Table::Hack のインストールと最初の使い方

連絡先: bin4tsv@gmail.com

2018 年 6 月 24 日 (日)

この文書は、CPAN にアップロードされたモジュール Table::Hack について、インストール方法と最初の使い方を解説する。

実際の利用場面において、よく読むべき箇所は数ページ程度であるで、残りは本書の太字の部分をさっと目で追うだけになるかもしれない。

本文書の全体は約 25 ページで、考えられるあらゆる状況に対する基本的な対処法、及び、モジュール作成者の意図の解説が記載されている。

目次

第 I 部
Table::Hack のインストール

1 前提となること (インストール前に)

1.1 本文書の作成方針

本文書は、「全く知識が無くてもすぐ使えるマニュアル」にはなっていない。**「初心者であってもキーワードを元にインターネットで調べたり試行錯誤を何度か繰り返せば意味の分かる手順書」**かつ**「いくらか知識のある人がさっと読んで要領の理解しやすい手順書」**になっていることを意図して書かれている。

1.2　想定する計算機環境および利用者のスキル

Perl 5.14 以降[*1]が使える環境であればどんな環境でも使えるように、本モジュール Table::Hack は作られている。すなわち、大学、政府および国際機関、インフラ系企業、金融機関などで、システム更新が近年なされていない場合でも、計算機環境が先進的でもそうでなくても、個人の環境でも、問題なく動くように試みられている。そのため、Perl 5 の知識があれば、意図しない動作をしない場合に迅速に対処できる程度に、プログラムは簡素に作られている。[*2]

本モジュールの利用に際し、利用者が **Perl 言語のプログラミングをする能力は必要とはしない**。(それでも何か不具合があった場合に、意図した通りの動作をさせるためには、Perl 言語の知識は役に立つ。) むしろ Unix 環境もしくは Linux 環境で bash や zsh などのシェルについて基本的な使い方をマスターしていることは、利用のしやすさに大きく影響する。リダイレクションでファイルの読み書きが出来ること、様々な基本的な Unix コマンドを使いこなしていること、パイプで複数のコマンドを連結して使うこと、さらにはプロセス置換で余計な中間ファイルを作らずにいろいろな操作が出来ることは、本モジュールを使いこなす上ではやや大事である (A.1 節を参照)。

Unix でも Linux でも Windows でも、Mac OS X でも使えるが、本文書作成者は、Mac OS X 上の Unix 環境及びクラウドサービスの AWS (Amazon Web Service) の EC2 の Linux 端末で検証を行っている。bash と zsh のシェルで検証をしているが、それ以外の csh などのシェルを使う場合は、読者が下記の文書の意図を読み取って独自調査をしてインストール作業とコマンドの利用をする必要がある。

2　インストールの方法

この 2 節では、Table::Hack のインストール法及び関連するソフトウェアのインストール法を記載している。下記のインストールの方法のどれかが完了したら、次の 4 節の「各コマンド使い始める」を参照のこと。

> この節は文の量が多いが、cpan コマンドを知っている人は、ここから先は何も読まなくても、画面の指示を追うだけでおそらく 3 分〜15 分以内にインストールは完了するであろう。(Perl の cpan は、Ruby の gem、Python の pip に相当する。)
>
> この節はあらゆる状況に対応して、さまざまなインストール方法の選択肢を示している。OECD 非加盟国の政府で統計業務に使えることを想定したためである。

[*1] Perl のバージョンの確認方法は `perl -v` を実行することである。5.14 は 2011 年に公開。2018 年の最新版は 5.26 である。

[*2] Perl 5.14 でも新しすぎる場合は、手でプログラムをたとえば 5.1(Perl5.1 は 1994 年公開) と書き換えて、エラーを起こす部分の機能を制限または修正するなどの方法が考えられる。モジュールが提供するファイルで Perl 言語で書かれているプログラムは、何かエラーがある場合は、基本的には、そのエラーを起こしたファイルのみを手を加えれば修正が可能である (エラーを起こしているプログラム以外のファイルを修正する必要は、本モジュール Table::Hack の場合、原則的に無いはずである)。そして、最初の方に `use 5.014` とあるので、そこでエラーが起こる場合はそこから直していくことになる。

2.1　インストール方法の選択

本モジュールのインストールの方法は、以下のどれか 1 つである。

1. cpan を使う方法
2. cpanm を使う方法
3. Build による方法
4. ディレクトリ scripts 下のプログラムにパスを通す方法
5. MetaCPAN でモジュールのページから Browse を辿ってファイル一覧からコピーする方法

上記 5 通りのインストールの方法を決めるには、表 1 による比較が参考になるであろう。

本モジュール `Table::Hack` をインストールする方法 5 個の比較

本 module の install 法→	1. cpan コマンドの方法	2. cpanm コマンドの方法	3. Build による方法
作業前に必要な環境:	cpan install 済みの環境	cpanm install 済みの環境	`Module::Build` 事前 install
その環境の準備の手間:	cpan の初期設定に数分間	cpanm の install	おそらく cpan や cpanm に頼る
管理者 (root) 権限の必要性:	必要がある場合が多い	install 先 directory による	install 先 directory による
本 module の install の手間:	`cpan` module 名 で一発 `:-)`	`cpanm` module 名 で一発 `:-)`	**download, 解凍, コマンド数回**
ありがちな追加の手間:	cpan 設定の理解と右の (#)	cpanm の install 法の選択と (#)	`Module::Build` の install(#)
install 先 directory の指定法	cpan 起動して o conf など	`cpanm -l` foo	`Build install` **`--install_base`** foo
アンインストールの方法:	難度高 (別環境で変更箇所を特定)	`cpanm -U Table::Hack`	書換/書き加えのファイルの追跡
計算機環境の破損の危険:	実例を聞いたことは無いが、root 権限で目で追いきれない操作があるのでありうる。		
既存のコマンド上書きの危険:	あり得る	あり得る	あり得る
既存コマンドと名前衝突:	あり得る	あり得る	あり得る

本 module の install 法→	4. 手作業展開とパス設定	5. MetaCPAN から逐次ファイルコピー
作業前に必要な環境:	Perl のみ `:-)`	左に加えウェブブラウザ
その環境の準備の手間:	皆無 `:-)`	
管理者 (root) 権限の必要性:	不要 `:-)`	不要 `:-)`
本 module の install の手間:	3. の方法に加え **PATH 設定作業**	ブラウザからコピーしてすぐ利用可能
ありがちな追加の手間:	**コマンド格納 directory の検討**	ファイル複数が必要なら反復作業が必要。
install 先 directory の指定法	手作業; `mkdir` など	左と同様だが GUI 操作であろう
アンインストールの方法:	PATH 設定の書き加えの除去など	ファイルの消去
計算機環境の破損の危険:	install 作業時にまず起きない	左と同様
既存のコマンド上書きの危険:	手作業で確認するので起きにくい	左と同様
既存コマンドと名前衝突:	手で名前を書き換えて回避可能	左と同様

表 1　この表の情報は確実とは限らない。誇張が混入している可能性もありうる。文字幅を減らすため、インストール、ディレクトリは英単語で表記した。CPAN モジュールインストールの方法として「標準的」な方法は上記 5 通りの方法の内左側の 3 個であるが、環境設定をしたことがない人が試しに使うには、一見手間がかかっても、最も右の方法が結局は容易で確実であるように思われる。

cpan がインストールされていない場合それをインストールするには通常 root 権限 (管理者権限) を必要とするし、そのインストールには数分間画面を注視するような 作業を要する。近年、**cpanm** の利用も多いと思われるが、それが未インストールの場合は cpanm のインストール作業が数分間必要である。`Module::Build` の **Build** を使う方法と**パス**を通す方法は、利用者が MetaCPAN のサイトから本モジュールをダウンロードして、`tar zxvf` などのコマンドで解凍する作業を伴う。どれを採用するのが最適であるかについては、利用者の計算機環境での権限、本モジュールを迅速にインストールしたいかどうか、計算機環境の変更を最小限にしたいかどうか、本モジュールを試しに使うかもしくは恒久的に使うか、利用者の外部ライブラリの管理ポリシーなどに依存する。

インストール直後に記録/記憶すべきこと:

下記の情報は記録し、後日必要な場合にすぐ利用可能な形で保存をすることが望ましい。

- どんな方法でインストールをしたか
- どのディレクトリにプログラムをインストールしたか (手動設定をした場合など)
- インストール時に見つけた不具合
- 誤操作をしそうに思われた箇所

→ `Table::Hack` の更新/アンインストール/他の関連プログラムのインストールの際に必要となる。

コマンド操作が正常終了か異常終了か知る方法:

直前に実行したコマンドが正常に終了したのかエラーであったのかを調べるには、下記の方法がある。

- エラーを示す文言を目でよく探す。異常ではない警告 (Warning) との区別に気をつける必要がある。
- bash, csh, zsh の場合は `echo $?` を実行。0 ならば正常終了。それ以外の値は異常終了。

