解がある人なら比較的データの獲得は容易であるように感じた。

２．３．２　シボレス認証という壁

サーバー内の入手したいファイル一覧のリンクはシボレス認証でログインする必要があ

る。学内ではCアカウント認証といえばわかりやすいだろう。学外の方には学認サービスが提供している認証方式だといえばわかるだろうか。Webスクレイビングの際にも認証を受けてからでないとデータの場所まで届かない。学内なのでLDAP等で認証サーバーへアクセスする方法も考えられるが徳島大学情報センター等では推奨されない。そこでブラウザを遠隔操作する必要がある。

２．３．３　ブラウザを遠隔操作

ブラウザにも種類がありそれぞれバージョンによって細かな違いがある。Internet Explorer 11ならエクセルから操作はできる。ただし必ず実行できるとは限らない。また，EdgeやChrome， Firefoxで動かそうとするとWebDriverをインストールしなければいけない。さらにスクレイビング用にSeleniumというツールも入れたい。Selenium自体はWebアプリケーションのテストの為に開発されたが，ブラウザを思い通りに操作することができる。これらの環境は一部の技術系の教職員には許容できても，一般の事務に導入するのは抵抗がある。

一応，簡単にテストしてみた。その結果，エクセル(VBA)で Internet Explorer 11 を動かして筆者のデータをダウンロードできた。処理中に動作が不安定で，処理に失敗することもあり，運用するには大幅な改善が必要であると感じた。処理中はファイル保存ダイアログなど自動的に押されたりして，知らない人ならウィルス感染したのではないかと思う状態である。

２．３．４　ブラウザのアドオン

一括ダウンロードをするという観点でブラウザのアドオンを試した。 Chrome にDownThemAllというダウンロード用の拡張機能をつけた。シボレス認証後のリストでDownThemAllを使って拡張子がcsvのものだけダウンロードする。ダウンロードしたファイルは別途エクセル等のフォルダに移動する。これについても以前ならファイル構造も含めて一括ダウンロードする手段があったのだが，そのようなアドオン等が最新のブラウザのバージョンに対応しなくなっている。筆者も以前はあるアドオンがあるという理由でFirefoxをずっと使っていたのだが，最近はいろいろと場面によりブラウザを使い分けている。

２．４　バージョン３

CSVファイルの仕様が大幅に変更になり，個人ごとに作成されていたCSVが作成されず，約3000名分のデータが一つになったファイルになった。ファイル名も「month-o_YYYYMM.csv」となった。それ以外でもバージョン２で増えた項目が減った。

バージョン１，バージョン２において「月別」シート・「個人別」シートは「詳細」シートを使っていた。つまり「詳細」シートに引数にファイル名，個人IDとする関数を用意して，それを呼び出すことによって「詳細」シートの内容を更新し，そのセル内容をそれぞれのシートでデータを作製していた。そしてバージョン３になると個人ID自体が当初の想定の数十人規模を大きく上回る約3000人を対象としている。また，1か月分のすべてのデータが一つのCSVファイルにあるのでファイルサイズも大きく読み取るのも時間がかかると予想した。試しに「list」シートに登録しているリストを3000名に増やして「詳細」シートを実行してみた。時間はかかるがまだ許容できるレベルだった。しかし「月別」シート・「個人別」シートになると時間がかかりすぎた。試しに実行してみた時に1時間くらいかかった時に，高速化する必要を感じた。さらにCSVファイルに約3000名分31日のダミーデータを作成したが，10MBを超えるファイルになり，1日くらいかかりそうなので試すことさえ出来なかった。そこでVBAのコードを見直し，高速な処理になるように編集をした。その内容は後述するように高速化手法として調査してまとめた。

最終的に，約3000名分31日のダミーデータで「月別」シートですべての個人について項目の最大や平均をまとめたデータを表示するのに，私のパソコンで4，5分程度になった。

シート構成もバージョン１に比較して以下の項目が増えた。

- 「temp」シートはCSVファイルをテーブルクエリで読み込む作業用シートで非表示である。
- 「候補」シートは「個人別」シートや「詳細」シートで個人を限定する為に絞り込むフィルタを作った。その動作用のシートで非表示である。

