MCMD利用マニュアル

対応 NYSOL バージョン: Ver. 3.0

履歴:

2017 年 4 月 30 日：Ver. 3.0 対応

2016 年 6 月 30 日：集計処理コマンドについての注意点追記、パラメータ微修正

2015 年 12 月 24 日：Ver. 2.4 対応、assert 機能追加

2014 年 10 月 6 日：Ver. 2.0 対応

2014 年 4 月 30 日：例題の Error 修正など

2014 年 3 月 10 日：nysol パッケージへの統合に伴いインストール方法変更

2014 年 2 月 15 日：mxml2csv 追加, 微修正

2013 年 11 月 2 日：mpadding,mvnullto,mvdelnull 追加, 微修正

2013 年 9 月 20 日：初期リリース

2017 年 9 月 24 日

目次

第 1 章　変更点 9
1.1 ver.2 での変更点 . . . 10
1.2 ver.3 での変更点 . . . 11

第 2 章　M コマンド 15
2.1 M コマンドとは . . . 16
2.2 インストール . . . 17
2.3 はじめよう . . . 20
2.4 CSV . . . 23
2.5 データ型 . . . 27
2.6 項目の指定 . . . 28
2.7 データ本体がない場合の動作 . . . 34
2.8 マルチバイト文字 . . . 35
2.9 パラメータ指定 . . . 37
2.10 Environment variable . . . 53
2.11 キーブレイク処理 . . . 55

第 3 章　便利ツール 57
3.1 bash completion . . . 58

第 4 章　コマンドリファレンス 61
4.1 maccum 累積計算 . . . 63
4.2 marff2csv arff 形式から csv 形式への変換 . . . 65
4.3 mavg 項目値の平均 . . . 67
4.4 mbest 指定行の選択 . . . 69
4.5 mbucket 件数均等化バケット分割 . . . 71
4.6 mcat 併合 . . . 75
4.7 mchgnum 数値範囲による置換 . . . 79
4.8 mchgstr 文字列の置換 . . . 82
4.9 mchkcsv CSV データのチェック . . . 85
4.10 mcombi 組合せ計算 . . . 89
4.11 mcommon 参照ファイルによる行選択 . . . 91
4.12 mcount 行数カウント . . . 94
4.13 mcross クロス集計 . . . 95
4.14 m2cross 1 対 N のクロス集計 . . . 97
4.15 mcsv2arff CSV を ARFF 形式に変換 . . . 99
4.16 mcsv2json CSV を JSON 形式へ変換 . . . 101
4.17 mcsvconv CSV を多様なフォーマットに変換 . . . 103

4.18 mcut 項目の選択 . . . 107
4.19 mdata データセットの出力 . . . 109
4.20 mdelnull NULL 値行の削除 . . . 111
4.21 mdformat 日付時刻抽出 . . . 113
4.22 mdsp 画面表示 . . . 115
4.23 mduprec レコードの複写 . . . 117
4.24 mfldname 項目名の変更 . . . 119
4.25 mfsort 項目ソート . . . 121
4.26 mhashavg ハッシュ法による項目値の平均 . . . 122
4.27 mhashsum ハッシュ法による項目値の合計 . . . 124
4.28 minput 画面入力 . . . 127
4.29 mjoin 参照ファイルの項目結合 . . . 129
4.30 mkeybreak キーブレイク箇所 . . . 131
4.31 mmbucket 多次元均等化バケット分割 . . . 133
4.32 mminput 画面フォーム入力 . . . 138
4.33 mmseldsp 複数選択画面入力 . . . 140
4.34 mmvavg 移動平均の算出 . . . 142
4.35 mmvsim 移動窓の類似度計算 . . . 145
4.36 mmvstats 移動窓の統計量の計算 . . . 146
4.37 mnewnumber 連番データの新規生成 . . . 148
4.38 mnewrand 乱数データの新規生成 . . . 150
4.39 mnewstr 固定文字列データの新規生成 . . . 152
4.40 mnjoin 参照ファイル項目の自然結合 . . . 153
4.41 mnormalize 基準化 . . . 155
4.42 mnrcommon 参照ファイルの複数範囲条件による行撰択 . . . 157
4.43 mnrjoin 参照ファイルの複数範囲条件による自然結合 . . . 159
4.44 mnullto NULL 値の置換 . . . 161
4.45 mnumber 連番 . . . 163
4.46 mpadding (行補完) コマンド . . . 167
4.47 mpaste 参照ファイル項目の行番号マッチング結合 . . . 170
4.48 mproduct 参照ファイルの直積結合 . . . 172
4.49 mrand 擬似乱数 . . . 173
4.50 mrjoin 参照ファイルの範囲条件による結合 . . . 176
4.51 msed 正規表現による文字列置換 . . . 179
4.52 msel 条件式による行選択 . . . 183
4.53 mseldsp 選択画面入力 . . . 185
4.54 mselnum 数値範囲による行選択 . . . 187
4.55 mselrand ランダムに行を選択 . . . 189
4.56 mselstr 文字列による行選択 . . . 191
4.57 msep レコードの分割 . . . 196
4.58 msep2 連番-項目値表の出力を伴った行の分割 . . . 198
4.59 msetstr 文字列項目の追加 . . . 200
4.60 mshare 構成比の計算 . . . 202
4.61 mshuffle レコード分割 . . . 203
4.62 msim 二変数間の類似度の計算 . . . 206

4.63 mslide 行ずらし . . . 210
4.64 msortf レコードの並べ換え . . . 213
4.65 msplit 区切り文字による項目分割 . . . 218
4.66 mstats 一変数の統計量算出 . . . 220
4.67 msum 項目値の合計 . . . 222
4.68 msummary 1 変数の統計量の計算 . . . 223
4.69 mtab2csv TSV から CSV データへの変換 . . . 225
4.70 mtee 複数ファイルへのコピー . . . 227
4.71 mtonull NULL 値へ置換 . . . 229
4.72 mtra 縦型データをベクトル項目に変換 . . . 231
4.73 mtrafld クロス表をトランザクション項目に変換 . . . 233
4.74 mtraflg クロス表をトランザクション項目に変換 . . . 236
4.75 muniq レコードの単一化 . . . 238
4.76 mvcat ベクトルの併合 . . . 239
4.77 mvcommon ベクトル要素の参照選択 . . . 241
4.78 mvcount ベクトルサイズの計算 . . . 243
4.79 mvdelim ベクトル要素の区切り文字変更 . . . 244
4.80 mvdelnull ベクトルの NULL 要素の削除 . . . 246
4.81 mvjoin ベクトル要素の参照結合 . . . 248
4.82 mvnullto ベクトル要素の NULL 置換 . . . 250
4.83 mvreplace ベクトル要素の参照置換 . . . 252
4.84 mvsort ベクトル要素のソート . . . 254
4.85 mvuniq ベクトル要素の単一化 . . . 256
4.86 mwindow スライド窓の生成 . . . 257
4.87 mxml2csv xml から csv への変換 . . . 260

第 5 章 mcal 263
5.1 mcal 項目間演算 . . . 264
5.2 式の構成要素 . . . 265
5.3 定数 . . . 265
5.4 項目値 . . . 265
5.5 ワイルドカード . . . 266
5.6 前行の項目値 . . . 266
5.7 前行の結果項目値 . . . 266
5.8 算術演算子 . . . 267
5.9 比較演算子 . . . 267
5.10 論理演算子 . . . 267
5.11 演算子の優先順位 . . . 267
5.12 関数 . . . 268
5.13 日付型と時刻型について . . . 268
5.14 abs 絶対値 . . . 273
5.15 acos コサインの逆関数 . . . 274
5.16 age 年令 . . . 275
5.17 and 論理積 . . . 276
5.18 argsize 引数の数 . . . 277
5.19 asin サインの逆関数 . . . 278

5.20 atan タンジェントの逆関数 . . . 279
5.21 atan2 座標の角度 . . . 280
5.22 avg 平均 . . . 281
5.23 binomdist 二項分布の累積確率 . . . 282
5.24 berrand ベルヌーイ乱数 . . . 283
5.25 bottom 終端行 . . . 284
5.26 capitalize 先頭文字大文字変換 . . . 285
5.27 cat 文字列併合 . . . 286
5.28 ceil 切り上げ . . . 287
5.29 cos コサイン . . . 289
5.30 cosh 双曲線余弦 . . . 290
5.31 countnull 合計 . . . 291
5.32 d2julian: date-ユリウス通日変更 . . . 292
5.33 day 日 . . . 293
5.34 date 年月日 . . . 294
5.35 degree 角度 . . . 295
5.36 diff 期間 . . . 296
5.37 dist 距離 . . . 298
5.38 distgps GPS 距離 . . . 299
5.39 dow 曜日 . . . 300
5.40 e ネイピア数 . . . 302
5.41 exp 指数関数 . . . 303
5.42 factorial 階乗 . . . 304
5.43 fixlen 固定長変換 . . . 305
5.44 fldsize 項目数 . . . 306
5.45 floor 切り捨て . . . 307
5.46 format 書式付き出力 . . . 309
5.47 fract 小数部 . . . 310
5.48 gcd 最大公約数 . . . 311
5.49 hasspace 空白類文字検索 . . . 312
5.50 heron 三角形の面積 . . . 313
5.51 hour 時 . . . 314
5.52 if 条件選択 . . . 315
5.53 int 整数部 . . . 317
5.54 match 検索 . . . 318
5.55 isnull NULL 値判定 . . . 320
5.56 julian ユリウス暦変換 . . . 321
5.57 lcm 最小公倍数 . . . 323
5.58 leapyear 閏年判定 . . . 324
5.59 left 先頭切り出し . . . 325
5.60 length 文字列長 . . . 326
5.61 line 行番号 . . . 327
5.62 ln 自然対数 . . . 328
5.63 log 対数 . . . 329
5.64 log10 常用対数 . . . 330

5.65 log2 底が 2 の対数 . . . 331
5.66 max 最大値 . . . 332
5.67 mid 部分文字列切り出し . . . 333
5.68 min 最小値 . . . 334
5.69 minute 分 . . . 335
5.70 month 月 . . . 336
5.71 not 否定 . . . 338
5.72 now 現在時刻 . . . 339
5.73 null NULL 値 . . . 340
5.74 nrand 正規乱数 . . . 341
5.75 or 論理和 . . . 342
5.76 pi 円周率 . . . 343
5.77 power 累乗 . . . 344
5.78 product 積 . . . 345
5.79 radian ラジアン . . . 346
5.80 rand 一様乱数 . . . 347
5.81 randi 整数一様乱数 . . . 348
5.82 regexlen マッチ文字数 . . . 349
5.83 regexm 全体マッチ . . . 351
5.84 regexpfx マッチ文字列のプレフィックス . . . 352
5.85 regexpos マッチ位置 . . . 353
5.86 regexrep マッチ文字列の置換 . . . 355
5.87 regexs マッチ . . . 356
5.88 regexsfx マッチ文字列のサフィックス . . . 357
5.89 regexstr マッチ文字列 . . . 358
5.90 right 末尾切り出し . . . 359
5.91 round 四捨五入 . . . 360
5.92 second 秒 . . . 362
5.93 sign 符号 . . . 363
5.94 sin サイン . . . 364
5.95 sinh 双曲線正弦 . . . 365
5.96 sqrt 平方根 . . . 366
5.97 sqsum 平方和 . . . 367
5.98 sum 合計 . . . 368
5.99 t2julian(日時) . . . 369
5.100 tan タンジェント . . . 371
5.101 tanh 双曲線逆正接 . . . 372
5.102 time 時分秒 . . . 373
5.103 today 本日の日付 . . . 374
5.104 tolower 小文字変換 . . . 375
5.105 top 先頭行 . . . 376
5.106 toupper 大文字変換 . . . 377
5.107 tseconds 経過秒数 . . . 378
5.108 uxt UNIX 時刻変換 . . . 379
5.109 week 週 . . . 381

5.110 xor 排他的論理和 383
5.111 year 西暦年 384
5.112 型変換 385

索引 386

第 1 章

変更点

1.1 ver.2 での変更点

1.1.1 自動並べ替え機能の追加

MCMD Ver. 1.0 では、たとえば msum コマンドでキー項目を指定して集計する場合や、mjoin コマンドで複数のファイルをキー項目で結合する場合、事前に msortf コマンドを使用してキー項目を並べ替えておく必要があった。

多くのコマンドで事前の並べ替えが必要であり、また並べ替えを忘れてもエラーになることなく誤った結果を出力して処理を終えるため、スクリプトにバグを生みやすいという難点があった。そこで、k=オプションでキー項目を指定するすべてのコマンドで、項目の並べ替えを自動的に行う機能を追加したのが Ver. 2.0 の最大の変更点である。どの項目で並べ替えられたのかが CSV ヘッダの項目名にも追加されるため（詳細は項目の並べ替え情報参照)、各コマンドは並べ替えが必要であるときのみ並べ替えを行う。

Ver. 1.0 での実行例

msum k=**顧客** コマンドを実行する前に、msortf コマンドで**顧客**項目を並べ替えておく必要があった。

```
$ more dat1.csv
顧客,金額
A,10
B,10
A,20
B,15
B,20
$ msortf i=dat1.csv f=顧客 | msum k=顧客 f=金額:金額合計 o=rsl1.csv
#END# kgsortf f=顧客 i=dat1.csv
#END# kgsum f=金額:金額合計 k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客,金額合計
A,30
B,45
```

Ver. 2.0 での実行例

msum コマンドが並べ替えの要否を判断し、必要であれば自身で並べ替えるため、msortf コマンドを省略しても正しい結果が得られる。msum コマンドが**顧客**項目で並べ替えた証左として、項目名に %0 が付加されている。

```
$ more dat1.csv
顧客,金額
A,10
B,10
A,20
B,15
B,20
$ msum i=dat1.csv k=顧客 f=金額:金額合計 o=rsl1.csv
#END# kgsum f=金額:金額合計 k=顧客 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
顧客 %0,金額合計
A,30
B,45
```

1.1.2 仕様が変更されたコマンド

表 1.1 に、Ver. 1.0 から仕様が変更されたコマンドを示す (k=パラメータに自動並べ替え機能が追加されたのみのコマンドを除く)。行の順序が処理結果に影響するコマンドに s=パラメータが追加されたほか、いずれも行の並べ替えに

関する仕様の変更である。

表 1.1 仕様が変更されたコマンド

コマンド	説明	変更内容
maccum	累積計算	s=パラメータを追加 (必須)
mbest	指定行の選択	s=パラメータを追加 (必須)
mcombi	組合せ計算	s=パラメータを追加
mkeybreak	キーブレイク箇所	s=パラメータを追加
mmvavg	移動平均の算出	s=パラメータを追加 (必須)
mmvsim	移動窓の類似度計算	s=パラメータを追加 (必須)
mmvstats	移動窓の統計量の計算	s=パラメータを追加 (必須)
mnumber	連番	-B 使用時以外は s=パラメータ必須に
mpadding	行補完コマンド	type=パラメータがなくなり、 補完方法は f=で指定するように
mrjoin	参照ファイルの範囲条件による結合	r=パラメータで並べ替え方法を指定するように
mslide	行ずらし	s=パラメータを追加 (必須)
mtra	縦型データをベクトル項目に変換	s=パラメータを追加
mwindow	スライド窓の生成	wk=パラメータで並べ替え方法を指定するように

1.1.3 従来のスクリプトを実行するには

-q オプションを使用すると、k=パラメータで指定した項目の自動並べ替えが無効化される。表 1.1 で s=パラメータが必須とされたコマンドで s=パラメータを省略できるようにもなるため、各コマンドは Ver. 1.0 と同等の動作をする。よって、Ver. 1.0 で使用していたスクリプトについては、k=パラメータが指定された全コマンドに-q オプションを付加することで Ver. 2.0 でも同等の結果が得られる (ただし mpadding コマンドについては、type=パラメータがなくなっているので、f=パラメータでの指定に変更する必要がある)。

修正前のスクリプト

Ver. 2.0 環境で実行すると、maccum コマンドに s=パラメータがないというエラーが出る。

```
msortf i=customer.csv f=custID,date |
maccum k=custID f=amount o=accum.csv
#ERROR# parameter s= is mandatory without -q (kgaccum); kgaccum f=amount i=test.csv k=custID
```

修正例

maccum コマンドは s=パラメータが必須となったが、-q オプションを追加することで Ver. 1.0 と同等の使い方ができる。

```
msortf i=customer.csv f=custID,date |
maccum -q k=custID f=amount o=accum.csv
```

1.2 ver.3 での変更点

MCMD を NYSOL パッケージから独立させ GitHuB に登録したことにより、バージョンを 2.x から 3.0 へ上げた。コマンド仕様の大きな変更はなく、以下に示す細かな点が改善されている。また入力系コマンドを始めとして幾つかの新たなコマンドが追加されている。

1.2.1 -params パラメータの追加

全コマンド共通で-params パラメータを追加した。主にシステム利用を目的としたもので、指定できるパラメータの一覧を標準出力に出力する。

```
$ mcut -params
-assert_diffSize
-assert_nullin
-nfn
-nfni
-nfno
-params
-r
-x
f=
i=
o=
precision=
tmpPath=
```

1.2.2 環境変数 KG_msgTimebyfsec 追加

この環境変数を true に設定すると終了メッセージの時刻がマイクロ秒単位で表示される。

```
$ export KG_msgTimebyfsec=true
```

1.2.3 mchkcsv の機能強化

- UTF BOM を自動的に除去する。
- -diag で内容が英語で出力され、-diagl で日本語で出力されるように変更。

1.2.4 mcal の機能強化

- c=,a=に複数指定できるように変更。
- `$t{},#t{},0t` でマイクロ秒までを扱えるように変更。小数点 (6 桁) でマイクロ秒を表現している。それに伴い、`diffusecond(T,T),tuseconds(T),usecond(T),useconds(T),unow()` 関数を追加。
- `if(B,B,B)` のように、ブール値を返す使い方ができるようにする。

1.2.5 mcat の機能強化

- `-skip_zero` を指定することで、`-nfn` を指定していない場合でも 0 バイトファイルでエラーにならない。
- `flist=`を指定することで、併合するファイルリストを CSV データとして指定することができるようにした。flist=fileName:fldName で指定する。
- `kv=`を指定することで、パス名に含めた”key-value”の文字列を抜き出し項目名とその値としてデータに付加する機能を追加。

1.2.6 追加コマンド

以下に示す画面入力系のコマンドが 5 つ追加された。ただし、これらのコマンドは開発バージョンであり、今後仕様が変更される可能性がある。

- minput :入力画面を表示する。
- mminput： 複数入力枠による入力画面を表示する。
- mdsp： 画面の指定位置に文字列を表示する。
- mseldsp： 画面に単一選択入力窓を表示する。
- mmseldsp： 画面に複数選択入力窓を表示する。

その他にも以下に示すコマンドが追加された。

- mshuffle :ハッシュキーによるファイル分割
- mcsvconv： CSV を多様なフォーマットに変換
- mcsv2json： CSV を json に変換

第 2 章

M コマンド

2.1　M コマンドとは

M コマンド (MCMD とも表記する) とは、大規模表構造データ (CSV) を高速に処理する目的で開発されたコマンド群である。M コマンドの M は、発案者である松田康之氏のイニシャルに拠っている。M コマンドは、単一の機能 (例えば、並べ替えや表の結合など) に特化したコマンドを約 80 種類提供している。全てのコマンドは共通して、標準入力から CSV データを読み込み、標準出力に結果を書き込むだけの非常にシンプルな処理方式に従っている。そして単一の機能を持ったコマンドをパイプによって接続することで多様な処理を実現する。M コマンドを使えば、標準的な PC であっても、数億件規模のデータ処理が可能である。M コマンドの基本的な利用方法の習得にはさほど時間は必要とせず、集中的に取り組めば数週間でかなり自由にデータ加工ができるようになる。

2.2 インストール

2.2.1 要件

MCMD は、Linux, MacOSX, Bash on Ubuntu on Windows といった、代表的な OS での動作確認はとれている。その他にも、UNIX 系の OS であれば上記のツールさえあれば利用できるであろう。以下に動作が確認できている OS およびそのバージョン一覧を示す。

- MacOS 10.9.5(Marverics) 以上
- CentOS 7.3 1611
- Ubuntu 16.04 LTS
- Bash on Ubuntu on Windows(Windows 10)

ソースコードのコンパイルには以下に示すツールが必要となる。

- c++ コンパイラ
- autotools
- boost C++ Library
- lib2xml Library

2.2.2 コンパイル＆インストール

MacOS X

```
## autotools と boost を導入していなければ brew 等でインストールする。
$ brew install autoconf
$ brew install automake
$ brew install libtool
$ brew install boost

## ソースのダウンロード
$ git clone https://github.com/nysol/mcmd.git
$ cd mcmd

## コンパイルとインストール
$ aclocal
$ autoreconf -i
$ automake --add-mising
$ ./configure
$ make
$ sudo make install
```

CentOS

```
## autotools,boost,libxml2 を導入していなければ yum 等でインストールする。
$ sudo yum update
$ sudo yum groupinstall "Development Tools"
$ sudo yum install boost-devel
$ sudo yum install libxml2-devel

## ソースのダウンロード
$ git clone https://github.com/nysol/mcmd.git
$ cd mcmd
```

```
## コンパイルとインストール
$ aclocal
$ autoreconf -i
$ ./configure
$ make
$ sudo make install
```

Ubuntu,Bash on Windows

```
## autotools,boost,libxml2 など必要なツールを導入していなければ apt-get 等でインストールする。
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential
$ sudo apt-get install libtool
$ sudo apt-get install automake
$ sudo apt-get install git
$ sudo apt-get install libboost-all-dev
$ sudo apt-get install libxml2-dev

## ソースのダウンロード
$ git clone https://github.com/nysol/mcmd.git
$ cd mcmd/

## コンパイルとインストール
$ aclocal
$ autoreconf -i
$ ./configure
$ make
$ sudo make install

# 共有ライブラリのパス設定
# 起動時に毎回設定するのであれば.bash_profile に記載しておく。
$ export LD_LIBRARY_PATH=/usr/local/lib
```

2.2.3　インストール完了の確認

各コマンドの利用方法については--help オプションを指定することで簡単なヘルプが表示される (図 2.1)。以下のようにコマンドヘルプが表示されたらインストールは完了している。Ver.2.4 からは--help オプションのデフォルトが英語に変更になった為、日本語で出力したい場合は、--helpl オプションを指定すると今まで通りに日本語の簡単なヘルプが出力される (図 2.2)。また、環境変数に KG_LOCALHELP=true を設定しておくと、--help で日本語を出力をデフォルトにすることができる。

また--version にて MCMD のバージョンが表示される (図 2.3)。このバージョンはコマンド毎のバージョンではなく、MCMD 全体のバージョンであることに注意する。したがって、全てのコマンドで同じバージョンが表示される。

2.2.4　インストールが出来ない場合の対処法

Anaconda をインストールした OS では一部、インストールが出来ない問題が確認されおり、コマンド検索 PATH の順序を変更することでエラーを回避できることが確認されている。変更を行った後、各 OS で記載されている手順に従ってインストールする。

```
$ mcut --help

MCUT - SELECT COLUMN

Extract the specified column(s). The specified column is removed with -r
option.

Format

mcut f= [-r] [-nfni] [i=] [o=] [-assert_diffSize] [-assert_nullin]
[-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
                         :
                         :
```

図 2.1 ヘルプの英語表示 (デフォルト)

```
$ mcut --helpl

mcut 項目の選択
===============
指定した項目を選択する。
-r オプションを付けると、指定した項目を削除する。

書式
----
mcut f= [-r] [i=] [o=] [-nfn] [-nfno] [-x] [--help]  [--helpl] [--version]

パラメータ
----------
  f=      抜き出す項目名
          項目名をコロンで区切ることで、出力項目名を変更することができる。
          ex. f=a:A,b:B
  -r      項目削除スイッチ
          f=で指定した項目を削除し、それ以外の項目が抜き出される。
  -nfni   入力データの１行目を項目名行とみなさない。よって項目番号で指定しなければならない。
          以下のように、新項目名を組み合わせて指定することで項目名行を追加出力することが可能となる。
          例）f=0:日付,2:店,3:数量
                  :
                  :
```

図 2.2 ヘルプの日本語表示

```
$ mcut --version
lib Version 1:1:0:0
```

図 2.3 バージョンの表示

```
# ~/.bash_profile の PATH 設定を以下の通り修正する。
# 修正前）
export PATH="/Users/username/anaconda/bin:$PATH"

# 修正後）
export PATH="$PATH:/Users/username/anaconda/bin"
```

2.3　はじめよう

簡単な例から始めよう。まずは CSV データがなければ始まらない。MCMD では、多くの種類のデータセットを出力するための mdata コマンドを備えている。図 2.4 にその利用例を示す。$で始まる行はコマンドラインでの入力を表し、その下に実行結果が示されている。mdata コマンドは、データセットの種類を引数にとり、標準出力に内容を出力する (詳細は mdata の章を参照)。以下の例では、標準出力の内容をリダイレクトして man0.csv という名前のファイルに出力している。

```
$ mdata -man0 >man0.csv
#END# kgData -man0
$ more man0.csv
顧客,金額
A,5200
B,4000
B,3500
A,2000
B,800
```

図 2.4　mdata コマンドによるデータの生成

MCMD の全てのコマンドは、終了すれば#END#から始まるメッセージを出力する（エラー終了すると#ERROR#から始まるメッセージが出力される)。また、more はファイルの内容をページ送りで表示するコマンドであり、データの内容を示すために利用している。本マニュアルで用いる例は、全て以上の形式に従って記述されている。

man0.csv データを利用して、金額を顧客別に合計する例を図 2.5 に示す。msum コマンドでは、man0.csv データを読み込み (i=)、顧客項目を集計キーにして (k=)、金額項目を合計し (f=)、結果を出力ファイル output.csv に書き込んでいる (o=)。出力された CSV の項目名「顧客」に %0 が付いているが、これは msum が顧客項目で並べ替えたことを示すものであり項目名の一部ではなく、今のところは特に気にする必要はない。

```
$ msum k=顧客 f=金額 i=man0.csv o=output.csv
#END# kgsum f=金額 i=man0.csv k=顧客 o=output.csv
$ more output.csv
顧客 %0,金額
A,7200
B,8300
```

図 2.5　顧客別金額合計

もう少し複雑な例を示しておこう。図 2.6 に示された例では、顧客別にどのような商品を何個購入したか、マトリックス形式で集計する。#で始まる行はコメントであり入力する必要はない。

mcut コマンドは指定した項目を切り出すだけの機能を持つコマンドで、mcount コマンドは行数をカウントするコマンドである。そして mcross コマンドはクロス集計を行う。ここでは、各コマンドの詳細な動きよりも、入力データが各コマンドによってどのように加工されていくかその流れを確認されたい。以上のように、M コマンドでは 80 種類以上のコマンドを組み合わせる事で多様なデータ加工を実現するのである。

以上の例では、各コマンドの結果をワークファイルファイル (xxa,xxb,xxc) に出力したが、パイプ (|) で連結することで、以下のようにワークファイルを使うことなく実行することも可能である (図 2.7)。パイプにより、あるコマンドの出力結果は次のコマンドの入力へと次々に受け渡されていく。

```
$ mdata -man1 >man1.csv
#END# kgData -man1
$ more man1.csv
顧客,日付,商品
A,20130916,a
A,20130916,c
A,20130917,a
A,20130917,e
B,20130916,d
B,20130917,a
B,20130917,d
B,20130917,f
$ mcut f=顧客,商品 i=man1.csv o=xxa
#END# kgcut f=顧客,商品 i=man1.csv o=xxa
$ more xxa
顧客,商品
A,a
A,c
A,a
A,e
B,d
B,a
B,d
B,f
# 顧客商品別に行数をカウントする。
$ mcount k=顧客,商品 a=件数 i=xxa o=xxb
#END# kgcount a=件数 i=xxa k=顧客,商品 o=xxb
$ more xxb
顧客%0,商品%1,件数
A,a,2
A,c,1
A,e,1
B,a,1
B,d,2
B,f,1
# 商品を項目にしたクロス集計を実行。購入されていない商品の個数は0にしている。
$ mcross k=顧客 s=商品 f=件数 v=0 i=xxb o=xxc
#END# kgcross f=件数 i=xxb k=顧客 o=xxc s=商品 v=0
$ more xxc
顧客%0,fld,a,c,d,e,f
A,件数,2,1,0,1,0
B,件数,1,0,2,0,1
# 余分な項目"fld"を除いている。
$ mcut f=fld -r i=xxc o=output.csv
#END# kgcut -r f=fld i=xxc o=output.csv
$ more output.csv
顧客%0,a,c,d,e,f
A,2,1,0,1,0
B,1,0,2,0,1
```

図 2.6 顧客別商品購入数量マトリックス

```
$ mdata -man1 | mcut f=顧客,商品 | mcount k=顧客,商品 a=件数 | mcross k=顧客 s=商品 f=件数 v=0 |
mcut f=fld -r o=output.csv
#END# mdata -man1
#END# kgcut f=顧客,商品
#END# kgcount a=件数 k=顧客,商品
#END# kgcross f=件数 k=顧客 s=商品 v=0
#END# kgcut -r f=fld o=output.csv
$ more output.csv
顧客 %0,a,c,d,e,f
A,2,1,0,1,0
B,1,0,2,0,1
```

図 2.7　パイプで連結した例

2.4 CSV

MCMD が処理する表構造データは図 2.8 に例示されるような CSV(Comma Separated Values) フォーマットである。CSV は表構造データのフォーマットのデファクトスタンダードであり、アプリケーションプログラム間でのデータ交換用フォーマットとして 広く利用されている。

```
商品コード, 商品名, 分類, 価格
0899781, パン, 食品,128
8879674, オレンジジュース, 飲料,98
3244565, チーズ, 食品,350
6711298, 茶碗, 食器,168
```

図 2.8 CSV データの例

しかしながら、CSV は標準化協会や企業主導で作成された標準フォーマットではなく、 それ故にベンダー毎に CSV の扱い方法が異なっているのが現状である。その中で 2005 年 10 月にインターネット標準である RFC4180 として CSV フォーマットが 提案されたのは注目すべき動きである。図 2.9、RFC4180 の中で ABNF 表現による CSV の定義を示す。

```
(A) file = [header CRLF] record *(CRLF record) [CRLF]
(B) header = name *(COMMA name)
(C) record = field *(COMMA field)
(D) name = field
(E) field = (escaped / non-escaped)
(F) escaped = DQUOTE *(TEXTDATA / COMMA / CR / LF / 2DQUOTE) DQUOTE
(G) non-escaped = *TEXTDATA
(H) COMMA = %x2C
(I) CR = %x0D ;as per section 6.1 of RFC 2234 [2]
(J) DQUOTE = %x22 ;as per section 6.1 of RFC 2234 [2]
(K) LF = %x0A ;as per section 6.1 of RFC 2234 [2]
(L) CRLF = CR LF ;as per section 6.1 of RFC 2234 [2]
(M) TEXTDATA = %x20-21 / %x23-2B / %x2D-7E
日本語翻訳：
```

図 2.9 CSV の ABNF による定義

図 2.9 中の各行の意味は以下のとおりである。

- (A) ファイル (file) は，ヘッダ (header) と 1 行以上のレコード (record) から構成される。ヘッダはなくてもよい。ヘッダとレコードの末尾には改行 (CRLF) が付く。最終レコードの改行 (CRLF) は任意である。
- (B) ヘッダ (header) は 1 つ以上の名前 (name) で構成され，カンマ (COMMA) で区切られる。
- (C) レコード (record) は一つ以上の項目 (field) で構成されており，
- (D) 名前 (name) は項目 (field) である。
- (E) 項目 (field) はエスケープ (escaped) か，非エスケープ (non-escaped) のいずれかである。
- (F) エスケープ (escaped) は，ダブルクォーツで囲まれた 0 個以上のテキスト文字 (TEXTDATA)，カンマ (COMMA)，改行文字 (CR もしくは LF)，もしくは 2 つの連続したダブルクォーツである。
- (G) 非エスケープ (non-escaped) は 0 個以上のテキスト文字 (TEXTDATA) である。
- (H) コンマは 16 進数アスキーコード 2C である。
- (I) キャリッジリターン (CR) は 16 進数アスキーコード 0D である。
- (J) ダブルクォーツ (DQUOTE) は 16 進数アスキーコード 22 である。
- (K) ラインフィード (LF) は 16 進数アスキーコード 0A である。

- (L) 改行ラインフィードはキャリッジリターン + ラインフィードである。
- (M) テキスト文字 (TEXTDATA) は 16 進数アスキーコードで 20〜21，23〜2B，もしくは 2D〜7E である。

2.4.1　M コマンド独自の定義

M コマンドでは上記の CSV の定義に対して以下の制約を追加している。

- 項目数は全行同じでなければならない。
- 1 行の最大長に制限を設ける (デフォルトでは 1024000 バイト (1MB),10MB まで拡張可能, ただしソースコード kgConfig.h の `KG_LimitRecLen` の値を変更すれば拡張可能)。
- 改行は LF のみとする。
- 最終レコードであっても改行は必須とする。
- テキスト文字として 80〜FF を付け加える (マルチバイト文字を扱うため)。

利用する CSV ファイルが上記の定義を満たしているかどうかを確かめるには mchkcsv コマンドを利用すればよい。

2.4.2　コマンドに共通した入出力の手順

多くの MCMD で行われる CSV ファイルの入出力の手順は概ね以下に示す手順に従っている。

1. ファイルからブロック単位でメモリに読み込む。
2. エスケープ文字を考慮しながらカンマ区切りの項目単位の文字列に分解する。
3. エスケープ文字を解釈しオリジナルデータに変換する (DQUOTE を外す)。
4. コマンドの特有の処理を実行し，出力バッファに書き込む。
5. エスケープが必要であれば，エスケープを付加する。
6. バッファが一杯になればファイルに出力する。

2.4.3　項目の並べ替え情報

msortf コマンドを使用して項目の並べ替えを行った場合や、他のコマンドで `s=`パラメータや `k=`パラメータを使用した場合、指定の項目でデータが並べ替えられる。その時、出力される CSV ファイルの項目名には、並べ替えに関する情報が付加される。具体的には、並べ替え項目として指定した順に `%0`、`%1`、. . . と 0 から始まる番号が付加される。数値として並べ替えたときには `n` が、降順で並べ替えたときには `r` がそれぞれ追加される。これらの付加情報はコマンドが効率的な処理のために内部的に利用するものであり、ユーザが項目名の一部として指定してはならない。

```
$ cat dat1.csv
id,item,price
2,b2,200
1,a1,100
2,b1,120
$ msortf f=id,price%n i=dat1.csv
id%0,item,price%1n
1,a1,100
2,b1,120
2,b2,200
$ msum k=id f=price i=dat1.csv
id%0,item,price
1,a1,100
2,b1,320
```

なお、これらの付加情報を削除したければ、以下に示すように `mfldname -q` を実行すればよい。

```
$ cat dat2.csv
id%0,item,price%1n
1,a1,100
2,b1,120
2,b2,200
$ mfldname i=dat2.csv
id,item,price
1,a1,100
2,b1,120
2,b2,200
```

なお、項目名の並べ替え情報と実際のデータの並び順が整合していない場合（実際には並べ替えられていないのに、項目名に %0 が付加されているなど)、各コマンドの動作は不定となる。

2.4.4 注意点

以下に CSV データで注意すべき点について、例を交えながら説明する。

カンマを含むデータ

カンマを含むデータはダブルクォーツで囲われる。以下は、f1,f2 の 2 項目から構成される CSV ファイルで、0 行目[*1]の f1 項目はカンマを含んでいるのでダブルクォーツで囲われている。

```
f1,f2
"abc,def",2
xyz,2
```

ダブルクォーツを含むデータ

ダブルクォーツを含むデータはダブルクォーツで囲われ，かつ連続するダブルクォーツとして表現される。以下は、f1,f2 の 2 項目から構成される CSV ファイルで、0 行目と 1 行目の f1 項目はダブルクォーツを含んでおり、オリジナルのデータはそれぞれ「abc"def」,「"」である。

```
f1,f2
"abc""def",2
""""",2
```

改行を含むデータ

改行を含むデータもダブルクオーツで囲うことで処理可能である。以下の例の 0 行目の f1 項目は、abc の後に改行が含まれているが、ダブルクオーツで囲われているため、行末ではなくデータの一部として識別される。

```
f1,f2
"abc
def",1
```

必要のないダブルクオーツ

以下のようにダブルクオーツで囲う必要ないデータをダブルクオーツで囲っていた場合、コマンドの出力時には外される。

[*1] MCMD では統一的に先頭行 (項目名行を除いた最初の行) を 0 行目と呼称する。

```
$ more data.csv
f1,f2
"abc",efg
abc,"efg"
$ mcut f=f1,f2 i=data.csv
f1,f2
abc,efg
abc,efg
```

2.5　データ型

MCMD で扱う CSV データはプレーンテキストであり、全てのデータは文字列で表されている。よってその文字列をどのようなデータ型として扱うかはコマンドによって決まる。例えば、`msum` コマンドで `f=`で指定した項目データは、コマンド内部で文字列から数値へと変換される。MCMD で扱うことのできる型は、表 2.1 に示される通り、数値型、文字列型、日付型、時刻型、論理型、ベクトル型の 6 つである。

表 2.1　MCMD が扱う 6 つのデータ型

データ型	CSV データ表記例	変換内容
数値型	10, 2.5, 1.5E+10	倍精度実数
文字列型	abc, あいう	文字列
日付型	20130920	日付オブジェクト [*1](グレゴリオ暦,8 桁固定長)
時刻型	20130920151154.123456	日付オブジェクト [*1](グレゴリオ暦,8 桁固定長)
		+POSIX 時刻 [*2](6 桁の時分秒 + 最大小数 6 桁のマイクロ秒)
	151154.123456	日付が指定されていない場合は、内部的には本日日付が付加される
論理型	1,0	”1”を真、”0”を偽の bool 値に変換する
ベクトル型	a c b, 1 5 11	スペースで区切られた文字列を、上記のいずれかのデータ型に変換したもの

[*1] boost ライブラリの boost::gregorian::date クラスを利用
[*2] boost ライブラリの boost::posix_time::ptime クラスを利用

また、表 2.2 に各データ型として扱う代表的なコマンドを示しておく。

表 2.2　各データ型を扱う代表的なコマンド

データ型	コマンド	内容
数値型	msum	数値項目の合計計算
	msim	2 つの項目の類似度計算
文字列型	mjoin	参照ファイルの項目の結合
	mcombi	組み合せの列挙
日付型	mcal の age 関数	年齢計算
	mcal の leapyear 関数	閏年の判定
時刻型	mcal の now 関数	現在時刻の出力 (秒単位)
	mcal の unow 関数	現在時刻の出力 (マイクロ秒単位)
	mcal の diffminute 関数	分単位での時刻差の計算
論理型	mcal の and 関数	論理積の計算
	mcal の if 関数	条件による値の設定
ベクトル型	mvsort	ベクトル要素の並べ替え
	mvuniq	ベクトル要素の単一化

2.6　項目の指定

MCMD では、CSV データの先頭行を項目名として扱うことも可能であるし、また項目名行がなくても項目番号で項目を指定することもできる。項目名行の扱いに関係するパラメータは-nfn,-nfno,-nfni,-x の４つある。以下では、例を示しながら、その利用方法について説明する。なお、項目番号は左から 0,1,2 のように 0 から始まることに注意する。

例 1: -nfn **指定**

-nfn(no field names) を指定すると，先頭行を項目名行と見なさない。そして項目は必ず番号で指定する (番号は 0 から始まることに注意する)。

```
$ more dat2.csv
a,2
b,5
b,4
$ msum -nfn k=0 f=1 i=dat2.csv o=rsl1.csv
#END# kgsum -nfn f=1 i=dat2.csv k=0 o=rsl1.csv
$ more rsl1.csv
a,2
b,9
```

例 2: -nfno **指定**

-nfno(no field names for output) を指定すると，入力データの先頭行は項目名行として扱うが，出力データには項目名を出力しない.

```
$ more dat1.csv
key,val
a,2
b,5
b,4
$ msum k=key f=val -nfno i=dat1.csv o=rsl2.csv
#END# kgsum -nfno f=val i=dat1.csv k=key o=rsl2.csv
$ more rsl2.csv
a,2
b,9
```