→ インストールの際の各段階で異常終了が発生して解決が出来なければ、別の方法を選択することになる。

コマンド操作の時間を測定する方法:

- 実行するコマンドの直前に `time` の文字列を置く。例、`time cpan Table::Hack`。
- 実行時間を知りたいのに既に実行してしまった場合は、即座に `Ctrl-z` 押下 (コントールキーを押したまま z キーも押す) によりジョブ中断を行い、`date; fg`[*3]`; date` のコマンドの実行により、再開時の時刻と終了時の時刻を表示させるようにする。既に経過した秒数と加算することで実行時間を推定することが出来る。

→ インストールにかかった時間の記録は、意外と重要な情報であろう。

2.2 インストールの手順 (5 通りから選ぶ)

この節で下記は 主に Unix/Linux 環境を仮定する。しかし他の OS であっても十分に参考になるであろう。

2.2.1 方法 1: cpan でインストールする場合

1. **cpan 自体をインストールする必要がある場合:**

   cpan コマンドがインストール済みであれば、ここに記載の作業は不要なので、次へ。cpan コマ

[*3] `fg` はストップしたジョブを実行 (再開) させるコマンドである。フォアグラウンドの略である。

ンドのインストール方法は OS ごと、Linux であればディストリビューションごとに異なるので注意。`yum install cpan` または `yum install perl-CPAN` などで `cpan` をインストール (root 権限が必要な場合は、このコマンドの先頭に `sudo` を付加する)。

2. **cpan を初めて起動する場合:** (数分間以上かかる。画面に大量のメッセージが流れる。)
   `cpan` を実行して、質問に答えながら進める (Enter キーだけを打鍵して進めても大抵は問題無い)。
   1. 最初の質問で「出来るだけ自動化するか」と聞かれるので、肯定すると良い。
   2. 次の質問で管理者権限を使うか、手動にするか、local::lib を使うか聞かれる。
   3. その次におそらく、インターネット接続のことを聞かれる。
   4. `/root/.bashrc` などの設定ファイルへの環境変数の設定の書込みの許可を求められる。
3. `Module::Build` が未インストールの場合は、それをインストール。2.3 節を参照。
   これが済んでいないと、次で少し意味の分かりにくいエラーが表示されるかも知れない。
4. `cpan Table::Hack` で完了する。もしくは `sudo cpan Table::Hack` で完了する。
5. インストール完了確認は `saikoro` コマンドの実行と 1 から 6 の値の乱数生成の確認である。

2.2.2 方法 2: cpanm でインストールする場合

1. **cpanm コマンド自体をインストールする必要がある場合:**
   - cpan が利用可能な場合には `cpan App::cpanminus` を実行する。
   - Mac の場合は brew がインストールされていれば `brew install cpanminus`。
   - Linux はディストリビューションごとに推奨方法が異なるので、それぞれ調べること。
     `yum` コマンドを使う場合は `time sudo yum -y install perl-App-cpanminus` などを実行。
2. `Module::Build` が未インストールの場合は、それをインストール。2.3 節を参照。
3. root 権限を使わずに `Table::Hack` をインストールするには、`local::lib` をインストールすると良い。
   - まず `cpanm --local-lib=~/perl5 local::lib` を実行。
   - 次に `eval $(perl -I ~/perl5/lib/perl5/ -Mlocal::lib)` を実行。
   - これで、ログアウトするまでローカルにモジュールがインストールされるようになる。
   - その状態で次の項目を実行。
4. `cpanm Table::Hack` で完了。もしくは `cpanm -S Table::Hack` (`sudo cpanm Table::Hack` と同等)。
5. インストール完了確認は `saikoro` コマンドの実行と 1 から 6 の値の乱数生成の確認である。

2.2.3 方法 3: Module::Build の Build でインストールする場合

1. `Module::Build` が未インストールの場合は、それをインストール。2.3 節を参照。
2. MetaCPAN のサイトから `Table::Hack` を検索する。
3. Download と書かれた項目をブラウザ上で探して、**ダウンロード**。
   - Download をクリックして、ダウンロードすると、ブラウザのダウンロード用のディレクトリに格納されるであろう。
   - Download を右クリックして「リンク先のアドレスをコピー」して、そのアドレスに対して `wget` を行っても良い。
     コピーしたものをコマンドラインで `wget` の後ろにペーストすると次のようになるであろう。
     ```
     wget https://cpan.metacpan.org/authors/id/T/TU/TULAMILI/Table-Hack-0.11.tar.gz
     ```

4. **解凍する。** `tar zxvf` some`.tar.gz` を実行することで解凍は可能[*4]。
5. 解凍先のディレクトリに入って (cd コマンドを使う)、下記を順に実行。
   (1) `perl Build.PL`　# `Module::Build` が未インストールの場合はエラーとなる。
   (2) `./Build`
   (3) `./Build test`
   (4) `./Build install` もしくは `./Build install --install_base` foo (foo は `~/perl5` など)
6. foo/`bin` にパスが通っていることを確認。もしくはパスを通す (通し続ける) 設定をする。
7. インストール完了確認は `saikoro` コマンドの実行と 1 から 6 の値の乱数生成の確認である。

2.2.4　方法 4: ディレクトリ scripts 下のプログラムに手動でパスを通す場合

"手作業" で本モジュールをインストールする方法: **(全行程の概要)**

1. MetaCPAN のサイトで `Table::Hack` を検索して**ダウンロード** (ブラウザで Download を探してクリック)。
2. **解凍する。** `tar zxvf` some`.tar.gz` により解凍は可能。
3. 解凍先のディレクトリに入って (`cd` コマンド)、`scripts` 直下のファイルを確認 (`ls` コマンド) :
   - これからコマンドとして使うファイル名なので、後の必要に応じてすぐ把握可能とするため。
   - (必要に応じ) `init.sh` 以外の各ファイルのパーミッションが実行可能に設定されていることを確認。[*5]
   - 他の実行可能ファイルと名前が**衝突**することがないかどうかを後で必要に応じ確認可能とするため。[*6]
4. `scripts` 直下の全ファイルの**格納先のディレクトリ**を決める (例: `~/bin` や `/home/you/bin4tsv`)。
   - そのコピー先のディレクトリがまだ無い場合は、 `mkdir` コマンドで作成。
   - そのコピー先に既に他のファイルが存在する場合は、`scripts` 下のファイルと名前が衝突しないかを確認する。衝突する場合は、`scripts` 直下のファイル名を適宜変更する (`mv -i` コマンドを使う)。
5. **パス PATH の設定:** (格納されたディレクトリにあるファイルが今後必要なときにすぐ実行出来るようにする。)
   - ここで前記で決めた格納先のディレクトリを以下、 somewhere とする。
   - Windows の場合は、**システム環境変数** の Path の値を編集し、somewhere を書き足す。[*7]
   - Unix/Linux の OS の場合: 使っているシェルに応じて、Bash ならば `~/.bash_profile` に、Zsh ならば `~/.zshrc` に、次の行を (ログイン時に毎回実行されるように) 書き加える : [*8]

     `source` somewhere`/init.sh`　# ← 直接実行ではなく、あるファイルに挿入。
     # init.sh はそれを格納するディレクトリを変数 `PATH` に追加する働きがある。
   - 一時的に PATH を設定する場合: Bash または Zsh の場合、次を実行。[*9]: `PATH=`somewhere`:$PATH` もしくは `init.sh` を `source` コマンドにより実行する (`source init.sh` など)。

[*4] `tar` コマンドの使い方は環境により異なることがある。A.1 節を参照してヘルプを読むと良い。

格納先のディレクトリの決め方に関して

- ~/bin4tsv と決めてしまう方法がある。(~ はホームディレクトリを意味する。)
- `echo $PATH | tr : "\n"` で検討する方法もある。自分で以前独自に作ったディレクトリがあればそこに追加することもあり。
- `man hier` のコマンドを実行して、Unix の**ファイルシステムレイアウト**を知っておくと、上記のコマンドの実行結果で現れたディレクトリ名の意味が分かるであろう。なお、Build の方法で root 権限でインストールすると、本モジュールの提供するプログラムファイルは /usr/local/bin に配置される。
- パスに書かれたディレクトリはその中のファイル数を考えると参考になるかもしれない。表 2 を参照。
- プログラムの**優先順位**の問題を考えるには、下記を実行。

  `which -a $(ls)` または `which -a ‘ls‘ #` 解凍したファイルの scripts ディレクトリ下で行う。

  — このコマンドは、次の上書きを防ぐ問題の簡易な検査になっている。
- 既存のディレクトリに書き込む場合に**上書き**を防ぐためには、 下記のコマンドを解凍したファイル直下でを実行して、scripts 下にあるファイルと、パスが既に通っているディレクトリにあるファイルと名前が一致しないか、確認する。

  `grep -f <( ls scripts ) <(find -L $(echo $PATH | tr : ”\n” ) )`

  上記をもっと厳格に行う (部分一致では無く完全一致のみを出力する様にする) には、

  `grep -f <( ls scripts | perl -pe ’chomp;$_=”^$_\$\n”’ ) <(find -L $(echo $PATH | tr : ”\n” ) )`