２．４．１　絞り込むフィルタについて

シートで個人IDを決める為に氏名で選択肢から選ぶ形にしている。3000候補から一つ選ぶのは手間がかかる。そこで，その選択肢の候補を作成する為に「候補」シートを作っている。絞り込む為のセルの文字列からIDの上位桁から絞り込む。例えば0と入れれば初めに1のものは候補から外れる。選択肢数も多くても困るので最大で50にした。使用者としては候補が出てこないならさらに01という風に範囲を絞り込めばよい。

２．５　バージョン４から６

バージョン４ではCSVファイルの仕様が変更になり，項目が削除された。学内限定サーバーにてシボレス認証により入手できるCSVファイルがなくなった。

バージョン５ではCSVファイルの仕様が変更になり，項目が追加され，バージョン４で削除された項目をエクセル文書で補完するように変更された。シート構成もバージョン４に比較して項目を補完する為のシートを二つ追加した。追加したシートからVLOOKUPにてデータを補完した。

バージョン６では集計表示に関して若干の変更があった。

３．高速化手法について

エクセルのVBA高速化で調査をしていくと，かなり古くから様々な方が速度比較してノウハウを報告している。そして古い報告は実はあまり役に立たないこともわかった。それはエクセルのバージョンが上がるごとに様々な処理が改善されているからだ。またパソコンの処理速度も向上しているのもあるだろう。私のパソコン環境はWindows10で稼働するノートPCでExcel 2016が入っている。購入からは執筆時点で4年経過している。4年といえば故障が多くなり買い替え時といわれている。現状で動作に不満がないのは機種選定の判断を褒めたいと勝手に思っている。とにかく，この高速化手法も執筆時点での高速化手法であることに留意しながら，読者の方は考え方やコツを吸収していただきたいと筆者は考える。

３．１　更新しない

３．１．１　画面を更新しない

セルの内容変更，書式設定，シート表示切り替えなどの処理があると，画面が頻繁に更新される。この更新処理が遅くなる大きな原因となる。ちょっと遅い程度のプログラムなら他の高速化手法を検討するより前に画面更新制御のコードを入れると，ほぼ解決する。画面を更新しないので長い演算中はエクセルがフリーズしたような印象で，応答なしの状態になる。コードを表１に示す。

表１　画面を更新しない

```
Application.ScreenUpdating = False
' ここに処理
Application.ScreenUpdating = True
```

３．１．２　ステータスバーを更新しない

画面更新制御と同じようなことでステータスバーの表示を中止する。画面を更新しないが処理の進捗状況をステータスバーに「進捗状況50%」などと表示するとユーザーには優しいプログラムになる。間欠的に表示する方法もあるが,割り切って表示しない方が速い。コードを表２に示す。

表２　ステータスバーを更新しない

```
Application.DisplayStatusBar = False
' ここに処理
Application.StatusBar = "処理中..."
' ここに処理
Application.DisplayStatusBar = True
```

３．１．３　シートの再計算

シートに数式がある場合はセルが変更される度に再計算が行われる。この再計算を自動

から手動に切り替える。途中で再計算が必要な場合には再計算するようにCalculate関数を実行する。計算範囲もRange等で限定した方が処理は少なくなる。

もう一つの方法としてWorksheetオブジェクトのEnableCalculationプロパティを使う方法がある。手動等の指定と関係なく再計算ができなくなる。コードを表３・４に示す。

表３　シートの再計算

```
Application.Calculation = xlCalculationManual
' ここに処理
Application.Calculation = xlCalculationAutomatic
```

表４　シートの再計算（禁止）

```
ActiveSheet.EnableCalculation = False
' ここに処理
ActiveSheet.Calculate'再計算されない
ActiveSheet.EnableCalculation = True
```

３．１．４　イベント発生

プロジェクト内にイベントが発生する場合はイベントの発生自体をOFFにしてイベント発生を停止した方が高速化できる場合がある。コードを表５に示す。

表５　イベント発生

```
Application.EnableEvents = False
' ここに処理
Application.EnableEvents = True
```