例 3: -nfni **指定**

-nfni(no field name for input) の指定は mcut でのみ可能であるオプションである．このオプションは-nfno と逆の働きをする．すなわち，入力データの先頭行は項目名行として扱わないが，出力データには項目名を出力する．よって、入力項目番号に続けて出力項目名を":"に続けて指定する必要がある.

```
$ mcut f=0:key,1:val -nfni i=dat2.csv o=rsl3.csv
#END# kgcut -nfni f=0:key,1:val i=dat2.csv o=rsl3.csv
$ more rsl3.csv
key,val
a,2
b,5
b,4
```

例 4: -x **指定**

項目名行がある CSV データに対して、項目番号で指定したい場合は-x オプションを利用する。

```
$ msum -x k=0 f=1 i=dat1.csv o=rsl4.csv
#END# kgsum -x f=1 i=dat1.csv k=0 o=rsl4.csv
$ more rsl4.csv
key%0,val
a,2
b,9
```

2.6.1 有効な項目名

項目名として利用可能な文字は以下の通りである。

- アルファベット (a-z,A-Z)
- 数字 (0-9)
- マルチバイト文字 (UTF-8 など)
- 記号

ただし、記号については、以下の 9 つの利用は避けることを推奨する。利用してエラーとなるわけではないが、いずれも MCMD の中で項目指定時の特殊文字として利用しておりその特殊用途を利用できなくなる可能性があるからである。

- , カンマ
- : コロン
- % パーセント
- * アスタリスク
- ? クエスチョンマーク
- & アンド
- \ バックスラッシュ
-] 四角括弧右
- [四角括弧左

2.6.2 有効な項目番号

項目番号の指定では、単純に項目番号をカンマで区切って列挙する以外にも、後ろの項目からの番号指定 ("L"を付ける) や範囲 (-) を指定することが可能である。例えば 0L とすれば、最後の項目を指定したことになり、2L とすれば、最後から数えて 2 番目の項目 (0 番から始まることに注意) を指定したことになる。また 0-5 と指定すれば、0 番項目から 5 番項目まで 6 つの項目を指定したことになる。すなわち 0,1,2,3,4,5 と指定したことと同等である。

例 1: **範囲指定**

「0-4」の指定により、「0,1,2,3,4」を指定したことになる。

```
$ more dat1.csv
ブランド,数量01,数量02,数量03,数量04,数量05,数量06,数量07,数量08,数量09,数量10
A,10,50,90,130,170,210,250,290,330,370
B,20,60,100,140,180,220,260,300,340,380
C,30,70,110,150,190,230,270,310,350,390
D,40,80,120,160,200,240,280,320,360,400
$ mcut -x f=0-4 i=dat1.csv o=rsl1.csv
#END# kgcut -x f=0-4 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
ブランド,数量01,数量02,数量03,数量04
A,10,50,90,130
```

```
B,20,60,100,140
C,30,70,110,150
D,40,80,120,160
```

例 2: **範囲指定逆順**

「4-0」の指定により、「4,3,2,1,0」を指定したことになる。

```
$ mcut -x f=4-0 i=dat1.csv o=rsl2.csv
#END# kgcut -x f=4-0 i=dat1.csv o=rsl2.csv
$ more rsl2.csv
数量 04, 数量 03, 数量 02, 数量 01, ブランド
130,90,50,10,A
140,100,60,20,B
150,110,70,30,C
160,120,80,40,D
```

例 3: **複数範囲指定**

「1-0,2-4」の指定により、「1,0,2,3,4」を指定したことになる。

```
$ mcut -x f=1-0,2-4 i=dat1.csv o=rsl3.csv
#END# kgcut -x f=1-0,2-4 i=dat1.csv o=rsl3.csv
$ more rsl3.csv
数量 01, ブランド, 数量 02, 数量 03, 数量 04
10,A,50,90,130
20,B,60,100,140
30,C,70,110,150
40,D,80,120,160
```

例 4: **末尾項目からの指定**

「2L」の指定により、項目の後ろから数えた 2 番項目 (数量 08) を指定したことになる。

```
$ mcut -x f=2L i=dat1.csv o=rsl4.csv
#END# kgcut -x f=2L i=dat1.csv o=rsl4.csv
$ more rsl4.csv
数量 08
290
300
310
320
```

例 5: **末尾項目からの指定と範囲指定**

「5-3L」の指定により、5 番項目〜後ろから 3 番目の項目、すなわち「5,6,7」を指定したことになる。

```
$ mcut -x f=5-3L i=dat1.csv o=rsl5.csv
#END# kgcut -x f=5-3L i=dat1.csv o=rsl5.csv
$ more rsl5.csv
数量 05, 数量 06, 数量 07
170,210,250
180,220,260
190,230,270
200,240,280
```

2.6.3 入力項目と出力項目

多くのコマンドで項目の指定には f=が利用される。f=の書式は、「入力項目:出力項目」で、出力項目名の指定を省略すれば、入力項目名が出力項目名として利用される。また、-x を指定することで f=0:**数量**のように、番号指定と混在させることも可能である。

例 1: **基本例**

「数量：売上数量」の指定により、項目名が「数量」から「売上数量」に変換されて出力される。

```
$ more dat1.csv
ブランド,数量
A,10
B,20
C,30
D,40
$ mcut f=ブランド,数量:売上数量 i=dat1.csv o=rsl1.csv
#END# kgcut f=ブランド,数量:売上数量 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
ブランド,売上数量
A,10
B,20
C,30
D,40
```

例 2: **追加項目名**

以下の maccum コマンドでは、「ブランド」項目で並べ替えた後、「数量」項目の値を累積し、「累積数量」項目として追加出力する。もし、「f=数量」とだけすれば、累積結果も「数量」という名の項目となり、オリジナルの「数量」項目とダブってしまいエラーとなる。

```
$ maccum s=ブランド f=数量:累積数量 i=dat1.csv o=rsl2.csv
#END# kgaccum f=数量:累積数量 i=dat1.csv o=rsl2.csv s=ブランド
$ more rsl2.csv
ブランド%0,数量,累積数量
A,10,10
B,20,30
C,30,60
D,40,100
$ maccum s=ブランド f=数量 i=dat1.csv o=rsl2.csv
#ERROR# same field name is specified: 数量 (kgaccum)
```

例 3: **番号指定との混在**

番号指定と出力項目名指定を混在させることも可能である。

```
$ mcut f=0,1:売上数量 -x i=dat1.csv o=rsl3.csv
#END# kgcut -x f=0,1:売上数量 i=dat1.csv o=rsl3.csv
$ more rsl3.csv
ブランド,売上数量
A,10
B,20
C,30
D,40
```

2.6.4 ワイルドカード

複数項目を指定する際にはには、項目名に"*"と"?"のワイルドカードを利用することができる。"*"は任意の長さの任意の文字列にマッチし、"?"は任意の 1 文字にマッチする。また、ワイルドカードの評価順は入力データ上の項目の並び順となることに注意する。例えば、入力データの項目の並びが、A5,A3,A4,A2,A1 であれば、f=A*は f=A5,A3,A4,A2,A1 と評価される。

例 1: **基本例**

「数量*」にて、「数量」で始まる項目名 (「数量 10」、「数量 11」、「数量 12」、「数量 123」) にマッチする。

```
$ more dat1.csv
ブランド,数量10,数量11,数量12,数量123
A,10,15,9,1
B,20,16,8,2
C,30,17,7,3
D,40,18,6,4
$ mcut f=数量* i=dat1.csv o=rsl1.csv
#END# kgcut f=数量* i=dat1.csv o=rsl1.csv
$ more rsl1.csv
数量10,数量11,数量12,数量123
10,15,9,1
20,16,8,2
30,17,7,3
40,18,6,4
```

例 2: ?**のワイルドカード**

「数量」で始まる項目名のうち、1 からはじまる任意の 1 文字にマッチする項目名が選択される。「数量 123」にはマッチしない。

```
$ mcut f=数量1? i=dat1.csv o=rsl2.csv
#END# kgcut f=数量1? i=dat1.csv o=rsl2.csv
$ more rsl2.csv
数量10,数量11,数量12
10,15,9
20,16,8
30,17,7
40,18,6
```

2.6.5 出力項目名の置換

出力項目名で指定された"&"は特殊な意味を持ち、入力項目名に置換される。例えば、f=abc:xx&xx では、出力項目名は xxabcxx に置換される。"&"は、出力項目名の任意の位置に指定することができ、またその指定数に制限はない。ただし、"&"記号は、シェルにおいて「バックグラウンド実行」と解釈されてしまうので、ダブルクォーツで囲うなどしてエスケープする必要がある。

例 1: **基本例**

ここでは、"&"が入力項目名である「**ブランド**」に置換され、「f=**ブランド**:**ブランドコード**」と指定したことと同等となる。

```
$ more dat1.csv
ブランド,数量10,数量11,数量12,数量123
A,10,15,9,1
```

```
B,20,16,8,2
C,30,17,7,3
D,40,18,6,4
$ mcut f="ブランド:&コード" i=dat1.csv o=rsl1.csv
#END# kgcut f=ブランド:&コード i=dat1.csv o=rsl1.csv
$ more rsl1.csv
ブランドコード
A
B
C
D
```

例 2: **ワイルドカードとの併用**

出力項目名指定における**売上**&の&が入力項目名 (例えば「数量 10」) に置き換わる。結果として、「**数量**」で始まる項目全てに対して「**売上**」を先頭に加えて出力することになる。

```
$ mcut f="ブランド,数量*:売上&" i=dat1.csv o=rsl2.csv
#END# kgcut f=ブランド,数量*:売上& i=dat1.csv o=rsl2.csv
$ more rsl2.csv
ブランド,売上数量 10,売上数量 11,売上数量 12,売上数量 123
A,10,15,9,1
B,20,16,8,2
C,30,17,7,3
D,40,18,6,4
```

2.6.6 集計処理コマンドについての注意点

集計処理コマンドでは、キー項目ごとに指定された項目が集計処理され、キー項目ごとに 1 行出力される際、指定した項目以外の項目はどのレコードが出力さるかについては不定である。例えば、msum コマンドで、顧客、日付、商品、金額という項目のデータを想定し、msum コマンドで顧客別に金額を集計するのに、`msum k=顧客 f=金額`と指定した場合、それ以外の日付、商品項目についてはどのレコードが出力されるか不定であることに注意する。

2.7　データ本体がない場合の動作

データ本体 (項目名行を除いたデータ) がないデータに対する動作は、項目名行ありの CSV データが入力の場合は、処理内容に応じた項目名が出力され、正常に終了する。一方で、項目名行なしの CSV データが入力の場合、データ本体がないということは 0 バイトファイルということになり、出力結果も 0 バイトファイルとなる。入力行数, 出力行数は共に 0 件である。

例 1: **項目名行ありデータ**

```
$ more dat1.csv
A,B,C
$ msetstr v="string" a=X i=dat1.csv o=rsl1.csv
#END# kgsetstr a=X i=dat1.csv o=rsl1.csv v=string
$ more rsl1.csv
A,B,C,X
```

例 2: **項目名行なしデータ**

```
$ more dat2.csv
$ msetstr v="string" -nfn i=dat2.csv o=rsl2.csv
#END# kgsetstr -nfn i=dat2.csv o=rsl2.csv v=string
$ more rsl2.csv
```

2.8 マルチバイト文字

MCMD が扱う漢字等のマルチバイト文字は基本的には UTF-8 を前提としている。SHIFT_JIS 等、異なるエンコーディングによるマルチバイト文字でも運用は可能であるが、一部の機能は正しく動作しないであろう。以下ではマルチバイト文字の扱いについての MCMD での処理方式について説明する。

MCMD では処理速度を重視する観点から、漢字コードはマルチバイト文字のまま扱っているために、エンコーディングによっては、文字列検索や置換の処理で思わぬ結果がもたらされることがある。例えば、SHIFT_JIS で「陰」は 0x8941 であるが、これは 2 バイト目がシングルバイト文字の「A」にあたる。そのため「陰」に対して「A」を「B」に置換する処理を付すと「隠」(0x8942) に変換されてしまう。UTF-8 ではこのような問題が起こらないようなコード体系を採用している。さらにマルチバイト文字と ASCII 文字が混在した文字列において文字数をカウントすることは、たとえ UTF-8 であろうと非常に困難である。

このような問題を避ける最良の方法は、ASCII コードも含めて全ての文字を固定長に変換してしまうことである。これがワイド文字と呼ばれるものである (MCMD では 32bit 固定長を採用)。ワイド文字への変換には、マルチバイト文字のエンコーディング方式が分かっている必要がある。変換プログラムは、環境変数 `LANG` に設定された値によって、その方式を識別している。環境変数は以下のように確認すればよい。

```
$ echo $LANG
ja_JP.UTF-8
```

MCMD の中の一部のコマンドは、データ処理に先立ち入力データを全てワイド文字に変換してから処理するオプション (`-W`) が提供されている。対応しているコマンド一覧を表 2.3 に示す。これらのコマンドは検索もしくは置換の機能であり、エンコーディングが UTF-8 であれば利用する必要はない。

表 2.3 ワイド文字への変換処理機能を持つコマンド一覧

コマンド名	機能	説明
mchgstr	置換	-W を指定することで f=で指定した項目データは内部でワイド文字に変換される。
mselstr	検索	部分文字列マッチング (-sub) を行う場合、
		-W を指定することで f=で指定した項目データは内部でワイド文字に変換される。
msed	置換	-W を指定することで f=で指定した項目データは内部でワイド文字に変換される。
mtonull	検索	部分文字列マッチング (-sub) を行う場合、
		-W を指定することで f=で指定した項目データは内部でワイド文字に変換される。

また、mcal、msel についてはワイド文字を扱う関数が用意されている (表 2.4)。これらの中で、`lengthw` などの文字数をカウントしたり文字位置を計算する関数においては、エンコーディングが UTF-8 であっても利用する必要がある。

その他、ワイド文字の扱いについては、以下の点に注意されたい。

- ワイド文字への変換オーバーヘッドがかかるため処理速度は犠牲になる。
- 項目名については例外なくワイド文字に変換して処理を行っているために問題とはならない。
- ファイル名はマルチバイト文字のまま扱っている。

表 2.4　ワイド文字への変換処理機能を持つ mcal 関数一覧

関数名	機能	説明
lengthw	文字数	対象文字列を処理前にワイド文字に変換する。
midw	部分文字列	〃
rightw	部分文字列	〃
leftw	部分文字列	〃
regexsw	正規表現マッチ	〃
regexmw	正規表現マッチ	〃
regexrepw	正規表現置換	〃
regexlenw	正規表現マッチ長	〃
regexposw	正規表現マッチ位置	〃
regexstrw	正規表現マッチ部分文字列	〃
regexpfxw	正規表現マッチプレフィックス	〃
regexsfxw	正規表現マッチサフィックス	〃

2.9 パラメータ指定

M コマンドで使われるパラメータの書式は、一般的な UNIX コマンドのものとは少し異なる。値を伴うパラメータは「**キーワード=値**」のように、キーワードと値をイコール記号で区切って指定する。またオプション型のパラメータは、「**-キーワード**」のように、先頭にマイナス記号一つを付ける。

また M コマンドで使われるパラメータには、概ね共通した意味で用いられるものが多い。以下では、それらのパラメータについて解説する。ただし、コマンドによっては全く異なる意味として実装されているケースもあるので注意されたい。

キーワード	意味
i=	入力ファイル名
o=	出力ファイル名
f=	入出力項目名
k=	キー項目名
s=	並べ替え項目名
a=	追加項目名
-nfn	項目名行のない CSV
-nfno	項目名行を出力しない
-x	項目番号による指定
-q	自動並べ替えの無効化
[-assert_diffSize]	入力、出力件数比較を行う
[-assert_nullkey]	キー項目に NULL 値が含まれているかどうかのチェックを行う
[-assert_nullin]	入力項目の f=または vf=で指定された項目の NULL 値チェックを行う
[-assert_nullout]	出力項目の NULL 値チェックを行う
precision=	有効桁数
tmpPath=	作業ファイル格納パス名
delim=	ベクトル型データの区切り文字
bufcount=	バッファの数
--help	英語ヘルプを表示する
--helpl	日本語ヘルプを表示する

2.9.1 i= 入力ファイル名

入力ファイル名を指定する。多くのコマンドでは単一のファイルのみ指定可能で、例外として mcat コマンドは複数のファイル名をカンマで区切って指定できる。また入力データを必要としないコマンド、例えば、mnewrand や mnewnumber などもある。

このパラメータが省略された時には標準入力からデータを読み込む。この機能があるために、パイプラインによる接続が可能となる。例えば、以下の例では、msum で i=を指定していないが、これは msortf の結果がパイプラインを介して標準入力としてデータが入力されるためである。

```
$ msortf f=a i=dat.csv | msum k=a f=b o=rsl.csv
```

また、上記と同様の処理を行うに当たって、気づきにくい間違いを以下に示そう。上記との違いは msum に i=パラメータが指定されている点である。この例は残念ながらエラーとはならない。msortf の結果は標準出力に出力され、msum は dat.csv から読み込んで処理を行う。よって、msortf を実行している意味が全くなくなっており、得られる結果も異なったものとなるであろう。

```
$ msortf f=a i=dat.csv | msum k=a f=b i=dat.csv o=rsl.csv
```

利用例

例 1: **基本例**

dat1.csv を入力データとして mcut は実行される。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
$ mcut f=顧客,金額 i=dat1.csv o=rsl1.csv
#END# kgcut f=顧客,金額 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
顧客,金額
A,10
A,20
```

例 2: **出力項目名の指定**

標準入力をリダイレクト ("<"記号) して読み込む。

```
$ mcut f=顧客,金額 o=rsl2.csv <dat1.csv
#END# kgcut f=顧客,金額 o=rsl2.csv
$ more rsl2.csv
顧客,金額
A,10
A,20
```

対応コマンド

mnewnumber,mnewrand など一部のコマンドを除いて全てのコマンドで利用できる。

2.9.2　o= 出力ファイル名

出力ファイル名を指定する。単一のファイルのみ指定可能である。ただし、例外として、mtee コマンドは複数の出力ファイルを指定でき、また出力データを必要としないコマンド、例えば、msep などもある。

このパラメータが省略された時には標準出力にデータを書き込む。この機能があるために、パイプラインによる接続が可能となる。例えば、以下の例では、msortf で o=を指定していないが、これは msortf の結果が標準出力を通じてパイプラインに出力されるためである。

```
$ msortf f=a i=dat.csv | msum k=a f=b o=rsl.csv
```

また、上記の似たような処理ではあるが、以下に示した例はうまく動作しない。上記との違いは msortf に o=パラメータが指定されている点である。msortf の結果は tmp.csv に出力されるが、標準出力に出力するデータがなく、パイプ接続された msum はいつまでも入力データが来るのを待つこととなり、一見動いているように見えて、いつまでも終了しない。

```
$ msortf f=a i=dat.csv o=tmp.csv | msum k=a f=b o=rsl.csv
```

少し複雑な例であるが、上記の例は以下のように mtee コマンドを利用することでうまく動作するようになる。

```
$ msortf f=a i=dat.csv | mtee o=tmp.csv | msum k=a f=b o=rsl.csv
```

mtee コマンドは、標準入力を o=で指定されたファイルおよび標準出力に書き出す。結果として、msortf の結果は

tmp.csv に書きこまれ、msum も mtee よりパイプラインを通じてデータ供給を受け、最終結果を rsl.csv に書き出す。

利用例

例 1: **基本例**

mcut の実行結果は o=で指定した rsl1.csv に出力される。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
$ mcut f=顧客,金額 i=dat1.csv o=rsl1.csv
#END# kgcut f=顧客,金額 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
顧客,金額
A,10
A,20
```

例 2: **リダイレクト**

標準出力をリダイレクト (">"記号) して書き込む。

```
$ mcut f=顧客,金額 i=dat1.csv >rsl2.csv
#END# kgcut f=顧客,金額 i=dat1.csv
$ more rsl2.csv
顧客,金額
A,10
A,20
```

対応コマンド

sep など一部のコマンドを除いて全てのコマンドで利用できる。

2.9.3 f= 入出力項目名

処理対象となる入力項目名の指定をおこなう。例えば、mcut においては「選択される項目名」、magg においては「集計される項目名」、mjoin においては「結合される項目名」を指定する。また複数の項目名は、f=a,b,c のようにカンマで区切って指定する。

さらに、指定された項目毎に出力項目名を指定できるコマンドもある。出力項目名は、f=a:A,b:B のように、入力項目名の後にコロンで区切って指定する。出力項目名が省略されたときは、入力項目名と同じ項目名が利用される。

利用例

例 1: **基本例**

項目 val1 と val2 を切り出す。

```
$ more dat1.csv
id,val1,val2
A,1,2
B,2,3
C,3,4
$ mcut f=val1,val2 i=dat1.csv o=rsl1.csv
#END# kgcut f=val1,val2 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
val1,val2
1,2
```

```
2,3
3,4
```

例 2: **出力項目名の指定**

val1,val2 を集計し、sum1,sum2 という項目名で出力する。

```
$ msum f=val1:sum1,val2:sum2 i=dat1.csv o=rsl2.csv
#END# kgsum f=val1:sum1,val2:sum2 i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,sum1,sum2
C,6,9
```

対応コマンド

mcut, msum, mcat, mjoin など

2.9.4　k= キー項目名

キー項目を指定する（複数項目指定可）。キー項目とは、集計の単位として指定したり、またファイルの結合時に 2 ファイル間の共通項目として指定したりする項目である。

たとえば msum コマンドでは、同一キーごとに合計処理を実施する（集計キーブレイク処理）。また mjoin コマンドでは、2 つのデータファイルについて、キー項目の大小を見比べて結合処理を実施する（結合キーブレイク処理）。

k=パラメータが指定されたとき、コマンドはまずその項目を文字列昇順で並べ替えた上で、それぞれの処理を実行する（ただし、mhashsum コマンドのような例外もある）。

なおキーブレイク処理の詳細は、キーブレイク処理を参照のこと。項目の並べ替えが頻繁に発生するとパフォーマンスの低下を招くため、キーブレイク処理の内容と必要性を理解した上で、並べ替えの回数を少なくするスクリプトを記述することが望ましい。

利用例

例 1: **基本例**

id 項目別に val 項目の値を合計する。

```
$ more dat1.csv
id,val
A,1
B,1
B,2
A,2
B,3
$ msum i=dat1.csv k=id f=val o=rsl1.csv
#END# kgsum f=val i=dat1.csv k=id o=rsl1.csv
$ more rsl1.csv
id%0,val
A,3
B,6
```

例 2: **結合処理**

dat1.csv の id を結合キーに、ref1.csv の name 項目を結合する。

```
$ more dat1.csv
id,val
```

```
A,1
B,1
B,2
A,2
B,3
$ more ref1.csv
id,name
A,nysol
B,mcmd
$ mjoin i=dat1.csv k=id m=ref1.csv f=name o=rsl4.csv
#END# kgjoin f=name i=dat1.csv k=id m=ref1.csv o=rsl4.csv
$ more rsl4.csv
id%0,val,name
A,1,nysol
A,2,nysol
B,1,mcmd
B,2,mcmd
B,3,mcmd
```

対応コマンド

msum, mslide, mjoin, mrjoin, mcommon など

2.9.5 s= 並べ替え項目名

並べ替え項目名を指定する（複数指定可）。

maccum などいくつかのコマンドは、行（レコード）の順序が処理結果に影響を与える。s=パラメータを指定すると、コマンドはあらかじめその項目で行を並べ替えたのち、それぞれの処理を実行する。

項目の並べ替え方法（並び順）は、数値/文字列、昇順/降順の組み合わせで 4 通り指定できる。指定方法は、項目名のあと % に続けて n と r を以下の通り組み合わせる。

文字列昇順:**項目名** (% 指定なし)、文字列逆順:f=**項目名** %r、数値昇順:f=**項目名** %n、数値降順:f=**項目名** %nr。

利用例

例 1: **基本例**

id 項目で並べ替えた後、val 項目の累計を計算する。

```
$ more dat1.csv
id,val
A,1
B,1
B,2
A,2
B,3
$ maccum s=id k=id f=val:val_accum i=dat1.csv o=rsl1.csv
#END# kgaccum f=val:val_accum i=dat1.csv k=id o=rsl1.csv s=id
$ more rsl1.csv
id,val,val_accum
A,1,1
A,2,3
B,1,1
B,2,3
B,3,6
```

例 2: **並べ替え方法を指定する**

val 項目を数値降順で並べ替えた後、val 項目の累計を計算する。

```
$ more dat1.csv
id,val
A,1
B,1
B,2
A,2
B,3
$ maccum s=id,val%nr k=id f=val:val_accum i=dat1.csv o=rsl1.csv
#END# kgaccum f=val:val_accum i=dat1.csv k=id o=rsl1.csv s=id,val%nr
$ more rsl1.csv
id,val,val_accum
A,2,2
A,1,3
B,3,3
B,2,5
B,1,6
```

対応コマンド

maccum, mbest, mmvavg, mnumber, mslide など

2.9.6　a= 追加項目名

新たに項目を追加するようなコマンドにおいて、その項目名を指定する。多くのコマンドは、追加する項目は一つであるため、ここで指定する項目名も一つであることが多い。中には、mcombi コマンドのように複数の項目を出力するものもあるが、その際はカンマで区切って複数の項目名を指定する。

利用例

例 1: **基本例**

精算日という新しい項目を追加する。

```
$ more dat1.csv
id
A
B
C
$ msetstr v=20070101 a=精算日 i=dat1.csv o=rsl1.csv
#END# kgsetstr a=精算日 i=dat1.csv o=rsl1.csv v=20070101
$ more rsl1.csv
id,精算日
A,20070101
B,20070101
C,20070101
```

例 2: **複数項目を追加**

id 項目のデータ A,B,C の 2 つの組合わせを 2 つの項目 (id1,id2) として出力する。

```
$ mcombi f=id n=2 a=id1,id2 i=dat1.csv o=rsl2.csv
#END# kgcombi a=id1,id2 f=id i=dat1.csv n=2 o=rsl2.csv
$ more rsl2.csv
id,id1,id2
C,A,B
C,A,C
C,B,C
```

対応コマンド

mcal, mcombi, mrand, msetstr など

2.9.7 -nfn 項目名行のない CSV(No Field Names の略)

このオプションを指定すると入力データの 1 行目を項目名行とみなさない。主に 1 行目に項目名がないデータの場合に利用される。このフラグを指定すると項目指定のときに項目名は利用できないので項目番号指定をすることになる。項目番号は 0 から始まる整数で指定することに注意する。`-nfn` オプションを指定すると、出力ファイルにも項目名は出力されない。

利用例

例 1: **基本例**

0 番目と 2 番目の項目を切り出す。

```
$ more dat1.csv
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mcut -nfn f=0,2 i=dat1.csv o=rsl1.csv
#END# kgcut -nfn f=0,2 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
A,10
A,20
B,15
B,10
B,20
```

対応コマンド

mchkcsv 以外全てのコマンドで利用できる。

2.9.8 -nfno 項目名行を出力しない (No Field Names for Output の略)

このオプションを指定すると出力データに項目名行を出力しない。`-nfn` とは違い、`i=`や `m=`で指定される入力データは項目名行を伴うデータを前提とする。

利用例

例 1: **基本例**

数量と金額項目を切りだすが、出力データには項目名行は出力されない。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mcut -nfno f=数量,金額 i=dat1.csv o=rsl1.csv
#END# kgcut -nfno f=数量,金額 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
```

```
1,10
2,20
1,15
3,10
1,20
```

対応コマンド

mchkcsv 以外全てのコマンドで利用できる。

2.9.9　-x 項目番号による指定

項目名行を伴う入力データに対して項目番号によって項目を指定したい場合にこのオプションを用いる。コロンで区切って出力項目名を指定することも可能である。

利用例

例 1: **基本例**

0 番目項目を集計キーとして 1 番目と 2 番目の項目を合計する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ msum -x k=0 f=1,2 i=dat1.csv o=rsl1.csv
#END# kgsum -x f=1,2 i=dat1.csv k=0 o=rsl1.csv
$ more rsl1.csv
顧客 %0,数量,金額
A,3,30
B,5,45
```

例 2: **出力項目名も利用可能**

1 番目と 2 番目の項目は、a,b という名前で出力する。

```
$ msum -x k=0 f=1:a,2:b i=dat1.csv o=rsl2.csv
#END# kgsum -x f=1:a,2:b i=dat1.csv k=0 o=rsl2.csv
$ more rsl2.csv
顧客 %0,a,b
A,3,30
B,5,45
```

例 3: -nfn **ではうまくいかない**

-nfn は、最初の行をデータ行としてみなすので、「数量」「金額」というデータを合計しようとしてしまい、うまくいかない。-x は、あくまでも最初の行は項目名行とみなす点が-nfn と異なる。

```
$ msum -nfn k=0 f=1,2 i=dat1.csv o=rsl3.csv
#END# kgsum -nfn f=1,2 i=dat1.csv k=0 o=rsl3.csv
$ more rsl3.csv
顧客,0,0
A,3,30
B,5,45
```

対応コマンド

mchkcsv 以外全てのコマンドで利用できる。

2.9.10 -q 自動並べ替えの無効化

k=パラメータで指定した項目による自動並べ替えを無効にしたい場合にこのオプションを用いる。s=オプションも省略可能となり、各コマンドは MCMD Ver. 1.0 と同等の動作をするようになる。

利用例

例 1: **基本例**

id 項目の値が連続しているときの累計を求める。-q オプションを指定することで、事前に k=パラメータで指定した項目で並べ替える機能を無効化している。

```
$ more dat1.csv
id,val
A,1
B,1
B,2
A,2
B,3
$ maccum -q k=id f=val:val_accum i=dat1.csv o=rsl1.csv
#END# kgaccum -q f=val:val_accum i=dat1.csv k=id o=rsl1.csv
$ more rsl1.csv
id,val,val_accum
A,1,1
B,1,1
B,2,3
A,2,2
B,3,3
```

対応コマンド

k=パラメータを持つすべてのコマンドで利用できる。

2.9.11 -assert_diffSize 入力、出力件数比較を行う

このパラメータを指定すると入力ファイルと出力ファイルの件数の比較を行い、入力ファイルと出力ファイルの件数が異なる場合に、**「#WARNING# ; the number of lines is different」**というメッセージを表示する。

利用例

(基本例)

例えば、mjoin（参照ファイルの項目結合）コマンドを利用する際に、入力ファイルのキー項目（k=パラメータで指定する項目）と参照ファイルのキー項目（K=パラメータで指定する項目）が完全に一致しているかどうかを確認したい場合を想定する。mjoin コマンドで NULL 値を出力する-n、-N パラメータを指定しない場合は、入力ファイルと参照ファイルで共通のキー項目のみが結合され、一致しないキー項目の値は除外される為、入力データと出力データの件数が異なる。その際、-assert_diffSize パラメータを指定しておくと、入力ファイルと出力ファイルの件数の比較を行い、入力ファイルと出力ファイルの件数が異なる場合に**「#WARNING# ; the number of lines is different」**というメッセージを表示するので入力ファイルと参照ファイルのキー項目が完全に一致していないことを確認することができる。

```
$ more dat1.csv
item,date,price
A,20081201,100
A,20081213,98
B,20081002,400
B,20081209,450
C,20081201,100

$ more ref1.csv
item,cost
A,50
B,300
E,200

$ mjoin k=item f=cost m=ref1.csv -assert_diffSize i=dat1.csv o=rsl1.csv
#WARNING# ; the number of lines is different
#END# kgjoin -assert_diffSize f=cost i=dat1.csv k=item m=ref1.csv o=rsl1.csv; IN=5 OUT=4

$ more rsl1.csv
item%0,date,price,cost
A,20081201,100,50
A,20081213,98,50
B,20081002,400,300
B,20081209,450,300
```

対応コマンド

marff2csv, mchkcsv, mcsv2arff, mnewnumber, mnewrand, mnewstr, msep, msep2, mtee, mxml2csv
上記以外の全てのコマンドで利用できる。

2.9.12　-assert_nullkey キー項目に NULL 値が含まれているかどうかのチェックを行う

このパラメータを指定すると、キー項目 (k=または K=パラメータで指定する項目）に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、「`#WARNING# ; exist NULL in key filed`」というメッセージを表示する。

利用例

(基本例)

例えば、msum（項目値の合計）コマンドを例にあげる。k=パラメータで指定した項目の値が同じ行に対して、f=パラメータで指定した集計項目の項目値を合計することを想定した場合、データによっては、k=パラメータで指定したキー項目の値に NULL 値が含まれている場合がある。-assert_nullkey パラメータを指定すると、キー項目の値に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、「`#WARNING# ; exist NULL in key filed`」というメッセージを表示するのでキー項目に NULL 値が含まれていたかどうかを確認することができる。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
,1,10
B,1,15
A,2,20
B,3,10
B,1,20

$ msum k=顧客 f=数量:数量合計,金額:金額合計 -assert_nullkey i=dat1.csv o=rsl1.csv
#WARNING# ; exist NULL in key filed
```

```
#END# kgsum -assert_nullkey f=数量:数量合計,金額:金額合計 i=dat1.csv k=顧客 o=rsl1.csv

$ more rsl1.csv
顧客 %0,数量合計,金額合計
,1,10
A,3,30
B,5,45
```

対応コマンド

maccum, mavg, mbest, mbucket, mcal, mcommon, mcount, mcross, mdelnull, mhashavg, mhashsum, mjoin, mkeybreak, mmbucket, mmvavg, mmvsim, mstats, mnjoin, mnormalize, mnrcommon, mnrjoin, mnumber, mpadding, mrand, mrjoin, mselnum, mselrand, mselstr, msep2, mshare, msim, mslide, mstats, msum, msummary, mtra, muniq, mwindow
上記のコマンドで利用できる。

2.9.13 -assert_nullin 入力項目の f=または vf=で指定された項目の NULL 値チェックを行う

このパラメータを指定すると、f=または vf=で指定された入力項目に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、「#WARNING# ; exist NULL in input data」というメッセージを表示する。

利用例

(基本例)

例えば、maccum（累積計算）コマンドを利用することを想定した場合、maccum コマンドは f=パラメータで指定した項目の値が NULL 値である場合は無視して計算を行う。f=パラメータで指定した項目の値に NULL 値が含まれているかどうかを確認したい場合、-assert_nullin パラメータを指定すると、f=パラメータで指定された入力項目の値に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、「#WARNING# ; exist NULL in input data」というメッセージを表示するので NULL 値が含まれていたかどうかを確認することができる。

```
$ more dat1.csv
顧客,数量,金額
A,1,
A,2,20
B,1,15
B,3,10
B,,20

$ maccum s=顧客 f=数量:数量累計,金額:金額累計 -assert_nullin i=dat1.csv o=rsl1.csv
#WARNING# ; exist NULL in input data
#END# kgaccum -assert_nullin f=数量:数量累計,金額:金額累計 i=dat1.csv o=rsl1.csv s=顧客

$ more rsl1.csv
顧客 %0,数量,金額,数量累計,金額累計
A,1,,1,
A,2,20,3,20
B,1,15,4,35
B,3,10,7,45
B,,20,,65
```

対応コマンド

marff2csv, mbest, mbucket, mchkcsv, mcommon, mcount, mdelnull, mfldname, mkeybreak, mnewnumber, mnewrand, mnewstr, mnrcommon, mnullto, mnumber, mrand, msel, mselrand, msep2, msetstr, msortf, mtee, muniq, mwindow, mxml2csv
上記のコマンドを除いて全てのコマンドで利用できる。

2.9.14　-assert_nullout 出力項目の NULL 値チェックを行う

このパラメータを指定すると、出力項目に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、**「#WARNING# ; exist NULL in output data」**というメッセージを表示する。ただし、計算項目など、入力データがそのまま出力されるものについてはチェックを行わない。

利用例

(基本例)

例えば、mslide(行ずらし）コマンドを例にあげる。k=パラメータで指定した項目単位に、f=パラメータで指定した項目の値を、t=パラメータで指定した行数ずらすことを想定した場合、データによっては、f=パラメータで指定した項目の行数よりも、t=パラメータで指定したずらす回数の方が多いケースがある。そのような処理をする場合に-n パラメータを指定することで、次（前）の行がなくても NULL 値を入れて出力することがある。出力項目に NULL 値が含まれているかどうかを確認したい場合、-assert_nullout パラメータを指定すると、出力項目の値に NULL 値が含まれているかどうかのチェックを行い、NULL 値が含まれていた場合に、**「#WARNING# ; exist NULL in output data」**というメッセージを表示するので出力項目に NULL 値が含まれていたかどうかを確認することができる。
下記の例では出力項目の syo_1,syo_2 コマンドの値に NULL 値が含まれているかどうかのチェックを行い、含まれている為**「#WARNING# ; exist NULL in output data」**が出力されている。

```
$ more dat1.csv
顧客,日付,商品,数量
A,20130406,a,1
A,20130408,b,1
A,20130416,c,1
B,20130407,k,2
C,20130408,d,1
C,20130409,e,4

$ mslide s=顧客,日付 k=顧客 f=商品:syo_ t=2 -n -assert_nullout i=dat1.csv  o=rsl1.csv
#WARNING# ; exist NULL in output data
#END# kgslide -assert_nullout -n f=商品:syo_ i=dat1.csv k=顧客 o=rsl1.csv s=顧客,日付 t=2

$ more rsl1.csv
顧客,日付,商品,数量,syo_1,syo_2
A,20130406,a,1,b,c
A,20130408,b,1,c,
A,20130416,c,1,,
B,20130407,k,2,,
C,20130408,d,1,e,
C,20130409,e,4,,
```

対応コマンド

mbest, mcat, mcombi, mcommon, mcount, mcsv2arff, mcut, mdelnull, mduprec, mfldname, mfsort, mnewnumber, mnewrand, mnewstr, mnrcommon, mnullto, mnumber, mproduct, mrand, msel, mselnum, mselrand, mselstr, msep, msep2, msetstr, msortf, mtee, mtonull, muniq, mvcount, mwindow, mxml2csv

上記のコマンドを除いてすべてのコマンドで利用できる。

2.9.15 precision= 有効桁数

内部的には C 言語における sprintf の書式「"%.*n*g"」を用いている。この書式は、データの桁数と指定した有効桁数によって、標準標記 (整数部. 小数部: ex. 123.456) と、指数表記 (仮数部 e± 指数部: ex. 1.23456e+02) を切り替える。切り替えの基準であるが、データを指数表記で表したときに、指数部が指定の有効桁数を超えるか、もしくは-5 以下の場合 (すなわち、小数点以下に 0 が 4 つ以上続く場合) に指数表記を採用する。

n は 1〜16 の整数が指定可能で、デフォルトは 10 である。$n < 1$ の場合は $n = 1$ にセットされ、$n > 16$ の場合は $n = 16$ にセットされる。

また、環境変数 KG_Precision を設定することでも有効桁数を変更できる。ただし、環境変数を変更すると、それ以降に実行するコマンド全てに反映されることに注意する。

利用例

例 1: **基本例**

id=1 は指数表現で 1.2345678e+08 であり、指数部が有効桁数 6 を超えているので指数表記となり、仮数部の有効桁数が 6 となっている。id=2 は指数表現で 1.23456789e+03 であり、指数部が有効桁数 7 を超えていないので標準標記となり、整数部 + 小数部の桁数が 6 となっている。id=4 は指数表現で 1.23456789e-04 であり、指数部が-4 未満ではないので標準標記となり、有効桁数が 6 となっている。id=5 は指数表現で 1.23456789e-05 であり、指数部が-4 未満となるため指数表記となり、仮数部の有効桁数が 6 となっている。

```
$ more dat1.csv
id,val
1,123456789
2,1234.56789
3,0.123456789
4,0.000123456789
5,0.0000123456789
$ mcal c='${val}' a=result precision=6 i=dat1.csv o=rsl1.csv
#END# kgcal a=result c=${val} i=dat1.csv o=rsl1.csv precision=6
$ more rsl1.csv
id,val,result
1,123456789,1.23457e+08
2,1234.56789,1234.57
3,0.123456789,0.123457
4,0.000123456789,0.000123457
5,0.0000123456789,1.23457e-05
```

例 2: presicion=2 **の場合**