PATH 上の優先順位に関して

- 本モジュール `Table::Hack` をインストールする目的は、これが提供するコマンドが動くようにすることであるから、そのコマンド (プログラムファイル) を格納したディレクトリ somewhere は既存のパスに記載されたディレクトリよりも優先順位を低くする理由は無い。従って、$PATH=$PATH:somewhere とするより$PATH=somewhere:$PATH とするのが妥当のように思われるので、init.sh ではそのようにしてある。
- しかし、$PATH の中の順序がきちんと管理されているように見えることは、管理上重要なことでもありうるので、実際の動作上に差し支えなければ、$PATH=$PATH:somewhere と書き換えても良い。

---

*3 ls -l コマンドの出力において、空白区切りの 1 列目で r,w,x のなどの文字の配列を確認する。r は読み取り可能、w は書込可能、x は実行可能を意味する。rwxrwxrwx や r-xr-xr-x のような文字列において、これら 9 文字は 3 文字ごとに左から、ユーザーの権限、グループの権限、その他の人の権限を意味する。この 9 文字の前後に、ディレクトリであるか否か、リンクファイルであるか否かを表す文字が付加される場合がある。

*4 `which -a` プログラム名 で検証可能。後の作業で既存ファイルへの上書き回避と、パス上の実行優先順位の検討が可能。

*5 セミコロン (;) で somewhere を後ろに追加する。システム環境変数の設定ができるようにするには、Windows 7 の場合は、「**システムのプロパティ**」を起動して「**詳細設定**」のメニューを選び「**環境変数**」のボタンを押す。(なお、システム環境変数の設定の方法は、インターネットで検索すれば容易に発見できる。)

*6 使っているシェルを確かめる場合は `echo $0` (ドル・ゼロ) を実行する。`-bash` と表示されたら bash である。

*7 = の前後に空白文字を含めてはいけない。

PATH 上のディレクトリが持つファイル数の数え方

453	/usr/local/bin
979	/usr/bin
36	/bin
248	/usr/sbin
61	/sbin
432	/Library/TeX/texbin
126	/opt/X11/bin

表 2　for i in ‘echo $PATH | tr : ”\n”‘; do echo ‘ls $i | wc -l; echo $i‘ ; done
— 上記の数は、ある環境での例である。

2.2.5　方法 5: MetaCPAN のサイトから直接ファイルをコピーしてそのまま使う場合

1. ウェブブラウザ (Chrome, Safari, Internet Explorer など) で `metacpan.org` のサイトを開く。
2. `Table::Hack` を検索する。
3. ウェブブラウザ画面の左上の方の ”Browse”をクリックする。
4. ファイルの一覧が現れるが、その中のフォルダーの 1 つである scripts をクリックする。
5. `Table::Hack` が提供するプログラムのファイルが現れるので、必要なものをクリックして、ファイルの中身を表示する。
6. そのファイルの文面について `#!/usr/bin/perl` から最後までをコピーする。
   ファイルの中身が一部折りたたまれてる場合がある。その場合は Show .. などと書かれているのでそれをクリックする。
7. クリップボードにコピーした内容を適当なファイルに書き込み、同じ名前、もしくは都合の良い名前でファイル名を付与する。
8. 必要に応じて PATH を通す。PATH を通さない場合は、そのプログラムファイルはパス名も今後使用する必要がある。
9. 使うプログラムが `saikoro` であれば、そのファイルが利用中のディレクトリにある場合は `./saikoro` などと指定して使うことになる。

2.3　別の CPAN モジュールのインストール

上記のインストール方法のいくつかは、`Module::Build` のインストールが必要である。また本モジュール `Table::Hack` 提供のコマンドを実行すると依存する CPAN モジュールが無い場合はエラーとなるので、そのモジュールのインストールが必要となる[*10]。下記を参照。

[*10] `List::MoreUtils` と `Text::CSV_XS` と `PerlIO::gzip` などのインストールが必要となりうる。また、コアモジュールであっても Linux ディストリビューションによっては削除している場合があるので、追加でインストールする対象になりうる。なお、コアモジュールかどうかを知るには、`Module::CoreList` をインストールして、`corelist List::Util List::MoreUtils` などのコマンドで確かめることができる。

- Module::Build 以外の CPAN モジュールをインストールする際は、下記の Module::Build の文字列の部分をそのモジュール名に置きかえて実行すること。
- perldoc Module::Build でヘルプが表示されたり、perl -MModule::Build -e1 が正常に実行できたりする場合は、Module::Build は既にインストール済み。
- cpan Module::Build もしくは cpanm Module::Build でインストールが出来る。この作業は数分間以上かかる。画面に大量のメッセージが流れる。
- Linux のディストリビューションによっては、yum や apt-get などによる他の方法によるインストール方法があるかもしれない。

2.4 Table::Hack のアンインストールの方法

- アンインストールは次に述べるように、やや難しい。従って、最初から仮想環境を使うか、もしくは最初からユーザーのディレクトリ内にインストールして、アンインストールしたい時に丸ごと (仮想環境またはディレクトリを) 消去するのが望ましい。
- cpan (方法 1.) もしくは Build(方法 3.) の方法を使った場合には、Table::Hack のアンインストールはおそらく難しい。最初に Table::Hack をインストールした時に書き換わったファイルを注意深く追跡して消去する、もしくは似た様なもう一つの環境で Table::Hack をインストールしてどのファイルが書き換わったか追跡するという方法が考えられる。
- cpanm を使った場合 (方法 2.) は cpanm -U Table::Hack で一発のコマンドで終わるが、cpanm --help に書いてある通りこの機能は実験的なものである。Table::Hack が提供する たとえば saikoro のプログラムファイルは、本モジュール作成者が別のモジュールでも提供しているので、そのモジュールをインストールしている場合、saikoro を消したくなくても消去してしまうであろう。
- 手作業による方法 4. または方法 5. による方法であれば、インストールした人がどのファイルを操作したか思い出せるであろうから、それらを全て元通りにすれば良い。

3 インストール上の補足事項

3.1 注意点となりうること (Perl に多少の経験がある人が気にするようなこと)

- **本モジュールが CPAN にある通常のモジュールと異なる点:** 本モジュールは、単数または複数の Perl 言語で書かれたプログラムファイルが単独で働くように作られている。各プログラムの中に、オンラインマニュアルヘルプも含まれているが、それらの文書は必ずしも POD 形式に従って書かれている訳ではない。通常のモジュールと異なり、ライブラリとなることを意図して作られていないので、本モジュールに含まれる *.pm ファイルを必要としているプログラムファイルは存在しない。
- **利用者の計算機環境に影響を与えないか:**
  1. **インストール時に既存の実行可能なプログラムとのファイル名の衝突** — 本モジュールのプログラムファイルの名前と別のプログラムのファイルが衝突[*11]した場合に、実行時にどちらが実行され

[*11] ファイル名の衝突とは「一致」と同じ意味であるが、この一致は望ましいものではないので、あえて衝突と表記する。なお、ファイル名とは、パス名と混同されがちだがここでは異なる。パス名とは、ディレクトリ名が階層的にファイル名の前に/を区切り文字にして連結した文字列である。

るかの問題、および、インストール時に上書きされる問題がある。cpan または cpanm の設定などに依存するが、/usr/local/bin/ に同名のファイルがあっても上書きがされないように、インストールするディレクトリを別の場所 (ローカルディレクトリなど) に変更することはできる。[*12] 本モジュールのインストール方法に記載した cpan, cpanm, Module::Build を使わない方法でインストールする場合には、パスの設定 (環境変数$PATH への書き込み) が通常は (特にその計算機に再びログインしてから本モジュールをすぐ使える状態にする場合は) 必要であるが、その場合でもそのパスに記載された複数のディレクトリの優先順位を検討する必要が発生しうる。[*13]

2. **実行時の汚染チェック:** 一般に、Perl インタプリタの起動時に -T または -t のスイッチを与えることで汚染チェックが可能である[*14]が、本モジュールにおいてそのような設定はまだ一部しか与えていない。
3. **本モジュールにテストプログラムは付属しているか:** 未実装である。[*15]　実行結果が正しいかどうかの検証には、下記の方法により確かめることが望ましい。
   (a) 単純な例を入力して、動作を理解し、意図した通りに動作するか確認すること。
   (b) 別のプログラムをうまく利用して異なる方法で結果をチェックをすること。
   (c) Perl で書かれている本モジュールのプログラムファイルを閲覧して、意図しない動作が起こりにくいことを確認すること。(閲覧して、プログラムが十分にシンプルに書かれていることに疑問がある場合は、連絡して欲しい。たとえば、同じような動作を 2 箇所以上で実装していたりすると、バグの原因になりやすい。)