３．２　ファイルの読み込み

一番遅くなる方法はおそらくファイルポインタを使って開いたファイルに対して一行ずつ読み取りカンマ区切りも自前で処理する方法だろうと予想される。テキストファイルなどならそのような処理がVBA入門には掲載されているだろう。

今回はテキストファイルではなくCSVファイルである。CSVファイル自体もエクセルで開くことができる。関連付けられて，デフォルトで「エクセルで開く」設定の方も多いのではないだろうか。バージョン１の段階からワークブックを開くようにしてCSVファイルを取り扱った。コードを表６に示す。

表６　ファイルオープン

```
Dim mFilename As String
Dim wb As Workbook
Set wb = Workbooks.Open(Filename:=mFilename,
            ReadOnly:=True)
```

ファイルを開く方法は新しくワークシート，つまりエクセルのウィンドウを開くことになる。「個人別」シートなどでは存在する年月の数だけエクセルのウィンドウを開くことになる。開く準備が水面下でされる為に時間がかかってしまう。そこで，データベース接続に使用されるクエリーテーブルを使うことにした。コードを表7に示す。作業用の非表示のシートを用意して，そこにデータを入れることにした。

表７　クエリーテーブル

```
Dim qt As QueryTable
Set Sh = Sheets("temp")
Sh.Cells.Clear
Set qt = Sh.QueryTables.Add(
      Connection:="TEXT;" & mFilename,
      Destination:=Sh.Range("A1"))
With qt
   .TextFilePlatform = 932   ' 文字コード(Shift_JIS)
   .TextFileParseType = xlDelimited ' 区切り文字
   .TextFileCommaDelimiter = True ' カンマ区切り
   .RefreshStyle = xlOverwriteCells ' セルに上書き
   .Refresh   ' データを表示
   .Delete    ' CSV との接続を解除
End With
```

ただ，一行ずつ読み取る方法は実は遅くないらしい。最近のPC性能なら問題ないようだ。参考文献２の「Line Inputは遅くない」に書かれているように遅いのはセルへの書き込みが原因だ。

３．３　関数を使う

３．３．１　VBA関数を使う

エクセルVBAにはいろいろな関数がある。それらの関数はかなり作りこまれていて処理

は非常に高速だ。だからVBAからそれらの関数を使う。あえてアルゴリズムの練習をする必要はない。

３．３．２　ワークシート関数を呼び出す

エクセルには関数がある。それらの関数はかなり作りこまれていて処理は非常に高速だ。だからVBAからそれらの関数を使う。技術レベル，経験値は上がるかもしれないが，車輪の再発明をしてもだれも褒めてくれない。「月別」シート・「個人別」シートにおいても最大・平均，累計最大を求めるのにMAXやSUMを使った。

３．４　配列を使う

大量のデータを扱う場合一つ一つ順番に処理するより一度配列にデータを入れて配列を操作する方が速い。最近のパソコン処理速度ならあまり配列で処理するからということが効いているわけではないようだ。

配列に入れるのが実は速くなる原因ではなく，セルへのアクセス（代入）がネックになっている。配列を引数にして関数内で処理した返り値は配列なのでセルへの代入が効率よくなるからだ。従って，平均や合計を求める際に配列に対して値を格納して配列を引数にしてワークシート関数を呼び出した。表示しない計算結果や一時的に使う作業用セルを使わないようにするとよい。

３．５　フィルタを使う

大量のデータを走査していくのは時間がかかる。だからフィルタを使って表示するデータを変更した。その際にはSUBTOTAL関数で非表示にした行は含めないで集計する場合の引数の指定方法が便利だった。表示されているデータを対象に繰り返し処理をした。表示されているデータを選択することが大切である。筆者は不慣れなのでチャレンジしなかったがピボットテーブルが使えるかもしれない。ピボットテーブルも最近は使い勝手がよくなってきているので，是非ともチェックしておきたい。

３．６　シートをクリアする

クエリーテーブルで使用する時にデータのある領域だけデータを削除しようとしたがシートごとクリアした方が速い場合がある。

３．７　その他

３．７．１　無駄なSelectをやめる

エクセルで操作するように操作対象を選択する,操作をするという2段階の使い方をやめて直接操作対象を指定して操作する。

VBA入門のレベルでコードを理解する際には操作対象をしっかりと把握しながら記述していく必要があるのでチュートリアルやサンプル例ではSelectを使うようになっている。入門レベルを突破したならSelectをやめるようにしていくことが大事だ。