```
$ mcal c='${val}' a=result precision=2 i=dat1.csv o=rsl2.csv
#END# kgcal a=result c=${val} i=dat1.csv o=rsl2.csv precision=2
$ more rsl2.csv
id,val,result
1,123456789,1.2e+08
2,1234.56789,1.2e+03
3,0.123456789,0.12
4,0.000123456789,0.00012
5,0.0000123456789,1.2e-05
```

例 3: **環境変数による指定**

環境変数によって設定すると、それ以降全てのコマンドがその設定値を使う。

```
$ export KG_Precision=4
$ mcal c='${val}' a=result i=dat1.csv o=rsl3.csv
#END# kgcal a=result c=${val} i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,val,result
1,123456789,1.235e+08
2,1234.56789,1235
3,0.123456789,0.1235
4,0.000123456789,0.0001235
5,0.0000123456789,1.235e-05
```

対応コマンド

msum, mcal などの実数値の演算を伴うコマンド全てで利用できる。

2.9.16 tmpPath= 作業ファイル格納パス名

コマンドが内部で用いる作業ファイルを格納するディレクトリ名を指定する。例えば、msortf は分割ソートにおいて並べ替えられた中間結果が一時ファイルとして保存される。指定がなければデフォルトとして/tmp が用いられる。一時ファイル名は、必ず__KGTMP から始まる。

作業ファイルは正常に終了すれば (エラー終了も含めて MCMD のコントロール下で正常に終了するという意味) 削除されるが、不測の事態、例えば停電やバグ終了の場合、消されず残る場合がある。データ量によっては、非常に多くの作業ファイルが生成される可能性があり (100 万ファイル以上!!)、その場合は、コマンドの動作が極端に遅くなる可能性があるので、定期的に作業パスのファイルを確認しておくべきである。なお、現在のところ、これらの不要ファイルの自動消去 (ガベージコレクション) の機能を実装する予定はない。

また、環境変数 KG_Tmp_Path を設定することで、作業ディレクトリを変更できる。ただし、環境変数を変更すると、それ以降に実行するコマンド全てに反映されることに注意する。

利用例

例 1: 基本例

カレントディレクトの直下の tmp ディレクトリを作業ファイル用のディレクトリとする。

```
$ msortf f=val tmpPath=./tmp i=dat1.csv o=rsl1.csv
#END# kgsortf f=val i=dat1.csv o=rsl1.csv tmpPath=./tmp
```

例 2: 環境変数による指定

環境変数によって設定すると、それ以降全てのコマンドがその設定値を使う。

```
$ export KG_TmpPath=~/tmp
$ msortf f=val i=dat1.csv o=rsl1.csv
#END# kgsortf f=val i=dat1.csv o=rsl1.csv
```

対応コマンド

msortf, mdelnull などのキー単位での選択を伴うコマンド、mbucket mnjoin mshare など、キー単位の処理において、データを複数パス走査する必要のあるコマンド。

2.9.17 delim= ベクトル要素の区切り文字

ベクトル型データについて、要素の区切り文字を指定する。デフォルトは半角スペースである。CSV の区切り文字であるカンマを指定することもできるが、ベクトルの区切り文字と混同しないよう、ベクトル全体がダブルクオーテーションで囲われる。

利用例

例 1: 基本例

コロンを区切り文字として、ベクトル項目 vec の要素を並べ替える。

```
$ more dat1.csv
vec
b:a:c
x:p
$ mvsort vf=vec delim=: i=dat1.csv o=rsl1.csv
#END# kgvsort delim=: i=dat1.csv o=rsl1.csv vf=vec
$ more rsl1.csv
vec
a:b:c
p:x
```

例 2: delim を指定しないと

delim を指定していないので b:a:c や x:p は一つの要素として解釈される。

```
$ mvsort vf=vec i=dat1.csv o=rsl2.csv
#END# kgvsort i=dat1.csv o=rsl2.csv vf=vec
$ more rsl2.csv
vec
b:a:c
x:p
```

例 3: カンマを区切り文字にする

区切り文字をカンマにした場合は、ベクトル全体がダブルクオーテーションで囲われることで CSV の区切り文字との区別がつけられる。

```
$ more dat2.csv
id,vec1,vec2
1,a,b
2,p,q
$ mvcat vf=vec1,vec2 a=vec3 delim=, i=dat2.csv o=rsl3.csv
#END# kgvcat a=vec3 delim=, i=dat2.csv o=rsl3.csv vf=vec1,vec2
$ more rsl3.csv
id,vec3
1,"a,b"
2,"p,q"
```

対応コマンド

mvcat, mvsort などベクトル型項目を扱うコマンド全てに適用できる。

2.9.18　bufcount= バッファの数

mbucket,mnjoin,mshare など、キー単位の処理において、データを複数パス走査する必要のあるコマンドにおいて利用する内部バッファの数 (ブロック数) を指定する。一つのバッファは 4MB で、デフォルトでは 10 ブロック (40MB) である。データがバッファに収まらない場合は一時ファイルに書き出されるため、キーのサイズが非常に大きい場合は、メモリに余裕があれば、このパラメータを調整することで処理速度の向上が期待できる。

利用例

例 1: **基本例**

参照ファイルのキーサイズが 80MB(4MB × 20) 以内であれば、一時ファイルは使われない。

```
$ mnjoin k=id m=ref.csv f=name i=dat.csv o=rsl.csv bufcount=20
#END# kgnjoin bufcount=20 f=name i=dat.csv k=id m=ref.csv o=rsl.csv
```

対応コマンド

mbucket, mnjoin, mshare など、キー単位の処理において、データを複数パス走査する必要のあるコマンド。

2.10 Environment variable

MCMD では、表 2.5 に示される環境変数の設定が可能で、その値に応じてコマンドの動作を変更することが可能である。

表 2.5 M コマンドで設定可能な環境変数一覧

変数名	デフォルト値	内容
KG_iSize	4096000	read 一回あたりのサイズ 入力バッファサイズは、KG_iSize の 4 倍のメモリが確保される。 ただし、kgsortf は 10 倍もしくはコマンドパラメータで設定した倍数確保される。 また、キーブロック単位で処理するコマンドのバッファサイズは KG_BlockCount を参照。 以下の条件を満たす必要がある。 KG_iSize=KG_MaxRecLen∗i (ただし、i は 1 以上の整数) KG_iSize≥KG_MaxRecLen∗2
KG_oSize	2048000	write 一回あたりのサイズ 出力バッファサイズは、KG_oSize 同じサイズのメモリが確保される。 KG_oSize=KG_MaxRecLen∗i (ただし、i は 1 以上の整数) KG_oSize≥KG_MaxRecLen∗2
KG_MaxRecLen	1024000	一行あたりの最大文字数 (上限:10240000) KG_iSize と KG_oSize に示された条件を満たす必要がある。
KG_BlockCount	128	キー単位ごとに処理 [注 1)] する際に使用するバッファ数 KG_iSize の欄で示したバッファサイズ ∗KG_BlockCount のバッファが確保される。 KG_MaxRecLen ∗2 + 4∗ KG_ioSize) ∗ KG_BlockCount
KG_TmpPath	/tmp	ライブラリ関数が用いるデフォルトの一時ファイル用ディレクトリ
KG_Precision	10	有効桁数
KG_VerboseLevel	4	M コマンドのエラーメッセージ出力レベル 0: メッセージを一切出力しない 1: + error メッセージ出力 2: + warning メッセージ出力 3: + end メッセージ出力 4: + msg メッセージ出力 (デフォルト)
KG_msgTimebyfsec	false	false: 終了メッセージの時刻を秒単位で表示する ture: 終了メッセージの時刻をマイクロ秒単位で表示する

注 1) キー単位ごとの処理とは、mnjoin の様に、同一キーのデータを一旦全てメモリに読み込む処理のこと。詳細は個々のコマンドのマニュアルを参照のこと。

以下、コマンドのメッセージを制御する KG_VerboseLevel の設定を変更する例を示す。

利用例

例 1: 終了メッセージ

デフォルトでは KG_Verbose=4 なので、正常終了時のメッセージもエラー終了メッセージも出力される。

```
$ more dat.csv
k,v
A,1
B,2
$ mcut f=k,v i=dat.csv o=out.csv
#END# kgcut f=k,v i=dat.csv o=out.csv
$ mcut x=k,v i=dat.csv o=out.csv
#ERROR# unknown parameter x= (kgcut)
```

例 2: **エラーメッセージのみ出力**

KG_Verbose=1 とすると、エラー終了メッセージは表示されるが、正常終了メッセージは表示されない。

```
$ export KG_VerboseLevel=1
$ mcut f=k,v i=dat.csv o=out.csv
$ mcut x=k,v i=dat.csv o=out.csv
#ERROR# unknown parameter x= (kgcut)
```

例 3: **メッセージを一切表示させない**

KG_Verbose=0 とすると、いずれのメッセージも表示されない。

```
$ export KG_VerboseLevel=0
$ mcut f=k,v i=dat.csv o=out.csv
$ mcut x=k,v i=dat.csv o=out.csv
```

2.11　キーブレイク処理

キーブレイク処理とは、その項目が並べ換わっていることを前提として、同一のキー項目値毎に一定の処理を行う処理方式のことを言う。キーブレイク処理は大きく分けて 2 つの処理に分けられる。一つは集計のためのキーブレイク処理 (以下「**集計キーブレイク処理**」と呼ぶ) で、他方は結合のためのキーブレイク処理 (以下「**結合キーブレイク処理**」と呼ぶ) である。

mjoin、mcommon などコマンド名に「join」か「common」を含むコマンドが結合キーブレイク処理を、それ以外のコマンドのうち k=パラメータを持つすべてのコマンドが集計キーブレイク処理を行っていると考えてよい。

たとえば集計キーブレイク処理を行う msum コマンドでは、キー項目の値の変化を検知することで、同一キー毎に合計処理を実行する。そのためには事前にキー項目で行の並べ替えをしておく必要があるので、(入力ファイルが事前に並べ替えられている場合を除き)msum コマンドは、内部でまず並べ替えをした上で、合計処理を行う。

結合キーブレイク処理はもう少し複雑で、たとえば mjoin コマンドは、2 つのデータファイルについて、キー項目の大小を見比べる。キー項目が小さいデータファイルは読み進め、キー項目値が同じであれば結合処理を実施する。このようにキー項目値の大小比較をしているため、結合のためのキーブレイク処理においては、事前に 2 つのデータファイルともキー項目で並べ替えられていることが前提となる。そのため mjoin コマンドは、まず内部で 2 つのデータファイルを並べ替える。

どちらのキーブレイク処理でも基本は**文字列昇順**による並べ替えを行うが、mrjoin のような数値範囲による結合キーブレイク処理においては、**数値昇順**で並べ替えを行う。

k=パラメータで項目を指定するだけで、各コマンドが自動的に並べ替えの要否を判断し、必要な場合は並べ替えを行うため、ユーザは原則としてファイルの並べ替えを意識する必要はない。ただ並べ替え処理が不要になったわけではなく、各コマンドが内部的に並べ替え処理を行っているという点に注意が必要である。スクリプトの構成によっては、並べ替え処理が頻繁に発生し、パフォーマンス低下の原因となる。

スクリプトの例

並べ替え処理が頻繁に発生するスクリプト

最初に xxcustomer ファイルを name 項目で並べ替えた状態で出力しているが、その後の mjoin コマンドはいずれも、id をキー項目として結合処理を行っている。この場合、3 つの mjoin コマンドがそれぞれ id 項目による並べ替えを行ってしまう。

```
mcut   i=customer.csv f=id,name |
msortf f=name o=xxcustomer

mjoin i=xxcustomer m=address.csv k=id f=address o=cust_address.csv
mjoin i=xxcustomer m=phone.csv   k=id f=phone   o=cust_phone.csv
mjoin i=xxcustomer m=age.csv     k=id f=age     o=cust_age.csv
```

並べ替え処理が最小限のスクリプト

次のように修正すれば、xxcustomer ファイルは id 項目で並べ替えられているので、続く 3 つの mjoin コマンドはいずれも並べ替え処理を省略できる。

```
mcut   i=customer.csv f=id,name |
msortf f=id o=xxcustomer

mjoin i=xxcustomer m=address.csv k=id f=address o=cust_address.csv
mjoin i=xxcustomer m=phone.csv   k=id f=phone   o=cust_phone.csv
mjoin i=xxcustomer m=age.csv     k=id f=age     o=cust_age.csv
```

第 3 章

便利ツール

3.1　bash completion

コマンドラインでのコマンド入力を支援するツールである bash-completion の MCMD 用設定ファイルをインストールすることにより、以下に示す入力補完機能が利用できるようになり、コマンドライン上での MCMD の利用が格段に便利になる。

- コマンド別に指定可能なパラメータ一覧と補完入力
- 入出力ファイル名の補完
- 入力 CSV データ上の項目名補完
- 選択肢パラメータの値補完

3.1.1　インストール

bash-completion がインストールされているかどうかは、以下に示すように complete コマンドが利用できるかどうかで確認できる。もしインストールされていないようであれば、開発のページを参照してインストールする。もしくはネット上から、OS 別により簡易なインストール方法の情報も得られるであろう。

```
$ complete --help
-bash: complete: --: invalid option
complete: usage: complete [-abcdefgjksuv] [-pr] [-o option] [-A action]
[-G globpat] [-W wordlist] [-P prefix] [-S suffix] [-X filterpat] [-F function]
[-C command] [name ...]
```

MCMD 用設定ファイルは、MCMD ソースツリー mcmd/unitilies の直下にある bash_completion_mcmd.sh ファイルを、home ディレクトリ直下に名前を変えてコピーする。

```
$ cp mcmd/unitilies/bash_completion_mcmd.sh ~/.bash_completion_mcmd.sh
```

そして、以下に示す用に、そのスクリプトの実行指示を~/.bash_profile に追加しておけば、ログインする度に自動的に設定が読み込まれるようになる。

```
. ~/.bash_completion_mcmd.sh
```

3.1.2　パラメータ補完

コマンド名を入力した後に TAB キーを 2 回押すと、そのコマンドで設定可能なパラメータ一覧が表示される。以下では msum コマンドを入力した後に TAB キーを二度押した時の状態が示されている。

```
$ msum [TAB キーを 2 回押す]
-assert_diffSize  -assert_nullkey   -n      -nfno     -q      f=      k=      precision=
-assert_nullin    -assert_nullout   -nfn    -params   -x      i=      o=      tmpPath=
```

上記の状態で、p を入力し TAB キーを押すと、p から始まるパラメータは precision=しかないので、そのキーワード文字列が補完される。なお、補完後はキーワードの後ろにスペース挿入されるので、続けて値を入力したければ一文字戻って入力する必要があることに注意する。

```
$ msum p[TAB キーを押す]
# 以下の通り補完される。
$ msum precision=
```

同様に、-a を入力後に TAB キーを押しても、-a から始まるパラメータは一意に決まらないため、一意に決まる部分である-assert_までが補完され、再度 TAB キーを押すと、-assert_始まる 4 つのパラメータが画面に表示される。

```
$ msum -a[TAB キーを押す]
-assert_diffSize  -assert_nullin    -assert_nullkey   -assert_nullout
```

3.1.3 入出力ファイル名の補完

MCMD では、ファイル名もしくはディレクトリ名を指定するパラメータとして i=,m=,o=などがあるが、それらのパラメータを入力した後に TAB キーを押すと、ファイル名の補完が可能となる。何も入力せず=に続けて TAB キーを押すとカレントディレクトリの直下にあるファイル一覧が表示される。何らかの文字列を入力後に TAB キーを押すと、その文字列から始まるファイル名が補完され、一意に決まらなければマッチする複数のファイル名が表示される。保管可能なファイル名は 1 つのみであるが、例外的に mcat コマンドは、カンマに続けてファイル名を複数補完入力することが可能である。

```
$ msum i=[TAB キーを 2 回押す]
test1.csv   test2.csv ...

$ mcat i=test1.csv,[TAB キーを 2 回押す]
test1.csv   test2.csv ...
```

3.1.4 項目名の補完

MCMD では項目名を指定するパラメータが多いが、入力ファイルが入力済みであれば、そのファイル上の項目名を補完入力することが可能となる。

```
$ cat test1.csv
codeA,codeB,date,value1,value2
A,aaa,20170101,3,120
A,bbb,20170102,6,100

$ msum i=test1.csv k=[TAB キーを 2 回押す]
codeA   codeB   date    value1  value2
~ $ msum i=test1.csv k=c[TAB キーを押す]
# 以下の通り補完される。
~ $ msum i=test1.csv k=code[TAB キーを押す]
codeA  codeB
~ $ msum i=test1.csv k=codeA f=value[TAB キーを 2 回押す]
value1  value2
~ $ msum i=test1.csv k=codeA f=value*
codeA%0,codeB,date,value1,value2
A,bbb,20170102,9,220
#END# kgsum f=value* i=test1.csv k=codeA; IN=2 OUT=1; 2017/04/28 13:09:13
```

mjoin のように入力ファイルの指定が 2 つある場合、項目指定のキーワード毎にどちらのファイルを利用するかが決まっているため、キーワードに対応するファイル上の項目名が補完される。

```
$ cat test1.csv
codeA,codeB,date,value1,value2
A,aaa,20170101,3,120
A,bbb,20170102,6,100

$ cat test2.csv
code,name
A,aaa
B,bbb
```

```
$ mjoin i=test1.csv m=test2.csv k=[TAB キーを 2 回押す]
codeA   codeB   date    value1  value2
$ mjoin i=test1.csv m=test2.csv k=codeA K=[TAB キーを 2 回押す]
codeA  name
$ mjoin i=test1.csv m=test2.csv k=codeA K=code f=[TAB キーを 2 回押す]
codeA  name
$ mjoin i=test1.csv m=test2.csv k=codeA K=code f=name
codeA%0,codeB,date,value1,value2,name
A,aaa,20170101,3,120,aaa
A,bbb,20170102,6,100,aaa
#END# kgjoin K=code f=name i=test1.csv k=codeA m=test2.csv; IN=2 OUT=2; 2017/04/28 13:46:08
```

3.1.5　選択肢の補完

mstats の c=など、決まった選択肢の中から一つを選択するタイプのパラメータについて補完入力が可能である。

```
$ msim c=[TAB キーを 2 回押す]
chi        confMax convMax cosine euclid  jaccard kendall oddsRatio phi      supportr yuleQ
cityblock confMin convMin covar  hamming kappa   lift    pearson   spearman ucovar   yuleY

~ $ mstats c=[TAB キーを 2 回押す]
USD     cv      kurt    mean    min     qrange  qtile3  sd      sum     ukurt   uskew   var
count   devsq   max     median  mode    qtile1  range   skew    ucount  usd     uvar
```

第 4 章

コマンドリファレンス

全コマンドに共通して示される書式の意味について解説する。全てのコマンドリファレンスにおいて、書式は以下の例のように示される。

書式

```
mjoin k= [f=] [K=] [-n] [-N] m=| i= [o=] [-nfn] [-nfno] [-x] [--help] [--helpl] [--version]
```

パラメータ

k=	**ここで指定した入力データの項目と** K=**パラメータで指定された** **参照データの項目が同じ行の項目結合が行われる。** NULL **値は，参照ファイルの** K=**で指定した項目のどの値にもマッチしない値として扱われる。**
f=	**結合する参照ファイル上の項目名リストを指定する。** **省略するとキー項目を除いた全ての項目が結合される。**
:	:

コマンド名に続いて、そのコマンドに指定可能なパラメータおよびオプションが列挙されている。多くのコマンドで共通する i=,-nfn などのパラメータやオプションについては、別の節にてまとめて記述されており、その節へのリンクが張られている。そして、各パラメータの説明が、書式の下に記述される。

[f=] のように四角括弧で囲われたパラメータは省略可能であることを意味する。一方で、k=のように四角括弧で囲われていないパラメータは必須であることを意味する。また [to=|size=] のように縦棒で区切られたパラメータは、いずれか一つのパラメータしか指定できないことを意味し、括弧で囲われていることで省略することも可能という意味になる (例えば mbest コマンド)。一方で m=|i=のように四角括弧で囲われていなければ、必ずいずれかのパラメータは指定しなければならない、すなわち選択必須のパラメータであることを意味する (例えば、上記の mjoin コマンド)。

また、あるオプションを指定していたときのみ必須となるパラメータなど、より条件が複雑なパラメータもあるが、それらは、各パラメータの説明欄にて解説している。

4.1 maccum 累積計算

f=パラメータで指定した項目の累積を計算し、新しい項目として追加する。k=を指定することで、キー単位毎に累積計算が可能となる。

書式

maccum f= s= [k=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の値が累積される。
項目の値が NULL 値である場合は無視される。
:(コロン）で新項目名を指定する必要がある。例）**f=数量:数量累計**

s= ここで指定した項目 (複数項目指定可) で並べ替えられた後、累積が計算される。
-q オプションを指定しないとき、s=パラメータは必須。

k= 累積の単位となる項目名リスト (複数項目指定可) を指定する。

利用例

例 1: 基本例

「数量」と「金額」項目の累積値を計算し、「数量累計」と「金額累計」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ maccum s=顧客 f=数量:数量累計,金額:金額累計 i=dat1.csv o=rsl1.csv
#END# kgaccum f=数量:数量累計,金額:金額累計 i=dat1.csv o=rsl1.csv s=顧客
$ more rsl1.csv
顧客 %0,数量,金額,数量累計,金額累計
A,1,10,1,10
A,2,20,3,30
B,1,15,4,45
B,3,10,7,55
B,1,20,8,75
```

例 2: キー項目を指定する例

「顧客」項目を単位に「数量」と「金額」項目の累積値を計算し、「数量累計」と「金額累計」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ maccum k=顧客 s=顧客 f=数量:数量累計,金額:金額累計 i=dat1.csv o=rsl2.csv
```

```
#END# kgaccum f=数量:数量累計,金額:金額累計 i=dat1.csv k=顧客 o=rsl2.csv s=顧客
$ more rsl2.csv
顧客,数量,金額,数量累計,金額累計
A,1,10,1,10
A,2,20,3,30
B,1,15,1,15
B,3,10,4,25
B,1,20,5,45
```

例 3: NULL 値を含む累計

「数量」と「金額」項目の累積値を計算し、「数量累計」と「金額累計」という項目名で出力する。NULL は無視される。結果も NULL が出力される。

```
$ more dat2.csv
顧客,数量,金額
A,1,10
A,,20
B,1,15
B,3,
B,1,20
$ maccum s=顧客 f=数量:数量累計,金額:金額累計 i=dat2.csv o=rsl3.csv
#END# kgaccum f=数量:数量累計,金額:金額累計 i=dat2.csv o=rsl3.csv s=顧客
$ more rsl3.csv
顧客 %0,数量,金額,数量累計,金額累計
A,1,10,1,10
A,,20,,30
B,1,15,2,45
B,3,,5,
B,1,20,6,65
```

関連コマンド

mshare：構成比を計算する。maccum と組み合わせて累積相対度数が計算できる。

mcal：前行の計算結果#{}を利用することで累計計算ができる。

4.2 marff2csv arff 形式から csv 形式への変換

arff 形式 (WEKA 用のデータフォーマット) のデータから csv 形式のデータへ変換する。

arff 形式データ

以下 arff 形式データのフォーマットを記載する。

```
@RELATION       タイトル

@ATTRIBUTE      項目名     string(文字列)
@ATTRIBUTE      項目名     date(日時  フォーマット:フォーマットは省略可能。
                                省略した場合は、"yyyy-MM-dd'T'HH:mm:ss")
@ATTRIBUTE      数量     numeric(数字)
@ATTRIBUTE      商品     {A,B}(カテゴリ型項目)

@DATA(実データ)
No.1,20081201,1,10,A
No.2,20081202,2,20,A
No.3,20081203,3,30,A
No.4,20081201,4,40,B
No.5,20081203,5,50,B
```

書式

marff2csv [i=] [o=] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

利用例

例 1: 基本例

arff 形式の顧客購買データを csv 形式のデータへ変換する。

```
$ more dat1.arff
@RELATION       顧客購買データ

@ATTRIBUTE      顧客     string
@ATTRIBUTE      日付     date yyyyMMdd
@ATTRIBUTE      数量     numeric
@ATTRIBUTE      金額     numeric
@ATTRIBUTE      商品     {A,B}

@DATA
No.1,20081201,1,10,A
No.2,20081202,2,20,A
No.3,20081203,3,30,A
No.4,20081201,4,40,B
No.5,20081203,5,50,B
$ marff2csv i=dat1.arff  o=rsl1.csv
#END# kgarff2csv i=dat1.arff o=rsl1.csv
$ more rsl1.csv
顧客,日付,数量,金額,商品
No.1,20081201,1,10,A
No.2,20081202,2,20,A
No.3,20081203,3,30,A
No.4,20081201,4,40,B
No.5,20081203,5,50,B
```

関連コマンド

mcsv2arff

参考資料

http://weka.wikispaces.com/ARFF

4.3 mavg 項目値の平均

f=パラメータで指定した項目の平均値を計算する。(**注**)k=と f=パラメータで指定した項目以外については、どの行が出力されるか¿は不定であることに注意してください。

書式

mavg f= [k=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の値が集計される。
:(コロン）で新項目名を指定可能。例）f=数量:数量平均
k= 集計の単位となる項目 (複数項目指定可) 名リストを指定する。
-n NULL 値が 1 つでも含まれていると結果も NULL 値とする。

利用例

例 1: **基本例**

「顧客」項目を単位に「数量」と「金額」項目の平均値を計算し、「数量平均」と「金額平均」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,5
A,2,20
B,1,15
B,,10
B,5,20
$ mavg k=顧客 f=数量:数量平均,金額:金額平均 i=dat1.csv o=rsl1.csv
#END# kgavg f=数量:数量平均,金額:金額平均 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客 %0,数量平均,金額平均
A,1.5,12.5
B,3,15
```

例 2: NULL **値がある場合の出力**

「顧客」項目を単位に「数量」と「金額」項目の平均値を計算し、「数量平均」と「金額平均」という項目名で出力する。-n オプションを指定することで、NULL 値が含まれている場合は、結果も NULL 値として出力する。

```
$ mavg k=顧客 f=数量:数量平均,金額:金額平均 -n i=dat1.csv o=rsl2.csv
#END# kgavg -n f=数量:数量平均,金額:金額平均 i=dat1.csv k=顧客 o=rsl2.csv
$ more rsl2.csv
顧客 %0,数量平均,金額平均
A,1.5,12.5
B,,15
```

例 3: 顧客項目を単位としない例

「数量」と「金額」項目の平均値を計算し、「数量平均」と「金額平均」という項目名で出力する。

```
$ mavg f=数量:数量平均,金額:金額平均 i=dat1.csv o=rsl3.csv
#END# kgavg f=数量:数量平均,金額:金額平均 i=dat1.csv o=rsl3.csv
$ more rsl3.csv
顧客,数量平均,金額平均
B,2.25,14
```

関連コマンド

mhashavg：集計キーを事前に並べ替えなくても計算できる。

msum：合計バージョン。

mstats：その他の多様な統計量を求めるのであればこれ。

4.4 mbest 指定行の選択

指定した行番号の行を選択する。行番号は 0 から始まることに注意する (項目名行は除いて、データ本体の先頭行が 0 行目)。行番号は from=と to=(もしくは size=) で指定する。

書式

mbest s= [R=] [from=] [to=|size=] [k=] [u=] [-r] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

s=	ここで指定した項目 (複数項目指定可) で並べ替えられた後、指定した行が選択される。 -q オプションを指定しないとき、s=パラメータは必須。
from=	選択する開始行番号 (0 以上の整数)【デフォルト値:0】
to=	選択する終了行番号 (0 以上の整数)【デフォルト値:0】 「from=の値」≤「to=の値」でなければならない。
size=	選択する行数【デフォルト値:1】 to=と size=の両方を同時に指定することはできない。
R=	行番号範囲リスト (複数項目指定可)【必須】*以前のバージョンで使用されていた範囲指定の方法 ここで指定した行番号の行が選択される。 _(アンダーバー) で範囲指定できる。 範囲指定の際には MIN(開始行以降),MAX(最終行まで) を使用できる。 ※ One Point：事前に目的とする行選択が行いやすいように並べ替えておくとよい。
k=	指定列の項目 (複数項目指定可) が同じ値の行ごとに、from=,to=,size=で指定した行番号の行を選択する。 -x,-nfn オプション使用時は、項目番号 (0〜) で指定可能。
u=	不一致データ出力ファイル名 指定の条件に一致しない行を出力するファイル名。
-r	条件反転 from=,to=(size=) パラメータで指定した行番号以外の行を選択する。

利用例

例 1: 基本例

この例では、「数量」と「金額」項目値の大きい順（降順）に並べ替えている。from=,to=,size=を指定しなければ先頭行 (0 行目) のみ選択する

```
$ more dat1.csv
顧客,数量,金額
A,20,5200
B,18,4000
C,15,3500
D,10,2000
E,3,800
$ mbest s=数量%nr,金額%nr i=dat1.csv o=rsl1.csv
#END# kgbest i=dat1.csv o=rsl1.csv s=数量%nr,金額%nr
$ more rsl1.csv
顧客,数量%0nr,金額%1nr
A,20,5200
```

例 2: **基本例** 2

「顧客」で並べ替えた後、先頭行 (0 行目) から 3 行選択する

```
$ mbest s=顧客 from=0 size=3 i=dat1.csv o=rsl2.csv
#END# kgbest from=0 i=dat1.csv o=rsl2.csv s=顧客 size=3
$ more rsl2.csv
顧客 %0, 数量, 金額
A,20,5200
B,18,4000
C,15,3500
```

例 3: **基本例** 3

並べ替えを行わず (もとのレコード順序を維持したまま)、0 行目から 1 行目まで選択する

```
$ mbest -q from=0 to=1 i=dat1.csv o=rsl3.csv
#END# kgbest -q from=0 i=dat1.csv o=rsl3.csv to=1
$ more rsl3.csv
顧客, 数量, 金額
A,20,5200
B,18,4000
```

例 4: **条件反転**

顧客の初回来店日以外の行を選択する。初回来店日は `fvd.csv` というファイルに出力する。

```
$ more dat2.csv
顧客, 日付, 金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
$ mbest s=顧客,日付 k=顧客 -r u=fvd.csv i=dat2.csv o=rsl4.csv
#END# kgbest -r i=dat2.csv k=顧客 o=rsl4.csv s=顧客,日付 u=fvd.csv
$ more rsl4.csv
顧客, 日付, 金額
A,20081207,20
A,20081213,30
B,20081209,50
$ more fvd.csv
顧客, 日付, 金額
A,20081201,10
B,20081002,40
```

関連コマンド

msel：`line()` 関数を使えば、同じようなことができる。

muniq：単純にキー項目を単一化したいだけならこれ。

mselnum：数値範囲によって行選択ができる。

4.5 mbucket 件数均等化バケット分割

f=で指定した数値項目を n=で指定した数の区間に分割する。区間の計算には 2 通りの方法があり、一つは、各区間に属する件数ができるだけ均等になるような区間を計算する (件数均等化バケット分割と呼ぶ)。他方は、区間の範囲が均等になるような区間を計算する (範囲均等化バケット分割と呼ぶ)。-rng を指定すると範囲均等分割となり、指定しなければ件数均等分割となる。f=に複数の項目を指定した場合は、それぞれの項目ごとにバケット分割を実行する。

書式

mbucket f= n= [-rng] [-r] [F=] [k=] [O=] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の値に基づいて分割をおこなう。
分割対象の項目名:出力する項目名

n= 分割数
f=で指定した項目それぞれの分割数をカンマで区切って指定する。
ただし 1 つの数字を指定した場合は全ての項目の分割数として扱われる。

F= 出力形式【デフォルト値:0】
バケットの名前を出力形式。
0:バケット番号のみを表示する。
1:バケットの範囲のみを表示する。
2:バケット番号とバケット範囲の両方を表示する。

k= バケット分割を行う単位となる項目 (複数項目指定可) 名リスト。

O= バケット範囲出力ファイル
f=パラメータで指定した各項目の各バケットの数値範囲を出力するファイル名。

-rng バケットの範囲均等指定
バケットの範囲が均等になるように分割する。

-r バケットの番号を逆順に出力する。

利用例

例 1: 基本例

x,y 項目それぞれで、件数ができるだけ均等になるように 2 分割する。その際、各バケットの数値範囲を rng1.csv に出力する。

```
$ more dat1.csv
id,x,y
A,2,7
B,6,7
C,5,6
D,7,5
E,6,4
F,1,3
G,3,3
H,4,2
I,7,2
J,2,1
$ mbucket f=x:xb,y:yb n=2 O=rng1.csv i=dat1.csv o=rsl1.csv
#END# kgbucket O=rng1.csv f=x:xb,y:yb i=dat1.csv n=2 o=rsl1.csv
```

```
$ more rsl1.csv
id,x,y,xb,yb
A,2,7,1,2
B,6,7,2,2
C,5,6,2,2
D,7,5,2,2
E,6,4,2,2
F,1,3,1,1
G,3,3,1,1
H,4,2,1,1
I,7,2,2,1
J,2,1,1,1
$ more rng1.csv
fieldName,bucketNo,rangeFrom,rangeTo
x,1,,4.5
x,2,4.5,
y,1,,3.5
y,2,3.5,
```

例 2: **範囲均等化分割**

-rng オプションを指定すると範囲均等化分割となる。

```
$ mbucket f=x:xb,y:yb n=2 -rng O=rng2.csv i=dat1.csv o=rsl2.csv
#END# kgbucket -rng O=rng2.csv f=x:xb,y:yb i=dat1.csv n=2 o=rsl2.csv
$ more rsl2.csv
id,x,y,xb,yb
A,2,7,1,2
B,6,7,2,2
C,5,6,2,2
D,7,5,2,2
E,6,4,2,2
F,1,3,1,1
G,3,3,1,1
H,4,2,2,1
I,7,2,2,1
J,2,1,1,1
$ more rng2.csv
fieldName,bucketNo,rangeFrom,rangeTo
x,1,,4
x,2,4,
y,1,,4
y,2,4,
```

例 3: **キー項目を指定した例**

id 項目を集計キーとして、x,y 項目それぞれを件数均等化バケット分割する。n=2,3 と指定することで、x 項目の分割数は 2 に、y 項目の分割数は 3 となる。出力形式はバケット番号とバケット範囲の両方を表示する (F=2)。

```
$ more dat2.csv
id,x,y
A,2,7
A,6,7
A,5,6
B,7,5
B,6,4
B,1,3
C,3,3
C,4,2
C,7,2
C,2,1
```

```
$ mbucket k=id f=x:xb,y:yb n=2,3 F=2 i=dat2.csv o=rsl3.csv
#END# kgbucket F=2 f=x:xb,y:yb i=dat2.csv k=id n=2,3 o=rsl3.csv
$ more rsl3.csv
id%0,x,y,xb,yb
A,2,7,1:_3.5,2:6.5_
A,6,7,2:3.5_,2:6.5_
A,5,6,2:3.5_,1:_6.5
B,7,5,2:3.5_,3:4.5_
B,6,4,2:3.5_,2:3.5_4.5
B,1,3,1:_3.5,1:_3.5
C,3,3,1:_3.5,3:2.5_
C,4,2,2:3.5_,2:1.5_2.5
C,7,2,2:3.5_,2:1.5_2.5
C,2,1,1:_3.5,1:_1.5
```

問題の定式化[*1]

いま、n 個のデータ $x_1, x_2, \cdots, x_n$(このデータ集合を D とする) が与えられている。これらのデータを k 個のグループ (バケットと呼ぶ) に分割し、得られる k 個のバケットに入っているデータの個数が一様になるようにしたい。一様性の評価基準としては分散を用いる。

$$X = \{x_i \mid 1 \ \leq \ i \ \leq \ n \ , \ x_i \ \in \ D\}$$

とおく。X の異なる値を小さい順に並べたものを $v_1, v_2, \ldots, v_n$ とする。X のバケット分割とは、$\{v_1, v_2, \ldots, v_n\}$ を区間に分割することを表し，その区間を、$I_1, I_2, \ldots, I_k$ とおき、D_j、n_j を以下のように定める。

$$D_j = \{x_i \mid 1 \ \leq \ i \ \leq \ n \ , \ x_i \ \in \ I_j\}$$

$$n_j = |D_j|$$

一様さの基準である分散は以下のように定められる。

$$Var = \sum_{j=1}^{k} \ (n_j - \bar{n})$$

ここで、$\bar{n} = n/k$ である。展開すると、

$$Var \ = \ \sum_{j=1}^{k} \ (n_j^2 - 2n_j\bar{n} + \bar{n}^2) \quad = \ \sum_{j=1}^{k} \ n_j^2 \ - \ k\bar{n}^2$$

となる。$k\bar{n}^2$ は分割の仕方によらず定数なので、Var を最小化するには上式の第 1 項のみを考えたらよい。つまり、

$$Var' = \ \sum_{j=1}^{k} \ n_j^2$$

を最小化するような区間分割を求めることを考えたらよい。

アルゴリズム

再帰方程式を用いることによって動的計画法により解ける。詳細は以下のとおりである。$DP(m, h)$ を $v_1, v_2, \ldots, v_m$ を h 個の区間 $I_1, I_2, \ldots, I_h$ に分割したときの $\sum_{j=1}^{h} \ n_j^2$ の最小値とする。最終目標は $DP(n, k)$ を求めることである。このとき，$DP(m, h)$ は以下の方程式を満たす。

$$DP(m, \ h) \ = \ \min_{g=h-1, \ \ldots, m-1} \ \{ \ DP(g, h-1) \ + \ |\{ \ x_i \mid v_{g+1} \ \leq \ x_i \ \leq \ v_m \ \}|^2 \}$$

[*1] 定式化およびアルゴリズムは加藤直樹教授 (京都大学工学研究科) による。

この再帰方程式を解く。ただし，初期値は

$$DP(m,\ 1)\ =\ |\{\ x_i \mid v_1\ \leq\ x_i\ \leq\ v_m\ \}|^2\ ,\ m = 1, ..., n$$

である。再帰方程式は $DP(m,2)(m = 1,\ldots,n), DP(m,3)(m = 1,\ldots,n),\ldots, DP(m,k-1)(m = 1,\ldots,n)$ の順に解いていく。最後は,

$$DP(n,\ k)\ =\ \min_{g=k-1,\ \ldots,n-1}\ \{\ DP(g,k-1)\ +\ |\{\ x_i \mid v_{g+1}\ \leq\ x_i\ \leq\ v_n\ \}|^2\}$$

により，$DP(n,k)$ を求める

関連コマンド

mmbucket：多次元のセルで件数均等化分割をする場合はこちらを使う。

4.6 mcat 併合

i=パラメータで指定した全ファイルのレコードを、指定した順に併合する。ワイルドカードでファイル名を指定した場合は、ファイル名のアルファベット順に併合される。

書式

mcat [f=] [-skip_fnf] [-nostop|-skip|-force] [i=] [o=] [-stdin] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

i=	入力ファイル名リストを指定する。 複数のファイルをカンマで区切って指定する。ワイルドカードを用いることができる。
f=	併合する項目名を指定する。 指定を省略すれば i=で指定した 1 つ目のファイルの項目名が使われる。
-skip_fnf	i=で指定したファイルが存在しなくてもエラー終了しない。 ただし、全ファイルがなければエラーとなる。
-nostop	-nostop ,-skip,-force は、指定の項目名がなかったときの動作を制御するフラグである。 -nostop は、指定の項目名がなければ null を出力する。 -nfn が同時に指定された場合，項目数が異なればエラー終了する。
-skip	指定の項目名がなければそのファイルは併合しない。 -nfn が同時に指定された場合、項目数が異なればそのファイルは併合しない。
-force	指定の項目名がなければ，項目番号で強制併合する。 指定の項目番号がなければ null を出力する。
-stdin	標準入力も併合する。
-add_fname	併合元のファイル名を最終項目として追加する。 標準入力は/dev/stdin という名称になる。 項目名は"fileName"固定なので、入力データに同一の項目名があるとエラーとなる。