3.2　本モジュール Table::Hack の仕組み

- **どのバージョンの Perl に依存するか:**

  現状 5.14 以前に合わせている。(2018 年 4 月現在の Perl の最新バージョンは 5.26 である。) 世界のさまざまな機関で利用可能となるよう、2011 年以降の Perl であれば十分であろうと考えたため。技術的な要請として、乱数シードを設定する srand の動作が 5.14 以降で変更となったためでもある。[*16]それでも、出来るだけ古い計算機環境でも利用可能とすべく、5.1 でも動くように、case, say, state などの利用は出来るだけ避けるか避けようとしている。同様の理由で splice でのリファレンスの利用もしていない。

- **本モジュールに含まれるプログラムが どのモジュールをよくインポートしているか:**
  - strict
  - warnings → Perl 5.6 からコアモジュール。プラグマである。
  - **Getopt::Std** → オプションスイッチのオプションとそのパラメータを取得するため、および、--help でオンラインマニュアルの出力をするため。Perl 5 の最初からコアモジュールである。

[*12] ただし、その場合は cpan, cpanm, Module::Build についての単なる利用者であること以上の知識が必要とされる。
[*13] echo $PATH | tr : "\n" や which -a コマンド名 のようなコマンドを使って検討することになるであろう。
[*14] コマンドラインで perldoc perlrun を実行すると、そのスイッチの意味が記載されたマニュアルが表示される。
[*15] 通常 Perl のテストはライブラリ (*.pm 形式のファイル) に対して行う。Test::More などのモジュールはそのように設計されているように思われる (異なるかも知れないが、少なくともチュートリアルではそのようだ)。Perl のスクリプトに対して、汎用性の高いテストの方法を現在検討中である。
[*16] perldoc -f srand で参照が可能。

  - FindBin → 実行中のプログラムの名前を $Script で参照するため。Perl 5.3.7 からコアモジュール。
  - **Term::ANSIColor** → 出力に ANSI シーケンスカラーで色を付けるため。Perl 5.6 からコアモジュール。
  - List::Util → min や max 、sum 、sum0 関数を使う為。Perl 5.7.3 からコアモジュール。

- 本モジュールのインストールに必要なモジュール:

  Module::Build の CPAN モジュールの事前インストールが、Table::Hack を cpan, cpanm もしくは Build の方法でインストールする際に、必要である。

第 II 部
Table::Hack を使い始める

> 各コマンドが端末で動作することを確認すれば、インストール完了と見なすことができる。コマンドラインにて、saikoro を通常のコマンドのように走らせると良い。

ここでは、提供されているいくつかのコマンドの内、5 個について紹介する。Table::Hack に収録されている全コマンドの解説については、本文書の最後にある付録 B 節を参照。

4　各コマンドを使い始める

提供されている全てのプログラムについて、第一行目は #!/usr/bin/perl で始まる文字列で書かれている。これは、**シェバング**もしくは**シバング**と呼ばれ、たとえば latextable というコマンドについて perl latextable のように perl を付加して実行しなくても、単に latextable と書くだけで同じ動作をする。ただし、下記の条件を必要とする。

- そのプログラムが実行可能であること。

  (そうでないならパーミッションを変更。chmod +x filename を実行。)
- そのプログラムを格納しているディレクトリにパスが通っていること。
- perl が /usr/bin/perl にあって実行可能であること。

  (他の場所に perl があるならシェバングを書き換える必要がある。which -a perl で確認可能。)
- なお、シェバングが認識されない場合 perl コマンド名 で実行する。[*17]

[*17] 本モジュール Table::Hack が提供するプログラムファイルは.pl の拡張子がファイル名の末尾には無いが、Perl 言語で書かれたプログラムである。Perl 言語で書かれたプログラムファイルは、通常 .pl があるが、無くとも機能上は問題無い。混乱を覚える人もいるかもしれないが、Unix/Linux コマンドのように使われたいという意図を本モジュールの提供者は持つので、そのようにしている。

4.1 saikoro 一様乱数

saikoro は一様乱数を出力する。コマンドラインで下記を順次実行する。

saikoro → 乱数が表示される。サイコロのように 1 から 6 までの整数が出力される。
saikoro > /dev/null → 標準エラー出力のみが表示される。
saikoro -g 5x3 → 5 行 3 列の乱数が表示される。
saikoro -g 10x5 -y 91..100 → 91 から 100 の範囲の整数の一様乱数が、10 行 5 列の乱数で出力。
saikoro -s 123 → 乱数シードの設定。再現性が確保される (繰り返し実行しても同じ結果になる)。
saikoro -. 3 → 小数点以下 3 桁まで表示する。連続一様乱数になる。
saikoro --help → ヘルプマニュアルの表示
saikoro --help opt → ヘルプの内、オプションスイッチのみを表示。
saikoro --help en → 日本語では無く、英語のヘルプを表示。

saikoro -/ , -g 5x5 → 横方向の区切り文字の変更。CSV 形式として使える。
saikoro -g12x12 -y 1.5e5 → 科学的表記法 (1.5e5 = 150,000) も使える。
saikoro -g12x12 -y 1.5e5 | less -x7 → タブ間隔は less で表示中も -x N enter で変更可能。
tabs -6 → タブの表示間隔を 6 文字に変更する。元に戻す場合は、単に tabs を実行。

4.2 transpose 転置行列

transpose は標準入力から TSV 形式 (タブ文字区切り形式) で行列を読取り、その転置行列 (縦方向と横方向を逆転した行列) を標準出力に出力する。入力は数値で無くても良い。

transpose <(saikoro -s123) → <(..) はプロセス置換である。
saikoro -s123 -g3x5 | transpose → saikoro -s123 -g3x5 と比較せよ。
saikoro -s56 -g 7x8 | transpose
saikoro -/, -g3x5 | transpose -/ ";" → トリッキーな使い方なので、汎用性はあまり無い。

4.3 csel 列選択

csel は簡単な列処理を AWK 言語や cut コマンドよりも簡単に実行出来るように考えて作られた。標準出力に出力する。入力は数値で無くても良い。

saikoro -q -y23 -g4x5 -s67 > tmp01 ← 一時的なファイル tmp01 にデータを書き込む。
csel -p4,2 tmp01 ← 4 列目と 2 列目のみを出力。上記のと比較せよ。
csel -p4,2 < tmp01 ←リダイレクション (<)
< tmp01 csel -p4,2 ← 似た様なコマンドの末尾のみを変更したい場合、便利。
csel -d 2,4 tmp01 ← 2 列目と 4 列目の出力を抑制。
csel -d 2..4 tmp01 ← 2 列目から 4 列目までの出力を抑制。

`csel -d -1 tmp01` ← 最右列の出力を抑制
`csel -h -1 tmp01` ← 最右列を最も左 (先頭 head) に出力。
`csel -t -2,3 tmp01` ← 2,3 列目を最も右 (末尾 tail) に移動。
`csel -t -2,3 tmp01 | csel -~ -t 2,3` ← 列の並びを元に戻す。何か処理を挟む場合に便利。

4.4 csv2tsv CSV 形式→ TSV 形式

`csv2tsv` は CSV 形式 (RFC4180) を TSV 形式に変換する。このプログラムは コアモジュールでは無い CPAN モジュール `Text::CSV_XS` に依存している。(似た様な実装は容易に他の人も思いつくので、プログラム名が衝突する可能性は、本モジュールの他の 4 個のプログラムに比べ、最も高いと考えられる。)

`csv2tsv foo.csv > foo.tsv` ← CSV 形式の foo.csv を読取り TSV 形式に変換。
`csv2tsv` もしくは `cat | csv2tsv` ← 手で CSV 形式で入力する。
`csv2tsv -t '(TAB)'` ← CSV 形式のあるレコードがタブ文字を含む場合、何に変換するか指定。
`csv2tsv -n '(ENTER)'` ← C あるレコードが改行文字を含む場合、何に変換するか指定。
`csv2tsv -2` ← 出力の区切りが 2 行の改行文字になる。レコード中のの改行の様子を簡易に調べる。
`csv2tsv -~` ← 逆変換、つまり、TSV 形式のデータを CSV 形式に変換する。

`latextable` コマンドによりこのような表が容易に作成出来る。罫線の有無は選択可能。
Excel でも Google のワークシートでもコピペして LaTeX 用の表が作成出来る。