３．７．２　繰り返し

For Next と For Eachを比較して昔はFor Eachの方が速かったらしい。今はほとんど変わらないので考え方や使い方がしっくりくる方で記述すればよい。For Nextはカウンタで回していく形でC言語など得意な方はこちらの方が慣れているかもしれない。For Eachの場合はオブジェクト型またはバリアント型で対象のオブジェクトで回していく形である。オブジェクトの関数やプロパティを使うならFor Eachの方が私は使いやすいと感じる。

３．７．３　Withステートメントでまとめる

実行するコマンドが少ない方が高速になるということで, Withステートメントで対象(オブジェクト)をくくる方法がある。あまり速度は変わらない。それよりもオブジェクト変数を作りセットすると若干速くなるようだ。コードを表８に示す。

表８　オブジェクト変数を使う

```
Dim S As Worksheet
Set S = Workbooks("Sample.xlsx").Sheets("Sheet1")
S.Cells(1, 1) = "このように使う"
```

３．７．４　バリアント型にしない

変数の型を指定しない場合はバリアント型になる。変数自体をデバッグ時に見てみると

わかるようにいろいろなプロパティが付いている。LONG型で事足りるならLONGにした方がよい。ただし，今のPC性能ならあまり気にしなくてよい。また変数を何の為に使っているのかしっかり把握しているなら必要最小限な型は選択できるはずだと思う。

３．７．５　シート名を使わない

シートを特定する際にSheets(1)を使い，Sheets("Sheet1")と使わないことで少しだけ以前は速かった。インデックス値でFor Next で処理するなどの使い方以外ならシート名の方がわかりやすい。昔ほど速度差は気にする必要ないレベルなので私はシート名を使う。

３．７．６　セルの選択にはCells

Range("A1")などのように一つのセルを指定するならCells(1,1)の方が速い。A1がどの行，どの列なのか変換している時間が節約できる。繰り返し使うコード領域ならば是非Cellsにしておく。一度くらいしかCellsを使わない箇所については，コードの可読性の為にRangeを使った方がわかりやすいてよい。

３．７．７　Valueプロパティを書かない

セルの値は標準としてValueプロパティで表される。値の代入する際，される際にValueを記述すると処理が理解しやすく記述できる。しかし，Valueプロパティは省略できる。若干省略した方が速い。可読性が悪くならない程度に省略しよう。コードを表９に示す。

表９　Valueプロパティ

```
Range("A1").Value = "Kataoka"
Range("A1") = "Yoshiki"
```

３．７．８　$がない関数を使う

VBA関数の中にはLeft$関数のように$が付くものがある。$ありの関数は「文字列を返す」という意味だ。String型には$，Integer型には%，Long型には&，Single型には!，Double型には#，Currency型には@とバリエーションもある。これは過去の時代の産物で$ありを使うとNull判定処理が必要になるなどバグを含む可能性が高い。以前は$ありの関数の方が若干速かったらしいが，最近のパソコン処理速度ならほどんど関係ない。

３．７．９　余計な処理をしない

今回は日時のデータを扱った。エクセルではそれらはシリアル値として格納されている。だからいちいち時刻データ二つの差分を計算するのは引き算で充分だった。年月日と分けて時分秒の処理等と複雑な演算をする必要はなかった。私はバージョン３の段階で無駄な処理を省く際に気づいたがバージョン１の時点で気づきたかった。

４．最後に

テスト用のダミーファイルが93000行まであったが，ひと昔前のエクセルでは処理できない行数である。そのデータがアクセスではなくエクセルで扱えることにちょっと感動した。普段高速化する必要がないデータ量でしか活用することはなかったが，高速化を考えたコーディングができるようになると仕事効率も上がるのではないかと期待する。読者にとってヒントとなる報告になっていると幸いだ。

参考文献

[1] Excel VBA 高速化アプローチ
http://dev-clips.com/clip/vba/improve-performance-property/

[2] Office TANAKA VBA高速化テクニック
http://officetanaka.net/excel/vba/speed/