備考

- 複数ファイルの指定にワイルドカード (”*”と”?”) を利用することができる。ファイル名だけでなくディレクトリ名に対しても指定することができる。
- ホームディレクトリ記号 (~/) も利用可能。
- 併合される順序は i=で指定したファイルの出現順。ワイルドカードを指定した場合は、アルファベット順。標準入力は最初に併合される。

利用例

例 1: 同一項目名ファイルの併合

```
$ more dat1.csv
顧客，日付，金額
A,20081201,10
B,20081002,40
$ more dat2.csv
顧客，日付，金額
A,20081207,20
A,20081213,30
```

```
B,20081209,50
$ mcat i=dat1.csv,dat2.csv o=rsl1.csv
#END# kgcat i=dat1.csv,dat2.csv o=rsl1.csv
$ more rsl1.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
A,20081207,20
A,20081213,30
B,20081209,50
```

例 2: **項目名の異なるファイルの併合**

i=の最初のファイル dat1.csv の項目「顧客, 日付, 金額」の 3 項目を併合する。しかし、dat3.csv には「金額」項目がないので、エラーとなる。ただし、dat1.csv の内容は既に出力されていることに注意する。

```
$ more dat3.csv
顧客,日付,数量
A,20081201,3
B,20081002,1
$ mcat i=dat1.csv,dat3.csv o=rsl2.csv
#ERROR# field name [金額] not found on file [dat3.csv] (kgcat)
$ more rsl2.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
```

例 3: **項目名の異なるファイルの併合** 2

前例に-nostop オプションを付けると、項目が見つからないデータについては NULL 値を出力するようになり、途中でエラー終了することはなくなる。その他にも、項目が見つからなかった場合の動作を変更するオプションとして、skip,force がある。詳しくはパラメータの説明を参照されたい。

```
$ more dat3.csv
顧客,日付,数量
A,20081201,3
B,20081002,1
$ mcat -nostop i=dat1.csv,dat3.csv o=rsl3.csv
#END# kgcat -nostop i=dat1.csv,dat3.csv o=rsl3.csv
$ more rsl3.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
A,20081201,
B,20081002,
```

例 4: **項目名を指定して併合**

f=で項目名を指定すると、それら指定した項目のみを併合する。

```
$ mcat f=顧客,日付 i=dat2.csv,dat3.csv o=rsl4.csv
#END# kgcat f=顧客,日付 i=dat2.csv,dat3.csv o=rsl4.csv
$ more rsl4.csv
顧客,日付
A,20081207
A,20081213
B,20081209
A,20081201
```

```
B,20081002
```

例 5: **標準入力の併合**

-stdin を指定することで、dat2.csv を標準入力から追加する。

```
$ mcat -stdin i=dat1.csv o=rsl5.csv <dat2.csv
#END# kgcat -stdin i=dat1.csv o=rsl5.csv
$ more rsl5.csv
顧客,日付,金額
A,20081207,20
A,20081213,30
B,20081209,50
A,20081201,10
B,20081002,40
```

例 6: **ファイル名項目を追加**

-add_fname を指定すると、元ファイルの名前を fileName 項目で追加する。標準入力のファイル名は/dev/stdin となる。

```
$ mcat -add_fname -stdin i=dat1.csv o=rsl6.csv <dat2.csv
#END# kgcat -add_fname -stdin i=dat1.csv o=rsl6.csv
$ more rsl6.csv
顧客,日付,金額,fileName
A,20081207,20,/dev/stdin
A,20081213,30,/dev/stdin
B,20081209,50,/dev/stdin
A,20081201,10,dat1.csv
B,20081002,40,dat1.csv
```

例 7: **ワイルドカード指定**

カレントディレクトリに dat1.csv,dat2.csv,dat3.csv の 3 つの CSV ファイルがあったとして、それらを全て併合するのにワイルドカード dat*.csv を指定する。

```
$ more dat1.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
$ more dat2.csv
顧客,日付,金額
A,20081207,20
A,20081213,30
B,20081209,50
$ more dat3.csv
顧客,日付,数量
A,20081201,3
B,20081002,1
$ mcat -force i=dat*.csv o=rsl7.csv
#END# kgcat -force i=dat*.csv o=rsl7.csv
$ more rsl7.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
A,20081207,20
A,20081213,30
B,20081209,50
A,20081201,3
```

```
B,20081002,1
```

例 8: **同一ファイルの複数回併合**

同一ファイルを複数指定することも可能である。

```
$ mcat i=dat1.csv,dat1.csv,dat1.csv o=rsl8.csv
#END# kgcat i=dat1.csv,dat1.csv,dat1.csv o=rsl8.csv
$ more rsl8.csv
顧客,日付,金額
A,20081201,10
B,20081002,40
A,20081201,10
B,20081002,40
A,20081201,10
B,20081002,40
```

関連コマンド

msep：ちょうど逆の動きをする。

4.7 mchgnum 数値範囲による置換

f=パラメータで指定した項目について、R=パラメータで指定する数値範囲条件と v=パラメータで指定する置換文字列により、項目の値を置換する。

書式

mchgnum f= R= [O=|-F] [v=] [-A] [-r] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の数値を R=と v=パラメータで指定した
数値範囲リストおよび置換文字列リストに従って置換する。

R= 置換対象となる数値範囲を指定 (複数項目指定可) する (1.1,2.5 : 1.1 以上 2.5 未満)。
最小値、最大値として MIN,MAX を使うことができる (MIN,2.5 : 2.5 未満)。

O= 範囲外文字列
R=パラメータで指定した数値範囲リストのいずれとも合致しない値を
置換するときの文字列 (指定がなければ NULL 値となる) を指定する。

-F 範囲外を項目の値として出力
R=パラメータで指定した数値範囲リストのいずれとも
合致しない値は、その項目の値のまま出力する。

v= R=パラメータで指定した数値範囲に対応する置換文字列を指定する。
R=で指定した値の個数より 1 つ少い個数でなければならない。

-A このオプションにより、指定した項目を置き換えるのではなく、新たに項目が追加される。

-r R=パラメータの範囲を'〜より大きく〜以下' として扱う。
例えば、1.1_2.5 は「1.1 より大きく 2.5 以下」として扱う。

利用例

例 1: 基本例

quantity 項目の値が最小以上 10 未満を low、10 以上 20 未満を middle、20 以上最大未満を high という文字列に置換する。

```
$ more dat1.csv
customer,quantity
A,5
B,10
C,15
D,2
E,50
$ mchgnum f=quantity R=MIN,10,20,MAX v=low,middle,high i=dat1.csv o=rsl1.csv
#END# kgchgnum R=MIN,10,20,MAX f=quantity i=dat1.csv o=rsl1.csv v=low,middle,high
$ more rsl1.csv
customer,quantity
A,low
B,middle
C,middle
D,low
E,high
```

例 2: **パラメータ範囲にイコールをつける例**

quantity 項目の値が最小より多く 10 以下を low、10 より多く 20 以下を middle、20 より多く最大以下を high という文字列に置換する。

```
$ mchgnum f=quantity R=MIN,10,20,MAX v=low,middle,high -r i=dat1.csv o=rsl2.csv
#END# kgchgnum -r R=MIN,10,20,MAX f=quantity i=dat1.csv o=rsl2.csv v=low,middle,high
$ more rsl2.csv
customer,quantity
A,low
B,low
C,middle
D,low
E,high
```

例 3: **数値範囲リストに合致しない値を置換**

quantity 項目の値が 10 以上 20 未満を low、20 以上 30 未満を middle、30 以上最大未満を high、数量が 10 より小さい値は out of range という文字列に置換する。

```
$ mchgnum f=quantity R=10,20,30,MAX v=low,middle,high O="out of range" i=dat1.csv o=rsl3.csv
#END# kgchgnum O=out of range R=10,20,30,MAX f=quantity i=dat1.csv o=rsl3.csv v=low,middle,high
$ more rsl3.csv
customer,quantity
A,out of range
B,low
C,low
D,out of range
E,high
```

例 4: **新たな項目の追加**

quantity 項目の値が最小以上 10 未満を low、10 以上 20 未満を middle、20 以上最大未満を high という文字列に置換し evaluate という項目名で出力する。

```
$ mchgnum f=quantity:evaluate R=MIN,10,20,MAX v=low,middle,high -A i=dat1.csv o=rsl4.csv
#END# kgchgnum -A R=MIN,10,20,MAX f=quantity:evaluate i=dat1.csv o=rsl4.csv v=low,middle,high
$ more rsl4.csv
customer,quantity,evaluate
A,5,low
B,10,middle
C,15,middle
D,2,low
E,50,high
```

例 5: **範囲外を項目の値として出力**

quantity 項目の値が 10 以上 20 未満を low、20 以上 30 未満を middle、30 以上最大未満を high、数量が 10 より小さい値は置換しないでそのまま出力する。

```
$ mchgnum f=quantity R=10,20,30,MAX v=low,middle,high -F i=dat1.csv o=rsl5.csv
#END# kgchgnum -F R=10,20,30,MAX f=quantity i=dat1.csv o=rsl5.csv v=low,middle,high
$ more rsl5.csv
customer,quantity
A,5
B,low
C,low
D,2
```

```
E,high
```

関連コマンド

mchgstr：文字列の置換であればこちら。
msed：正規表現を使った置換が可能。

4.8　mchgstr 文字列の置換

f=パラメータで指定した項目について、c=パラメータで指定した置換条件で文字列を置換する。

書式

mchgstr c= f= [O=] [-A] [-F] [-sub] [-W] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

c=	置換対象となる文字列と置換文字列を指定する。
f=	ここで指定した項目 (複数項目指定可) の文字列を c=パラメータで指定した置換条件リストに従って置換する。
O=	c=パラメータで指定した置換条件リストのいずれとも合致しない値を 置換するときの文字列 (指定がなければ NULL 値となる) を指定する。
-A	このオプションにより、指定した項目を置き換えるのではなく、新たに項目が追加される。
-F	c=パラメータで指定した置換条件リストのいずれとも合致しない値は、 その項目の値のまま出力する。
-sub	検索を完全一致ではなく部分文字列マッチで比較する すなわち、f=パラメータで指定した項目の値に、 c=パラメータで指定した置換条件で文字列を置換する。
-W	-sub オプションが指定されているときにワイド文字として部分文字列マッチをおこなう。

利用例

例 1: 基本例

item の値が"01"を"A"に、"03"を"B"に、"04"を"C"に置換する。その他は NULL 値として出力する。

```
$ more dat1.csv
id,item
1,01
2,02
3,03
4,04
5,05
$ mchgstr f=item c=01:A,03:B,05:C i=dat1.csv o=rsl1.csv
#END# kgchgstr c=01:A,03:B,05:C f=item i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,item
1,A
2,
3,B
4,
5,C
```

例 2: 条件に合致しない値を置換する文字列の指定

O=パラメータを指定することで、置換条件に合致しない場合は"out of range"という文字列に置換して出力する。

```
$ mchgstr f=item c=01:A,03:B,05:C O="out of range" i=dat1.csv o=rsl2.csv
#END# kgchgstr O=out of range c=01:A,03:B,05:C f=item i=dat1.csv o=rsl2.csv
$ more rsl2.csv
```

```
id,item
1,A
2,out of range
3,B
4,out of range
5,C
```

例 3: **新しい項目として出力**

-A オプションを付けることで、新しい項目 (item info) として出力する。

```
$ mchgstr f=item:"item info" c=01:A,03:B,05:C O="out of range" -A i=dat1.csv o=rsl3.csv
#END# kgchgstr -A O=out of range c=01:A,03:B,05:C f=item:item info i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,item,item info
1,01,A
2,02,out of range
3,03,B
4,04,out of range
5,05,C
```

例 4: **条件外を項目の値として出力**

-F オプションを付けることで、置換条件に合致しない場合は、元の値をそのまま出力する。

```
$ mchgstr f=item c=01:A,03:B,05:C -F i=dat1.csv o=rsl4.csv
#END# kgchgstr -F c=01:A,03:B,05:C f=item i=dat1.csv o=rsl4.csv
$ more rsl4.csv
id,item
1,A
2,02
3,B
4,04
5,C
```

例 5: **条件を部分文字列マッチで置換**

-sub オプションをつけることで、部分文字列の置換となる。以下の例では、item 項目に文字列"01"が含まれていれば、それを"A"に置換する。

```
$ more dat2.csv
id,item
1,0111
2,0121
3,0231
4,0241
5,0151
$ mchgstr f=item c=01:A -sub i=dat2.csv o=rsl5.csv
#END# kgchgstr -sub c=01:A f=item i=dat2.csv o=rsl5.csv
$ more rsl5.csv
id,item
1,A11
2,A21
3,
4,
5,A51
```

例 6: ワイド文字での部分文字列マッチ

ワイド文字の部分文字列置換をする場合は-W オプションを用いる。ただし、UTF-8 エンコーディングを用いているのであれば-W をつけなくても正しく動作する。詳しくは「マルチバイト文字」の節を参照されたい。

```
$ more dat3.csv
id,city
1,奈良市
2,下市町
3,十津川村
4,五條市
5,山添村
$ mchgstr f=city c=市:01,町:02,村:02 -sub -W i=dat3.csv o=rsl6.csv
#END# kgchgstr -W -sub c=市:01,町:02,村:02 f=city i=dat3.csv o=rsl6.csv
$ more rsl6.csv
id,city
1,奈良01
2,下0102
3,十津川02
4,五條01
5,山添02
```

関連コマンド

mchgnum：数値範囲による置換ならばこちら。

msed：正規表現による置換が可能。

4.9 mchkcsv CSV データのチェック

MCMD が前提とする CSV の仕様を満たしていないデータを自動修復する (項目数の統一など)。また、-diag オプションを指定することで、CSV データのチェックのみ実行する。

書式

mchkcsv [a=] [-diagl-diag] [-r]— [i=] [o=] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

i=	入力ファイル名 このパラメータで指定した CSV データに不備な箇所がないかチェックを行い、 不備がある場合その箇所を自動的に補完する。 このパラメータが省略された時には標準入力が用いられる。
a=	入力データの項目名を無視し、ここで指定した項目名を出力する。 ここで指定した項目数が入力データの項目数より少ない場合は、 入力データの左から順番に指定の個数分の項目が出力される。 逆に、ここで指定した項目数が入力データの項目数より多い場合、不足分は NULL 値が出力される。
-diagl	このオプションを指定した場合、データの修正/補完は行わず、 データ項目の統計情報を出力するとともに、CSV お妥当性チェックの結果を標準出力に出力する。 チェックする内容は以下の通り。 a): 同一項目名がないか。 b): 項目名に不正な文字がないか。 c): 改行が LF か。 d): 最終行に改行が存在するか。 e): データファイル内に\0 がないか。 f): 一行の最大長を超過していないか。 g): 全行同じ項目数か。 h): 制御文字 (0x01 0x1F,0x7F) が入り込んでいないか。 i): TAB が入っていないか。 j): DQUOTE で囲われていない中で DQUOTE がない。 k): DQUOTE で囲われている中で単一の DQUOTE がないか。 l): 先頭に BOM(Bit Order Mark) が含まれていないか。
-diag	-diagl の英語版
-r	制御文字を無視 ここでは ASCII 文字コードの 0x00〜0x1f,0x7f(0x09,0x0a,0x0d は除く) を制御文字と定義している。 このオプションを指定しなければ制御コードは自動的に&#x という文字列に変換される。

利用例

例 1: データの補完

データの項目数が違う（2,4 行目が 3 項目しかない）問題のあるデータを M コマンドで利用できるデータに補完（4 項目に自動的に補完）する。

```
$ more dat1.csv
商品，日付，数量，金額
A,20081201,1,10
A,20081202,2,
```

```
A,*,3
B,20081201,4,40
B,20081203,50
$ mchkcsv i=dat1.csv o=rsl1.csv
#END# kgchkcsv i=dat1.csv o=rsl1.csv
$ more rsl1.csv
商品，日付，数量，金額
A,20081201,1,10
A,20081202,2,
A,*,3,
B,20081201,4,40
B,20081203,50,
```

例 2: **データの補完、項目名変更**

データの項目数が違う（3,5 行目が 3 項目しかない）問題のあるデータを M コマンドで利用できるデータに補完（4 項目に自動的に補完）する。その際に、入力データの左の項目から順番に「**商品，日付，数量，金額**」という項目名で出力する。

```
$ more dat2.csv
fld1,fld2,fld3,fld4
A,20081201,1,10
A,20081202,2,
A,*,3
B,20081201,4,40
B,20081203,50
$ mchkcsv a=商品，日付，数量，金額 i=dat2.csv o=rsl2.csv
#END# kgchkcsv a=商品，日付，数量，金額 i=dat2.csv o=rsl2.csv
$ more rsl2.csv
商品，日付，数量，金額
A,20081201,1,10
A,20081202,2,
A,*,3,
B,20081201,4,40
B,20081203,50,
```

例 3: **データの不備チェック、診断結果の出力**

CSV データに不備な箇所がないかチェックのみを行い、CSV ファイル診断結果を出力する。

```
$ mchkcsv -diagl i=dat1.csv o=rsl3.csv
#END# kgchkcsv -diagl i=dat1.csv o=rsl3.csv
$ more rsl3.csv
#=================================================
# CSV ファイル診断
# file name : dat1.csv
#-------------------------------------------------
# 結果の一文字目の意味
# # : 情報行（問題なし）
# ? : KGMOD にて扱えない問題点（?の後の文字は解説参照）
# よって、左端が全て#になれば OK
#=================================================
############################## ヘッダー情報（1 行目） ###
# 項目数 : 4
# 項目 No.  項目名
# 1   商品
# 2   日付
# 3   数量
# 4   金額
#
############## EOL(End Of Line) 情報（ヘッダー含む） ##
```

```
#         LF 改行行数 : 6 (LineNo: 0 1 2 ... )
#
################## データ行情報（ヘッダー含まない）###
#            総行数 : 5
#         総バイト数 : 66
#            平均長 : 13.2
#            最大長 : 16 (LineNo:2)
#            最小長 : 6 (LineNo:4)
# 注:長さは、改行文字も含めた長さ
#
################################## 項目数の一貫性 ###
?g 異なる項目数の行が発見されました。
?g  項目数:3 (LineNo:4)
?g  項目数:3 (LineNo:6)
#
######################################## 項目情報 ###
#  項目番号 [1] 項目名 [商品]
#    NULL 値の行数                 : 0
#    DQUOTE で囲われていない行数   : 5 (LineNo: 1 2 3 ... )
#    DQUOTE で囲われている行数     : 0
#
#  項目番号 [2] 項目名 [日付]
#    NULL 値の行数                 : 0
#    DQUOTE で囲われていない行数   : 5 (LineNo: 1 2 3 ... )
#    DQUOTE で囲われている行数     : 0
#
#  項目番号 [3] 項目名 [数量]
#    NULL 値の行数                 : 0
#    DQUOTE で囲われていない行数   : 5 (LineNo: 1 2 3 ... )
#    DQUOTE で囲われている行数     : 0
#
#  項目番号 [4] 項目名 [金額]
#    NULL 値の行数                 : 1 (LineNo: 2 )
#    DQUOTE で囲われていない行数   : 3 (LineNo: 1 2 4 )
#    DQUOTE で囲われている行数     : 0
#
################################### 問題点の解説 ###
# ?a : 同じ項目名があると項目番号を特定できない。
#   【対処方法】kgchkcsv a=x,y,z のように項目名を新たに指定する。
# ?b : 項目名に不正な文字があるとエラーになる
#   【対処方法】kgchkcsv a=x,y,z のように項目名を新たに指定する。
# ?c : KGMOD が扱う改行は高速化のため LF(UNIX 改行) のみ。
#       この問題は RFC4180 には準拠しておらず KGMOD 独自の制約である。
#   【対処方法】kgchkcsv にて全て LF に変換される。
#
# ?d : 最終行に LF や CR などの改行 (EOL) 文字が存在しない。
#       これは RFC4180 にも準拠していない。
#   【対処方法】kgchkcsv にて LF が付加される。
#
# ?e : データファイル内に'\0' が入り込んでいる。
#       テキストファイルでない可能性が高い。
#       RFC4180 には準拠していない。
#   【対処方法】kgchkcsv にて、"&#x00
#                kgchkcsv -r にて\0 は削除される。
#
# ?f : KGMOD が扱える一行の最大長を超過している。
#       現在の設定では 1024000 バイト以上の長さの行は扱えない。
#   【対処方法】環境変数を設定することで最大値を変更可能である。
#         ex) export KG_MaxRecLen=204800
#       ただし 10240000 バイトを越えては指定できない。
#       この問題は RFC4180 には準拠しており KGMOD 独自の制約である。
#
# ?g : KGMOD では全行同じ項目数を前提とする。
#       この問題は RFC4180 には準拠しており KGMOD 独自の制約である。
```

```
#   【対処方法】
#        1) kgchkcsv   データ HEADER の項目数に合わせる.
#             超過項目は捨てられ、足りない項目は null 値となる
#        2) kgchkcsv  a=x,y,z HEADER 行をスキップし、
#             指定した x,y,z を項目名として 1) と同様の処理を行う。
#
# ?h : 制御文字 (0x01~0x1F,0x7F) が項目値として入り込んでいる。
#      テキストファイルでない可能性が高い。
#      RFC4180 には準拠していない。
#   【対処方法】kgchkcsv にて、"&#x01
#               kgchkcsv -r にて制御文字は削除される。
#
# ?i : TAB は利用できない。
#      RFC4180 には準拠していない。
#   【対処方法】kgchksv にて、"&#x09
#               kgchkcsv -r にて TAB は削除される。
#
# ?j : DQUOTE で囲われていない中で DQUOTE が見つかった
#        ex) NG: xxx,oo"oo,xxx  -> OK: xxx,"oo""oo",xxx
#      RFC4180 には準拠していない。
#   【対処方法】kgchkcsv にて上記の変換を行う。
#
# ?k : DQUOTE で囲われている中で単一の DQUOTE が見つかった
#      ex) NG: xxx,"oo"oo",xxx  -> OK: xxx,"oo""oo",xxx
#      RFC4180 には準拠していない。
#   【対処方法】kgchkcsv にて上記の変換を行う。
# ?l : 先頭に BOM が含まれている
#   【対処方法】kgchkcsv にて BOM は除去される。
#-------------------------------------------------------------
```

関連コマンド

4.10 mcombi 組合せ計算

f=パラメータで指定した項目について、n=パラメータで指定した数の組み合せを求め、a=パラメータで指定した項目名で出力する。-p を指定することで順列として出力することも可能である。

書式

mcombi a= f= n= [s=] [k=] [-p] [-dup] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a=	新たに追加される項目の名前を指定する。
f=	組合せを求める項目名リスト（複数項目指定可）を指定する。 ここで指定した項目の値の全組合せを出力する。
n=	組合せの数を指定する。 組み合わせ数を大きくすると、出力レコードが爆発的に大きくなることに注意する。
s=	ここで指定した項目 (複数項目指定可) で並べ替えられた後、f=で指定した項目の組み合わせを求める。
k=	キー項目名リスト (複数項目指定可) 組合せを求める単位となる項目名リスト。
-p	組合せでなく順列を求める。
-dup	同一の値の組み合せも出力する

利用例

例 1: 基本例

customer 項目を単位に、item 項目の 2 アイテムの組み合わせを求め、item1,item2 という項目名で出力する。k=,f=で指定していない項目 (ここでは item 項目) は、キーの最終行の値が出力される。

```
$ more dat1.csv
customer,item
A,a1
A,a2
A,a3
B,a4
B,a5
$ mcombi k=customer f=item n=2 a=item1,item2 i=dat1.csv o=rsl1.csv
#END# kgcombi a=item1,item2 f=item i=dat1.csv k=customer n=2 o=rsl1.csv
$ more rsl1.csv
customer%0,item,item1,item2
A,a3,a1,a2
A,a3,a1,a3
A,a3,a2,a3
B,a5,a4,a5
```

例 2: 基本例 2

-dup オプションを指定すると同一のアイテムの組み合せも出力される。

```
$ mcombi k=customer f=item n=2 a=item1,item2 i=dat1.csv o=rsl2.csv -dup
#END# kgcombi -dup a=item1,item2 f=item i=dat1.csv k=customer n=2 o=rsl2.csv
$ more rsl2.csv
```

```
customer%0,item,item1,item2
A,a3,a1,a1
A,a3,a1,a2
A,a3,a1,a3
A,a3,a2,a2
A,a3,a2,a3
A,a3,a3,a3
B,a5,a4,a4
B,a5,a4,a5
B,a5,a5,a5
```

例 3: **順列を求める例**

customer 項目を単位に、item 項目の 2 アイテムの順列を求め、item1,item2 という項目名で出力する。

```
$ mcombi k=customer f=item n=2 a=item1,item2 -p i=dat1.csv o=rsl3.csv
#END# kgcombi -p a=item1,item2 f=item i=dat1.csv k=customer n=2 o=rsl3.csv
$ more rsl3.csv
customer%0,item,item1,item2
A,a3,a1,a2
A,a3,a2,a1
A,a3,a1,a3
A,a3,a3,a1
A,a3,a2,a3
A,a3,a3,a2
B,a5,a4,a5
B,a5,a5,a4
```

例 4: **順列を求める例**

item 項目を降順に並べ替えた後、item 項目の 2 アイテムの順列を求める。

```
$ mcombi k=customer f=item n=2 s=item%r a=item1,item2 -p i=dat1.csv o=rsl4.csv
#END# kgcombi -p a=item1,item2 f=item i=dat1.csv k=customer n=2 o=rsl4.csv s=item%r
$ more rsl4.csv
customer%0,item%1r,item1,item2
A,a1,a3,a2
A,a1,a2,a3
A,a1,a3,a1
A,a1,a1,a3
A,a1,a2,a1
A,a1,a1,a2
B,a4,a5,a4
B,a4,a4,a5
```

関連コマンド

4.11 mcommon 参照ファイルによる行選択

k=パラメータで指定した入力ファイルの項目値と m=パラメータで指定した参照ファイルの項目値を比較し、同じ値を持つ入力ファイルの行を選択する。

書式

mcommon k= [K=] [u=] [-r] m=| i= [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= 入力データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した入力データの項目と K=パラメータで指定された参照データの項目が同じ行が選択される。
同じ値が複数行連続していてもよい。

m= 参照ファイル名を指定する。
またこのパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

K= 参照データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行が選択される。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。
同じ値が複数行連続していてもよい。

u= 指定の条件に一致しない行を出力するファイル名。

-r 条件反転
k=パラメータで指定した入力ファイルの項目値と
m=パラメータで指定した参照ファイルの項目値を比較し、
同じ値を持たない入力ファイルの行を選択する。

利用例

例 1: 基本例

入力ファイルにある「顧客」項目と、参照ファイルにある「顧客」項目が同じ値を持つ入力ファイルの行を選択する。それ以外のデータは oth.csv に出力する。

```
$ more dat1.csv
顧客,数量
A,1
B,2
C,1
D,3
E,1
$ more ref1.csv
顧客,性別
A,女性
B,男性
E,女性
$ mcommon k=顧客 m=ref1.csv u=oth.csv i=dat1.csv o=rsl1.csv
#END# kgcommon i=dat1.csv k=顧客 m=ref1.csv o=rsl1.csv u=oth.csv
$ more rsl1.csv
顧客 %0,数量
A,1
B,2
E,1
$ more oth.csv
```

```
顧客 %0, 数量
C,1
D,3
```

例 2: 同じ値を持たない入力ファイルの行選択

-r オプションを付けることで、条件が逆転し、参照ファイルにない「顧客」を選択することになる。

```
$ mcommon k=顧客 m=ref1.csv -r i=dat1.csv o=rsl2.csv
#END# kgcommon -r i=dat1.csv k=顧客 m=ref1.csv o=rsl2.csv
$ more rsl2.csv
顧客 %0, 数量
C,1
D,3
```

例 3: 結合キー項目名が異なる場合

結合キーの項目名が異なる場合は、K=で指定する。

```
$ more ref2.csv
顧客 ID, 性別
A, 女性
B, 男性
E, 女性
$ mcommon k=顧客 K=顧客 ID i=dat1.csv m=ref2.csv o=rsl3.csv
#END# kgcommon K=顧客 ID i=dat1.csv k=顧客 m=ref2.csv o=rsl3.csv
$ more rsl3.csv
顧客 %0, 数量
A,1
B,2
E,1
```

例 4: キー項目に重複行がある場合の例

参照ファイルと入力ファイルのキー項目に重複行があっても選択可能。

```
$ more dat3.csv
顧客, 数量
A,1
A,2
A,3
B,1
D,1
D,2
$ more ref3.csv
顧客
A
A
D
$ mcommon k=顧客 m=ref3.csv -r i=dat3.csv o=rsl4.csv
#END# kgcommon -r i=dat3.csv k=顧客 m=ref3.csv o=rsl4.csv
$ more rsl4.csv
顧客 %0, 数量
B,1
```

関連コマンド

mselstr：参照ファイルの結合キーの種類数が少なければこのコマンドでも対応できる。

mnrcommon：参照ファイルの結合キーがユニークでなければこちらを使う。
mjoin：選択だけでなく、項目を結合したい場合はこのコマンド。

4.12　mcount 行数カウント

行数をカウントし、a=パラメータで指定した項目名で出力する。k=を指定すると、集計キー毎の件数をカウントし、k=を指定しなければ、全行数がカウントされる。

書式

mcount a= [k=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a=　新たに追加される項目の名前を指定する。
　　nfn オプション使用時は、必須ではない。
k=　キー項目名リスト (複数項目指定可)
　　カウントの単位となる項目名リスト。

利用例

例 1: 基本例

date 項目を単位に行数をカウントし、count という項目名で出力する。

```
$ more dat1.csv
date
20090109
20090109
20090110
20090109
20090110
$ mcount k=date a=count i=dat1.csv o=rsl1.csv
#END# kgcount a=count i=dat1.csv k=date o=rsl1.csv
$ more rsl1.csv
date%0,count
20090109,3
20090110,2
```

例 2: 集計キーなし

集計キーを指定しなければ全体の行数をカウントする。

```
$ mcount a=count i=dat1.csv o=rsl2.csv
#END# kgcount a=count i=dat1.csv o=rsl2.csv
$ more rsl2.csv
date,count
20090110,5
```

関連コマンド

mstats：c=count を指定することで、NULL 値でないデータ件数をカウントできる。

4.13 mcross クロス集計

クロス集計を行う。s=で指定した項目の値が項目名となるように横に展開され、k=で指定した値が行 id となり、f=で指定した項目がセルとして出力される。

書式

mcross f= s= [a=] [k=] [v=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定された項目の値がセルの値として出力される。
複数項目指定すると、複数行に展開される。
それら複数行を識別するための項目として fld 項目が出力され、
f=で指定した項目名が値として出力される。
この fld という項目名を変更したい場合は a=パラメータを使う。

s= 列項目名に展開する項目を指定する。
ここで指定された項目の値が項目名として出力される。

a= f=で指定した項目名がデータとして展開される項目名を指定する。
省略した場合は fld という項目名で出力される。

k= キー項目名リスト
ここで指定した項目を単位に横展開をおこなう。

v= NULL 値置換文字列
NULL 値があった場合、v=パラメータで指定する置換文字列により、項目の値を置換する。

利用例

例 1: 基本例

item 項目を単位に date 項目を横に展開し、quantity 項目を出力する。

```
$ more dat1.csv
item,date,quantity,price
A,20081201,1,10
A,20081202,2,20
A,20081203,3,30
B,20081201,4,40
B,20081203,5,50
$ mcross k=item f=quantity s=date i=dat1.csv o=rsl1.csv
#END# kgcross f=quantity i=dat1.csv k=item o=rsl1.csv s=date
$ more rsl1.csv
item%0,fld,20081201,20081202,20081203
A,quantity,1,2,3
B,quantity,4,,5
```

例 2: 元の入力データに戻す例

例 1 の出力結果を元に戻すには、同じく mcross を以下のよう用いればよい。

```
$ more rsl1.csv
item%0,fld,20081201,20081202,20081203
A,quantity,1,2,3
```

```
B,quantity,4,,5
$ mcross k=item f=2008* s=fld a=date i=rsl1.csv o=rsl2.csv
#END# kgcross a=date f=2008* i=rsl1.csv k=item o=rsl2.csv s=fld
$ more rsl2.csv
item%0,date,quantity
A,20081201,1
A,20081202,2
A,20081203,3
B,20081201,4
B,20081202,
B,20081203,5
```

例 3: 複数の値を出力

quantity,price の 2 項目を出力する。

```
$ mcross k=item f=quantity,price s=date i=dat1.csv o=rsl3.csv
#END# kgcross f=quantity,price i=dat1.csv k=item o=rsl3.csv s=date
$ more rsl3.csv
item%0,fld,20081201,20081202,20081203
A,quantity,1,2,3
A,price,10,20,30
B,quantity,4,,5
B,price,40,,50
```

例 4: 並びを逆順する例

横に展開する項目名の並びを逆順にする。

```
$ mcross k=item f=quantity,price s=date%r i=dat1.csv o=rsl4.csv
#END# kgcross f=quantity,price i=dat1.csv k=item o=rsl4.csv s=date%r
$ more rsl4.csv
item%0,fld,20081203,20081202,20081201
A,quantity,3,2,1
A,price,30,20,10
B,quantity,5,,4
B,price,50,,40
```

関連コマンド

mtra：横展開するイメージは同じだが、mtra は 1 つのベクトル項目として出力する。

4.14 m2cross 1 対 N のクロス集計

1 対 N のクロス集計を行う。s=を指定した場合には項目の値が項目名となるように横に展開され、f=で指定した項目がセルとして出力される。a=を指定した場合 (2 項目指定) には指定した値が項目名となり、1 項目に f=で指定した項目名が、2 項目に f=で指定した項目値がそれぞれ縦展開される k=が指定されていた場合には、指定した値が行 id となり、id 単位で展開される。

書式

m2cross f= s=|a= [k=] [v=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定された項目の値がセルの値として出力される。
a=を使用するときのみ複数項目指定可。

s= 列項目名に展開する項目を指定する。
ここで指定された項目の値が項目名として出力される。

a= 2 項目指定する。
1 項目目に f=で指定した項目名がデータとして展開される項目名を指定する。
2 項目目に f=で指定した項目値の項目名を指定する

k= キー項目名リスト
ここで指定した項目を単位に展開をおこなう。

v= NULL 値置換文字列
NULL 値があった場合、v=パラメータで指定する置換文字列により、項目の値を置換する。

利用例

例 1: 基本例

item 項目を単位に date 項目を横に展開し、quantity 項目を出力する。

```
$ more dat1.csv
item,date,quantity
A,20081201,1
A,20081202,2
A,20081203,3
B,20081201,4
B,20081203,5
$ m2cross k=item f=quantity s=date i=dat1.csv o=rsl1.csv
#END# kg2cross f=quantity i=dat1.csv k=item o=rsl1.csv s=date
$ more rsl1.csv
item%0,20081201,20081202,20081203
A,1,2,3
B,4,,5
```

例 2: 元の入力データに戻す例

例 1 の出力結果を元に戻すには、同じく m2cross を以下のよう用いればよい。

```
$ more rsl1.csv
item%0,20081201,20081202,20081203
```

```
A,1,2,3
B,4,,5
$ m2cross f=2008* a=date,quantity i=rsl1.csv o=rsl2.csv
#END# kg2cross a=date,quantity f=2008* i=rsl1.csv o=rsl2.csv
$ more rsl2.csv
item%0,date,quantity
A,20081201,1
A,20081202,2
A,20081203,3
B,20081201,4
B,20081202,
B,20081203,5
```

例 3: 並びを逆順する例

横に展開する項目名の並びを逆順にする。

```
$ m2cross k=item f=quantity s=date%r i=dat1.csv o=rsl4.csv
#END# kg2cross f=quantity i=dat1.csv k=item o=rsl4.csv s=date%r
$ more rsl4.csv
item%0,20081203,20081202,20081201
A,3,2,1
B,5,,4
```

関連コマンド

mcross：イメージは同じだが、mcross は N 対 N クロス集計として出力する。

4.15 mcsv2arff CSV を ARFF 形式に変換

csv 形式のデータから arff 形式 (WEKA 用のデータフォーマット) のデータへ変換する。arff では、属性の型を指定する必要があり、d=でカテゴリ型項目を、n=で数値型項目を、s=で文字列型項目を、そして D=で日付型項目をそれぞれ指定する。日付型項目名に %t を付ければ時刻を含んだ値と見なし、付けなければ日付のみの値と見なす。

書式

mcsv2arff n=|d=|D=|s= [T=] i= [o=] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

n= 数値項目名 (複数項目指定可) を指定する。
d= カテゴリ項目名 (複数項目指定可) を指定する。
D= 日付 (時刻) 項目名 (複数項目指定可) リストを指定する。 [%t]
%t を指定しなかった場合：yyyyMMdd
%t を指定した場合 ：yyyyMMddHHmmss
s= 文字列項目名 (複数項目指定可) を指定する。
T= タイトルにする文字列を指定する。

利用例

例 1: csv 形式のデータを arff 形式のデータへ変換

「顧客」項目は文字列型、「商品」項目はカテゴリ型、「日付」項目は日付型 (時刻は含まない)、そして「数量」と「金額」項目は数値型として、arff 形式のデータへ変換する。

```
$ more dat1.csv
顧客,商品,日付,数量,金額
No.1,A,20081201,1,10
No.2,A,20081202,2,20
No.3,A,20081203,3,30
No.4,B,20081201,4,40
No.5,B,20081203,5,50
$ mcsv2arff s=顧客 d=商品 D=日付 n=数量,金額 T=顧客購買データ i=dat1.csv  o=rsl1.csv
#END# kgcsv2arff D=日付 T=顧客購買データ d=商品 i=dat1.csv n=数量,金額 o=rsl1.csv s=顧客
$ more rsl1.csv
@RELATION        顧客購買データ

@ATTRIBUTE        顧客        string
@ATTRIBUTE        日付        date yyyyMMdd
@ATTRIBUTE        数量        numeric
@ATTRIBUTE        金額        numeric
@ATTRIBUTE        商品        {A,B}

@DATA
No.1,20081201,1,10,A
No.2,20081202,2,20,A
No.3,20081203,3,30,A
No.4,20081201,4,40,B
No.5,20081203,5,50,B
```

例 2: csv 形式のデータを arff 形式のデータへ変換 (時刻を含む日付データ指定)

時刻を伴うデータは D=**日付** %t のように %t を加えて指定する。

```
$ more dat2.csv
顧客，商品，日付，数量，金額
No.1,A,20081201102030,1,10
No.2,A,20081202123010,2,20
No.3,A,20081203153010,3,30
No.4,B,20081201174010,4,40
No.5,B,20081203133010,5,50
$ mcsv2arff s=顧客 d=商品 D=日付 %t n=数量，金額 T=顧客購買データ i=dat2.csv  o=rsl2.csv
#END# kgcsv2arff D=日付 %t T=顧客購買データ d=商品 i=dat2.csv n=数量，金額 o=rsl2.csv s=顧客
$ more rsl2.csv
@RELATION         顧客購買データ

@ATTRIBUTE         顧客         string
@ATTRIBUTE         日付         date yyyyMMddHHmmss
@ATTRIBUTE         数量         numeric
@ATTRIBUTE         金額         numeric
@ATTRIBUTE         商品         {A,B}

@DATA
No.1,20081201102030,1,10,A
No.2,20081202123010,2,20,A
No.3,20081203153010,3,30,A
No.4,20081201174010,4,40,B
No.5,20081203133010,5,50,B
```

関連コマンド

marff2csv：逆変換

参考資料

http://weka.wikispaces.com/ARFF

4.16 mcsv2json CSV を JSON 形式へ変換

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

CSV データを JavaScript のオブジェクト表記構文である JSON 形式に変換する。本コマンドは以下のような特徴を持つ。詳細は例を参照のこと。

- JSON 上のデータ型は文字列のみに対応
- 配列とオブジェクト (key-value のペア) による出力が可能
- キー項目を指定することで、配列の入れ子構造で出力可能

書式

`mcsv2json [k=] [s=] f=|h=|p= -flat i= [o=] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]`