2	4	9	1	1	3	6	8	6	1	7	3	9	1	8	1	8	3	9	1	9	5	2	8	3	8	7	1	7	8
7	9	0	9	6	9	2	5	8	6	8	6	1	5	9	4	1	9	3	1	6	3	5	2	1	6	2	3	1	6
9	5	2	4	3	3	9	4	7	9	3	6	8	6	3	0	1	5	2	5	0	3	9	9	1	5	5	8	2	6
2	4	1	3	7	2	7	5	8	3	2	1	2	9	8	0	2	0	0	5	3	5	6	2	9	8	7	3	5	0
8	7	4	9	4	1	4	9	4	7	9	0	9	1	7	7	7	2	5	1	0	9	1	9	4	5	9	4	0	2
6	3	2	2	7	2	9	8	1	0	2	9	7	3	6	9	7	7	8	8	5	9	6	5	9	9	2	7	4	0
2	0	8	4	8	9	7	3	4	0	1	8	3	0	2	8	9	1	3	3	3	7	3	5	5	9	7	8	2	2
5	0	7	6	8	5	6	0	5	0	6	7	8	1	1	1	7	4	3	9	6	4	6	9	7	4	9	7	4	2
6	4	6	2	9	2	6	5	7	0	5	7	5	4	4	8	0	8	1	7	7	2	6	9	1	6	3	7	9	4
5	8	3	9	1	3	4	1	3	0	7	7	4	2	6	4	1	0	8	8	6	1	8	9	9	0	6	8	4	7
5	1	3	7	8	7	8	7	7	3	2	1	5	9	0	9	7	2	7	7	5	4	6	0	0	2	5	9	5	5
4	8	2	5	2	8	4	6	6	9	9	2	4	9	0	8	8	2	0	3	5	7	6	7	0	6	0	7	3	9
4	7	3	8	1	5	2	4	9	5	8	6	9	8	2	5	5	2	9	6	8	3	0	6	8	1	8	0	6	0
0	5	8	8	3	9	1	4	4	2	9	8	1	9	3	5	1	9	6	4	9	7	9	5	1	3	8	7	2	0
7	5	8	1	7	1	7	4	8	2	7	1	4	5	3	4	4	2	2	1	5	4	2	6	7	3	1	9	9	1
7	1	0	2	0	5	8	0	8	3	4	8	8	9	9	6	3	8	8	1	6	8	4	3	7	4	0	1	0	8
3	1	4	5	6	1	8	4	4	3	0	1	2	7	2	6	5	4	3	2	0	0	0	8	3	7	6	9	2	3
9	6	5	8	7	2	2	0	5	2	7	8	5	2	6	3	4	2	7	0	1	7	8	9	4	5	9	9	4	2
2	0	7	6	2	6	3	9	8	5	1	8	2	5	4	1	3	2	4	9	6	4	3	4	5	7	6	4	6	1
2	2	6	5	7	1	6	4	7	1	8	5	9	5	1	9	7	2	2	9	8	9	7	7	0	8	6	5	4	8
5	9	8	9	9	4	6	9	0	7	8	6	3	8	0	2	0	8	2	6	0	6	1	1	7	0	4	4	3	9
5	1	3	7	5	0	3	7	8	2	5	7	8	0	2	8	9	1	0	0	2	7	3	3	8	8	1	8	0	8
4	4	5	4	6	5	6	5	3	2	8	0	6	9	6	6	3	1	5	4	0	3	2	5	5	3	4	4	4	8
7	0	0	7	2	3	9	0	5	5	8	1	1	8	8	0	2	2	3	5	1	2	6	2	9	0	0	1	2	8
0	1	5	0	1	7	1	2	7	1	2	7	5	9	7	9	1	6	9	5	3	0	6	4	7	6	0	7	6	6
7	1	2	9	7	6	8	8	6	2	4	0	8	9	8	4	6	1	1	9	7	5	4	5	3	5	2	3	1	5
7	1	6	1	1	0	0	5	4	1	5	2	0	9	1	0	9	3	4	0	2	6	3	6	2	1	5	9	4	3
3	3	1	7	4	0	9	6	0	7	3	0	0	4	7	9	0	8	6	3	5	6	8	1	4	3	0	5	8	1
8	5	4	4	4	9	1	2	3	8	0	5	1	6	4	9	6	7	6	8	9	4	3	7	6	9	0	4	6	6
1	0	8	9	4	1	1	0	6	3	6	5	6	0	3	6	7	2	1	7	9	5	8	7	4	1	7	0	1	4

表 3 `saikoro -s123 -g 30x30 -y0..9 | latextable -m0.6 -\# . -1 -=`

`csv2tsv --help` ← ヘルプ

`csv2tsv --help opt` ← オプションのヘルプのみ見る。opt は opti, optio, option, options でも良い。

4.5 latextable LaTeX 用製表機

latextable は、エクセルや各種 SQL ソフトの出力結果から LaTeX の表 (table 環境) への使い易く多機能な変換器として作られた。様々な記号を含む文字列を LaTeX に使えるようにすぐ変換する目的に使うこともできる。

saikoro -g12x12 | latextable ← LaTeX を知っているなら何が起きたか分かるはず。
latextable ← この 1 つの単語を端末で入力して (Enter を押下する)。
そしてキーボードで何かを入力し最後に Ctrl+D を押下。
すると \begin{table} と \end{table} で囲まれた文字列が生成される。
要領がつかめたら、LaTeX の .tex 形式のファイルに記入して動作を確かめる。
エクセルや各種表計算ソフトからコピーをしてペーストをして、いろいろ試すと良い。
latextable --help opt ← どのようなスイッチオプションが使えるか確かめる。
latextable -s ← 入力の余分な空白を取り去る。(BigQuery をブラウザ経由で使う場合に便利。)
latextable -_ ← 半角と全角の空白、全角ハイフンの存在を分かり安く表示するようにする。
latextable -3 コンマ区切り 1 始まりの列番号 ← 数値を 3 桁毎に区切って右寄せする。
latextable -j ← 日本語の半角カナを半角の幅で表示するようにする。
latextable -m0.7 -x40mm ← 大きな表を 0.7 倍の大きさにし、センタリングをする時に使う。
latextable -C1 ← 任意の (UTF8 の) 文字列を、LaTeX 用に表とは関係無く簡潔に出力。
latextable -Cmn ← table 環境の caption で複数行にまたがる SQL 文を挿入する時便利。

下記で latextable を実行した例を示す。

```
latextable -s   # -s はデータの余分な空白セルを取り除くために（やむなく）付加した
      ↑↑この 1 行がコマンドラインに入力したコマンド↑↑  ↓↓この下はエクセルなどの表からのコピペ。↓↓
75 2007-02-22 12:30:17.000 UTC 34;Allen Morgan said tha 278
76 2007-02-22 13:13:39.000 UTC rhaps you have heard of 454
77 2007-02-22 13:28:06.000 UTC tp://www.miketaber.net/a 461
78 2007-02-22 14:48:18.000 UTC ’s too bad. Javascript-i 257
79 2007-02-22 16:52:25.000 UTC y omarish - i’m in on th 493
80 2007-02-22 17:19:04.000 UTC ry good idea. We can act 501
81 2007-02-22 17:29:42.000 UTC is is a cool service--I’504
82 2007-02-22 19:37:53.000 UTC cellent point; YC might 452
83 2007-02-22 21:07:47.000 UTC at a strange mix. About 530
84 2007-02-22 21:20:44.000 UTC won’t clutter it up. Lo 388
85 2007-02-22 21:22:59.000 UTC that case, I _am_ excit 540
86 2007-02-22 21:49:17.000 UTC e simplest repro case is 531
87 2007-02-22 21:50:03.000 UTC r the question:<p>Why wo 531
      ↑↑ データの入力はここで終わり。↑↑  ↓↓下記は出力。これをさらにコピペして LaTeX で使う。↓↓
13 lines are read and accepted.
\begin{table}
    \center
    %\hspace*{-0mm}
    %\rotatebox{0}{
    %\scalebox{1.}{
        \begin{tabular}{ | llll | }
            \hline
            75 & 2007-02-22 12:30:17.000 UTC & 34;Allen Morgan said tha & 278\\
            76 & 2007-02-22 13:13:39.000 UTC & rhaps you have heard of & 454\\
            77 & 2007-02-22 13:28:06.000 UTC & tp://www.miketaber.net/a & 461\\
            78 & 2007-02-22 14:48:18.000 UTC & ’s too bad. Javascript-i & 257\\
            79 & 2007-02-22 16:52:25.000 UTC & y omarish - i’m in on th & 493\\
            80 & 2007-02-22 17:19:04.000 UTC & ry good idea. We can act & 501\\
            81 & 2007-02-22 17:29:42.000 UTC & is is a cool service-{}-I’ & 504\\
            82 & 2007-02-22 19:37:53.000 UTC & cellent point; YC might & 452\\
            83 & 2007-02-22 21:07:47.000 UTC & at a strange mix. About & 530\\
            84 & 2007-02-22 21:20:44.000 UTC & won’t clutter it up. Lo & 388\\
            85 & 2007-02-22 21:22:59.000 UTC & that case, I \_am\_ excit & 540\\
            86 & 2007-02-22 21:49:17.000 UTC & e simplest repro case is & 531\\
            87 & 2007-02-22 21:50:03.000 UTC & r the question:$<$p$>$Why wo & 531\\  \hline
        \end{tabular}
    %}% closing scalebox
    %}% closing rotatebox
    %\hspace*{-0mm}
    \caption{}
    \label{tbl:547441}
\end{table}
```