パラメータ

`k=` JSON 上で配列をネストさせる項目名リスト。
3 つの項目を指定すれば、3 重の JSON 配列となる。

`s=` 値を並べる順序項目。`%n`(数値順),`%r`(逆順) も指定可能。

`f=` 指定した項目の値を JSON の配列として出力する

`h=` 指定した項目名をキーにした JSON オブジェクト (hash 構造) として出力する。

`p=` JSON オブジェクトのキーと値の項目名を 2 つ指定する。
2 項目は次のようにコロンで区切る。`p=key` **項目名** `1:value` **項目名** `1,key` **項目名** `2:value` **項目名** `2,...`

利用例

例 1: **配列として出力する例**

```
$ more dat1.csv
key1,key2,v1,v2
A,X,1,a
A,Y,2,b
A,Y,3,c
B,X,4,d
B,Y,5,e
$ mcsv2json f=v1,v2 i=dat1.csv
[["1","a"],["2","b"],["3","c"],["4","d"],["5","e"]]
#END# kgcsv2json f=v1,v2 i=dat1.csv
```

例 2: **オブジェクト** (key-value) **として出力する例**

```
$ mcsv2json h=v1,v2 i=dat1.csv
[{"v1":"1","v2":"a"},{"v1":"2","v2":"b"},{"v1":"3","v2":"c"},{"v1":"4","v2":"d"},{"v1":"5","v2":"e"}]
#END# kgcsv2json h=v1,v2 i=dat1.csv
```

例 3: **項目指定によってオブジェクトとして出力する例**

```
$ mcsv2json p=v2:v1 i=dat1.csv
[{"a":"1"},{"b":"2"},{"c":"3"},{"d":"4"},{"e":"5"}]
```

```
#END# kgcsv2json i=dat1.csv p=v2:v1
```

例 4: **キー項目を指定する例**

key1 項目が A の 3 行が一つの配列として出力され、続いて key1=B の 2 行が一つの配列として出力される。

```
$ mcsv2json k=key1 f=v1 i=dat1.csv
[[["1"],["2"],["3"]],
[["4"],["5"]]]
#END# kgcsv2json f=v1 i=dat1.csv k=key1
```

例 5: **キー項目のネスト例**

key1=A かつ key2=X の 1 行が一つの配列として出力され、key1=A かつ key2=Y の 2 行が一つの配列として出力され、それら 2 つの配列 (すなわち key1=A の行) がさらに一つの配列として括られる。

```
$ mcsv2json k=key1,key2 f=v1 i=dat1.csv
[[[["1"]],
[["2"],["3"]]],
[[["4"]],
[["5"]]]]
#END# kgcsv2json f=v1 i=dat1.csv k=key1,key2
```

例 6: **行を配列で括らずにフラットに出力する例**

```
$ mcsv2json f=v1,v2 -flat i=dat1.csv
["1","a","2","b","3","c","4","d","5","e"]
#END# kgcsv2json -flat f=v1,v2 i=dat1.csv
```

関連コマンド

参考資料

http://www.rfc-editor.org/rfc/rfc4627.txt

4.17 mcsvconv CSV を多様なフォーマットに変換

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

CSV データを多様なフォーマットに変換する。変換したいデータフォーマットのルールを記述したテキストファイルを用意し、ルールキーワードに従って指定した CSV のデータ項目を出力する。ルールファイルに記述できる項目名指定や出力タイミング指定は以下のとおりである。

項目名キーワード 項目名を % で括ると、CSV 上の対応する項目の値に置き換えられる。ex. **% 項目名 %**

出力タイミングキーワード CSV データの出力タイミングを決めるためのキーワードは以下の 3 種類ある。

LINEDATA %LINEDATA〜%LINEEND で囲われた項目名キーワード及びその他の文字列は、CSV の各行が読み込まれる度に出力される。

KEYHEAD %KEYHEAD〜%KEYEND で囲われた項目名キーワード及びその他の文字列は、k=で指定されたキー項目がキーブレイクした時にのみ出力される。LINEDATA より前に出力される。

KEYFOOT %KEYFOOT〜%KEYEND で囲われた項目名キーワード及びその他の文字列は、k=で指定されたキー項目がキーブレイクした時にのみ出力される。LINEDATA より後に出力される。

キーワド前後 上述の出力タイミングキーワードで囲われたブロックが出現する前の文字列、および最後のブロック以降の文字列は、CSV の読み込み前、および読み込み後に一度だけ出力される。

k=で複数の項目を指定した場合、KEYHAD と KEYFOOT は更に細かなタイミング制御も可能である。%KEYHEAD1 のように、番号を後ろに付けることで、k=の何番目の項目がキーブレイクしたかを指定できる。例えば、k=A,B,C において %KEYHEAD1 であれば、キー項目 A がキーブレイクしたタイミングで出力され、また %KEYHEAD2 であれば、キー項目 A,B がキーブレイクしたタイミングで出力される。そして %KEYHEAD3 もしくは単に %KEYHEAD であれば、全てのキー項目 A,B,C がキーブレイクしたタイミングで出力される。

出力タイミング制御キーワードを記載した行は、それ以外の文字を記述することはできない。

書式

mcsvconv [k=] [s=] m= i= [o=] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= キー項目名リスト。
s= k=で指定した項目の値が同じ行の中での並べ替え順を決める項目名リスト。
m= ルールファイル名

利用例

例 1: **基本例** 1

CSV の 2 項目 key1,v1 をスペース区切りで出力する。

```
$ more dat1.csv
key1,key2,v1,v2
A,X,1,a
A,Y,2,b
A,Y,3,c
B,X,4,d
```

```
B,Y,5,e
$ more form1.txt
%LINEDATA
%key1% %v1%
%LINEEND
$ mcsvconv m=form1.txt i=dat1.csv o=rsl1.txt
#END# kgcsvconv i=dat1.csv m=form1.txt o=rsl1.txt
$ more rsl1.txt
A 1
A 2
A 3
B 4
B 5
```

例 2: **基本例** 2

2 行にズラして出力する。

```
$ more dat1.csv
key1,key2,v1,v2
A,X,1,a
A,Y,2,b
A,Y,3,c
B,X,4,d
B,Y,5,e
$ more form2.txt
%LINEDATA
%key1% %v1%
       %key2% %v2%
%LINEEND
$ mcsvconv m=form2.txt i=dat1.csv o=rsl2.txt
#END# kgcsvconv i=dat1.csv m=form2.txt o=rsl2.txt
$ more rsl2.txt
A 1
       X a
A 2
       Y b
A 3
       Y c
B 4
       X d
B 5
       Y e
```

例 3: key **を指定した例**

```
$ more dat1.csv
key1,key2,v1,v2
A,X,1,a
A,Y,2,b
A,Y,3,c
B,X,4,d
B,Y,5,e
$ more form3.txt
Head Area
%KEYHEAD
KeyHead=%key1% %key2% %v1% %v2%
%KEYEND
%LINEDATA
%v1%-%v2%
%LINEEND
%KEYFOOT
```

```
KeyFoot=%key1% %key2% %v1% %v2%
%KEYEND
Foot Area
$ mcsvconv k=key1 m=form3.txt i=dat1.csv o=rsl3.txt
#END# kgcsvconv i=dat1.csv k=key1 m=form3.txt o=rsl3.txt
$ more rsl3.txt
Head Area
KeyHead=A X 1 a
1-a
2-b
3-c
KeyFoot=A Y 3 c
KeyHead=B X 4 d
4-d
5-e
KeyFoot=B Y 5 e
Foot Area
```

例 4: tex **データとして出力する例**

```
$ more dat1.csv
key1,key2,v1,v2
A,X,1,a
A,Y,2,b
A,Y,3,c
B,X,4,d
B,Y,5,e
$ more form3.txt
Head Area
%KEYHEAD
KeyHead=%key1% %key2% %v1% %v2%
%KEYEND
%LINEDATA
%v1%-%v2%
%LINEEND
%KEYFOOT
KeyFoot=%key1% %key2% %v1% %v2%
%KEYEND
Foot Area
$ mcsvconv k=key1 m=form4.txt i=dat1.csv o=rsl4.tex
#END# kgcsvconv i=dat1.csv k=key1 m=form4.txt o=rsl4.tex
$ more rsl4.tex
\documentclass{article}
\begin{document}
\begin{table}
\begin{tabular}{l|l|r|r}
\hline
key1 & key2 & v1 & v2 \\
\hline
A & X & 1 & a \\
  & X & 1 & a \\
  & Y & 2 & b \\
  & Y & 3 & c \\
\hline
B & X & 4 & d \\
  & X & 4 & d \\
  & Y & 5 & e \\
\hline
\end{tabular}
\end{table}
\end{document}
```

関連コマンド

mcsv2arff：CSV 形式のデータから arff 形式 (WEKA 用のデータフォーマット) のデータへ変換する。
mcsv2json：CSV データを JavaScript のオブジェクト表記構文である JSON 形式に変換する。

4.18 mcut 項目の選択

指定した項目を選択する。-r オプションを付けると、指定した項目を削除する。

書式

mcut f= [-r] [-nfni] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=	抜き出す項目名 項目名をコロンで区切ることで、出力項目名を変更することができる。 ex. f=a:A,b:B
-r	項目削除スイッチ f=で指定した項目を削除し、それ以外の項目が抜き出される。
-nfni	入力データの１行目を項目名行とみなさない。よって項目番号で指定しなければならない。 以下のように、新項目名を組み合わせて指定することで項目名行を追加出力することが可能となる。 例）f=0:日付,2:店,3:数量

利用例

例 1: **基本例**

「顧客」と「金額」項目を選択する。ただし、「金額」項目は「売上」と名前を変更して出力している。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mcut f=顧客,金額:売上 i=dat1.csv o=rsl1.csv
#END# kgcut f=顧客,金額:売上 i=dat1.csv o=rsl1.csv
$ more rsl1.csv
顧客,売上
A,10
A,20
B,15
B,10
B,20
```

例 2: **項目削除**

-r を指定することで、項目を削除できる。

```
$ mcut f=顧客,金額 -r i=dat1.csv o=rsl2.csv
#END# kgcut -r f=顧客,金額 i=dat1.csv o=rsl2.csv
$ more rsl2.csv
数量
1
2
1
```

```
3
1
```

例 3: 項目名なしデータ

ヘッダなし入力ファイルから、0,2 番目の項目を選択し、「顧客」と「金額」という名前で出力する。

```
$ more dat2.csv
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mcut f=0:顧客,2:金額 -nfni i=dat2.csv o=rsl3.csv
#END# kgcut -nfni f=0:顧客,2:金額 i=dat2.csv o=rsl3.csv
$ more rsl3.csv
顧客,金額
A,10
A,20
B,15
B,10
B,20
```

関連コマンド

mfldname：項目名を変更したいだけの場合は `mfldname` を使う。

4.19 mdata データセットの出力

様々なデータセットを生成する。データセットの詳細は以下を参照のこと。

書式

```
mdata [O=] -iris|-man0|-man1|-tutorial_en|-tutorial_jp|-yakiniku_en|-yakiniku_jp [--help] [--helpl] [--version]
```

パラメータ

O=	出力ファイル名。省略時は標準出力に出力される。 -tutorial_jp もしくは-tutorial_en が指定された場合はディレクトリ名となり、 省略した場合は、O=tutorial_jp もしくは O=tutorial_en が指定されたことになる。
-iris	萼片と花びらの大きさによって、アヤメの種類の分類モデルの構築を目的に構成されるデータセット 項目名:SepalLength,SepalWidth,PetalLength,PetalWidth,Species(ガク長, ガク幅, 花びら長, 花びら幅, 種) http://archive.ics.uci.edu/ml/datasets/Iris?ref=datanews.io
man0	本マニュアルの図 2.4 で使われている 5 行データ 項目名: 顧客, 金額
man1	本マニュアルの図 2.6 で使われている 8 行データ 項目名: 顧客, 日付, 商品
yakiniku_jp	焼肉店の販売データ (岡山フードサービス株式会社より提供された焼肉店福牛の販売データ)。 項目名: 日付, 時間, レシート, 商品, 単価, 数量, 金額 注 1) 時間は注文時刻で、同じレシート内でも追加注文時はその時の時刻が記録されている。 注 2) 金額=単価×数量
yakiniku_en	焼肉店の注文データの英語版 項目名: date,time,receipt,item,price,quantity,totalAmount
tutorial_jp	チュートリアルで利用されるスーパーマーケットの擬似購買データ。 dat.csv: 購買履歴データ 項目名:店, 日付, 時間, レシート, 顧客, 商品, 大分類, 中分類, 小分類, 細分類, メーカー, ブランド, 仕入単価, 単価, 数量, 金額, 仕入金額, 粗利金額 syo.csv: 商品マスター 項目名: 商品, 商品名, 大分類, 中分類, 小分類, 細分類, メーカー, ブランド, 仕入単価 cust.csv: 顧客マスター 項目名: 顧客, 生年月日, 性別 jicfs1,2,4,6.csv: 商品分類マスター 項目名: 大分類, 大分類名 (中分類, 中分類名)(小分類, 小分類名)(細分類, 細分類名)
tutorial_en	tutorial_jp データセットの英語版 dat.csv: 購買履歴データ 項目名:shop,date,time,receipt,customer,product,CategoryCode1,CategoryCode2,CategoryCode4, CategoryCode6,manufacturer,brand,unitCost,unitPrice,quantity,amount,costAmount,profit syo.csv: 商品マスター 項目名: product,productName,CategoryCode1,CategoryCode2,CategoryCode4,CategoryCode6, manufacturer,brand,unitCost cust.csv: 顧客マスター 項目名: customer,dob,gender jicfs1,2,4,6.csv: 商品分類マスター 項目名: CategoryCode1,Category1(CategoryCode2,Category2)(CategoryCode4,Category4) (CategoryCode6,Category6)

利用例

例 1 iris データセットの出力

iris データセットを標準出力に出力する。

```
$ mdata -iris
SepalLength,SepalWidth,PetalLength,PetalWidth,Species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
         :
```

例 2 チュートリアルデータセットの出力

チュートリアルデータセットを全てファイル出力する。

```
$ mdata -tutorial_en
#END# mdata -tutorial_en

$ ls -l tutorial_en
total 4704
-rw-r--r--  1 nysol  staff    20673  8 22 08:14 cust.csv
-rw-r--r--  1 nysol  staff  2281312  8 22 08:14 dat.csv
-rw-r--r--  1 nysol  staff      128  8 22 08:14 jicfs1.csv
-rw-r--r--  1 nysol  staff      529  8 22 08:14 jicfs2.csv
-rw-r--r--  1 nysol  staff     6630  8 22 08:14 jicfs4.csv
-rw-r--r--  1 nysol  staff    36400  8 22 08:14 jicfs6.csv
-rw-r--r--  1 nysol  staff    46466  8 22 08:14 syo.csv

$ more tutorial_en/dat.csv
customer,dob,gender
00000A,19711107,female
00000B,19461025,female
00000C,19660307,female
         :
```

例 3 焼肉データを出力

```
$ mdata -yakiniku_jp
日付, 時間, レシート, 商品, 単価, 数量, 金額
20070701,1123,10000, 焼肉ヘルシーセット,1410,1,1410
20070701,1152,10001, 和牛焼肉弁当,1240,1,1240
20070701,1202,10002, ランチコーヒー,130,2,260
              :
$ mdata -yakiniku_en
date,time,receipt,item,price,quantity,totalAmount
20070701,1123,10000,Low-fat BBQ set,1410,1,1410
20070701,1152,10001,Japanese grilled beef lunch box,1240,1,1240
20070701,1202,10002,Lunchtime coffee,130,2,260
         :
```

4.20 mdelnull NULL 値行の削除

f=パラメータで指定した項目について、NULL 値が含まれる行を削除 (撰択) する。

書式

mdelnull f= [k=] [u=] [-F] [-r] [-R] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= NULL 値の検索対象となる項目名（複数項目指定可）を指定する。
k= 削除 (撰択) する単位となるキー項目名（複数項目指定可）を指定する。
u= 不一致データ出力ファイル名を指定する。
-F 項目間 AND 条件
f=パラメータで複数項目を指定した場合、その全ての値が NULL 値の行を削除 (撰択) する。
-r 条件反転
削除ではなく選択する。
-R レコード間 AND 条件
k=パラメータを指定した場合、その全ての行が NULL 値の行を削除 (撰択) する。

利用例

例 1: 基本例

「数量」と「金額」項目が NULL 値の行を削除する。NULL 値の行は oth.csv に出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,,20
B,1,15
B,3,
C,1,20
$ mdelnull f=数量,金額 u=oth.csv i=dat1.csv o=rsl1.csv
#END# kgdelnull f=数量,金額 i=dat1.csv o=rsl1.csv u=oth.csv
$ more rsl1.csv
顧客,数量,金額
A,1,10
B,1,15
C,1,20
$ more oth.csv
顧客,数量,金額
A,,20
B,3,
```

例 2: NULL 値の行を選択

-r を指定することで、削除ではなく選択することになる。

```
$ mdelnull f=数量,金額 -r i=dat1.csv o=rsl2.csv
#END# kgdelnull -r f=数量,金額 i=dat1.csv o=rsl2.csv
```

```
$ more rsl2.csv
顧客,数量,金額
A,,20
B,3,
```

例 3: キー項目での NULL 値の行の削除

k=を指定することで、集計キー単位で削除することになる。以下では「顧客」項目を単位にして、「数量」と「金額」項目に NULL 値が一つでも含まれていれば削除する。

```
$ mdelnull k=顧客 f=数量,金額 i=dat1.csv o=rsl3.csv
#END# kgdelnull f=数量,金額 i=dat1.csv k=顧客 o=rsl3.csv
$ more rsl3.csv
顧客 %0,数量,金額
C,1,20
```

例 4: 項目間 AND 条件の例

「数量」と「金額」項目の両方が NULL 値の行を削除する。

```
$ more dat2.csv
顧客,数量,金額
A,1,10
A,,
B,1,15
B,3,
C,1,20
$ mdelnull f=数量,金額 -F i=dat2.csv o=rsl4.csv
#END# kgdelnull -F f=数量,金額 i=dat2.csv o=rsl4.csv
$ more rsl4.csv
顧客,数量,金額
A,1,10
B,1,15
B,3,
C,1,20
```

例 5: レコード間 AND 条件の例

「顧客」項目を単位にして、「数量」項目が全て NULL 値の行を削除する。

```
$ mdelnull k=顧客 f=数量 -R i=dat1.csv o=rsl5.csv
#END# kgdelnull -R f=数量 i=dat1.csv k=顧客 o=rsl5.csv
$ more rsl5.csv
顧客 %0,数量,金額
A,1,10
A,,20
B,1,15
B,3,
C,1,20
```

関連コマンド

mnullto：NULL 値を含む行を削除するのではなく、NULL 値を指定の文字列に変換する。

4.21 mdformat 日付時刻抽出

他のシステムからエクスポートした CSV データでは、日付時刻項目にスラッシュ記号やコロン記号が入っていることが多く、また月日や時が 1 桁で格納されている場合もある (例:2014/7/18 1:57)。このような項目をそのまま MCMD で扱おうとすると、日付計算や並べ替え、範囲検索がうまくいかない。

mdformat コマンドを使うことで、f=パラメータで指定した項目から、c=パラメータで指定したフォーマットに従って年月日・時分秒を抽出し、MCMD で扱うことが可能な日付型や時刻型に変換することができる。

書式

mdformat c= f= [-A] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 抽出対象となる項目名リスト (複数項目指定可) を指定する。
c= 文字列のフォーマットを指定する。フォーマットの変換指定文字参照
-A このオプションにより、指定した項目を置き換えるのではなく、新たに項目が追加される。

フォーマットの変換指定文字

c=パラメータで利用可能な変換指定文字を表 4.1 に示す。

表 4.1 変換指定文字

変換指定文字	意味
%Y	西暦を表す数 (4 桁以内)
%y	西暦を表す数 (2 桁以内)
%m	月を表す数字 (2 桁以内)
%d	日を表す数字 (2 桁以内)
%H	時間 (2 桁以内)
%M	分 (2 桁以内)
%S	秒 (2 桁以内)
%s	秒. マイクロ秒

利用例

例 1: 基本例

fld 項目から日付・時刻を抽出し変換する。fld 項目には「date:年月日 time:時分秒. マイクロ秒」の形式で日付・時刻が格納されているので、c=パラメータには「date:%Y%m%d time:%H%M%s」と指定している。

```
$ more dat1.csv
fld
date:20120304 time:121212
date:20101204 time:011309.1234
$ mdformat f=fld c='date:%Y%m%d time:%H%M%s' i=dat1.csv o=rsl1.csv
#END# kgdformat c=date:%Y%m%d time:%H%M%s f=fld i=dat1.csv o=rsl1.csv
$ more rsl1.csv
fld
20120304121212
20101204011309.1234
```

例 2: 項目の追加

fld1 項目、fld2 項目には「年/月/日」形式で日付が格納されているので、c=パラメータには「%Y/%m/%d」と指定している。-A オプションを使用し、変換結果を新たな f1、f2 項目に抽出する。

```
$ more dat2.csv
fld,fld2
2010/11/20,2010/11/21
2010/1/1,2010/1/2
2011/01/01,2010/01/02
2010/1/01,2010/1/02
$ mdformat f=fld:f1,fld2:f2 c=%Y/%m/%d i=dat2.csv -A o=rsl2.csv
#END# kgdformat -A c=%Y/%m/%d f=fld:f1,fld2:f2 i=dat2.csv o=rsl2.csv
$ more rsl2.csv
fld,fld2,f1,f2
2010/11/20,2010/11/21,20101120,20101121
2010/1/1,2010/1/2,20100101,20100102
2011/01/01,2010/01/02,20110101,20100102
2010/1/01,2010/1/02,20100101,20100102
```

例 3: 抽出がうまくいかない例

fld 項目には「年 月 日 時:分:秒」形式で日付が格納されているので、c=パラメータには「%Y %m %d %H:%M:%S」と指定している。しかし形式が異なる行は抽出に失敗している。

```
$ more dat3.csv
fld
2010 11 20 12:34:56
2011 01 01 23:34:56
2010  1 01 123455
$ mdformat f=fld:f1 c='%Y %m %d %H:%M:%S' i=dat3.csv -A o=rsl3.csv
#END# kgdformat -A c=%Y %m %d %H:%M:%S f=fld:f1 i=dat3.csv o=rsl3.csv
$ more rsl3.csv
fld,f1
2010 11 20 12:34:56,20101120123456
2011 01 01 23:34:56,20110101233456
2010  1 01 123455,
```

関連コマンド

4.22 mdsp 画面表示

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

座標 x=,y=で指定したターミナル上の位置に str=で指定した文字列を表示する。i=でファイル名を指定すると、その内容を表示する。str=と i=の両方が指定されれば i=が優先される。i=が複数行ある場合、全ての行が x=で指定した座標位置から表示される。

fc=で文字列の色を指定でき、また bg=で文字列の背景の色を指定できる。指定できる色は、black, red, green, yellow, blue, magenda, cyan, white、の 8 色である。fc=のデフォルトは black で、bg=のデフォルトは white である。また-bold を指定することで強調表示が可能である。

書式

```
mdsp x= y= str=|i= [fc=] [bg=] [-bold]  [--help] [--helpl] [--version]
```

パラメータ

x=	x 軸 (左から右への横方向) 表示開始位置 (1 以上の値) を指定する。
y=	y 軸 (上から下への縦方向) 表示開始位置 (1 以上の値) を指定する。
str=	表示する文字列
i=	表示する内容のファイル名
fc=	文字色
bg=	背景色
-bold	強調表示

利用例

例 1: **基本例**

ターミナルの x=10,y=5 の位置に文字列 abcd を表示する。

```
$ mdsp x=10 y=5 str=abcd

以下、画面イメージ
+--------------------------------------
|
|
|
|
|           abcd
|
|
```

例 2: **ファイルで指定する場合**

ターミナルの x=10,y=5 の位置に dat.txt の内容を表示する。

```
$ more dat.txt
abcd
efg
$ mdsp x=10 y=5 i=dat.txt

以下、画面イメージ
```

```
+--------------------------------------
|
|
|
|
|           abcd
|           efg
|
```

例 3: 色をつける例

ターミナルの x=10,y=5 の位置に文字列 abcd を、文字の色を赤、背景を青にして表示する。

```
$ mdsp x=10 y=5 str=abcd fc=red bc=blue

以下、画面イメージ
+--------------------------------------
|
|
|
|
|           abcd
|
|
```

関連コマンド

minput :入力画面を表示する。
mminput：複数入力枠による入力画面を表示する。
mseldsp：画面に単一選択入力窓を表示する。
mmseldsp：画面に複数選択入力窓を表示する。

4.23 mduprec レコードの複写

各レコードを複写する。複写する行数は n=で固定値を与えるか、もしくは f=で指定した項目の値により与える。

書式

mduprec f=|n= [i=] [o=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 複写行数をもつ項目名
ここで指定した項目の値の数分、その行を複写する。
n= 複写行数の指定
ここで指定した値の数分、行を複写する。

利用例

例 1: 基本例

「数量」項目の値の数分、データを複写し複数行のデータを生成する。対象項目が NULL 値の場合は複写しない。

```
$ more dat1.csv
store,val
A,2
B,
C,5
$ mduprec f=val i=dat1.csv o=rsl1.csv
#END# kgduprec f=val i=dat1.csv o=rsl1.csv
$ more rsl1.csv
store,val
A,2
A,2
C,5
C,5
C,5
C,5
C,5
```

例 2: 複写行数の指定

データを 2 行づつ複写した (n=2) データを生成する。

```
$ mduprec n=2 i=dat1.csv o=rsl2.csv
#END# kgduprec i=dat1.csv n=2 o=rsl2.csv
$ more rsl2.csv
store,val
A,2
A,2
B,
B,
C,5
C,5
```

関連コマンド

mcount：`mduprec` と逆の動きをする。
mwindow：一定数のレコードをずらしながら複写する。

4.24 mfldname 項目名の変更

f=で指定した項目名を変更する。又、n=で項目名を新規設定する。

書式

mfldname f=|n= [-nfni] [i=] [o=] [-assert_diffSize] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=	ここで指定された項目名のみを変更する。(現項目名:新項目名) 指定していない項目名は変更せずに元の項目名が出力される。
n=	ここで指定された項目名が新たに採用される。 データ項目数と同じ数の項目名を指定する必要がある。
-nfni	入力データの 1 行目を項目名行とみなさない。このオプションが指定された場合は f=は利用できない。

利用例

例 1: **基本例**

項目名の「顧客」を「cust」に、「10 月」を「Oct.」に変更する。

```
$ more dat1.csv
顧客,itemID,10 月
a,xx,11
b,yy,122
c,zz,
$ mfldname f=顧客:cust,10 月:Oct. i=dat1.csv o=rsl1.csv
#END# kgfldname f=顧客:cust,10 月:Oct. i=dat1.csv o=rsl1.csv
$ more rsl1.csv
cust,itemID,Oct.
a,xx,11
b,yy,122
c,zz,
```

例 2: **項目名変更**

項目名を x,y,z に変更する。

```
$ mfldname n=x,y,z i=dat1.csv o=rsl2.csv
#END# kgfldname i=dat1.csv n=x,y,z o=rsl2.csv
$ more rsl2.csv
x,y,z
a,xx,11
b,yy,122
c,zz,
```

例 3: **項目名行がないデータ**

```
$ more dat2.csv
a,xx,11
b,yy,122
```

```
c,zz,
$ mfldname -nfni n=x,y,z i=dat2.csv o=rsl3.csv
#END# kgfldname -nfni i=dat2.csv n=x,y,z o=rsl3.csv
$ more rsl3.csv
x,y,z
a,xx,11
b,yy,122
c,zz,
```

関連コマンド

mcut：mfldname と同じことができるが、一部の項目名を変更するには少し面倒。また mfldname の方が少しだけ高速。

4.25 mfsort 項目ソート

各行で f=で指定した複数項目の値を並べ替え (デフォルトでは文字列昇順)、その順序で出力する。項目名の並びは変化しないことに注意する。

書式

mfsort f= [-r] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ソート対象となる項目を複数指定する。単一の項目を指定してもよいが、結果は変わらない。
-n 数値順に並べる。
-r 逆順に並べる。

利用例

例 1: 例 1: 基本例

各行において v1,v2,v3 の値を昇順にならべ、その順番で v1,v2,v3 項目として出力する。

```
$ more dat1.csv
id,v1,v2,v3
1,b,a,c
2,a,b,a
3,b,,e
$ mfsort f=v* i=dat1.csv o=rsl1.csv
#END# kgfsort f=v* i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3
1,a,b,c
2,a,a,b
3,,b,e
```

例 2: 例 2: 降順

降順にしたければ-r を付ける。

```
$ mfsort f=v* -r i=dat1.csv o=rsl2.csv
#END# kgfsort -r f=v* i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3
1,c,b,a
2,b,a,a
3,e,b,
```

関連コマンド

4.26　mhashavg ハッシュ法による項目値の平均

hash 法を使って k=パラメータで指定した項目を単位にして、f=パラメータで指定した項目値の平均を計算する。mavg との違いは、キー項目の並べ変えが必要ないため、その分処理速度が速い。ただし、キーのサイズ (キー項目のとる値の種類数) が多い場合は処理速度が遅くなる。

書式

mhashavg f= [hs=] [k=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

f=	ここで指定された項目 (複数項目指定可) の平均が計算される。 :(コロン）で新項目名を指定可能。例）f=数量:数量平均
k=	ここで指定された項目をキーとして集計する (複数項目指定可)。
hs=	ハッシュサイズ【デフォルト値:199999】 ハッシュサイズを指定する。【デフォルト値:199999】 詳細に関しては mhashsum 参照
-n	NULL 値が 1 つでも含まれていると結果も NULL 値とする。

利用例

例 1: **基本例**

「顧客」項目を単位にして、「数量」と「金額」項目の平均を計算する。

```
$ more dat1.csv
顧客,数量,金額
A,1,
B,,15
A,2,20
B,3,10
B,1,20
$ mhashavg k=顧客 f=数量,金額 i=dat1.csv o=rsl1.csv
#END# kghashavg f=数量,金額 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客,数量,金額
A,1.5,20
B,2,15
```

例 2: NULL **値の出力**

-n オプションを指定することで、NULL 値が含まれている場合は、結果も NULL 値として出力する。

```
$ mhashavg k=顧客 f=数量,金額 -n i=dat1.csv o=rsl2.csv
#END# kghashavg -n f=数量,金額 i=dat1.csv k=顧客 o=rsl2.csv
$ more rsl2.csv
顧客,数量,金額
A,1.5,
B,,15
```

備考

動作速度に関しては、mhashsum のページにあるベンチマークの項を参照のこと。

関連コマンド

mavg：同じ機能をもつコマンドだが、内部的にキー項目の並べ替えを行う。
mhashsum：同じくハッシュ法を用いた合計計算。

4.27　mhashsum ハッシュ法による項目値の合計

hash 法を使って k=パラメータで指定した項目を単位にして、f=パラメータで指定した項目値を合計する。msum との違いは、キー項目の並べ替えが必要ないため、その分処理速度が速い。ただし、キーのサイズ (キー項目のとる値の種類数) が多い場合は処理速度が遅くなる。msum と mhashsum のどちらを利用するかはデータの内容からユーザーが判断する (後半に示す「ベンチマーク」参照)。

書式

mhashsum f= [hs=] [k=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

f=　ここで指定された項目 (複数項目指定可) が合計される。
　　:(コロン）で新項目名を指定可能。例）f=数量:数量合計
k=　ここで指定された項目 (複数項目指定可) をキーとして集計する。
hs=　ハッシュサイズ【デフォルト:199999】
　　処理対象データのキーサイズから，ユーザが消費メモリ量と速度を判断して指定する。指定する値としては素数がよい。
　　キーサイズが大きいデータに対してハッシュサイズが十分な大きさでなければ処理速度が遅くなる。
　　ハッシュサイズが十分に大きいと処理速度は速いが、
　　その分多くのメモリが必要になる (後半に示す「ベンチマーク」参照)。
　　必要なメモリ量の目安: K*(24+F*16) byte, K:キーのサイズ, F:f=で指定した項目数
-n　NULL 値が 1 つでも含まれていると結果も NULL 値とする。

利用例

例 1: **基本例**

「顧客」項目を単位にして、「数量」と「金額」項目の合計を計算する。

```
$ more dat1.csv
顧客,数量,金額
A,1,
B,,15
A,2,20
B,3,10
B,1,20
$ mhashsum k=顧客 f=数量,金額 i=dat1.csv o=rsl1.csv
#END# kghashsum f=数量,金額 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客,数量,金額
A,3,20
B,4,45
```

例 2: NULL **値出力**

-n オプションを指定することで、NULL 値が含まれている場合は、結果も NULL 値として出力する。

```
$ mhashsum k=顧客 f=数量,金額 -n i=dat1.csv o=rsl2.csv
#END# kghashsum -n f=数量,金額 i=dat1.csv k=顧客 o=rsl2.csv
$ more rsl2.csv
```

```
顧客,数量,金額
A,3,
B,,45
```

アルゴリズムの概要

mhashsum コマンドは分離連鎖法 (separate chaining) によるハッシュ法を用いて実装されている。この方法では，キーを格納する M 個の配列が用意される。キーを構成する文字列をハッシュ関数により 0 から M までの整数 (ハッシュ値) に変換し、配列の番地として利用する。2 つ以上の異なるキーが同一のハッシュ値を持つ (キーが衝突する) 場合は、リンクリストにより格納される。キーを格納すべき番地の探索は定数オーダーであるが、リストの探索は線形オーダーである。そのため、キーの衝突が多発するほど速度は低下する。mhashsum のハッシュサイズはデフォルトで 199999 であり、これはキーのサイズが 20 万までであればリストの平均サイズは 1 以下であることを意味する。実際には，データによっては、キーサイズが小さい場合であってもキーの衝突が多発するケースもあり得る。キーのサイズは `hs=`パラメータで変更できる (上限:1999999)。

ベンチマークテスト

ベンチマークテストの方法

mhashsum コマンド（ハッシュサイズ：199,999）と msum コマンド（事前に msort コマンドで並べ替えを行う）の計算速度を異なるキーのサイズのデータにおいて計算速度を比較する。キーのサイズとして 100 から 100 万までの 13 種類のデータを対象とした。利用データは，表 4.2 に示されるような key と fld の 2 項目からなる 500 万行の乱数データである。key 項目の値は 6 桁の数値からなる固定長データで fld は 3 桁の数字である。

表 4.2　ベンチマーク用データ (抜粋)

key	value
100020	120
100007	107
100029	129
100065	165
100030	130
100088	188
100055	155
100093	193
100072	172

ベンチマークに利用したコマンド

mhashsum による方法

```
$ time mhashsum k=key f=fld i=dat.csv o=dev/null/
```

msort+msum による方法

```
$ time msort i=dat.csv | msum k=key f=fld o=dev/null/
```

実験結果

実験結果を表 4.3 に示す。キーのサイズが小さいとき (〜10,000) の速度は、ソーティングする方法に比べて 5 倍程度高速である。キーのサイズが大きくなるに従ってその差は小さくなり、キーのサイズが 80 万を超えるあたりで逆転している。ハッシュサイズが 20 万であることから、リストの平均サイズが 4 の時に逆転していることになる。キーの

値の分布によっても異なるが、この値を mhashsum と msum を使い分ける目安とすればよいであろう。

表 4.3　mhashsum と msum(msort+msum) の速度比較結果

キーサイズ	mhashsum(a)(秒)	msort+msum(b)(秒)	比 (a/b)
100	0.672	2.955	0.227
1,000	0.731	3.981	0.184
10,000	0.814	4.201	0.194
100,000	1.793	4.291	0.418
200,000	2.241	4.336	0.517
300,000	2.604	4.394	0.593
400,000	2.993	4.448	0.673
500,000	3.380	4.497	0.752
600,000	3.793	4.579	0.828
700,000	4.128	4.618	0.894
800,000	4.514	4.667	0.967
900,000	4.901	4.707	1.041
1,000,000	5.352	4.771	1.122

ベンチマーク環境

iMac, Mac OS X 10.5 Leopard, 2.8GHz Intel Core 2 Duo, 4GB メモリ

関連コマンド

msum：同じ機能をもつコマンドだが、内部的にキー項目の並べ替えを行う。

mhashavg：同じくハッシュ法を用いた平均計算。

4.28 minput 画面入力

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

座標 x=,y=で指定したターミナル上の位置に len=で指定した長さの入力枠を表示し、入力した内容を o=で指定したファイルに出力する。入力画面でエンターキーを押すと、終了ステータス 0 を返して終了し、エスケープキーを押すと、終了ステータス 1 を返して終了する。いずれのキーで終了しても、そこまでに入力した内容はファイルに出力される。

出力される CSV ファイルは、一項目一行のデータとなる。入力枠に何も入力せずに終了した場合は、null 値が出力される (すなわち改行だけが出力される)。f=を指定すれば、項目名を出力できる。f=を省略すれば、項目名ヘッダーは出力されない。

座標は左上が x=1,y=1 である (エスケープシーケンスの仕様)。x=もしくは y=で指定した値が 1 より小さい場合は、1 を指定したものとして動作する。またターミナルの範囲を超えた座標が指定された場合の動作は不定である。

書式

```
minput x= y= len= [f=]  o= [-nfn] [-nfno] [-x] [--help] [--helpl] [--version]
```

パラメータ

x=	x 軸 (左から右への横方向) 表示開始位置 (1 以上の値) を指定する。
y=	y 軸 (上から下への縦方向) 表示開始位置 (1 以上の値) を指定する。
len=	入力欄の半角文字数
f=	出力項目名

利用例

例 1: 基本例

ターミナルの x=10,y=5 の位置に幅 12 文字の入力領域を示し、入力枠に”abcd”と入力して enter キーを押し、入力結果を rsl1.csv に保存する。

```
$ minput x=10 y=5 len=12 o=rsl1.csv
$ more rsl1.csv
abcd


以下、画面イメージ
+--------------------------------------
|
|
|
|
|           [abcd        ]
|
|
```

例 2: 項目名を指定する基本例

例 1 と同様で f=で項目名を指定する。

```
$ minput x=10 y=5 len=12 f=name o=rsl1.csv
$ more rsl1.csv
```

```
name
abcd
```

例 3: 終了ステータスを判定する例

例 1 と同じパラメータで実行し、終了ステータスを判定して異なる動作をするスクリプトの例。

```
$ more scp.sh
rm -f rsl2.csv
clear
minput x=10 y=5 len=12 o=rsl2.csv
if [ $? = 0 ] ; then
  clear ; echo "end by enter key"
else
  clear ; echo "end by escape key"
fi

# abcd と入力後 enter キーを入力した場合の結果
$ bash scp.sh
end by enter key
$ more rsl2.csv
abcd

# abcd と入力後 escape キーを入力した場合の結果
$ bash scp.sh
end by escape key
$ more rsl2.csv
abcd
```

関連コマンド

mminput：複数入力枠による入力画面を表示する。

mdsp：画面の指定位置に文字列を表示する。

mseldsp：画面に単一選択入力窓を表示する。

mmseldsp：画面に複数選択入力窓を表示する。

4.29 mjoin 参照ファイルの項目結合

k=パラメータで指定した入力ファイルの項目値と参照ファイルの項目値を比較し、同じ値を持つ f=パラメータで指定した参照ファイルの項目値を結合する。参照ファイルのキー項目は単一化されている必要がある。参照ファイルに同じキー項目の値が複数ある場合は、mnjoin コマンドを利用すればよい。また、f=を省略すると、参照ファイルのキー項目以外全ての項目を結合する。

書式

mjoin k= [f=] [K=] [-n] [-N] m=| i= [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= ここで指定した入力データの項目と K=パラメータで指定された
参照データの項目が同じ行の項目結合が行われる。
NULL 値は，参照ファイルの K=で指定した項目のどの値にもマッチしない値として扱われる。

f= 結合する参照ファイル上の項目名リストを指定する。
省略するとキー項目を除いた全ての項目が結合される。

m= 参照ファイル名を指定する。
このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

K= 参照データ上の突き合わせる項目名リスト
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行の項目結合が行われる。
NULL 値は，入力ファイルの k=で指定した項目のどの値にもマッチしない値として扱われる。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。