上記の\begin{table}から\end{table}までを LaTeX を使う為に *.tex ファイルに記入してコンパイルすると、表 4 が現れる。手作業でいろいろな編集をすることと比較すると、非常に楽である。

75	2007-02-22 12:30:17.000 UTC	34;Allen Morgan said tha	278
76	2007-02-22 13:13:39.000 UTC	rhaps you have heard of	454
77	2007-02-22 13:28:06.000 UTC	tp://www.miketaber.net/a	461
78	2007-02-22 14:48:18.000 UTC	's too bad. Javascript-i	257
79	2007-02-22 16:52:25.000 UTC	y omarish - i'm in on th	493
80	2007-02-22 17:19:04.000 UTC	ry good idea. We can act	501
81	2007-02-22 17:29:42.000 UTC	is is a cool service--I'	504
82	2007-02-22 19:37:53.000 UTC	cellent point; YC might	452
83	2007-02-22 21:07:47.000 UTC	at a strange mix. About	530
84	2007-02-22 21:20:44.000 UTC	won't clutter it up. Lo	388
85	2007-02-22 21:22:59.000 UTC	that case, I _am_ excit	540
86	2007-02-22 21:49:17.000 UTC	e simplest repro case is	531
87	2007-02-22 21:50:03.000 UTC	r the question:<p>Why wo	531

表 4　Google BigQuery で次のコマンドを実行した結果を `latextable -s` のコピペで入力した結果を LaTeX のファイルにコピペした結果が上の表である。英数字以外の記号が含まれていて、正しく LaTeX のコマンドにする手間が掛かる場合でも、すぐに `latextable` に変換出来る。次のコマンドも `latextable` で作成している。→ **with T as (select time_ts , substr(text ,3,24) text , parent, id from 'bigquery-public-data.hacker_news.comments' where text like '% € %' or not regexp_contains(substr(text,1,30), '^[0-9a-zA-Z ,?) !\']*$')) select * except (id) from T where length (text) < 40 and length(text) >8 and text not like '%.' order by id**

5　不具合 (バグ等) を発見した場合/機能を追加したい場合

全てのソフトウェアで言われるように、バグの無いプログラムは無い。現状においても、本モジュールの作成者は、バグを見つけているし、それを仕様とごまかすつもりはない。本モジュール `Table::Hack` についてのバグについては以下の傾向があるであろう。

1. 各プログラムファイルについて、最近、もしくは、最後に追加した機能は、不具合は発生しよいであろう。十分なテストをしていないためである。
2. 各プログラムファイルについて、最初の方に設計され実装がされた機能は、何度も使われているので、バグはあまりないだろう。しかし、プログラムに編集を別の機能のために編集を加えたために、巻き添えでバグを誘発することは、ありそうである。
3. 各プログラムにスイッチ `--help` を追加して表示されるオンラインヘルプマニュアルは、書き漏らしや書き間違いがあるかもしれない。
4. 別のモジュール、特に本モジュール作成人による別のモジュールをインストールすると、プログラムファイルを上書きをすると考えられるので、古いプログラムにしてしまう可能性もある。
5. `Table::Hack` が提供するコマンドに不具合があった場合、それはほぼ全てそのコマンドの機能を実現している 1 つのプログラムファイルが引き起こしている。`saikoro` のバグは `saikoro` というファイル内に存在し、`latextable` のバグは `latextable` というファイル内に存在し、それ以外の場所のバグで発生することは極めて可能性が低い[*18]。

[*18] それでも Perl 言語または OS の制御をしているファイルが一部書き換わっていたという事象は考えられるし、Perl もソフトウェアであるから多数の人のチェックを受けていてもバグは残っている可能性はあって、それが `Table::Hack` の提供コマンドの実行

もしも利用者自身がそのような不具合を修正したい場合は、Perl 言語で書かれたプログラムファイルを修正する必要がある。現状 `Table::Hack` の各コマンドは機能がその 1 つのプログラム内に集約されているので、Perl 言語の知識があれば修正は容易であろう。Perl 言語プログラミングの参照先は A.2 節。

なお、不具合を減らすための方策としては、下記が考えられる。ただし、最後の方は、あまり標準的なプログラミングでは使わない方法である。

- テスト/診断機能:
  - 自動テストのプログラムを作る。`Test::More` などを利用して。
  - 不具合を見つかりやすいように、該当するプログラム内に検証機能を追加する。
  - 提供機能の各機能に対して、ダブルチェックが可能なようにする。
- プログラムの保守性を高める:
  - プログラムの構造を、工夫して分かり安く保つ。
  - 最低限、各変数については、その役割をコメントに書く。
  - プログラム内において、(1 行で収まらないような) 同じような機能は 2 回以上書かないようにする。
  - 関数内の関数を多用する。[*19] 普通の関数が多いと、問題の切り出しに時間がかかるため。
  - 演算子の優先順位を利用して、括弧 () を出来るだけ使わないようにする。修正時のタイピングの数が減るし、優先順位を熟知している場合は可読性が向上する。

付録 A　コマンド習得/プログラミングの必要が生じた場合

インターネットの接続が遅い、プログラミングの出来る人が周囲にいない、もしくは、国際協力で外国に派遣されデータ分析を要請されている同僚に最低限のプログラミングを教える必要性が生じた、などの状況を想定し，下記を記す。

A.1　Unix または Linux でシェルの bash などを使う場合に

A.1.1　各コマンドの機能の調べ方

必要に応じ、下記 4 個を行う。コマンドによりどれで十分なヘルプの参照が可能であるかは異なる。

- `command --help` のように、調べたいコマンドの後ろに`--help` を付加して実行する。
- `man command` のように調べたい `command` (例、`cat` など) の前に `man` をつけて実行する。
- `help for` のように、ある種の基本的なコマンドは (この場合は `for`) は `man` でなくて `help` を使う。
- `info iconv | cat` または `info iconv | less` のように info を使うと、詳細な説明が読める場合があるので、念のためこれを参照する (この場合は iconv のコマンドを例に説明した)。

  単に `info iconv` だと info のキーボード操作の習得を必要とする (例, h キーでヘルプ, q で終了)。したがって、パイプ (`|` 下記で説明) を使った説明をここでは示した。

結果に影響することはありうる。可能性は低いが、それが疑われる場合の最終手段は、再インストールである。

[*19] my ではなくて our を多用することになるかもしれない。

A.1.2 オプションに関して

1. Unix/Linux の各コマンドは、オプションと呼ばれる-(ハイフン) と決められた文字を連結させたもので指定されるいろいろな機能の指定が可能である。(オプションはさらにパラメータを伴うことがある。)

   → `cat -n foo` の場合、ファイル `foo` の中身が `-n` の指定により行番号を行頭に付加して表示される。

   → `grep -B 2 -A 1 string foo` の場合、string にマッチするファイル foo の各行と、さらにその直前 (before) の 2 行とその直後 (after) の 1 行が出力される。
2. オプションでは無い引数は多くの場合、ファイル名 (の並び) と見なされる。
3. オプションは各コマンドごとに決められている。オプションにパラメータが伴うかどうかも各オプションごとに決められている。それを知るには、マニュアルを参照すると良い。
4. オプションで無い引数が現れると、通常は、それより後は、オプションのような文字列が現れてもオプションとは認識されない。

   → 仮に `cat foo -n` を実行しても、`-n` は `cat` にとってただのファイル名と認識されて、オプションとは見なされない。したがって `-n` というファイル名のファイルを探して、普通は存在しないので無いことになり、`cat` はエラーを起こす。
5. オプションのように見えるファイル名を持つファイルが存在する場合に、それをコマンドの引数として渡すには、そのファイル名の直前に `--` の文字列を置く。

   `cat -- -n` とすると、ファイル名`-n` というファイルの中身が (思ったように) 表示される。単に `cat -n` とすると、`cat` は標準入力からの入力を待つ状態になってしまう。

A.1.3 パイプとリダイレクション、コマンド置換、プロセス置換について

- **パイプ** `|` を 2 つのコマンド文の間に挟むことで、前のコマンド文の標準出力を次のコマンド文の標準入力に渡すことができる。
- **リダイレクション**とは `<` または `>` などのことで、この 2 つはそれぞれ、ファイルの中身をコマンドの標準入力に渡すことと、コマンド文の標準出力をファイルに書き込んだりする。

  リダイレクションには様々なものがあるので、必要に応じて、覚えていくと良い。`>>` で追加書込み、`2>` で標準エラー出力の書込みなどがある。シェルと OS により違うものがあり、調べて工夫して使うこと。
- **コマンド置換**は、バッククオーテーションでコマンド文を囲む方法と、`$(`と`)` で囲む方法がある。囲まれたコマンド文の出力結果が、あたかもただの文字列のように振る舞うので、囲みの外のコマンドとうまく結合させて用いる。
- **プロセス置換** (process substitution) は `cat <( echo 123 )` のように、`<(`と`)` に別のコマンド文を挟んで使う。そのコマンド文の出力結果が、あたかもひとつのファイルの中身に書かれたものであるかのように振る舞う。無駄に中間ファイルを残す必要が無くなるので便利である。