-n 参照データにない入力データを NULL 値として出力するフラグ。

-N 入力データにない参照データを NULL 値として出力するフラグ。

利用例

例 1: 基本例

入力ファイルにある item 項目と、参照ファイルにある item 項目を比較し同じ値の場合、cost 項目を結合する。

```
$ more dat1.csv
item,date,price
A,20081201,100
A,20081213,98
B,20081002,400
B,20081209,450
C,20081201,100
$ more ref1.csv
item,cost
A,50
B,300
E,200
$ mjoin k=item f=cost m=ref1.csv i=dat1.csv o=rsl1.csv
#END# kgjoin f=cost i=dat1.csv k=item m=ref1.csv o=rsl1.csv
$ more rsl1.csv
item%0,date,price,cost
A,20081201,100,50
A,20081213,98,50
B,20081002,400,300
```

```
B,20081209,450,300
```

例 2: 未結合データ出力

入力ファイルにある item 項目と、参照ファイルにある item 項目を比較し同じ値の場合、cost 項目を結合する。その際、参照データにない入力データと参照データにない範囲データを NULL 値として出力する。

```
$ mjoin k=item f=cost m=ref1.csv -n -N i=dat1.csv o=rsl2.csv
#END# kgjoin -N -n f=cost i=dat1.csv k=item m=ref1.csv o=rsl2.csv
$ more rsl2.csv
item%0,date,price,cost
A,20081201,100,50
A,20081213,98,50
B,20081002,400,300
B,20081209,450,300
C,20081201,100,
E,,,200
```

関連コマンド

mnjoin：参照ファイルのキーに重複がある場合は mnjoin を使う。

mpaste：行番号による結合を行う。

mcommon：結合でなく単に選択するだけなら mcommon を使えばよい。

4.30 mkeybreak キーブレイク箇所

k=パラメータで指定した項目をキー項目について、先頭と終端に印を付ける。先頭は top 項目に、終端は bot 項目に 1 を出力する。先頭/終端でない行は NULL 値を出力する。

書式

mkeybreak k= [s=] [a=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= 集計キーとなる項目名リスト（複数項目指定可）を指定する。
s= ここで指定した項目 (複数項目指定可) で並べ替えた後、先頭・終端に印を付ける。
a= 先頭と終端の印を出力する項目名を指定する。【デフォルト値:top,bot】

利用例

例 1: **基本例**

k1 項目で並べ替えた後、k1 キー項目の先頭 (top 項目) と終端 (bottom 項目) に印 (1) をつける。

```
$ more dat1.csv
id,k1,k2,val
1,A,a,1
2,A,b,2
3,A,b,3
4,B,a,4
5,B,a,5
$ mkeybreak k=k1 i=dat1.csv o=rsl1.csv
#END# kgkeybreak i=dat1.csv k=k1 o=rsl1.csv
$ more rsl1.csv
id,k1%0,k2,val,top,bot
1,A,a,1,1,
2,A,b,2,,
3,A,b,3,,1
4,B,a,4,1,
5,B,a,5,,1
```

例 2: 2 **項目キー**

k1・k2 項目で並べ替えた後、k1 キー項目の先頭 (top 項目) と終端 (bottom 項目) に印 (1) をつける。

```
$ mkeybreak s=k1,k2 k=k1 i=dat1.csv o=rsl2.csv
#END# kgkeybreak i=dat1.csv k=k1 o=rsl2.csv s=k1,k2
$ more rsl2.csv
id,k1,k2,val,top,bot
1,A,a,1,1,
2,A,b,2,,
3,A,b,3,,1
4,B,a,4,1,
5,B,a,5,,1
```

関連コマンド

4.31 mmbucket 多次元均等化バケット分割

f=で指定した複数の数値項目を次元とした件数均等化バケット分割を行う。例えば、f=a,b,c そして n=5 と指定すると、mbucket コマンドと同様に、項目 a,b,c それぞれを 5 つの区間に分割するが、mmbucket では、項目 a,b,c の 3 次元空間における各バケット (バケット数は $5^3 = 125$ 個になる) に属する件数ができるだけ均等になるような区間を計算する

書式

mmbucket f= n= [F=] [k=] [O=] [-ms] [-r] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の値を分割する。
複数指定すれば、その数の次元に基づく均等化バケット分割を行う。
1 項目のみ指定すれば mbucket と同じ結果になる。
-x,-nfn オプション使用時は、項目番号 (0〜) で指定可能。

n= f=で指定した項目数と同じ個数分指定する。
ただし 1 つの数字を指定した場合、f=で指定した全ての項目に、同じ分割数が適用される。

F= 出力形式を指定する。【デフォルト値:1】
バケットの名前を出力形式。
0:バケット番号のみを表示する。
1:バケットの範囲のみを表示する。
2:バケット番号とバケット範囲の両方を表示する。

k= バケット分割を行う単位となる項目名リスト (複数項目指定可) を指定する。

O= f=パラメータで指定した各項目の各バケットの数値範囲を出力するファイル名を指定する。

-ms 各項目を順次バケット分割していく時の開始項目を変えることで複数回のバケット分割をトライし、
よりよい解を求める。詳細は、以下の「アルゴリズムの概要」を参照のこと。

-r バケット番号を逆順に出力する。

利用例

例 1: 基本例

x、y 項目の件数ができるだけ多次元均等になるように 2 分割する。その際、各バケットの数値範囲を rng.csv という名前のファイルに出力する。

```
$ more dat1.csv
id,x,y
A,2,7
B,6,7
C,5,6
D,7,5
E,6,4
F,1,3
G,3,3
H,4,2
I,7,2
J,2,1
$ mmbucket f=x:xb,y:yb n=2,2 O=rng.csv i=dat1.csv o=rsl1.csv
calculating on dimension ... #0 #1 done. VAR=30 updated!
```

```
calculating on dimension ... #0 #1 done. VAR=28 updated!
calculating on dimension ... #0 #1 done. VAR=28
#END# kgmbucket O=rng.csv f=x:xb,y:yb i=dat1.csv n=2,2 o=rsl1.csv
$ more rsl1.csv
id,x,y,xb,yb
A,2,7,1,2
B,6,7,2,2
C,5,6,2,2
D,7,5,2,2
E,6,4,2,1
F,1,3,1,1
G,3,3,1,1
H,4,2,2,1
I,7,2,2,1
J,2,1,1,1
$ more rng.csv
fieldName,bucketNo,rangeFrom,rangeTo
x,1,,3.5
x,2,3.5,
y,1,,4.5
y,2,4.5,
```

例 2: **出力形式**

id 項目を単位に件数ができるだけ多次元均等になるように x,y 項目を 2 分割する。出力形式はバケット番号とバケット範囲の両方を表示する。

```
$ more dat2.csv
id,x,y
A,2,7
A,6,7
A,5,6
B,7,5
B,6,4
B,1,3
C,3,3
C,4,2
C,7,2
C,2,1
$ mmbucket k=id f=x:xb,y:yb n=2,2 F=2 i=dat2.csv o=rsl2.csv
calculating on dimension ... #0 #1 done. VAR=3 updated!
calculating on dimension ... #0 #1 done. VAR=3
calculating on dimension ... #0 #1 done. VAR=3 updated!
calculating on dimension ... #0 #1 done. VAR=3
calculating on dimension ... #0 #1 done. VAR=6 updated!
calculating on dimension ... #0 #1 done. VAR=6
#END# kgmbucket F=2 f=x:xb,y:yb i=dat2.csv k=id n=2,2 o=rsl2.csv
$ more rsl2.csv
id%0,x,y,xb,yb
A,2,7,1:_3.5,2:6.5_
A,6,7,2:3.5_,2:6.5_
A,5,6,2:3.5_,1:_6.5
B,7,5,2:3.5_,2:4.5_
B,6,4,2:3.5_,1:_4.5
B,1,3,1:_3.5,1:_4.5
C,3,3,1:_3.5,2:1.5_
C,4,2,2:3.5_,2:1.5_
C,7,2,2:3.5_,2:1.5_
C,2,1,1:_3.5,1:_1.5
```

アルゴリズムの概要[*2]

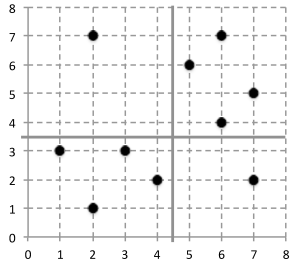
2 つの数値項目からなるデータセット D があるものとし、2 つの項目名を A, B とする。A および B の数値範囲をそれぞれ K 個、L 個に分割するものとする。この分割により、データセットは $K \cdot L$ 個に分割される。mmbucket は分割された各領域 (バケット) に入るデータの個数が一様になるような分割方法を求める。一様性の評価基準としては分散を用いるものとする。分散値が最小のバケット分割を求めるものであるが、ここでは、以下に述べるようなヒューリスティックスに基づいており、最適性は保証していない。アルゴリズムは以下のように記述される。

1. 項目 A について件数均等化 1 次元バケット分割をおこなう。
2. 1. で得られた A の分割を固定して、B の数値範囲のバケット分割をおこなう。このときの評価基準は 2 次元バケット分割の分散値である。方法は、件数均等化 1 次元バケット分割と同様に動的計画法を用いている。
3. 次にステップ 2 で得られた B の分割を固定して、A の数値範囲のバケット分割をおこなう。このときの評価基準はステップ 2 と同様に、2 次元バケット分割の分散値である。
4. 以上のステップ 2,3 を分散値の改良が生じなくなるまで繰り返す。
5. 最終的に得られた分割を出力する。

実装においては、ステップ 1 において、A と B の役割を入れ替えた場合も走らせて、解を求め、両者で得られた 2 つの解のうち優れた方を出力するようにしている。また、3 次元以上の場合の均等化バケット分割も同様のアルゴリズムに基づいている。

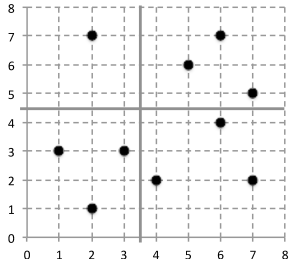
mbucket と mmbucket の比較

多次元バケット分割がどのように分割を行うかを示すために、1 次元バケット分割との比較を通して以下で説明する。表 4.4 には、二つの数値属性 x,y を持つ、id が A から J までの 10 件データが示されている。このデータについて、属性 x,y をそれぞれ 2 分割し計 4 つのバケットに分割することを考えよう。属性 x と y のそれぞれを 1 次元バケット分割することで得られた結果が x1,y1 項目として示されており、その値を用いて 4 つのバケットに分割した様子が図に示されている。そして多次元バケット分割のアルゴリズムを用いて得られた結果が x2,y2 項目として示されており、その分割の様子が図に示されている。
分散値 (各バケットに属するデータ件数の二乗和：詳しくは mbucket の定式化を参照) は、1 次元バケット分割においては、$Var'_a = 1^2 + 4^2 + 4^2 + 1^2 = 34$ となり、2 次元バケット分割では $Var'_b = 1^2 + 3^2 + 3^2 + 3^2 = 28$ となる。1 次元バケット分割においては、x,y それぞれ独立で考えた場合には、分散を最小化する最適解となっている (すなわち 5 件ずつに分割している) が、2 次元の分割として評価すると、2 次元バケット分割に比べて劣った分割となる。このように多次元バケット分割を使うことで 1 次元バケット分割の組み合わせよりも優れた分割が可能となる。しかしながら、次項で示すように、多次元分割はデータの内容によっては多大な計算時間が必要となるケースがあることに注意する。



		x	
		~4.5	4.5~
y	~3.5	1	4
	3.5~	4	1

図 4.1 1 次元分割 (mbucket) × 2 による分割



		x	
		~3.5	3.5~
y	~4.5	1	3
	4.5~	3	3

図 4.2 2 次元分割 (mmbuket) による分割

[*2] このアルゴリズムは加藤直樹教授 (京都大学工学研究科) による。

表 4.4　サンプルデータと mbucket,mmbucket による出力結果

データ			mbucket		mmbucket	
id	x	y	x1	y1	x2	y2
A	2	7	1	2	1	2
B	6	7	2	2	2	2
C	5	6	2	2	2	2
D	7	5	2	2	2	2
E	6	4	2	2	2	1
F	1	3	1	1	1	1
G	3	3	1	1	1	1
H	4	2	1	1	2	1
I	7	2	2	1	2	1
J	2	1	1	1	1	1

多様な項目での精度実験

mbucket と mmbucket の精度を実データを用いて比較してみる。利用するデータは、エルニーニョ現象の解明を目的とした TAO プロジェクトで収集されているデータで、赤道付近に設置されたセンサーが収集した海洋データおよび気象データである。表 4.7 にデータの一部を示す。

表 4.5　精度実験用データセット

id	year	month	day	date	latitude	longitude	zonwinds	merwinds	humidity	air_temp.	sstemp
4060	93	5	9	930509	-0.02	-109.96	-2.1	2.1	81.2	26.8	27.02
4061	93	5	10	930510	-0.02	-109.96	-3.4	1.4	84.2	26.95	26.91
4062	93	5	11	930511	-0.02	-109.96	-3.8	2.2	84.9	26.98	26.78
4063	93	5	12	930512	-0.02	-109.96	-3	1.5	86.9	26.93	26.74
4064	93	5	13	930513	-0.02	-109.96	-4.5	1.9	87.6	27.01	26.82
4065	93	5	14	930514	-0.02	-109.96	-5	1.3	85.6	26.96	26.68

latitude:緯度, longitude:経度, zonwinds:東西風速, merwinds:南北風速, humidity:湿度, air_temp.:気温, sstemp:海面温度

このデータの数値属性項目 (緯度から海面温度までの 7 項目) を使いバケット分割の精度実験を行った。レコード行数は 93,935 行である。各項目の統計は表 3 に示す通りである。バケット分割においては「値の種類数」が計算時間に大きく影響を及ぼす。

表 4.6　数値データの各種統計

計量	緯度	経度	東西風速	南北風速	湿度	気温	海面温度
値の種類数	482	924	228	206	385	1104	1201
算術平均	0.305	-70.8	-3.35	-0.046	81.3	27.1	27.9
標準偏差	4.77	128.7	3.42	3.021	5.28	1.674	1.87
最小値	-8.33	-180	-10.7	-10.6	52.1	17.5	18.2
中央値	0.01	-125	-4.1	-0.1	81.3	27.5	28.4
最大値	9.05	170.0	14.3	13	99.9	31.5	31.0

7 つの属性から 2 項目の組み合わせ全 21 通りについて 1 次元バケット分割 (mbucket) に基づく方法と 2 次元バケット分割 (mmbucket) に基づく方法の分散値 (バケットに属する件数の二乗和) の比較が表 4.7 に示されている。例えば一行目に示された結果は、気温×湿度のバケット分割の比較で、mbucket での分散値が 408274409、mmbucket での分散値が 406438211 であり、その比 (mmbucket/mbucket) が 0.996 であることが示されている。続いて、分割数を 10 × 10、15 × 15、20 × 20 としたときの比が示されている。結果は属性の組み合わせによって異なり、「湿度×東西風速」のように全く改善の見られない属性の組み合わせもあれば、「緯度×経度 (15 × 15)」のように 19%(1-0.81) の精度向上を示す属性もみられる。

表 4.7　mbucket と mmbucket による 2 次元分割精度の比較

項目 1	項目 2	分散値 (5 × 5 分割)		分散値の比 (mmbucket/mbucket)			
		mbucket	mmbucket	5 × 5	10 × 10	15 × 15	20 × 20
気温	湿度	408274409	406438211	0.996	0.987	0.988	0.981
	緯度	374125463	371258775	0.992	0.988	0.982	0.964
	経度	490727955	454663065	0.927	0.949	0.946	0.929
	南北風速	396436813	394724219	0.996	0.991	0.993	0.989
	海面温度	816199131	747215787	0.915	0.897	0.883	0.862
	東西風速	382069455	381225143	0.998	0.998	0.997	0.996
湿度	緯度	368492959	367941709	0.999	0.994	0.994	0.990
	経度	372116309	370591351	0.996	0.991	0.993	0.986
	南北風速	355658757	355658757	1.000	1.000	1.000	1.000
	海面温度	380382203	379546293	0.998	0.991	0.991	0.983
	東西風速	357729819	357697283	1.000	1.000	1.000	1.000
緯度	経度	365459223	361754113	0.990	0.923	0.812	0.816
	南北風速	371478669	370970251	0.999	0.988	0.985	0.982
	海面温度	392810425	389589403	0.992	0.987	0.979	0.952
	東西風速	364521077	364406663	1.000	0.999	0.991	0.992
経度	南北風速	414154185	408431667	0.986	0.976	0.976	0.962
	海面温度	510979465	463576537	0.907	0.945	0.939	0.934
	東西風速	400021527	392710641	0.982	0.983	0.982	0.973
南北風速	海面温度	393233841	392432943	0.998	0.995	0.994	0.993
	東西風速	359290669	359074015	0.999	1.000	1.000	0.997
海面温度	東西風速	442877539	438326669	0.990	0.988	0.984	0.986

速度比較

次に分割数による速度の違いみるための実験を行った。前節と同様のデータを用い、気温×海面温度および緯度×経度について、分割数を 5 から 40 まで 5 刻みで設定し、その実行時間を計測した。その結果が表 4.8 に示されている。mbucket の実行時間は分割数に関係なくほぼ一定である。一方で 2 次元分割については分割数が多くなるに従って実行時間も長くなっている。これはアルゴリズム内部で必要となる多次元直交数値範囲の選択において効率的なアルゴリズムを実装していないためである。これは改善可能であるが、次期バージョンでの課題としたい。

表 4.8　サンプルデータと mbucket,mmbucket による出力結果

	気温×海面温度		緯度×経度	
分割数	mbucket	mmbucket	mbucket	mmbucket
5	0.221	2.21	0.216	0.67
10	0.227	3.90	0.216	1.67
15	0.233	10.4	0.230	3.28
20	0.231	26.1	0.228	7.13
25	0.232	32.9	0.237	13.3
30	0.236	46.7	0.236	11.4
35	0.237	62.3	0.240	15.2
40	0.237	80.1	0.237	25.8

関連コマンド

mbucket：複数項目指定しても、それぞれの項目で 1 次元バケット分割を実行する。

4.32　mminput 画面フォーム入力

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

i=で与えられたテキストファイルを画面フォームとして読み込み、データ入力画面を表示する。画面フォーム上の文字列は、そのままのイメージで画面出力され、角括弧 ([]) で囲われた領域は自由入力欄として表示される。入力欄は複数あってもよい。利用者が入力したデータは CSV として o=で指定したファイルに出力する。出力されるデータは一行データで、入力欄が複数ある場合は、複数項目の CSV として出力される。

入力枠に何も入力せずに終了した場合は、null 値が出力される。f=を指定すれば、項目名を出力できる。f=を省略すれば、項目名ヘッダーは出力されない。

またターミナルの範囲を超えた文字列や入力枠が指定された場合の動作は不定である。

書式

```
mminput i= [f=]  o= [-nfn] [-nfno] [-x] [--help] [--helpl] [--version]
```

パラメータ

i=	画面フォームのテキストファイル名
f=	出力項目名

利用例

例 1: 基本例

名前と住所を入力する画面を表示し、項目名 name,address として rsl1.csv に出力する。

```
$ more screen.txt

     name    :[               ]
     address:[               ]

$ mminput i=screen.txt f=name,address o=rsl1.csv
$ more rsl1.csv
name,address
Taro,Japan


以下、画面イメージ
+---------------------------------------
|
|      name    :[Taro            ]
|      address:[Japan            ]
|
```

例 2: 終了ステータスを判定する例

例 1 と同じパラメータで実行し、終了ステータスを判定して異なる動作をするスクリプトの例。

```
$ more scp.sh
rm -f rsl3.csv
clear
mminput i=screen.txt f=name,address o=rsl3.csv
```

```
if [ $? = 0 ] ; then
  clear ; echo "end by enter key"
else
  clear ; echo "end by escape key"
fi

# Taro と Japan を入力後 enter キーを入力した場合の結果
$ bash scp.sh
end by enter key
$ more rsl3.csv
name,address
Taro,Japan

# Taro と Japan を入力後 escape キーを入力した場合の結果
$ bash scp.sh
end by escape key
$ more rsl3.csv
name,address
Taro,Japan
```

関連コマンド

minput :入力画面を表示する。

mdsp：画面の指定位置に文字列を表示する。

mseldsp：画面に単一選択入力窓を表示する。

mmseldsp：画面に複数選択入力窓を表示する。

4.33　mmseldsp 複数選択画面入力

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

座標 x=,y=で指定したターミナル上の位置に i=、もしくは seldata=で指定した文字列リストの選択画面を表示し、ユーザが選んだ文字列を o=で指定したファイルに出力する。mseldsp コマンドでは、利用者は一つの選択肢しか選択できないが、mmseldsp は複数の選択肢を選択できる。選ばれた複数の文字列は、複数行の CSV 項目として出力される。入力枠に何も入力せずに終了した場合は、null 値が出力される (すなわち改行だけが出力される)。f=を指定すれば、項目名を出力できる。f=を省略すれば、項目名ヘッダーは出力されない。選択肢の数が多くて画面をはみ出る場合は、height=でスクロール窓の行数を指定すればよい。

選択画面でエンターキーを押すと、終了ステータス 0 を返して終了し、エスケープキーを押すと、終了ステータス 1 を返して終了する。いずれのキーで終了しても、選択画面で選ばれた内容はファイルに出力される。

座標は左上が x=1,y=1 である (エスケープシーケンスの仕様)。x=もしくは y=で指定した値が 1 より小さい場合は、1 を指定したものとして動作する。またターミナルの範囲を超えた座標が指定された場合の動作は不定である。

書式

```
mmseldsp x= y= [height=] i=|seldata= o= [-nfn] [-nfno] [-x] [--help] [--helpl] [--version]
```

パラメータ

x=	x 軸 (左から右への横方向) 表示開始位置 (1 以上の値) を指定する。
y=	y 軸 (上から下への縦方向) 表示開始位置 (1 以上の値) を指定する。
height=	選択肢を表示する行数。
i=	選択肢の文字列を項目として持つ CSV ファイル名
f=	選択肢の文字列を項目として持つ CSV ファイル名
seldata=	カンマで区切られた選択肢の文字列リスト

利用例

例 1: 基本例

ターミナルの x=10,y=2 の位置に sel.txt の内容を表示し、利用者が選んだ文字列を rsl1.txt に出力する。

```
$ more sel.txt
apple
pineapple
grape
orange
$ mmseldsp x=10 y=2 i=sel.txt o=rsl1.txt
# 利用者が一行目を選んだとする。
$ mose rsl1.txt
apple
orange


以下、画面イメージ
+--------------------------------------
|
|          apple
|          pineapple
|          grape
```

```
|          orange
|
```

例 2: **引数で与える例**

例 1 と同様で、選択肢の文字列を seldata=で与える。

```
$ mmseldsp x=10 y=2 seldata=apple,pineapple,grape,orange o=rsl2.txt
# 利用者が二行目を選んだとする。
$ mose rsl2.txt
apple
grape


以下、画面イメージ
+--------------------------------------
|
|
|          apple
|          pineapple
|          grape
|          orange
|
```

例 3: **終了ステータスを判定する例**

例 2 と同じパラメータで実行し、終了ステータスを判定して異なる動作をするスクリプトの例。

```
$ more scp.sh
rm -f rsl3.csv
clear
mmseldsp x=10 y=2 seldata=apple,pineapple,grape,orange o=rsl3.csv
if [ $? = 0 ] ; then
  clear ; echo "end by enter key"
else
  clear ; echo "end by escape key"
fi

# apple を選択後 enter キーを入力した場合の結果
$ bash scp.sh
end by enter key
$ more rsl3.csv
apple

# apple を選択後 escape キーを入力した場合の結果
$ bash scp.sh
end by escape key
$ more rsl3.csv
apple
```

関連コマンド

minput :入力画面を表示する。

mminput：複数入力枠による入力画面を表示する。

mdsp：画面の指定位置に文字列を表示する。

mseldsp：画面に単一選択入力窓を表示する。

4.34　mmvavg 移動平均の算出

移動平均 (moving average) を算出する。移動平均としては、単純移動平均 (SMA)、線形荷重移動平均 (WMA)、指数平滑移動平均 (EMA) の 3 種類の移動平均が計算可能である。

ある時 t における値を x_t で表したとき、m 期の各種移動平均は式 (4.1,4.2,4.3) で定義される。

$$SMA_t = \frac{1}{m}\sum_{i=0}^{m-1} x_{t-i} \tag{4.1}$$

$$WMA_t = \sum_{i=0}^{m-1} \frac{m-i}{S} x_{t-i}, \quad S = \sum_{i=1}^{m} i \tag{4.2}$$

$$EMA_t = \alpha x_t + (1-\alpha) EMA_{t-1} \tag{4.3}$$

書式

`mmvavg [s=] [k=] [n=] f= [t=] [-exp|-w] [alpha=] [skip=]` [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] `[--help] [--helpl] [--version]`

パラメータ

`s=`	ここで指定した項目 (複数項目指定可) で並べ替えられた後、移動平均が計算される。 `-q` オプションを指定しないとき、`s=`パラメータは必須。
`k=`	ここで指定された項目 (複数項目指定可) を単位として集計する。
`f=`	移動平均を求める項目名リスト (複数項目指定可) を指定する。
`t=`	期間数を 1 以上の整数で指定する。 `-exp` 指定時に `alpha=`を指定すれば `t=`は指定できない。
`-w`	線形加重移動平均を求める。
`-exp`	指数平滑移動平均を求める。
`alpha=`	`-exp` が指定された時の平滑化係数を実数値で与える。 省略時は `alpha=2/(t=`**の値** `+1)`。
`skip=`	出力を抑制する最初の行数。 デフォルト値: `skip=(t=`**の値**`-1)`, `-exp` が指定された場合は `skip=0`

利用例

例 1: **基本例**

最初の行は期数に満たないため出力されない。

```
$ more dat1.csv
id,value
1,5
2,1
3,3
4,4
5,4
6,6
```

```
7,1
8,4
9,7
$ mmvavg s=id f=value t=2 i=dat1.csv o=rsl1.csv
#END# kgmvavg f=value i=dat1.csv o=rsl1.csv s=id t=2
$ more rsl1.csv
id%0,value
2,3
3,2
4,3.5
5,4
6,5
7,3.5
8,2.5
9,5.5
```

例 2: **基本例** 2

最初の行は期数に満たないため出力されない。

```
$ mmvavg s=id f=value t=2 -w i=dat1.csv o=rsl2.csv
#END# kgmvavg -w f=value i=dat1.csv o=rsl2.csv s=id t=2
$ more rsl2.csv
id%0,value
2,2.333333333
3,2.333333333
4,3.666666667
5,4
6,5.333333333
7,2.666666667
8,3
9,6
```

例 3: **基本例** 3

指数平滑移動平均 (`-exp`) の場合は最初の行から出力される。

```
$ mmvavg s=id f=value t=2 -exp i=dat1.csv o=rsl3.csv
#END# kgmvavg -exp f=value i=dat1.csv o=rsl3.csv s=id t=2
$ more rsl3.csv
id%0,value
1,5
2,2.333333333
3,2.777777778
4,3.592592593
5,3.864197531
6,5.288065844
7,2.429355281
8,3.47645176
9,5.82548392
```

例 4: **キーを指定する例**

```
$ more dat2.csv
id,key,value
1,a,5
2,a,1
3,a,3
4,a,4
5,a,4
```

```
6,b,6
7,b,1
8,b,4
9,b,7
$ mmvavg s=key,id k=key f=value t=2 i=dat2.csv o=rsl4.csv
#END# kgmvavg f=value i=dat2.csv k=key o=rsl4.csv s=key,id t=2
$ more rsl4.csv
id,key,value
2,a,3
3,a,2
4,a,3.5
5,a,4
7,b,3.5
8,b,2.5
9,b,5.5
```

例 5: 指定した期に満たなくても出力する例

```
$ more dat3.csv
key,value
a,1
a,2
a,3
a,4
a,5
b,6
b,1
b,4
b,7
$ mmvavg -q k=key f=value t=2 skip=0 i=dat3.csv o=rsl5.csv
#END# kgmvavg -q f=value i=dat3.csv k=key o=rsl5.csv skip=0 t=2
$ more rsl5.csv
key,value
a,1
a,1.5
a,2.5
a,3.5
a,4.5
b,6
b,3.5
b,2.5
b,5.5
```

関連コマンド

mmvstats：平均だけでなく、各種統計量を指定可能。

mmvsim：2 変量の統計量を計算する。

mwindow：動窓のデータを作成するので、そのデータを使えば mmvstats で計算できない統計量も計算可能。

4.35 mmvsim 移動窓の類似度計算

移動窓を設定し、各種類似度 (2 変量の統計量) を計算する。msim コマンドの移動窓バージョンとして考えればよい。msim との違いは、指定できる類似度は一つだけで、また類似度計算の対象項目は 2 つのみである。

書式

mmvsim [s=] [k=] f= c= a= [t=] [skip=] -n [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

s=	ここで指定した項目 (複数項目指定可) で並べ替えられた後、各種類似度が計算される。 -q オプションを指定しないとき、s=パラメータは必須。
k=	ここで指定された項目 (複数項目指定可) を単位として集計する。
f=	集計項目名リスト (複数項目指定可) を指定する。
t=	期間数を 1 以上の整数で指定する。
c=	類似度 (以下のリストから一つだけ) 指定する。 covar|ucovar|pearson|spearman|kendall|euclid| cosine|cityblock|hamming|chi|phi|jaccard|support|lift 詳細な定義は msim コマンドを参照のこと。
skip=	出力を抑制する最初の行数を指定する。【デフォルト値:skip=(t=の値-1)】

利用例

例 1: **基本例**

x、y 項目についてのピアソンの積率相関係数を 3 期を窓として計算する。

```
$ more dat1.csv
t,x,y
1,14,0.17
2,11,0.2
3,32,0.15
4,13,0.33
5,8,0.1
6,19,0.56
$ mmvsim s=t t=3 c=pearson f=x,y a=sim i=dat1.csv o=rsl1.csv
#END# kgmvsim a=sim c=pearson f=x,y i=dat1.csv o=rsl1.csv s=t t=3
$ more rsl1.csv
t%0,x,y,sim
3,32,0.15,-0.8746392857
4,13,0.33,-0.6515529194
5,8,0.1,-0.1164257338
6,19,0.56,0.9986254289
```

関連コマンド

msim：移動窓を設定せずに類似度計算を行う。

mwindow：動窓のデータを作成するので、そのデータを使えば mmvstats で計算できない統計量も計算可能。

mmvavg：移動平均に限定した計算を行う。

4.36　mmvstats 移動窓の統計量の計算

移動窓を設定し、各種統計量 (1 変量) を計算する。mstats コマンドの移動窓バージョンとして考えればよい。

書式

```
mmvstats [s=] [k=] f= [t=] c= [skip=] -n  [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]
```

パラメータ

s=	ここで指定した項目 (複数項目指定可) で並べ替えられた後、各種統計量が計算される。 -q オプションを指定しないとき、s=パラメータは必須。
k=	ここで指定された項目 (複数項目指定可) を単位として集計する。
f=	集計項目名リスト (複数項目指定可) を指定する。
t=	期間数を 1 以上の整数で指定する。
c=	統計量 (以下のリストから一つだけ指定可) sum|mean|devsq|var|uvar|sd|usd|cv|min| |max|range|skew|uskew|kurt|ukurt 詳細な定義は mstats コマンドを参照のこと。
skip=	出力を抑制する最初の行数

利用例

例 1: 基本例

移動窓の合計を計算する。最初の行は期数に満たないため出力されない。

```
$ more dat1.csv
id,value
1,5
2,1
3,3
4,4
5,4
6,6
7,1
8,4
9,7
$ mmvstats s=id f=value t=2 c=sum i=dat1.csv o=rsl1.csv
#END# kgmvstats c=sum f=value i=dat1.csv o=rsl1.csv s=id t=2
$ more rsl1.csv
id%0,value
2,6
3,4
4,7
5,8
6,10
7,7
8,5
9,11
```

関連コマンド

mmvavg：移動平均に限定した計算を行う。
mwindow：動窓のデータを作成するので、そのデータを使えば `mmvstats` で計算できない統計量も計算可能。
mmvsim：移動窓の類似度 (2 変量統計量) の計算を行う。

4.37　mnewnumber 連番データの新規生成

S=パラメータで指定した開始数値もしくはアルファベットにより、I=パラメータで指定した間隔で連番もしくはアルファベット連番を新規作成し、a=パラメータで指定した項目名で出力する。アルファベット連番とは、A から Z の 26 文字を用いた 26 進数のこと (A,B,···,Z,AA,AB,···,AZ,BA,BB,···,ZZ,AAA,AAB,···)。

書式

```
mnewnumber a= [I=] [S=] [l=] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

a=　新規に作成する連番行の項目名を指定する。
　　-nfn,-nfno オプション指定時は指定の必要はない。
I=　連番をふる間隔を指定する。【デフォルト値:1】
S=　開始数値/アルファベット (大文字)【デフォルト値:1】
　　連番の開始数値もしくはアルファベットを指定する。
　　数値を指定した場合は数値の連番がふられる。
　　アルファベットを指定した場合はアルファベット連番がふられる。(小文字は指定できない)
l=　作成するデータ行数を指定する。【デフォルト値:10】

利用例

例 1: 基本例

1 から始まる間隔が 1 の連番をふり、No. という項目名で新規に 5 行のデータを作成する。

```
$ mnewnumber a=No. I=1 S=1 l=5 o=rsl1.csv
#END# kgNewnumber I=1 S=1 a=No. l=5 o=rsl1.csv
$ more rsl1.csv
No.
1
2
3
4
5
```

例 2: 開始番号と間隔の変更

10 から始まる間隔が 5 の連番をふり、No. という項目名で新規に 5 行のデータを作成する。

```
$ mnewnumber a=No. I=5 S=10 l=5 o=rsl2.csv
#END# kgNewnumber I=5 S=10 a=No. l=5 o=rsl2.csv
$ more rsl2.csv
No.
10
15
20
25
30
```

例 3: **アルファベット連番**

A から始まる間隔が 1 のアルファベット連番をふり、No. という項目名で新規に 5 行のデータを作成する。

```
$ mnewnumber a=No. I=1 S=A l=5 o=rsl3.csv
#END# kgNewnumber I=1 S=A a=No. l=5 o=rsl3.csv
$ more rsl3.csv
No.
A
B
C
D
E
```

例 4: **ヘッダ行なしで新規作成**

B から始まる間隔が 3 のアルファベット連番をふり、ヘッダなしで新規に 11 行のデータを作成する。

```
$ mnewnumber  -nfn  I=3 l=11 S=B o=rsl4.csv
#END# kgNewnumber -nfn I=3 S=B l=11 o=rsl4.csv
$ more rsl4.csv
B
E
H
K
N
Q
T
W
Z
AC
AF
```

関連コマンド

mnewrand：新たに乱数を生成する。
mnewstr：固定文字列を生成する。

4.38　mnewrand 乱数データの新規生成

0.0 から 1.0 の範囲の実数乱数を生成する。-int を指定することで、整数乱数を生成することもできる。
乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

書式

mnewrand a= [max=] [min=] [S=] [l=] [-int] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a=	新規に作成するデータの項目名を指定する。 -nfn,-nfno オプション指定時は指定の必要はない。
max=	乱数の最大値を指定する。【デフォルト値:INT_MAX】 このパラメータを指定するときは-int も指定しなければならない。
min=	乱数の最小値を指定する。【デフォルト値:0】 このパラメータを指定するときは-int も指定しなければならない。
S=	乱数の種を指定する。【デフォルト値:現在時刻】
l=	行数【デフォルト値:10】 新規作成する乱数データの行数を指定する。
-int	整数乱数を生成する

利用例

例 1: 基本例

実数乱数を 10 行生成する。乱数の種は 1 に固定しているので、いつ実行しても乱数系列は同じになる。

```
$ mnewrand a=rand S=1 o=rsl1.csv
#END# kgnewrand S=1 a=rand o=rsl1.csv
$ more rsl1.csv
rand
0.4170219984
0.9971848081
0.7203244893
0.9325573612
0.0001143810805
0.1281244478
0.3023325677
0.9990405154
0.1467558926
0.2360889763
```

例 2: 整数乱数

最小値が 0、最大値が 1000、乱数の種が 1 の整数乱数を 5 行作成する。

```
$ mnewrand a=rand -int max=1000 min=0 l=5 S=1 o=rsl2.csv
#END# kgnewrand -int S=1 a=rand l=5 max=1000 min=0 o=rsl2.csv
$ more rsl2.csv
rand
417
```

```
998
721
933
0
```

例 3: ヘッダ行なしで出力

-nfn でヘッダーなしのデータが生成される。

```
$ mnewrand -nfn l=5 S=1 o=rsl3.csv
#END# kgnewrand -nfn S=1 l=5 o=rsl3.csv
$ more rsl3.csv
0.4170219984
0.9971848081
0.7203244893
0.9325573612
0.0001143810805
```

関連コマンド

mnewnumber：連番を生成する。

mnewstr：固定文字列を生成する。

4.39　mnewstr 固定文字列データの新規生成

v=パラメータで指定した文字列データを新規作成し、a=パラメータで指定した項目名で出力する。一度に複数の項目を生成することも可能。

書式

```
mnewstr a= [v=] [l=] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

a=　新規に作成するデータの項目名を指定する。
　　複数の項目を生成する場合は、項目名をカンマで区切る。
　　-nfn,-nfno オプション指定時は指定の必要はない。
v=　新しく作成する文字列を指定する。
　　複数の項目を生成する場合は、値をカンマで区切る。a=で指定した個数と同数でなければならない。
l=　新規作成する乱数データの行数を指定する。【デフォルト値:10】

利用例

例 1: 基本例

custNo と A0001 という文字列データを 5 行作成し、attribute,code という名前の項目名で出力する。

```
$ mnewstr a=attribute,code v=custNo,A0001 l=5 o=rsl1.csv
#END# kgnewstr a=attribute,code l=5 o=rsl1.csv v=custNo,A0001
$ more rsl1.csv
attribute,code
custNo,A0001
custNo,A0001
custNo,A0001
custNo,A0001
custNo,A0001
```

関連コマンド

mnewnumber：連番を生成する。
mnewrand：乱数を生成する。

4.40 mnjoin 参照ファイル項目の自然結合

k=パラメータで指定した入力ファイルの項目値と参照ファイルの項目値を比較し、同じ値の場合 m=パラメータで指定した参照ファイルにある f=パラメータで指定した項目値を自然結合する。mjoin コマンドとの違いは、参照ファイル上のキー項目に重複があってもよい点である。あるキー値について、入力ファイル上に n 件、参照ファイル上に m 件のレコードがあった場合、$n \times m$ 件のレコードが出力されることになる。また、f=を省略すると、参照ファイルのキー項目以外全ての項目を結合する。

書式

mnjoin k= [f=] [K=] [-n] [-N] m=| i= [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= 入力データ上の突き合わせる項目名リストを指定する。
ここで指定した入力データの項目と K=パラメータで指定された
参照データの項目が同じ行の項目結合が行われる。

f= 結合する参照ファイル上の項目名リストを指定する。
省略するとキー項目を除いた全ての項目が結合される。

m= 参照ファイル名を指定する。
このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

K= 参照データ上の突き合わせる項目名リスト
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行の項目結合が行われる。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。

-n 参照データにない入力データを NULL 値として出力するフラグ。

-N 入力データにない参照データを NULL 値として出力するフラグ。

利用例

例 1: 基本例

入力ファイルにある item 項目と、参照ファイルにある item 項目を比較し同じ値の場合、cost 項目を結合する。入力ファイル、参照ファイル共に item=A が 2 行あり、結果、出力ファイルには 2×2=4 行の item=A が出力されている。

```
$ more dat1.csv
item,date,price
A,20081201,100
A,20081213,98
B,20081002,400
B,20081209,450
C,20081201,100
$ more ref1.csv
item,cost
A,50
A,70
B,300
E,200
$ mnjoin k=item f=cost m=ref1.csv i=dat1.csv o=rsl1.csv
#END# kgnjoin f=cost i=dat1.csv k=item m=ref1.csv o=rsl1.csv
$ more rsl1.csv
item%0,date,price,cost
A,20081201,100,50
```

```
A,20081201,100,70
A,20081213,98,50
A,20081213,98,70
B,20081002,400,300
B,20081209,450,300
```

例 2: 未結合データ出力

-n を指定することで、参照ファイルにマッチしない入力ファイルの行 (item="C"の行) も出力し、-N を指定することで、入力ファイルにマッチしない参照ファイルの行 (item="E"の行) も出力する。

```
$ more ref2.csv
item,cost
A,50
B,300
E,200
$ mnjoin k=item f=cost m=ref2.csv -n -N i=dat1.csv o=rsl2.csv
#END# kgnjoin -N -n f=cost i=dat1.csv k=item m=ref2.csv o=rsl2.csv
$ more rsl2.csv
item%0,date,price,cost
A,20081201,100,50
A,20081213,98,50
B,20081002,400,300
B,20081209,450,300
C,20081201,100,
E,,,200
```

関連コマンド

mjoin：参照ファイルのキーが単一化されているのであれば mjoin を使うと若干高速。

mproduct：結合キー関係なく全行の組み合せで結合する。1 行だけからなる参照ファイルを入力ファイル全行に結合する目的で利用することが多い。

4.41 mnormalize 基準化

f=パラメータで指定した項目を、c=パラメータで指定した基準化の方法で基準化する。

書式

mnormalize c= f= [k=] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

c= 以下に示す基準化の方法のいずれかを指定する。
z：z 得点：$z_i = (x_i - m)/u$ (x_i: i 番目のデータ, m :算術平均, u :標準偏差)
Z：偏差値：$Z_i = 50 + 10 \times z_i$
range：最小値を 0, 最大値を 1 に線形変換 $r_i = (x_i - \min_x)/(\max_x - \min_x)$

f= ここで指定された項目が基準化される。
:(コロン）で新項目名を指定する必要がある。例）f=数量:数量基準値

k= キー項目名リスト
ここで指定された項目を単位に基準化を行う。

利用例

例 1: 基本例

「顧客」を単位にして「数量」と「金額」項目を基準化（z 得点）し、「数量基準値」と「金額基準値」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mnormalize c=z k=顧客 f=数量:数量基準値,金額:金額基準値 i=dat1.csv o=rsl1.csv
#END# kgnormalize c=z f=数量:数量基準値,金額:金額基準値 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客,%0,数量,金額,数量基準値,金額基準値
A,1,10,-0.7071067812,-0.7071067812
A,2,20,0.7071067812,0.7071067812
B,1,15,-0.5773502692,0
B,3,10,1.154700538,-1
B,1,20,-0.5773502692,1
```

例 2: 偏差値

```
$ mnormalize c=Z k=顧客 f=数量:数量基準値,金額:金額基準値 i=dat1.csv o=rsl2.csv
#END# kgnormalize c=Z f=数量:数量基準値,金額:金額基準値 i=dat1.csv k=顧客 o=rsl2.csv
$ more rsl2.csv
顧客,%0,数量,金額,数量基準値,金額基準値
A,1,10,42.92893219,42.92893219
A,2,20,57.07106781,57.07106781
```

```
B,1,15,44.22649731,50
B,3,10,61.54700538,40
B,1,20,44.22649731,60
```

例 3: 0 から 1 への線形変換

```
$ mnormalize c=range k=顧客 f=数量:数量基準値,金額:金額基準値 i=dat1.csv o=rsl3.csv
#END# kgnormalize c=range f=数量:数量基準値,金額:金額基準値 i=dat1.csv k=顧客 o=rsl3.csv
$ more rsl3.csv
顧客 %0,数量,金額,数量基準値,金額基準値
A,1,10,0,0
A,2,20,1,1
B,1,15,0,0.5
B,3,10,1,0
B,1,20,0,1
```

関連コマンド

4.42 mnrcommon 参照ファイルの複数範囲条件による行撰択

参照ファイルの範囲条件にマッチする入力ファイルの行を選択する。k=パラメータで指定した入力ファイルの項目値と K=パラメータで指定した参照ファイルの項目値が同じ行について、r=でパラメータで指定した項目値が R=パラメータで指定した 2 項目の値の範囲条件 (項目 1 以上項目 2 未満) にマッチすれば選択する。数値として処理したい場合は r=パラメータの項目名のあとに %n をつけること。

書式

mnrcommon [k=] R= r= [K=] [u=] [-r] m=| i= [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= 入力データ上の突き合わせる項目名リスト (複数項目指定可) を指定する。
ここで指定した入力データの項目と K=パラメータで指定された参照データの項目が同じ行の項目結合が行われる。

m= 参照ファイル名を指定する。
このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

R= 参照ファイル上の範囲項目名 (start,end) を指定する。
第一項目の NULL 値は無限小，第二項目の NULL 値は無限大として扱われる。

r= 範囲比較される入力ファイル上の項目名を指定する。[%n]
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行が選択される。
数値として処理したい場合は r=パラメータの項目名のあとに %n をつける。

K= 参照データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行の項目結合が行われる。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。

u= 指定の条件に一致しない行を出力するファイル名。

-r 条件反転
R=パラメータで指定した行番号以外の行を選択する。

利用例

例 1: 基本例

日付項目の値が 20080203 で、「金額」項目の値が 5 以上 15 未満の行、および 40 以上 50 未満の行を選択する。

```
$ more dat1.csv
日付,金額
20080123,10
20080203,10
20080203,20
20080203,45
20080410,50
$ more ref1.csv
日付,金額 F,金額 T
20080203,5,15
20080203,40,50
$ mnrcommon k=日付 m=ref1.csv R=金額 F,金額 T r=金額 %n i=dat1.csv o=rsl1.csv
#END# kgnrcommon R=金額 F,金額 T i=dat1.csv k=日付 m=ref1.csv o=rsl1.csv r=金額 %n
$ more rsl1.csv
日付 %0,金額
20080203,10
```

```
20080203,45
```

例 2: **条件反転**

-r を付けると選択条件は反転する。

```
$ mnrcommon k=日付 m=ref1.csv R=金額 F, 金額 T r=金額 %n -r i=dat1.csv o=rsl2.csv
#END# kgnrcommon -r R=金額 F, 金額 T i=dat1.csv k=日付 m=ref1.csv o=rsl2.csv r=金額 %n
$ more rsl2.csv
日付 %0, 金額
20080123,10
20080203,20
20080410,50
```

関連コマンド

mcommon：範囲でなく文字列マッチで選択したい場合はこのコマンドを使う。

mnrjoin：選択ではなく参照ファイルの項目を結合する。

4.43 mnrjoin 参照ファイルの複数範囲条件による自然結合

範囲により参照ファイルの項目を結合 (join) する。r=パラメータで指定した項目値が、m=パラメータで指定した参照ファイルの R=パラメータで指定した 2 項目の値の範囲条件 (項目 1 以上項目 2 未満) にマッチすれば f=パラメータの項目を結合する。マッチする行が複数あれば、それらの行全てが出力され、ちょうど自然結合のような動きをする。範囲比較される値は、デフォルトで文字列と見なされる。数値として処理したい場合は r=パラメータの項目名のあとに %n をつける。

書式

mnrjoin R= r= [k=] [K=] [f=] [-n] [-N] m=| i= [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 結合する参照ファイル上の項目名リスト (複数項目指定可) を指定する。
省略すると K=で指定された項目以外の項目を全て結合する。

m= 参照ファイル名を指定する。
このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

R= 範囲項目名リスト (二項目限定)
参照ファイル上の範囲項目名 (start,end) を指定する。
第一項目の NULL 値は無限小，第二項目の NULL 値は無限大として扱われる。

r= 範囲比較される項目名 [%n]
入力ファイル上の項目名を指定する。
数値として処理したい場合は r=パラメータの項目名のあとに %n をつける。

k= 入力データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した入力データの項目と K=パラメータで指定された参照データの項目が同じ行の項目結合が行われる。

K= 参照データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行の項目結合が行われる。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。

-n 参照データにない入力データを NULL 値として出力するフラグ。

-N 入力データにない参照データを NULL 値として出力するフラグ。

例えば、パラメータを k=key K=Key r=val%n R=range i=dat.csv m=ref.csv と指定するのであれば、dat.csv データは、msortf f=key,val%n の条件で、また ref.csv データは、msortf f=Key,range%n の条件によって並べ替えておかなければならない。

利用例

例 1: 基本例

日付項目の値が 20080203 で、「金額」項目の値が 5 以上 15 未満の入力データ行には avg=150 を、40 以上 50 未満の行には avg=200 を結合する。

```
$ more dat1.csv
date,price
20080123,10
20080123,20
20080203,10
20080203,35
20080410,50
```

```
$ more ref1.csv
date,priceF,priceT,avg
20080203,5,15,150
20080203,40,50,200
$ mnrjoin k=date f=avg m=ref1.csv R=priceF,priceT r=price%n i=dat1.csv o=rsl1.csv
#END# kgnrjoin R=priceF,priceT f=avg i=dat1.csv k=date m=ref1.csv o=rsl1.csv r=price%n
$ more rsl1.csv
date%0,price,avg
20080203,10,150
```

例 2: 未結合データ出力

-n を指定することで、参照ファイルにマッチしない入力ファイルの行 (avg=が NULL 値の行) も出力し、-N を指定することで、入力ファイルにマッチしない参照ファイルの行 (price=が NULL 値の行) も出力する。いわゆる外部結合である。

```
$ mnrjoin k=date f=avg m=ref1.csv R=priceF,priceT r=price%n -n -N i=dat1.csv o=rsl2.csv
#END# kgnrjoin -N -n R=priceF,priceT f=avg i=dat1.csv k=date m=ref1.csv o=rsl2.csv r=price%n
$ more rsl2.csv
date%0,price,avg
20080123,10,
20080123,20,
20080203,10,150
20080203,35,
20080203,,200
20080410,50,
```

関連コマンド

mrjoin：参照データの結合キー (K=項目) に重複がなければ mrjoin を使う。

4.44 mnullto NULL 値の置換

f=パラメータで指定した項目について NULL 値を v=パラメータで指定した文字列に置換する。

書式

mnullto f= [v=|-p] [O=] [-A] [i=] [o=] [-assert_diffSize] [-nfn] [-nfno] [-x] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= ここで指定した項目 (複数項目指定可) の NULL 値が置換される。
v= ここで指定した文字列に NULL 値を置換する。
-p 前の行の値で置換する。
v=パラメータと同時に指定できない。
k= -p を指定した時にのみ意味があり、ここで指定した項目値の単位で置換処理を行なう。
s= -p を指定した時にのみ意味があり、k=項目内での並び順を指定する。
O= NULL 値以外を置換したい場合は、ここで値を指定する。指定しなければ NULL 値以外は置換しない。
-A このオプションにより、指定した項目を置き換えるのではなく、新たに項目が追加される。
-A オプションを指定した場合は必ず、
:(コロン）で新項目名を指定する必要がある。例）f=数量:置換後の項目名

利用例

例 1: 基本例

birthday 項目が NULL 値の場合は"no value"に置換する。

```
$ more dat1.csv
customer,birthday
A,19690103
B,
C,19500501
D,
E,
$ mnullto f=birthday v="no value" i=dat1.csv o=rsl1.csv
#END# kgnullto f=birthday i=dat1.csv o=rsl1.csv v=no value
$ more rsl1.csv
customer,birthday
A,19690103
B,no value
C,19500501
D,no value
E,no value
```

例 2: NULL 値以外の置換

birthday 項目が NULL 値の場合は、"no value" 値がある場合は"value"置換し entry という項目名に変更して出力する。

```
$ mnullto f=birthday:entry v="no value" O="value" i=dat1.csv o=rsl2.csv
#END# kgnullto O=value f=birthday:entry i=dat1.csv o=rsl2.csv v=no value
$ more rsl2.csv
```

```
customer,entry
A,value
B,no value
C,value
D,no value
E,no value
```

例 3: **新しい項目を出力**

birthday 項目が NULL 値の場合は"no value"、値がある場合は"value"に置換し entry という項目名で出力する。

```
$ mnullto f=birthday:entry v="no value" O="value" -A i=dat1.csv o=rsl3.csv
#END# kgnullto -A O=value f=birthday:entry i=dat1.csv o=rsl3.csv v=no value
$ more rsl3.csv
customer,birthday,entry
A,19690103,value
B,,no value
C,19500501,value
D,,no value
E,,no value
```

例 4: **前行の値に置換**

```
$ more dat3.csv
id,seq,val
A,1,1
A,3,2
A,2,
B,2,3
B,1,
$ mnullto f=val -p i=dat3.csv o=rsl4.csv
#END# kgnullto -p f=val i=dat3.csv o=rsl4.csv
$ more rsl4.csv
id,seq,val
A,1,1
A,3,2
A,2,2
B,2,3
B,1,3
```

例 5: **キー項目を指定した場合の例**

```
$ mnullto k=id s=seq f=val -p i=dat3.csv o=rsl5.csv
#END# kgnullto -p f=val i=dat3.csv k=id o=rsl5.csv s=seq
$ more rsl5.csv
id%0,seq%1,val
A,1,1
A,2,1
A,3,2
B,1,
B,2,3
```

関連コマンド

mdelnull：置換ではなく、行を削除したい場合はこちら。

mchgstr：NULL 値でなく文字列を置換したい場合に使用する。

4.45 mnumber 連番

数字連番もしくはアルファベット連番 (A,B,...,Z,AA,AB,...,AZ,BA,BB,...,ZZ,AAA,AAB,...) ををふり、a=パラメータで指定した項目名で出力する。

書式

mnumber a= [e=] [I=] [k=] [s=] [S=] [-B] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a= 新たに追加される項目の名前を指定する。【但し、-nfn,-nfno オプション指定時は必要なし】

e= 同 Rank の処理方法
同一キー同一ソート項目値への処理方法を指定する。
指定しない場合は、デフォルトとして (e=seq) 並び順で No もしくは文字列が付け加えられる。
seq:同 Rank の場合は並び順で No もしくはアルファベット連番を付け加える。
same:同 Rank の場合は同じ No もしくはアルファベットを付け加える。
skip:同 Rank の場合は同じ No を振り、
次の No はスキップするように No もしくはアルファベット連番を付け加える。
(注意)e={same/skip}を指定する場合は、s=パラメータを同時に指定しなければならない。

I= 連番の間隔を指定する。

k= 連番もしくは連文字をふる単位となる項目名リスト (複数項目指定可) を指定する。【集計キーブレイク処理】
(注意) 指定する場合は事前に k=パラメータで指定する連番、
もしくは連文字をふる単位となる項目順に並べ替えておく必要がある。

s= ここで指定した項目 (複数項目指定可) で並べ替えられた後、連番が追加される。
-B オプション指定時以外は必須。

S= 開始 No
連番の開始 No を指定する。
大文字のアルファベットが指定された場合はアルファベット連番となる。
ただし、アルファベット連番の場合、間隔 (I=) に負の値は指定できない。

-B キー毎に連番もしくはアルファベット連番をふる。
あるキー内では全行同じ No もしくはアルファベットがふられる。

利用例

例 1: 数字の連番

Customer 項目名順（昇順）に連番を振り「No」という項目名で出力する。

```
$ more dat1.csv
Customer,Val,Sum
A,29,300
B,35,250
C,15,200
D,23,150
E,10,100
$ mnumber s=Customer a=No i=dat1.csv o=rsl1.csv
#END# kgnumber a=No i=dat1.csv o=rsl1.csv s=Customer
$ more rsl1.csv
Customer%0,Val,Sum,No
A,29,300,0
B,35,250,1
```

```
C,15,200,2
D,23,150,3
E,10,100,4
```

例 2: Date **項目順の連番**

Date 項目順（昇順）に連番をふる。その際、同じ Date には同じ No を振り「No」という項目名で出力する。

```
$ more dat2.csv
Date
20090101
20090101
20090102
20090103
20090103
$ mnumber k=Date a=No -B i=dat2.csv o=rsl2.csv
#END# kgnumber -B a=No i=dat2.csv k=Date o=rsl2.csv
$ more rsl2.csv
Date%0,No
20090101,0
20090101,0
20090102,1
20090103,2
20090103,2
```

例 3: Sum **項目順の連番 (同** Rank **は同じアルファベットをふる)**

Sum 項目の多い順（降順）にアルファベットの A から順に連文字を振り「Rank」という項目名で出力する。また、同 Rank の場合は同じアルファベット文字を振ることにする。

```
$ more dat3.csv
Customer,Val,Sum
A,3,300
B,1,250
C,2,250
D,1,150
E,1,100
$ mnumber a=Rank e=same s=Sum%nr S=A  i=dat3.csv o=rsl3.csv
#END# kgnumber S=A a=Rank e=same i=dat3.csv o=rsl3.csv s=Sum%nr
$ more rsl3.csv
Customer,Val,Sum%0nr,Rank
A,3,300,A
B,1,250,B
C,2,250,B
D,1,150,C
E,1,100,D
```

例 4: Sum **項目順の連番 (同** Rank **は並び順で** No **をふる)**

Sum 項目の多い順（降順）に連番を振り「Rank」という項目名で出力する。その際、同 Rank の場合は並び順で No を振ることにする。

```
$ mnumber a=Rank e=seq s=Sum%nr i=dat3.csv o=rsl4.csv
#END# kgnumber a=Rank e=seq i=dat3.csv o=rsl4.csv s=Sum%nr
$ more rsl4.csv
Customer,Val,Sum%0nr,Rank
A,3,300,0
B,1,250,1
C,2,250,2
```

```
D,1,150,3
E,1,100,4
```

例 5: Sum 項目順の連番 (同 Rank は同じ No をふる)

Sum 項目の多い順（降順）に連番を振り「Rank」という項目名で出力する。その際、同 Rank の場合は同じ No を振ることにする。

```
$ mnumber a=Rank e=same s=Sum%nr i=dat3.csv o=rsl5.csv
#END# kgnumber a=Rank e=same i=dat3.csv o=rsl5.csv s=Sum%nr
$ more rsl5.csv
Customer,Val,Sum%0nr,Rank
A,3,300,0
B,1,250,1
C,2,250,1
D,1,150,2
E,1,100,3
```

例 6: Sum 項目順の連番 (同 Rank の場合は同じ RankNo を振り、次の No はスキップ)

Sum 項目の多い順（降順）に連番を振り「Rank」という項目名で出力する。その際、同 Rank の場合は同じ RankNo を振り、次の No はスキップするように No を振ることにする。

```
$ mnumber a=Rank e=skip s=Sum%nr i=dat3.csv o=rsl6.csv
#END# kgnumber a=Rank e=skip i=dat3.csv o=rsl6.csv s=Sum%nr
$ more rsl6.csv
Customer,Val,Sum%0nr,Rank
A,3,300,0
B,1,250,1
C,2,250,1
D,1,150,3
E,1,100,4
```

例 7: 10 から始まる連番

Sum 項目の小さい順（昇順）に 10 から始まる連番を振り「Score」という項目名で出力する。その際、同 Rank の場合は同じ RankNo を振り、次の No はスキップするように No を振ることにする。

```
$ more dat4.csv
Customer,Val,Sum
A,1,100
B,1,150
C,1,250
D,2,250
E,3,300
$ mnumber a=Score e=same s=Sum%n S=10 i=dat4.csv o=rsl7.csv
#END# kgnumber S=10 a=Score e=same i=dat4.csv o=rsl7.csv s=Sum%n
$ more rsl7.csv
Customer,Val,Sum%0n,Score
A,1,100,10
B,1,150,11
C,1,250,12
D,2,250,12
E,3,300,13
```

例 8: 10 から始まる 5 つ飛びの連番

Sum 項目の小さい順番（昇順）に 10 から始まる 5 つ飛びの連番を振り「Score」という項目名で出力する。また、同 Rank の場合は同じ No を振ることにする。

```
$ mnumber a=Score e=same s=Sum%n S=10 I=5 i=dat4.csv o=rsl8.csv
#END# kgnumber I=5 S=10 a=Score e=same i=dat4.csv o=rsl8.csv s=Sum%n
$ more rsl8.csv
Customer,Val,Sum%0n,Score
A,1,100,10
B,1,150,15
C,1,250,20
D,2,250,20
E,3,300,25
```

関連コマンド

mnewnumber：新たに連番データを生成する場合に使う。

mbest：行番号による選択であれば、mnumber を使わずともこのコマンドで。

4.46 mpadding (行補完) コマンド

k=パラメータで指定した項目をキーとして、f=パラメータで指定した項目の値が連続するようにレコードを作成する。v=パラメータを指定した場合は、k=,f=で指定した以外の項目値を指定した文字列でパディングし、-n オプション指定時は、null でパディングする。(v=,-n 共に指定がなければ直前の項目値でパディングする)

書式

mpadding [k=] f= [v=] [S=] [E=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k= ここで指定された項目をキーとする。
f= 連続パディング対象項目名
ここで指定された項目の値が連続するようにレコードをパディングする。
数字としてパディングするときは、no%n のように %n を指定する。
日付としてパディングするときは %d、時刻としてパディングするときは %t を指定する。
降順でパディングしたいときは no%d%r のように r を追加する。
v= パディング用文字列指定
k=,f=で指定した以外の項目値を指定した文字列で出力する。
S= 開始値
f=で指定した項目の値の開始値を指定する。
E= 終了値
f=で指定した項目の値の終了値を指定する。
-n パディングに null 値指定
k=,f=で指定した以外の項目値を null で出力する。

利用例

例 1: 基本例

no 項目が整数 (%n) として連続するようにレコードをパディングする。1 と verb—4—の間に 2,3 を、4 と 2 の間に 3 が挿入されている。

```
$ more dat1.csv
no
3
6
8
$ mpadding f=no%n i=dat1.csv o=rsl1.csv
#END# kgpadding f=no%n i=dat1.csv o=rsl1.csv
$ more rsl1.csv
no%0n
3
4
5
6
7
8
```

例 2: **開始値、終了値の指定**

行間のパディングだけでなく、先頭行/終端行の前後もパディングする。前後の範囲は S=,E=で指定する。

```
$ mpadding f=no%n S=1 E=10 i=dat1.csv o=rsl2.csv
#END# kgpadding E=10 S=1 f=no%n i=dat1.csv o=rsl2.csv
$ more rsl2.csv
no%0n
1
2
3
4
5
6
7
8
9
10
```

例 3: **日付パディング**

date 項目が日付 (%d) として連続するようにレコードをパディングする。k=,f=で指定した以外の項目は、直前の行の項目値でパディングする。

```
$ more dat2.csv
date,dummy
20130929,a
20131002,b
20131004,c
$ mpadding f=date%d i=dat2.csv o=rsl3.csv
#END# kgpadding f=date%d i=dat2.csv o=rsl3.csv
$ more rsl3.csv
date%0,dummy
20130929,a
20130930,a
20131001,a
20131002,b
20131003,b
20131004,c
```

例 4: **パディング用文字列指定**

v=にてパディング文字列を指定することもできる。

```
$ mpadding f=date%d v=padding i=dat2.csv o=rsl4.csv
#END# kgpadding f=date%d i=dat2.csv o=rsl4.csv v=padding
$ more rsl4.csv
date%0,dummy
20130929,a
20130930,padding
20131001,padding
20131002,b
20131003,padding
20131004,c
```

例 5: **パディングに** NULL **値を指定**

-n を指定して NULL 値でパディングすることも可能。

```
$ mpadding f=date%d -n i=dat2.csv o=rsl5.csv
#END# kgpadding -n f=date%d i=dat2.csv o=rsl5.csv
$ more rsl5.csv
date%0,dummy
20130929,a
20130930,
20131001,
20131002,b
20131003,
20131004,c
```

関連コマンド

4.47　mpaste 参照ファイル項目の行番号マッチング結合

入力ファイルと参照ファイルを行番号でマッチングさせて結合する。データ件数が異なる場合は、少い方のデータに合わせる。-n や-N を指定することでマッチングできな行も NULL 値で結合することが可能である。

書式

mpaste [f=] -n -N m=| i= [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=	結合する参照ファイル上の項目名リスト (複数項目指定可)。 省略するとキー項目を除いた全ての項目が結合される。
m=	参照ファイル名を指定する。 このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)
-n	入力ファイルにあって参照ファイルにない場合に NULL 値を出力する。
-N	参照ファイルにあって入力ファイルにない場合に NULL 値を出力する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id1
1
2
3
4
$ more ref1.csv
id2
a
b
c
d
$ mpaste m=ref1.csv i=dat1.csv o=rsl1.csv
#END# kgpaste i=dat1.csv m=ref1.csv o=rsl1.csv
$ more rsl1.csv
id1,id2
1,a
2,b
3,c
4,d
```

例 2: **行数が異なる例**

入力ファイルと参照ファイルで行数が異なる場合は、少いファイルの行数に合わせる。

```
$ more ref2.csv
id2
a
b
$ mpaste m=ref2.csv i=dat1.csv o=rsl2.csv
#END# kgpaste i=dat1.csv m=ref2.csv o=rsl2.csv
```

```
$ more rsl2.csv
id1,id2
1,a
2,b
```

例 3: 外部結合

-n を指定すると、参照ファイルの行数が少なくても、対応しない入力ファイルの行も NULL 値を結合して出力する。

```
$ mpaste m=ref2.csv -n i=dat1.csv o=rsl3.csv
#END# kgpaste -n i=dat1.csv m=ref2.csv o=rsl3.csv
$ more rsl3.csv
id1,id2
1,a
2,b
3,
4,
```

例 4: 結合項目を指定

```
$ more ref3.csv
id2,val
a,R0
b,R1
c,R2
d,R3
$ mpaste f=val m=ref3.csv i=dat1.csv o=rsl4.csv
#END# kgpaste f=val i=dat1.csv m=ref3.csv o=rsl4.csv
$ more rsl4.csv
id1,val
1,R0
2,R1
3,R2
4,R3
```

関連コマンド

mjoin：行番号でなく、キー項目で結合する。

4.48　mproduct 参照ファイルの直積結合

入力データ 1 行に対して、m=パラメータで指定した参照データの f=パラメータで指定した項目全行を結合する。

書式

```
mproduct [f=] m=| i= [o=] [bufcount=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=]
[--help] [--helpl] [--version]
```

パラメータ

f=　結合する参照ファイル上の項目名リスト (複数項目指定可)。
　　省略するとキー項目を除いた全ての項目が結合される。
m=　参照ファイル名を指定する。
　　このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

利用例

例 1: 基本例

入力ファイルにある customer 項目に対して、参照ファイルにある date 項目全行を結合する。

```
$ more dat1.csv
customer
A
B
$ more ref1.csv
date
20090101
20090201
20090301
$ mproduct f=date m=ref1.csv i=dat1.csv o=rsl1.csv
#END# kgproduct f=date i=dat1.csv m=ref1.csv o=rsl1.csv
$ more rsl1.csv
customer,date
A,20090101
A,20090201
A,20090301
B,20090101
B,20090201
B,20090301
```

関連コマンド

mnjoin：結合キーを指定しての mproduct のような結合を行う。

4.49 mrand 擬似乱数

0.0 から 1.0 の範囲の実数の擬似乱数、もしくは範囲指定による整数の擬似乱数を生成し、a=パラメータで指定した項目名で出力する。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

書式

mrand [k=] a= [max=] [min=] [S=] [-int] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k=	指定したキー項目について、同じキー値には同じ乱数値が振られる。
a=	新たに追加される項目の名前を指定する。【但し、-nfn,-nfno オプション指定時は必要なし】
max=	乱数の最大値【デフォルト値:INT_MAX】 0〜2^{32}(約 21 億) の範囲の整数が指定可能 このパラメータを指定するときは-int も指定しなければならない。
min=	整数乱数の最小値【デフォルト値:0】 0〜2^{32}(約 21 億) の範囲の整数が指定可能 このパラメータを指定するときは-int も指定しなければならない。
S=	乱数の種【デフォルト値:現在時刻】 同じ乱数の種を指定すれば、同じ乱数系列となる。 S=を指定しなければ、時刻 (ミリ (1/1000 秒単位) に応じた異なる乱数の種が利用される。 指定可能な乱数の種 (-2147483648〜2147483647)
-int	整数乱数を生成

利用例

例 1: 基本例

0.0 から 1.0 の範囲の実数乱数を生成する。

```
$ more dat1.csv
顧客
A
B
C
D
E
$ mrand a=rand i=dat1.csv o=rsl1.csv
#END# kgrand a=rand i=dat1.csv o=rsl1.csv
$ more rsl1.csv
顧客,rand
A,0.1769897994
B,0.6509473284
C,0.2095655163
D,0.2693608883
E,0.849216162
```

例 2: **基本例** 2

-int で整数乱数

```
$ mrand a=rand -int i=dat1.csv o=rsl2.csv
#END# kgrand -int a=rand i=dat1.csv o=rsl2.csv
$ more rsl2.csv
顧客,rand
A,2049311444
B,523788637
C,1710064583
D,214062764
E,303309628
```

例 3: **最小値、最大値を決めた乱数の生成**

最小値が 10、最大値が 100 の整数の乱数を生成し、rand という項目名で出力する。

```
$ mrand a=rand -int min=10 max=100 S=1 i=dat1.csv o=rsl3.csv
#END# kgrand -int S=1 a=rand i=dat1.csv max=100 min=10 o=rsl3.csv
$ more rsl3.csv
顧客,rand
A,47
B,100
C,75
D,94
E,10
```

例 4: **キー単位の乱数生成**

以下の例は、顧客 A,B,C,D の 4 人について同じ顧客には同じ乱数値を振る。

```
$ more dat2.csv
顧客
A
A
A
B
B
C
D
D
D
$ mrand k=顧客 -int min=0 max=1 a=rand i=dat2.csv o=rsl4.csv
#END# kgrand -int a=rand i=dat2.csv k=顧客 max=1 min=0 o=rsl4.csv
$ more rsl4.csv
顧客 %0,rand
A,1
A,1
A,1
B,0
B,0
C,1
D,1
D,1
D,1
```

関連コマンド

mselrand：ランダムに行を選択する。
mnewrand：入力ファイルなしに、乱数データを新たに生成する。

4.50　mrjoin 参照ファイルの範囲条件による結合

範囲により参照ファイルの項目を結合 (join) する。r=パラメータで指定した項目値が、参照ファイル上にある範囲条件 (項目値以上、次行の項目値未満) にマッチすれば f=パラメータで指定した項目値を結合する。より複雑な範囲条件で結合したければ mnrjoin を使う。範囲条件数が少なければ mchgnum の利用を考えるとよい。

書式

mrjoin r= [k=] [K=] [R=] [f=] [-n] [-lo] [m=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 結合する参照ファイル上の項目名リスト (複数項目指定可)。
省略すると K=で指定された項目以外の項目を全て結合する。

m= 参照ファイル名を指定する。
このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)

r= 範囲比較される項目名 [%n]
入力ファイル上の項目名を指定する。
ここでここで指定した項目 (複数項目指定可) で並べ替えられた後、結合が行われる。
%n が指定されると、数値範囲として解釈し、指定がなければ文字列範囲として解釈する。
ここで指定する項目に NULL 値があってはならない。NULL 値があった場合の動作は不定である。

R= 参照ファイル上の範囲項目名。
省略時は r=パラメータと同名として扱われる。

k= 入力データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した入力データの項目と K=パラメータで指定された参照データの項目が同じ行の項目結合が行われる。

K= 参照データ上の突き合わせる項目名リスト (複数項目指定可)
ここで指定した参照データの項目と k=パラメータで指定された入力データの項目が同じ行の項目結合が行われる。
参照データ上に k=パラメータで指定した入力データ上の項目と同名の項目が存在する場合は指定する必要はない。

-n 参照データにない入力データを NULL 値として出力するフラグ。

-lo left open interval
R= パラメータで指定した範囲を左半開区間（より大きい〜以下）と解釈する。

利用例

例 1: 基本例

price を範囲で分類項目 low、middle、high を結合する。

```
$ more dat1.csv
price
8
15
35
50
90
200
$ more ref1.csv
range,category
10,low
35,middle
80,high
100,
```

```
$ mrjoin r=price%n m=ref1.csv R=range f=category i=dat1.csv o=rsl1.csv
#END# kgrjoin R=range f=category i=dat1.csv m=ref1.csv o=rsl1.csv r=price%n
$ more rsl1.csv
price%0n,category
15,low
35,middle
50,middle
90,high
```

例 2: **基本例** 2

```
$ mrjoin -lo r=price%n m=ref1.csv R=range f=category i=dat1.csv o=rsl2.csv
#END# kgrjoin -lo R=range f=category i=dat1.csv m=ref1.csv o=rsl2.csv r=price%n
$ more rsl2.csv
price%0n,category
15,low
35,low
50,middle
90,high
```

例 3: **基本例** 3

```
$ mrjoin -n r=price%n m=ref1.csv R=range f=category i=dat1.csv o=rsl3.csv
#END# kgrjoin -n R=range f=category i=dat1.csv m=ref1.csv o=rsl3.csv r=price%n
$ more rsl3.csv
price%0n,category
8,
15,low
35,middle
50,middle
90,high
200,
```

例 4: **商品別に異なる範囲を設定して結合**

```
$ more dat2.csv
item,price
A,10
A,20
B,10
B,20
$ more ref2.csv
item,price,category
A,10,low
A,15,high
A,100,
B,10,low
B,35,high
B,100,
$ mrjoin k=item r=price%n m=ref2.csv f=category i=dat2.csv o=rsl4.csv
#END# kgrjoin f=category i=dat2.csv k=item m=ref2.csv o=rsl4.csv r=price%n
$ more rsl4.csv
item%0,price%1n,category
A,10,low
A,20,high
B,10,low
B,20,low
```

関連コマンド

mchgnum：数値範囲を指定して値を置換/追加する。
mjoin：数値範囲ではなく文字列一致による結合の場合はこのコマンドを使う。
mnrcommon：結合ではなく選択する場合はこのコマンドを使う。

4.51 msed 正規表現による文字列置換

f=パラメータで指定した項目について、c=パラメータで指定した正規表現に一致する内容を v=パラメータ指定した文字列で置換する。

書式

msed c= f= v= [-A] [-g] [-W] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 置換対象となる項目名リスト (複数項目指定可) を指定する。
c= 置換したい文字列についての正規表現を指定する。
正規表現の使用方法参照
v= c=パラメータで指定した正規表現にマッチした部分文字列が、
ここで指定した文字列に置換される。
マッチ結果を用いた置換も可能で、指定方法は以下の通り。
$& : マッチした文字列
$‘ : 置換対象文字列の先頭から、マッチした文字列の直前までの文字列
$’ : マッチした文字列の直後から、置換対象文字列の最後までの文字列
$N : N 番目の部分マッチ文字列 (N>=1)
-A このオプションにより、指定した項目を置き換えるのではなく、
新たに項目が追加される。
-g 正規表現にマッチする全ての部分文字列を置換対象とする。
-W ワイド文字として正規表現による文字列置換を行う。

正規表現の使用方法

c=パラメータで指定する正規表現を表 4.9 から表 4.12 に示す。

表 4.9 正規表現 1 文字マッチ

正規表現	意味	値例	c=,v=例	結果例
.	任意の一文字	abbbcc	c=. v=X -g	XXXXXX
[abc]	a,b,c のいずれか一文字	abbbcc	c=[ac] v=X -g	XbbbXX
[^abc]	a,b,c 以外の任意の一文字	abbbcc	c=[^ac] v=X -g	aXXXcc
[a-z]	a から z の範囲の任意の一文字	abbbcc	c=[a-b] v=X -g	XXXXcc
[^a-z]	a から z の範囲外の任意の一文字	abbbcc	c=[^a-b] v=X -g	abbbXX
\t	タブ文字			
\w	単語構成文字 ([0-9a-zA-Z_])	ab#cd&ef	c=\w v=X -g	XX#XX&XX
\W	単語構成文字以外	ab#cd&ef	c=\w v=X -g	abXcdXef
\s	空白文字 ([\t])	ab cd ef	c=\s v=X -g	abXcdXef
\S	空白文字以外	ab cd ef	c=\s v=X -g	XX XX XX
\d	数字構成文字文字 ([0-9])	ab12c0	c=\d v=X -g	abXXcX
\D	数字構成文字文字以外	ab12c0	c=\d v=X -g	XX12X0

表 4.10　正規表現繰り返し

正規表現	意味	値例	c=,v=例	結果例
a*	a の 0 個以上の繰り返し	abbbcc	c=ab* v=X	Xcc
a+	a の 1 個以上の繰り返し	abbbcc	c=ab+ v=X	Xcc
a?	a の 0 個または 1 個の出現	abbbcc	c=ab? v=X	Xbbcc
a{M,N}	a の M 個以上 N 個以下の繰り返し	abbbbbcc	c=ab{3,4} v=X	Xbcc
a{M}	a の M 個以上の繰り返し	abbbbbcc	c=ab{3} v=X	Xbbcc
a | b	a または b	abbbc	c=(ab) | (bc) v=X	XbX
?	繰り返し記号の後ろに付けて最短マッチ	abbbc	c=ab*? v=X	Xbbbc

表 4.11　正規表現位置指定

正規表現	意味	値例	c=,v=例	結果例
^	行頭にマッチする	abac	c=^a v=X -g	Xbac
$	行末にマッチする	acac	c=c$ v=X -g	acaX
\b	単語頭または単語末にマッチ	aac ba ac bac	c=\ba v=X -g	Xac bX Xc bac
\B	単語中にマッチ	aac ba ac bac	c=\Ba v=X -g	aXc ba ac bXc

表 4.12　その他

正規表現	意味	値例	c=,v=例	結果例
(expr)	グループ化			
\1,..,\9	後方参照	abbcababc	c=(ab)(bc)\1 v=x	Xabc
(?=expr)	expr にマッチする直前位置にマッチ			
(?!expr)	expr にマッチしない直前位置にマッチ			

利用例

例 1: **基本例**

zipCode 項目の値が 00 から始まる 4 桁文字列を####に置換する。

```
$ more dat1.csv
customer,zipCode
A,6230041
B,6240053
C,6330032
D,6230087
E,6530095
$ msed f=zipCode c=00.. v=#### i=dat1.csv o=rsl1.csv
#END# kgsed c=00.. f=zipCode i=dat1.csv o=rsl1.csv v=####
$ more rsl1.csv
customer,zipCode
A,623####
B,624####
C,633####
D,623####
E,653####
```

例 2: **項目名指定**

zipCode の値が 00 から始まる 4 桁の数字を####に置換し、zipCode4 という項目名で出力する。

```
$ msed f=zipCode:zipCode4 c='00\d\d' v=#### i=dat1.csv o=rsl2.csv
#END# kgsed c=00\d\d f=zipCode:zipCode4 i=dat1.csv o=rsl2.csv v=####
$ more rsl2.csv
```

```
customer,zipCode4
A,623####
B,624####
C,633####
D,623####
E,653####
```

例 3: **グローバル置換**

zipCode の値が 0 を全て-にグローバル置換する。

```
$ msed f=zipCode c=0 v=- -g i=dat1.csv o=rsl3.csv
#END# kgsed -g c=0 f=zipCode i=dat1.csv o=rsl3.csv v=-
$ more rsl3.csv
customer,zipCode
A,623--41
B,624--53
C,633--32
D,623--87
E,653--95
```

例 4: **部分置換**

item の先頭の fruit を削除する。先頭一致 (^) を指定しているので、最後の行の grapefruit は削除されていないことに注意する。

```
$ more dat2.csv
item,price
fruit:apple,100
fruit:peach,250
fruit:pineapple,300
fruit:orange,450
fruit:grapefruit,500
$ msed f=item c='^fruit' v= -g i=dat2.csv o=rsl4.csv
#END# kgsed -g c=^fruit f=item i=dat2.csv o=rsl4.csv v=
$ more rsl4.csv
item,price
:apple,100
:peach,250
:pineapple,300
:orange,450
:grapefruit,500
```

例 5: **マッチ結果を用いた置換**

v=の中で$&を用いれば、マッチした文字列 (連続した b) に置き換わる。

```
$ more dat3.csv
str1
abc
abbc
ac
$ msed f=str1 c='b+' v='#$&#' i=dat3.csv o=rsl5.csv
#END# kgsed c=b+ f=str1 i=dat3.csv o=rsl5.csv v=#$&#
$ more rsl5.csv
str1
a#b#c
a#bb#c
ac
```