  Ctrl+C などによって発生するシグナルに対する挙動が、`echo 123 | cat` の場合は echo に cat よりも早く働くが、プロセス置換の場合はその逆のことが多い。つまり、同じ機能をしていても、シグナルに対する挙動が違うので、注意 (何らかの目的で利用可能かもしれないが、その動作が保証されているかどうかは正式な文書を参照する必要があるであろう)。

A.1.4 TSV 形式 (タブ文字区切りの形式) を扱う上で習熟すべきと思われるコマンド

`less`、`wc`、`gawk`、`cut`、`head`、`tail`、`grep`、`sed`、`od`、`iconv`、`nkf`、`lv`、`file`、`diff`、`sdiff`、`sort`、`uniq`、`paste`、`column`、`fold`、`tabs`、`time`、`unbuffer`

A.2 Perl 言語でプログラミングをしたい場合

- Perl 言語の関数のマニュアルは `perldoc -f 関数名` ですぐ参照できる。例、`perldoc -f srand`
- Perl 言語の演算子については `perldoc perlop` ですぐ参照できる。演算子の優先順位の確認にも便利。
- Perl 言語の「チートシート」(カンニングペーパーのように狭い文面で Perl のプログラミングの重要事項を記載したメモ) は `perldoc perlcheat`。
- `perl` コマンド自身のさまざまなオプションを知りたい場合は `perldoc perlrun`。
- その他全てのドキュメントの一覧を見たい場合は `perldoc perltoc`。
- やや高レベルの Perl 言語運用スキルを身に付けたい場合は、本文書の最後に記載した参考文献 [2][3] を参照。

付録 B 本モジュール作成者が提供するモジュール

B.1 Table::Hack に収録したプログラム — 2018 年 6 月 19 日現在

全プログラムをここでは紹介する。それらの簡潔な機能一覧は表 5 を参照。各プログラムは、コマンドラインインターフェースのコマンドと同じように使えるように作られている。`--help` をコマンドの後ろに付けることで、すぐにヘルプを参照が可能 (例、`saikoro --help`)。`--help opt` にすることでオプションの一覧だけを短めに参照が出来て便利。コマンドによっては英語マニュアルを付属していて、`--help en` で参照が可能。各コマンドは Unix 哲学 [4] に基づいて設計している。

---

expskip 　ファイルを読取りつつ 1, 2, 3, 5, 10, 20, 50, 100, 200, 500.. 行目を表示する。1, 10, 100, 1000, 1 万, 10 万.. 行目のみを表示する様にすることも出来る。大きなテキストファイルの中身を最初に確認するときに便利。最初と最後の 3 行は特に指示が無い限り出力。

colorplus 　古くから使われるアスキーカラーシーケンスの仕組みを使って、コマンドラインインターフェースの文字に**着色**するいろいろな方法を提供。`-3` により、テキストの数の部分だけ 3 桁区切りで着色して、数を読みやすくする。`-0` で全てのアスキーカラーシーケンスコードを除去する。なお、ビューアの `less` で使う場合は `less -R` とするか、シェル変数`$LESS` に`-R` の値を設定すると良い。

colsummary 　TSV ファイルに対して、各列の次の統計情報をこのコマンド一発で整理して出力する。(1) 何通りの異なる値が出現したか (2) 出現値の最小値と最大値 (3) 最頻値いくつか (4) 最頻値と低頻度値の両方の頻度の様子 (5) 数値と見なした場合の平均値 (6) 値の桁数の範囲。入力の一行目が、列名の並びであってデータの値でない場合に、`--` という記号を用いたオプションで適切な処理をさせるようにすることが可能。

freq 　入力を改行文字で区切られて各行が与えられていると見なして、その各行の値が、それぞれ何個現れた

Table::Hack が提供するコマンドの一覧

コマンド名	主な機能
expskip	1, 10, 100, 1000,.. 番目の行だけを出力したりする。
colorplus	数字や横広のテーブルの文字に着色して見やすくする。
colsummary	表形式データの各列の普通に知りたい情報を一発で整理。
freq	どんな文字列の行が何回現れたか。一元分割表。
sampler	各行を確率抽出する。
csel	列選択。AWK/cut より簡単/柔軟。
venn	各行を要素と見なし数個のファイル間で包含関係を見る。
crosstable	クロス集計表。二元分割表。
latextable	どんな文字列/TSV でも LaTeX で使えるコマンド/表に変換。
madeafter	ファイルの時刻情報 3 種を一発で表示する。
colchop	行列の各セルの文字列長を制限したり折りたたんだり。
saikoro	一様分布の乱数 (行列) を生成する。
transpose	表の縦方向と横方向を交換する。転置行列。
csv2tsv	CSV 形式 (RFC4180 準拠) をタブ文字区切りの TSV 形式に。

表 5　上記のコマンドは試しに使えば、すぐに意味が明解になるであろう。

かを出力する。つまり、一元分割表を出力する。基本機能は `sort | uniq -c` と同じであるが、`freq` はソートを伴わずビンを用意して集計する方式なので数百万行程度以上でも遅さを感じさせず、またさまざまなオプションスイッチによる機能を提供している。

-f　頻度によって出力表をソート
-n　入力の各行の文字列を数と見なしてソート
-r　ソート順を逆転
-=　入力の 1 行目をソートの対象外と指定
-s　頻度の累積和を出力するようにできる。
-y **数値範囲**　頻度の数値条件を指定。例、`-y 1` で 1 回しか出現しない値。`-y 2..` で複数回出現する値の様子

Unix/Linux の sort や uniq コマンドは、非アスキー文字に対して動作しないことがあるが、このコマンドはそのようなことは起きにくい (UTF-8 では起きない)。

sampler　`-r` により抽出確率を数値で指定すると、入力の各行に対して**指定確率で抽出操作**を行う。復元抽出と非復元抽出の両方をシュミレートできる。(このコマンドは、本モジュール提供者の都合でまだあまり洗練されていないスイッチをいくつか含む。また各行で乱数生成を行うのではなく、何行読み飛ばすかの計算をするごとに 1 個の乱数生成をするように工夫をしたかったが、未実装である。)

csel　`AWK` 言語または `cut` コマンドよりも使いやすい**列選択を指定**できるコマンドである。

-p **数列**　TSV 形式のファイルが来た場合に、特定の列だけを表示するには `-p 3,2,1` のように指定する。`cut -f 3,2,1` では 1, 2, 3 列目の順序のまま出力されてしまい不便だが、それを回避できる。

.. (範囲指定) `-p 10..16` と範囲指定も可能なので `awk '{print $11,$12,$13,$14,$15,$16}'` のような多数の打鍵が、回避できる。

-d 数列 `-d` により指定列だけを出力しない。

-h 数列 `-h` により指定列を先頭 (最左) 列に移動。

-t 数列 `-t` により指定列を末尾 (最右) 列に移動できるので、何かと便利だろう。(数値列と文字列の混合した表に対して、画面で見やすくするために簡単なコマンド操作で文字列だけを最右列に持って行きたいことは多い。)

-～ 上記のオプションで指定した操作に対して「**対象群の要素の逆元**」としての操作をするように指定することができる。たとえば`-t 2,4` は 2 列目と 4 列目を全体が何列か分からないとしても最も末尾 (右側) の列に移動する操作であったが、`-～t 2,4` により、その操作で移動した列を元に戻す指定となる。

venn いくつかのファイルの中身の各行の値がどのように重なるかを知るために用いる。`venn file1 file2 file3` のように使う。N 個のファイルがあるとき、あるファイルのある行の値がどのファイル i に含まれているかのパターンを $2^N - 1$ 通りに分類し、各パターンに異なる行の値が何回出現したか、各ファイル i に計何回出現したかを表形式で出力する。同じ符号体系をどの程度共有しているかを把握することで、DB のテーブルのどの列とどの列が結合出来るのか把握する時に有用である。

crosstable TSV2 列の入力から **2 元分割表**を出力する。入力の各行が i(TAB)j の値であるとき、出力は $n \times m$ の行列の左側に縦に出現した i の値が並び、上側に横に出現した j の値が並び、行列の中は対応する値が何回出現したかを表す。オプションを使って `crosstable -3` とすると、各行が n(TAB)i(TAB)j の入力を受け付けて、出力する行列の中の数値は対応する n の合計値となる。`freq` が 1 次元だとすると `crosstable` はその 2 次元版である。このような形式の出力の 2 元分割表はとても有用であるにも関わらず、簡単に出力する Unix/Linux コマンドも SQL も無いのでこのコマンドを作成をした。

latextable エクセルやグーグルドライブのスプレッドシート、各種 SQL の出力結果をコピーとペーストの操作で容易に LaTeX で使える table 環境に変換する大変有用な機能を `latextable` は提供する。[*20][*21]

madeafter 多数のファイルに対して、atime, mtime, ctime を一度に即座に表示できるインターフェースを提供。有用と思われるシチュエーション: (1) ファイルの時刻情報の意味をとっさに理解したい時 (2) 多数のファイルがいつどのように操作されたかさくっと把握したい時 (3) クラウドのファイル同期サービスに不具合がある時。

colchop 特定の長さで切ったり折りたたんで次行に持っていったりする機能を提供する。長さの指定には，バイト長、UTF-8 文字列長、もしくは半角以外は 2 と換算する方法のいずれかの方法で指定が可能。CPAN モジュール `Text::VisualWidth::UTF8` に依存している。クライアントからもらったデータのテキストファイルの一行目が列名の並びで、それらの文字列長が長くて、`less` で閲覧するにも不便なことがあるが、それを低減する。

[*20] 多数の細かい表をひとつの文書にまとめることは、データを分析して得た知見をまとめる為に有用な作業である。それ無しには、過去の分析からの知見を後日参照することが厄介になってしまう。

[*21] 2018 年にいたってなお 10 個か 20 個のテーブルを安定して安心して扱えるソフトウェアは LaTeX 以外にあまり無いように思われる。よく使われているオフィス系ソフトはすぐにファイルサイズが大きくなったり、操作に時間がかかるようになったり、起動に時間が掛かったり、動作が不安定になって作成中の文書が失われたりすることが未だによく起こる。

saikoro　いろいろな**一様乱数** (の行列) を生成する。`-g 3x4` で 3 行 4 列を出力する様にする、`-y 10..99` で各乱数の数値範囲を指定するようにする、`-. 3` で小数点以下 3 桁まで出力するようにする、などのいろいろな指定が可能。`Table::Hack` が提供する様々なコマンドの機能を試すために TSV ファイルの代わりをさせることに多用する人もいるかもしれない。

transpose　**転置行列**の出力 (数値行列で無くて文字列行列でも良い)。つまり表の縦方向と横方向を交換する。

csv2tsv　RFC4180 で規定されている CSV 形式の入力を TSV 形式に変換する。CSV 形式はダブルクォーテーション囲みも可能で各セルはタブ文字も改行文字も含みうるので、データ処理上気にしなければいけない問題になるが、そのような場合にタブ文字と改行文字をどう置きかえるかも指定が可能。本モジュール `Table::Hack` が提供するコマンドの多くは、別に TSV に限らず、ダブルクォーテーション囲みを考慮しない場合の CSV 形式もしくは決まった文字で区切られている表形式のデータには `-/ ,` のような指定で対応しているが、それだけでは解決できないこともあるので用意した。

B.2　共通してよく用いられるオプション

--help　オンラインですぐに参照できるマニュアル。ここに書いてある以上のことを知るには、コマンドを実行するか、ソースファイルを読むことである。

--help opt　ヘルプを参照しているコマンドの、オプションの説明のみを出力。必要な時に少ない文量のヘルプが表示されるので便利。`opt` は `opti` でも `optio` でも `option` でも `options` でも良い。

--help en　存在すれば英語でマニュアルを表示する。`en` は `eng` でも `english` でも大文字であっても良い。

--help en nopod　英語のマニュアルを POD を使わないで表示する。

-=　多くの TSV ファイル (および CSV ファイル) は 1 行目がデータの値では無く、列名の並びであるので、その場合に、各コマンドが適切な処理をするようにする。単に読み飛ばす、列名として再利用するなど。

-/ ,　タブ文字以外が区切り文字である場合にそれを指定する。コンマ (,) で無ければ該当部分を別の文字に置きかえる。ただし、RFC4180 で規定されているように CSV 形式のデータをダブルクォーテーション囲みを考慮して処理することはできない。そのような場合は、本モジュールが提供する `csv2tsv` で先に形式を変換すること。

-～　ちょっとした何かの機能を逆転させる。クロス集計表の縦方向と横方向を交換する場合など。

-? foo　未定義値についてどんな値を割り当てるかを指定する。

-%　割合を用いること、百分率を用いることの指定。

-:　入力の各行または行番号の情報を出力させることを指定する場合によく用いる。

-@ n　n 秒ごとに何行の入力が読み取れたかを表示させるための指定に用いる。(まだあまり実装できていない。初期の実装だと、n 行を読み取るごとにその時刻情報を標準エラーに出力していて処理速度を低下させている。)

-. n　小数点以下の桁数を指定する場合に用いる。

-s n　乱数シードを指定する場合によく用いる。

-g n　結果を何個得たいかの個数を指定する場合によく用いる。

-n　「数値であること」「数値を指定すること」を上記 6 通りでは指定しがたい場合に用いる。

-#　上記 7 通りで対応できない数の指定、もしくは特に異なる数を出力指定したい場合。シェルはよく#

をコメントとして認識するので、実際に使う時には `-\#` と書いて指定することになるだろう。

-1　標準エラー出力に出すような情報を抑制する指示。もしくはドライランの様な処理の指定。

-q　出力を抑制気味にしたい場合に用いる。(以前設計したコマンドには多い。おそらく今後は`-1` に統一される。)

-z　gzip などの圧縮形式のファイルの入力の指定。高速化のため。(あまり実装できていない。)

B.3　Table::Hack と似た様なコンセプトを持つ別のモジュール

CLI::Files::Utils　数十から数百万のファイル群を操作する上で有用な機能をいくつか実装。`dirhier` コマンドは引数に与えられたディレクトリ (群) に対してそこから n 階層下に何個のファイルがあるかを $n = 1, 2, 3, ..$ に対して一覧にして示す。1 個のディレクトリに数万個のファイルがあると `ls` コマンドですら操作性が悪くなるが、このコマンドにより非常に多数のファイルを持つディリクトリの構造を解読するのが楽になる。

CLI::Numbers::Hack　数値データに対するちょっとした有用なコマンドを集録。`cumsum` で累積和を計算。`minmax` で数列に対して、最小と最大のそれぞれ n 個ずつを抽出 ($n > 1$ の需要はあるのになかなかすぐ計算するユーティリティが見つからないので作成した)。`quantile` で分位点を計算する。層別に分位点も計算できるので、「商品ごとの価格の分位点を算出する」ことが容易になる。`denomfind` は N 人に対するアンケート調査の集計結果があって、割合の数がいくつか示されている場合に、その数から N を推定するユーティリティーである。「この 6 個の割合の数がパーセント表示の小数点第 2 位が四捨五入されているとすると、回答者が 250 人未満なら 131 人に違いない」のような判断が出来る。

CLI::Table::Key::Finder　TSV 形式の表のデータがあった場合、どの列がキー列であるか探るためのコマンドを集録。`alluniq` は全ての行の文字列が異なるものであるかどうかを判定する。もしも $k\,(k \geq 2)$ 回出現する異なる文字列が d_k 個あった場合には $k = 1, 2, 3, ..$ に対して d_k の値とそれに該当する具体的な文字列を表示する。`keyvalues` は 2 列の表をキーバリュー形式と見なし、各キーが異なるバリューをいくつ持つかを出力する。`wisejoin` は 2 個の表を結合するもので、Unix/Linux の join と SQL の join と同じような機能を提供する (使い易くて便利な機能を多数追加してある)。`colpairs` は B 列の表に対してあらゆる 2 列の組みあわせ $B \times (B-1)/2$ 通りについて、異なる値が何通り現れたかなどを $B \times B$ のマトリックスで表示する。どの 2 列の組みあわせが重要であるかの探索などに用いる。`piececount` は指定されたパターンにマッチする行数を数えたりする。データの形式のチェックに使える。

CLI::TextLines::Utils　`idmaker` 行頭に連番を割り振るが、同じ内容の行が再び出現したら、既に割り当てた連番を与える。`shuffler` 行単位でシャッフルをする。様々なオプションがある。`ngram` 文字単位の n-gram でランキングをする。文字列の頻出パターンを探索するためのもの。`hashtotal` 各行の SHA1 ハッシュ値の先頭 4 バイトを整数化したものを加算する。多くの DB はエクスポートの際に行の順序は意識しないので、行の内容が行の順序に関係なく一致するかを確認するためのもの。ハッシュ変換の乱数の一様性を仮定して、2 つのファイルの行の内容が何行異なるかを確率的に推定することもできる。

参考文献

[1] "A Hacking Toolset for Big Tabular Files", in Proceedings of 2016 IEEE International Conference on Big Data, pp. 2902 – 2910, T. Shimono.

[2] 続・初めての Perl, Randal L.Schwartz, biran d foy, Tom Phoenix 著 (12 章 Perl ディストリビューションを作るには, 21 章 CPAN への投稿)

[3] PERL HACKS プロが教えるテクニック& ツール 101 選, chromatic, Damian Conway, Cutris ”Ovid” Poe 著 (4 章 モジュールの操作)

[4] UNIX の 1/4 世紀, A Quarter Century of UNIX, Peter H. Salus 著, QUIPU LLC 訳

[5] LaTeX2ε 美文書作成入門, 改訂第 6 販, 奥村晴彦, 黒木裕介著