例 6: グローバルマッチとの組み合せ

グローバルマッチにすると、個々のマッチ毎に v=の内容が評価される。

```
$ msed f=str1 c=b v='#$&#' -g i=dat3.csv o=rsl6.csv
#END# kgsed -g c=b f=str1 i=dat3.csv o=rsl6.csv v=#$&#
$ more rsl6.csv
str1
a#b#c
a#b##b#c
ac
```

例 7: プレフィックス置換

$‘にて、マッチした箇所の前の文字列 (プレフィックス) に置換される。

```
$ msed f=str1 c=b v='#$‘#' i=dat3.csv o=rsl7.csv
#END# kgsed c=b f=str1 i=dat3.csv o=rsl7.csv v=#$‘#
$ more rsl7.csv
str1
a#a#c
a#a#bc
ac
```

例 8: サフィックス置換

$’ にて、マッチした箇所の後の文字列 (サフィックス) に置換される。

```
$ msed f=str1 c=b v="#$'#" i=dat3.csv o=rsl8.csv
#END# kgsed c=b f=str1 i=dat3.csv o=rsl8.csv v=#$'#
$ more rsl8.csv
str1
a#c#c
a#bc#bc
ac
```

関連コマンド

mchgstr：単純な文字列マッチによる置換であればこのコマンドを利用する。

mcal：正規表現を扱う関数がいくつか用意されている。

4.52 msel 条件式による行選択

c=パラメータで指定した計算式で計算をおこない、結果が真であれば、その行を選択する。なお mcal と同じ計算式が利用できるので、詳細は mcal を参照されたい。

書式

msel c= [u=] [-r] [i=] [o=] [-assert_diffSize] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

c= 用意された関数を組み合わせて計算する式を指定する。
詳細は mcal を参照。
o= 指定の条件に一致する行を出力するファイル名を指定する。
u= 指定の条件に一致しない行を出力するファイル名を指定する。
-r 条件反転
選択ではなく削除する。

利用例

例 1: 基本例

「金額」項目の値が 40 より大きい行を選択する。それ以外のデータは unmatch1.csv に出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,30
B,3,40
B,1,50
$ msel c='${金額}>40' u=unmatch1.csv i=dat1.csv o=match1.csv
#END# kgsel c=${金額}>40 i=dat1.csv o=match1.csv u=unmatch1.csv
$ more match1.csv
顧客,数量,金額
B,1,50
$ more unmatch1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,30
B,3,40
```

例 2: NULL 値の選択規制

msel コマンドでは c=で与えられた式を評価した結果が NULL 値の場合その行は選択されない。また、アンマッチ出力ファイルが u=によって指定されていれば、そのファイルに出力される。以下の例では項目 b に-1、NULL 値、1 を持つ 3 行のデータについて、0 より大きい行を選択しているが、NULL 値を含む行はアンマッチ出力ファイルへと出力される。

```
$ more dat2.csv
a,b
```

```
A,-1
B,
C,1
$ msel c='${b}>0' i=dat2.csv o=match2.csv u=unmatch2.csv
#END# kgsel c=${b}>0 i=dat2.csv o=match2.csv u=unmatch2.csv
$ more match2.csv
a,b
C,1
$ more unmatch2.csv
a,b
A,-1
B,
```

例 3: -r **オプション指定**

真偽は逆転するが NULL 値の評価に変わりはない。すなわち NULL 値の行は選択されない。以下の例では、上の例と同様のデータおよび選択条件で-r をつけている。真偽の選択条件は逆転しているが、NULL 値を含む行は上記の例と同様にアンマッチファイルへと出力されていることがわかる。

```
$ msel -r c='${b}>0' i=dat2.csv o=match3.csv u=unmatch3.csv
#END# kgsel -r c=${b}>0 i=dat2.csv o=match3.csv u=unmatch3.csv
$ more match3.csv
a,b
A,-1
$ more unmatch3.csv
a,b
B,
C,1
```

関連コマンド

mselnum：簡単な数値範囲による行選択はこちら。

mselstr：簡単な文字列マッチによる行選択はこちら。

mcal：行選択でなく、計算の結果を項目として出力する。

4.53 mseldsp 選択画面入力

注) 本コマンドは開発バージョンであり、仕様が変更される可能性があります。

座標 x=,y=で指定したターミナル上の位置に i=、もしくは seldata=で指定した文字列リストの選択画面を表示し、ユーザが選んだ文字列を o=で指定したファイルに出力する。利用者は一つの選択肢しか選択できない。複数の選択肢を選択せたければ mmseldsp コマンドを利用する。出力される CSV ファイルは、一項目一行のデータとなる。入力枠に何も入力せずに終了した場合は、null 値が出力される (すなわち改行だけが出力される)。f=を指定すれば、項目名を出力できる。f=を省略すれば、項目名ヘッダーは出力されない。選択肢の数が多くて画面をはみ出る場合は、height=でスクロール窓の行数を指定すればよい。

選択画面でエンターキーを押すと、終了ステータス 0 を返して終了し、エスケープキーを押すと、終了ステータス 1 を返して終了する。いずれのキーで終了しても、選択画面で選ばれた内容はファイルに出力される。

座標は左上が x=1,y=1 である (エスケープシーケンスの仕様)。x=もしくは y=で指定した値が 1 より小さい場合は、1 を指定したものとして動作する。またターミナルの範囲を超えた座標が指定された場合の動作は不定である。

書式

```
mseldsp x= y= [height=] i=|seldata= o= [-nfn] [-nfno] [-x] [--help] [--helpl] [--version]
```

パラメータ

x=	x 軸 (左から右への横方向) 表示開始位置 (1 以上の値) を指定する。
y=	y 軸 (上から下への縦方向) 表示開始位置 (1 以上の値) を指定する。
height=	選択肢を表示する行数。
i=	選択肢の文字列を項目として持つ CSV ファイル名
f=	選択肢の文字列を項目として持つ CSV ファイル名
seldata=	カンマで区切られた選択肢の文字列リスト

利用例

例 1: 基本例

ターミナルの x=10,y=3 の位置に sel.txt の内容を表示し、利用者が選んだ文字列を rsl1.txt に出力する。

```
$ more sel.txt
1:そう思う
2:どちらでもない
3:そう思わない
$ mseldsp x=10 y=3 i=sel.txt o=rsl1.txt
# 利用者が一行目を選んだとする。
$ mose rsl1.txt
1:そう思う


以下、画面イメージ
+--------------------------------------
|
|
|           1:そう思う
|           2:どちらでもない
|           3:そう思わない
```

```
|
```

例 2: 引数で与える例

例 1 と同様で、選択肢の文字列を seldata=で与える。

```
$ mseldsp x=10 y=3 seldata=1:そう思う,2:どちらでもない,3:そう思わない o=rsl2.txt
# 利用者が二行目を選んだとする。
$ mose rsl2.txt
2:どちらでもない


以下、画面イメージ
+-------------------------------------
|
|
|           1:そう思う
|           2:どちらでもない
|           3:そう思わない
|
```

例 3: 終了ステータスを判定する例

例 1 と同じパラメータで実行し、終了ステータスを判定して異なる動作をするスクリプトの例。

```
$ more scp.sh
rm -f rsl3.csv
clear
mseldsp x=10 y=3 seldata=aaaa,bbbb,cccc o=rsl3.csv
if [ $? = 0 ] ; then
  clear ; echo "end by enter key"
else
  clear ; echo "end by escape key"
fi

# aaaa を選択後 enter キーを入力した場合の結果
$ bash scp.sh
end by enter key
$ more rsl3.csv
aaaa

# bbbb を選択後 escape キーを入力した場合の結果
$ bash scp.sh
end by escape key
$ more rsl3.csv
bbbb
```

関連コマンド

minput :入力画面を表示する。

mminput : 複数入力枠による入力画面を表示する。

mdsp : 画面の指定位置に文字列を表示する。

mmseldsp : 画面に複数選択入力窓を表示する。

4.54 mselnum 数値範囲による行選択

f=で指定した項目の値が、c=で指定した数値範囲にマッチする行を選択する。以下に示すように多様な選択条件を指定することが可能である。このコマンドで指定できないより複雑な条件 (文字列マッチとの組み合せなど) を設定するのであれば msel コマンドを利用すればよい。OR 条件、AND 条件およびキー指定についての詳細は mselstr コマンドを参照されたい。

- 開区間、閉区間、半開区間、無限区間の制定が可能である。
- c=に複数の範囲を指定すれば、いずれかの範囲にマッチすれば選択される (OR 条件)。
- f=に複数項目を指定すれば、いずれかの項目の値がマッチすれば選択される (OR 条件)。
- f=の OR 条件を AND 条件に変更することも可能 (-F オプション)。
- k=を指定することでキー単位で選択することが可能。
- キー単位選択の場合、複数レコードの AND 条件を指定可能 (-R オプション)。

典型的な例を表 4.13〜4.16 に示す。

表 4.13 入力データ

key	val
a	1
a	-3
b	3
b	6

表 4.14 val 項目が 1 以上 3 以下の行を選択

```
mselnum f=val c='[1,3]'
```

key	val
a	1
b	3

表 4.15 val 項目が 1 以上 3 未満の行を選択

```
mselnum f=val c='[1,3)'
```

key	val
a	1

表 4.16 5 以上の行を選択

```
mselnum f=val c='[5,)'
```

key	val
b	6

表 4.17 1 以下もしくは 5 以上の行を選択

```
mselnum f=val c='(,1],[5,)'
```

key	val
a	1
a	-3
b	6

書式

mselnum f= c= [k=] [u=] [-F] [-r] [-R] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

- `f=`　検索対象となる項目名リスト (複数項目指定可) を指定する。
- `c=`　`f=`パラメータで指定した項目の値が、ここで指定した文字列リスト (複数範囲指定可) の 1 つにマッチすれば選択される。
- `k=`　撰択する単位となるキー項目 (複数項目指定可) を指定する。
- `o=`　指定の条件に一致する行を出力するファイル名を指定する。
- `u=`　指定の条件に一致しない行を出力するファイル名を指定する。
- `-F`　`f=`パラメータで複数項目を指定した場合、その全ての値がマッチする行を撰択する。
- `-r`　条件反転
  選択ではなく削除する。
- `-R`　`k=`パラメータを指定した場合、その全ての行がマッチすれば行を撰択する。

利用例

例 1: **基本例**

`val` 項目が 2 以上 5 以下の行、すなわち `id=2,5` の行を選択する。

```
$ more dat1.csv
id,val
1,5.1
2,5
3,-2.0
4,
5,2.0
$ mselnum f=val c='[2,5]' i=dat1.csv o=rsl1.csv
#END# kgselnum c=[2,5] f=val i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val
2,5
5,2.0
```

例 2: **片側範囲**

`val` 項目が 2 以上の行、すなわち `id=1,2,5` の行を選択する。

```
$ mselnum f=val c='[2,]' i=dat1.csv o=rsl2.csv
#END# kgselnum c=[2,] f=val i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val
1,5.1
2,5
5,2.0
```

関連コマンド

msel：複雑な条件指定による選択を行う場合に利用する。

mselstr：文字列マッチによる選択。

4.55 mselrand ランダムに行を選択

c=パラメータもしくは p=パラメータで指定した行数をランダムに選択する (非復元抽出)。k=を指定した場合、同一キーの行から指定の行数をランダムに選択し、また同時に-B オプションを指定すると、キー単位で選択する。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

書式

mselrand c=|p= [k=] [S=] [u=] [-B] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

c= 各キーの値毎に選択する行数を指定する。
p=パラメータを指定しない場合の指定は必ず指定する必要がある。
p= 各キーの値毎に選択する割合をパーセントで指定する。
c=パラメータを指定しない場合の指定は必ず指定する必要がある。
k= 指定する項目（複数項目指定可）の値が同じ行から、一定行数ランダムに選択する。
S= 同じ乱数の種は同じシーケンスの乱数をふる。
指定しない場合は、時刻に応じた異なる乱数の種が利用される。
指定可能な乱数の種 (-2147483648〜2147483647)
u= 指定の条件に一致しない行を出力するファイル名を指定する。
-B キー単位選択

利用例

例 1: **基本例**

一人の顧客につきランダムに 1 行を選択する。

```
$ more dat1.csv
顧客,日付,金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
$ mselrand k=顧客 c=1 S=1 i=dat1.csv o=rsl1.csv
#END# kgselrand S=1 c=1 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客 %0,日付,金額
A,20081201,10
B,20081002,40
```

例 2: **ランダムに一定割合の行を選択**

一人の顧客につきランダムに 50% の行を選択する。また、それ以外の不一致データは oth2.csv というファイルに出力する。

```
$ mselrand k=顧客 p=50 S=1 u=oth2.csv i=dat1.csv o=rsl2.csv
#END# kgselrand S=1 i=dat1.csv k=顧客 o=rsl2.csv p=50 u=oth2.csv
$ more rsl2.csv
```

```
顧客 %0, 日付, 金額
A,20081201,10
B,20081002,40
$ more oth2.csv
顧客 %0, 日付, 金額
A,20081207,20
A,20081213,30
B,20081209,50
```

例 3: キー単位の選択

以下の例は、顧客 A,B,C,D の 4 人からランダムに 2 人を選ぶ。顧客 D が選ばれると、顧客 D の行は全て出力される。

```
$ more dat2.csv
顧客, 日付, 金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
C,20081210,60
D,20081201,70
D,20081205,80
D,20081209,90
$ mselrand k=顧客 c=2 S=1 -B i=dat2.csv o=rsl3.csv
#END# kgselrand -B S=1 c=2 i=dat2.csv k=顧客 o=rsl3.csv
$ more rsl3.csv
顧客 %0, 日付, 金額
C,20081210,60
D,20081201,70
D,20081205,80
D,20081209,90
```

関連コマンド

msel：正規乱数も使える。

mrand：ランダム選択でなく、乱数項目を付け加える。

4.56 mselstr 文字列による行選択

f=で指定した項目の値が、v=で指定した文字列に一致すれば、その行を選択する。

典型例を表 4.18〜4.20 に示す。表 4.19 では key に関係なく val が"y"である行を選択する。表 4.20 では、val が"x"の行を含んでいる同一キーの行全て、すなわち key 項目が"a"である行全てを選択する。すなわち key 項目が"b"の行はいずれも"x"を含んでいないので選択されない。

表 4.18 入力データ

key	val
a	x
a	y
b	y
b	z

表 4.19 f=val v=y

key	val
a	y
b	y

表 4.20 k=key f=val v=x

key	val
a	x
a	y

また、以下に示すように多様な選択条件を指定することも可能である。このコマンドで指定できない複雑な条件 (例えば正規表現など) を設定するのであれば msel コマンドを利用すればよい。

- v=に複数の文字列を指定すれば、いずれかの文字列にマッチすれば選択される。
- f=に複数項目を指定すれば、いずれかの項目の値がマッチすれば選択される。
- 複数項目のマッチ条件を AND 条件とすることも可能 (-F オプション)。
- 完全一致だけでなく、先頭一致、末尾一致、部分一致の指定も可能 (-head,-tail,-sub オプション)。
- k=を指定することでキー単位で選択することが可能。
- キー単位選択の場合、複数レコードの AND 条件を指定可能 (-R オプション)。

いま同じキーのデータとして 2 項目 2 行からなるデータ (表 4.21) に対して、

```
mselstr k=key f=fld1,fld2 v=s1,s2
```

を実行した場合、オプション-R,-F の指定の有無によるマッチ条件を表 4.22 に示す。

表 4.21 入力データ

key	fld1	fld2
k	v_{a1}	v_{a2}
k	v_{b1}	v_{b2}

表 4.22 表 4.21 で示されるデータに、mselstr k=key f=fld1,fld2 v=v1,v2 を実行した時の、-R,-F オプションの指定の有無によるマッチ条件の違い。条件にマッチすれば全行 (2 行) 出力され、アンマッチなら 1 行も出力されない。

-F	-R	マッチ条件
		((v_{a1} == s1 or v_{a1} == s2) or (v_{a2} == s1 or v_{a2} == s2)) or ((v_{b1} == s1 or v_{b1} == s2) or (v_{b2} == s1 or v_{b2} == s2))
-F		((v_{a1} == s1 or v_{a1} == s2) and (v_{a2} == s1 or v_{a2} == s2)) or ((v_{b1} == s1 or v_{b1} == s2) and (v_{b2} == s1 or v_{b2} == s2))
	-R	((v_{a1} == s1 or v_{a1} == s2) or (v_{a2} == s1 or v_{a2} == s2)) and ((v_{b1} == s1 or v_{b1} == s2) or (v_{b2} == s1 or v_{b2} == s2))
-F	-R	((v_{a1} == s1 or v_{a1} == s2) and (v_{a2} == s1 or v_{a2} == s2)) and ((v_{b1} == s1 or v_{b1} == s2) and (v_{b2} == s1 or v_{b2} == s2))

書式

mselstr f= v= [k=] [u=] [-F] [-r] [-R] [-sub] [-head] [-tail] [-W] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl]

[--version]

パラメータ

f=	検索対象となる項目名リスト (複数項目指定可) を指定する。
v=	f=パラメータで指定した項目の値が、ここで指定した文字列リスト (複数項目指定可) の 1 つにマッチすれば選択される。
k=	選択する単位となるキー項目 (複数項目指定可) を指定する。
o=	指定の条件に一致する行を出力するファイル名を指定する。
u=	指定の条件に一致しない行を出力するファイル名を指定する。
-F	f= パラメータで複数項目を指定した場合、その全ての値がマッチする行を撰択する。
-r	条件反転 選択ではなく削除する。
-R	k= パラメータを指定した場合、その全ての行がマッチすれば行を撰択する。
-sub	検索を完全一致ではなく部分文字列マッチで比較する。 すなわち、f=パラメータで指定した項目の値に、 v=パラメータで指定の文字列が部分文字列として含まれていればその行を撰択する。
-head	先頭文字列マッチオプション
-tail	末尾文字列マッチオプション
-W	-sub,-head,-tail オプションが指定されているときにワイド文字として部分文字列マッチをおこなう。

利用例

例 1: **基本例**

「商品」項目の値が apple、orange に完全一致する行を選択し、rsl1.csv に出力する。u=oth1.csv を指定すれば、それ以外の行は oth1.csv に出力する。pineapplejuice は完全一致ではないので、oth1.csv に出力される。

```
$ more dat1.csv
商品,金額
apple,100
milk,350
orange,100
pineapplejuice,500
wine,1000
$ mselstr f=商品 v=apple,orange u=oth1.csv i=dat1.csv o=rsl1.csv
#END# kgselstr f=商品 i=dat1.csv o=rsl1.csv u=oth1.csv v=apple,orange
$ more rsl1.csv
商品,金額
apple,100
orange,100
$ more oth1.csv
商品,金額
milk,350
pineapplejuice,500
wine,1000
```

例 2: **行の削除**

-r オプションを指定することで、例 1 とは逆に、商品項目の値が apple、orange に完全一致する行を削除し、rsl2.csv に出力する。

```
$ mselstr f=商品  v=apple,orange -r i=dat1.csv o=rsl2.csv
#END# kgselstr -r f=商品 i=dat1.csv o=rsl2.csv v=apple,orange
$ more rsl2.csv
商品,金額
```

```
milk,350
pineapplejuice,500
wine,1000
```

例 3: **キー単位での選択**

orange を購入したことのある顧客を選択する k=**顧客**を指定することで、orange を購入したことのある顧客の他に購入した商品の行を含めて選択する。それ以外の行は oth2.csv に出力する。

```
$ more dat2.csv
顧客,商品,金額
A,apple,100
A,milk,350
B,orange,100
B,orange,100
B,pineapple,500
B,wine,1000
C,apple,100
C,orange,100
$ mselstr k=顧客 f=商品 v=orange u=oth2.csv i=dat2.csv o=rsl3.csv
#END# kgselstr f=商品 i=dat2.csv k=顧客 o=rsl3.csv u=oth2.csv v=orange
$ more rsl3.csv
顧客 %0,商品,金額
B,orange,100
B,orange,100
B,pineapple,500
B,wine,1000
C,apple,100
C,orange,100
$ more oth2.csv
顧客 %0,商品,金額
A,apple,100
A,milk,350
```

例 4: **部分一致**

「商品」項目の値が apple に部分一致するの行を選択し、rsl4.csv に出力する。部分一致であるため pine(apple)juice も rsl4.csv に出力される。

```
$ mselstr f=商品 v=apple -sub i=dat1.csv o=rsl4.csv
#END# kgselstr -sub f=商品 i=dat1.csv o=rsl4.csv v=apple
$ more rsl4.csv
商品,金額
apple,100
pineapplejuice,500
```

例 5: **ワイド文字の部分一致**

「商品」項目の値がワイド文字の「柿」、「桃」、「葡萄」の行を選択 (部分一致) 選択項目にワイド文字が使用されている場合にバイト単位のマッチングを使用すると、マルチバイト文字をまたいだ文字列にマッチングする可能性がある。その為、ワイド文字が選択項目に含まれる場合は-W オプションを使用して、ワイド文字を使用していることを意図的に示す必要がある。

```
$ more dat3.csv
商品,金額
果物:柿,100
果物:桃,250
果物:葡萄,300
```

```
果物:梨,450
果物:苺,500
$ mselstr f=商品 v=柿,桃,葡萄 -sub -W i=dat3.csv o=rsl5.csv
#END# kgselstr -W -sub f=商品 i=dat3.csv o=rsl5.csv v=柿,桃,葡萄
$ more rsl5.csv
商品,金額
果物:柿,100
果物:桃,250
果物:葡萄,300
```

例 6: **商品の購入日と前回の購入日が** 2013 **年の商品データを選択**

-F オプションを指定することで、同じ商品を 2013 年内に購入したことのある (購入日と前回購入日両方が 2013 年) 商品行を選択し、rsl6.csv に出力する。それ以外の行は oth3.csv に出力する。

```
$ more dat4.csv
顧客,商品,金額,性別,購入日,前回購入日
A,apple,100,1,2013/01/04,2013/01/01
A,milk,350,1,2013/04/04,2011/05/06
B,orange,100,2,2012/11/11,2011/12/12
B,orange,100,2,2013/05/30,2012/11/11
B,pineapple,500,2,2013/04/15,2013/04/01
B,wine,1000,2,2012/12/24,2011/12/24
C,apple,100,2,2013/02/14,NULL
C,orange,100,2,2013/02/14,2013/01/31
D,orange,100,2,2011/10/28,NULL
$ mselstr f=購入日,前回購入日 -F -sub v=2013 u=oth3.csv i=dat4.csv o=rsl6.csv
#END# kgselstr -F -sub f=購入日,前回購入日 i=dat4.csv o=rsl6.csv u=oth3.csv v=2013
$ more rsl6.csv
顧客,商品,金額,性別,購入日,前回購入日
A,apple,100,1,2013/01/04,2013/01/01
B,pineapple,500,2,2013/04/15,2013/04/01
C,orange,100,2,2013/02/14,2013/01/31
$ more oth3.csv
顧客,商品,金額,性別,購入日,前回購入日
A,milk,350,1,2013/04/04,2011/05/06
B,orange,100,2,2012/11/11,2011/12/12
B,orange,100,2,2013/05/30,2012/11/11
B,wine,1000,2,2012/12/24,2011/12/24
C,apple,100,2,2013/02/14,NULL
D,orange,100,2,2011/10/28,NULL
```

例 7: **商品の購入日と前回の購入日が** 2013 **年の顧客データの抽出**

k=**顧客**を指定することで、同じ商品を 2013 年内に購入したことのある顧客の他に購入した商品の行を含めて選択する。それ以外の行は oth4.csv に出力する。

```
$ mselstr k=顧客 f=購入日,前回購入日 -F -sub v=2013 u=oth4.csv i=dat4.csv o=rsl7.csv
#END# kgselstr -F -sub f=購入日,前回購入日 i=dat4.csv k=顧客 o=rsl7.csv u=oth4.csv v=2013
$ more rsl7.csv
顧客%0,商品,金額,性別,購入日,前回購入日
A,apple,100,1,2013/01/04,2013/01/01
A,milk,350,1,2013/04/04,2011/05/06
B,orange,100,2,2012/11/11,2011/12/12
B,orange,100,2,2013/05/30,2012/11/11
B,pineapple,500,2,2013/04/15,2013/04/01
B,wine,1000,2,2012/12/24,2011/12/24
C,apple,100,2,2013/02/14,NULL
C,orange,100,2,2013/02/14,2013/01/31
$ more oth4.csv
顧客%0,商品,金額,性別,購入日,前回購入日
```

```
D,orange,100,2,2011/10/28,NULL
```

例 8: 2013 年度の新規顧客情報の抽出

-R オプションを指定することで、購入日、前回購入日両方が 2013 年,NULL(前回購入なし) の顧客情報を抽出する。つまり 2013 年の新規顧客データを選択し、rsl8.csv に出力する。それ以外の行は oth5.csv に出力する。

```
$ mselstr k=顧客 f=購入日,前回購入日 -F -R -sub v=2013,NULL u=oth5.csv i=dat4.csv o=rsl8.csv
#END# kgselstr -F -R -sub f=購入日,前回購入日 i=dat4.csv k=顧客 o=rsl8.csv u=oth5.csv v=2013,NULL
$ more rsl8.csv
顧客 %0,商品,金額,性別,購入日,前回購入日
C,apple,100,2,2013/02/14,NULL
C,orange,100,2,2013/02/14,2013/01/31
$ more oth5.csv
顧客 %0,商品,金額,性別,購入日,前回購入日
A,apple,100,1,2013/01/04,2013/01/01
A,milk,350,1,2013/04/04,2011/05/06
B,orange,100,2,2012/11/11,2011/12/12
B,orange,100,2,2013/05/30,2012/11/11
B,pineapple,500,2,2013/04/15,2013/04/01
B,wine,1000,2,2012/12/24,2011/12/24
D,orange,100,2,2011/10/28,NULL
```

関連コマンド

msel： より複雑な条件で行選択を行う。

mcommon： 選択対象となる文字列の数が多いときは、参照ファイルを用意することで mcommon コマンドが使える。

4.57　msep レコードの分割

d=パラメータで指定したファイル名のデータに各レコードを出力する。指定するファイル名に項目名を埋め込むことができるので、結果としてレコード分割することになる。埋め込むファイル名は**${項目名}**によって指定する。例えば、d=./out/${date}.csv と指定すれば、カレントディレクトリの下の out ディレクトリの下に、date 項目の値別にファイルが作成されることになる。

内部的には、埋め込んだ項目の値をキーとして認識し、並べ替えが行われた後レコードが分割される。

書式

msep d= [-p] [f=] [i=] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

d=　異なるデータファイルに分割する項目名を指定する。
　　ここで指定した文字列をファイル名として各レコードが追記されていく。
　　項目名は**${項目名}**によって埋め込む。

-p　d= パラメータで指定したディレクトリ名が存在しなければ作成する。

利用例

例 1: **基本例**

dat という名前のディレクトリを作成し、そのディレクトリに日付項目値 date ごとに異なるファイルに出力する。

```
$ more dat1.csv
item,date,quantity,price
A,20081201,1,10
B,20081201,4,40
A,20081202,2,20
A,20081203,3,30
B,20081203,5,50
$ msep d='./dat/${date}.csv' -p i=dat1.csv
#END# kgsep -p d=./dat/${date}.csv i=dat1.csv
$ ls ./dat
20081201.csv
20081202.csv
20081203.csv
$ more ./dat/20081201.csv
item,date%0,quantity,price
A,20081201,1,10
B,20081201,4,40
$ more ./dat/20081202.csv
item,date%0,quantity,price
A,20081202,2,20
$ more ./dat/20081203.csv
item,date%0,quantity,price
A,20081203,3,30
B,20081203,5,50
```

関連コマンド

msep2：`msep` と同じような機能だが、ファイル名は連番で出力し、キー項目との対応表を別途ファイルに出力する。
mcat：`msep` で分割したファイルをこのコマンドで併合すると元に戻る。

4.58　msep2 連番-項目値表の出力を伴った行の分割

k=で指定された項目の値別にデータを分割 (separation) する。分割後のデータは自動的に採番される連番ファイル名に保存され、k=で指定された項目の値と分割後のファイル名との対応表が出力される。

書式

msep2 k= O= a= [-p] [i=] [o=] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k=	分割単位となる項目名リスト
O=	ここで指定されたディレクトに連番ファイル (0 から始まる連番) を作成する。
o=	k=で指定した項目値と連番ファイル名との対応表を CSV で出力する。 指定しなければ標準出力に出力される。
a=	o=に出力されるファイル名の項目名を指定する。
-p	O=で指定したディレクトリがなければ強制作成する。

利用例

例 1: 基本例

item 項目別にデータを分割する。出力ファイル名は 0 から始まる連番であり、どの番号がどのキーに対応しているかが table.csv に出力される。

```
$ more dat1.csv
item,no
A,1
A,1
A,2
B,1
B,2
$ msep2 k=item O=./output a=fileName o=table.csv i=dat1.csv
#END# kgsep2 O=./output a=fileName i=dat1.csv k=item o=table.csv
$ ls ./output
0
1
$ more table.csv
item%0,fileName
A,./output/0
B,./output/1
$ more output/0
item%0,no
A,1
A,1
A,2
$ more output/1
item%0,no
B,1
B,2
```

例 2: **複数キー項目**

複数のキー項目 item,no を設定しても同様に各ファイル名は連番で作成される。table.csv に複数のキー項目と番号の対応表が出力されている。

```
$ more dat1.csv
item,no
A,1
A,1
A,2
B,1
B,2
$ msep2 k=item,no O=./output2 a=fileName o=table.csv i=dat1.csv
#END# kgsep2 O=./output2 a=fileName i=dat1.csv k=item,no o=table.csv
$ ls ./output2
0
1
2
3
$ more table.csv
item%0,no%1,fileName
A,1,./output2/0
A,2,./output2/1
B,1,./output2/2
B,2,./output2/3
$ more output/0
item%0,no
A,1
A,1
A,2
```

関連コマンド

msep : ファイル名に項目名を組み込みたいときはこのコマンドを使う。

4.59　msetstr 文字列項目の追加

指定した文字列を項目として全行に追加する。複数項目の追加も可能。

書式

```
msetstr v= a=  [i=] [o=] [-assert_diffSize] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

v=　追加する文字列リスト。
　　値を何も指定しないと NULL 値が追加される。
a=　追加する項目名。
　　v=で指定した文字列の個数と同数の項目名を指定しなければならない。

利用例

例 1: 基本例

日付計算で必要となる基準日を (2007 年 01 月 01 日と定義した場合) すべての行に「20070101」という文字列を追加し「基準日」という項目名で出力する。

```
$ more dat1.csv
顧客,日付
A,20081202
A,20081204
B,20081203
$ msetstr v=20070101 a=基準日 i=dat1.csv o=rsl1.csv
#END# kgsetstr a=基準日 i=dat1.csv o=rsl1.csv v=20070101
$ more rsl1.csv
顧客,日付,基準日
A,20081202,20070101
A,20081204,20070101
B,20081203,20070101
```

例 2: 複数項目を追加

```
$ msetstr v=20070101,20070201 a=基準日 1,基準日 2 i=dat1.csv o=rsl2.csv
#END# kgsetstr a=基準日 1,基準日 2 i=dat1.csv o=rsl2.csv v=20070101,20070201
$ more rsl2.csv
顧客,日付,基準日 1,基準日 2
A,20081202,20070101,20070201
A,20081204,20070101,20070201
B,20081203,20070101,20070201
```

例 3: null 値項目追加

```
$ msetstr v= a=追加項目 i=dat1.csv o=rsl3.csv
#END# kgsetstr a=追加項目 i=dat1.csv o=rsl3.csv v=
$ more rsl3.csv
顧客,日付,追加項目
A,20081202,
A,20081204,
```

```
B,20081203,
```

関連コマンド

mcal：if 関数を使えば、行ごとに条件を判定して異なる固定文字列を追加できる。

4.60　mshare 構成比の計算

f=パラメータで指定した項目の構成比を計算し、新しい項目として追加する。

書式

mshare f= [k=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=　ここで指定された項目（複数項目指定可）の値のシェアが計算される。
:(コロン）で新項目名を指定する必要がある。例）f=数量:数量シェア
k=　シェア計算の単位となる項目名リスト（複数項目指定可）を指定する。
省略すると全行同じキーの値として処理される。

利用例

例 1: 基本例

「顧客」項目を単位に「数量」と「金額」項目のシェアを計算し、「数量シェア」と「金額シェア」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ mshare k=顧客 f=数量:数量シェア,金額:金額シェア i=dat1.csv o=rsl1.csv
#END# kgshare f=数量:数量シェア,金額:金額シェア i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客,%0,数量,金額,数量シェア,金額シェア
A,1,10,0.3333333333,0.3333333333
A,2,20,0.6666666667,0.6666666667
B,1,15,0.2,0.3333333333
B,3,10,0.6,0.2222222222
B,1,20,0.2,0.4444444444
```

関連コマンド

4.61 mshuffle レコード分割

f=で指定した項目の hash 値に従って指定した数のファイルに入力ファイルを分割する。分割数 (hash サイズ) を n とすると、f=で指定した「項目の値」v の hash 値は n の剰余 (v mod n) として計算される。「項目の値」は、データを文字列として考え、バイト単位の文字コードの合計値として計算される。f=を指定しなかった場合は、「項目の値」として行番号が用いられる。そして、各行は、得られた hash 値を名前に持ったファイルに出力される。以上の方法により、同じ項目データを持つ行は全て同一のファイルに出力されることが保証される。

また、v=で重みを指定することで、分割される各ファイルに複数の hash 値を割り当てることもできる。n=3,v=2,1,3 と指定すれば、hash サイズを重みの合計 $2+1+3=6$ とし、2 つの hash 値 (0,1) を分割ファイル 0 に、1 つの hash 値 (2) を分割ファイル 1 に、そして 3 つの hash 値 (3,4,5) を分割ファイル 2 に出力する。重みは hash 値の割当数の重みであり、出力行数の重みではないことに注意する。

書式

```
mshuffle n=|v= d= [f=] [i=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

- `d=` 出力するファイル名の接頭辞を指定する
  ここで指定した値＋連番 (hash 値) が実際に出力されるファイル名になる
- `f=` 分割単位となるキーを指定する
  ここで指定した項目値が等しいものは同じファイルに出力される
- `n=` 分割するファイル数を指定する
- `v=` 分割するファイルごとにデータ量の重みを指定する

利用例

例 1: **基本例**

指定した項目の値 (顧客) が同じであれば同一のファイルに出力にされるように 2 つのファイルに分割する

```
$ more dat2.csv
顧客,日付,金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
C,20081003,60
C,20081219,20
$ mshuffle f=顧客 d=./dat/d n=2 i=dat2.csv
#END# kgshuffle d=./dat/d f=顧客 i=dat2.csv n=2
$ ls ./dat
d_0
d_1
$ more ./dat/d_0
顧客,日付,金額
B,20081002,40
B,20081209,50
$ more ./dat/d_1
顧客,日付,金額
A,20081201,10
A,20081207,20
```

```
A,20081213,30
C,20081003,60
C,20081219,20
```

例 2: f=**を指定しない例**

f=を指定せず 2 つのファイルに分割する。行番号の hash 値を用いるので、2 つのファイルの行数はほぼ等しくなる。

```
$ more dat2.csv
顧客,日付,金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
C,20081003,60
C,20081219,20
$ mshuffle d=./dat/d n=2 i=dat2.csv
#END# kgshuffle d=./dat/d i=dat2.csv n=2
$ ls ./dat
d_0
d_1
$ more ./dat/d_0
顧客,日付,金額
A,20081207,20
B,20081002,40
C,20081003,60
$ more ./dat/d_1
顧客,日付,金額
A,20081201,10
A,20081213,30
B,20081209,50
C,20081219,20
```

例 3: v=,f=**の指定**

v=2,1 を指定することで、ファイル 0(d_0) には 2 つの hash 値を割り当て、ファイル 1(d_1) には 1 つの hash 値を割り当てて分割する。

```
$ more dat2.csv
顧客,日付,金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
C,20081003,60
C,20081219,20
$ mshuffle f=顧客 d=./dat/d v=2,1 i=dat2.csv
#END# kgshuffle d=./dat/d f=顧客 i=dat2.csv v=2,1
$ ls ./dat
d_0
d_1
$ more ./dat/d_0
顧客,日付,金額
B,20081002,40
B,20081209,50
C,20081003,60
C,20081219,20
$ more ./dat/d_1
顧客,日付,金額
```

```
A,20081201,10
A,20081207,20
A,20081213,30
```

例 4: v=の指定

例 3 を f=の指定なしで実行する。行番号の hash 値を用いるので、この場合は出力行数の比と重みの比がほぼ等しくなる。

```
$ more dat2.csv
顧客,日付,金額
A,20081201,10
A,20081207,20
A,20081213,30
B,20081002,40
B,20081209,50
C,20081003,60
C,20081219,20
$ mshuffle d=./dat/d v=2,1 i=dat2.csv
#END# kgshuffle d=./dat/d i=dat2.csv v=2,1
$ ls ./dat
d_0
d_1
$ more ./dat/d_0
顧客,日付,金額
A,20081201,10
A,20081213,30
B,20081002,40
C,20081003,60
C,20081219,20
$ more ./dat/d_1
顧客,日付,金額
A,20081207,20
B,20081209,50
```

関連コマンド

msep：項目値によるレコードの分割

4.62　msim 二変数間の類似度の計算

f=パラメータで指定した項目の二変数間の類似度 (距離) を c=パラメータで指定した類似度 (距離) 関数で計算し類似度行列として出力する。

書式

msim c= f= [a=] [k=] [n=] [-d] [i=] [o=] [bufcount=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

k=　ここで指定された項目 (複数項目指定可) を単位として求める。
f=　ここで指定された項目全ての二項目間の類似度を求める。
c=　類似度 (距離) 名リスト (複数項目指定可)
　次項に示した類似度 (距離) 名を指定する。
　項目名は以下のように，(:) コロンに続けて指定して変更可能。
　コロンに続く名称を省略した場合は類似度 (距離) 関数名がそのまま項目名として利用される。
　例) msim f=x,y,z c=pearson:**ピアソン積率相関係数**,euclid:**ユークリッド距離**,cosine:**コサイン**
　類似度=covar|ucovar|pearson|spearman|kendall|euclid|cosine|
　　cityblock|hamming|chi|phi|jaccard|supportr|lift|confMax|
　　confMin|yuleQ|yuleY|kappa|oddsRatio|convMax|convMin
a=　2 変数の名称を示す項目名を指定する。カンマで区切って 2 つ指定する。
　省略すると fld1,fld2 が使われる。
-d　対角行列、上三角行列を出力する。
　-d オプションが指定されないと類似度行列の下三角行列のみ出力されるが、
　-d オプションを指定することにより対角行列及び上三角行列も出力される。

類似度 (距離) の定義

実数ベクトル

サイズが同じ 2 つの実数ベクトル $\mathbf{x} = (x_1, x_2, \cdots, x_n), \mathbf{y} = (x_1, x_2, \cdots, x_n)$ に関する類似度 (もしくは距離) の定義を表 4.23 に示す。

0-1 ベクトル

値として 0 もしくは 1 をとる 2 つの 0-1 ベクトル $\mathbf{a} = (a_1, a_2, \cdots, a_n), \mathbf{b} = (b_1, b_2, \cdots, b_n)$ に関する類似度の定義を表 4.25 に示す。表の中で使われている記号 f_{jk} は、a_i, b_i がとる値 (0,1) の組み合せ別の件数で、表 4.24 に示されている。

また $P(\cdot)$ の意味は以下に示すとおりである。

利用例

例 1: 基本例

x、y、z 項目の 2 項目間の組み合わせについてピアソンの積率相関係数とコサインを計算する。

```
$ more dat1.csv
x,y,z
14,0.17,-14
```

表 4.23 実数ベクトルの類似度一覧

パラメータ値	内容	距離/類似度	定義式	範囲
covar	共分散	類似度	$\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$	$-\infty \sim \infty$
ucovar	不偏共分散	類似度	$\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$	$-\infty \sim \infty$
pearson	ピアソンの積率相関係数	類似度	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y})^2}}$	$-1.0 \sim 1.0$
spearman	スピアマンの順位相関係数	類似度	$\mathbf{x},\mathbf{y}$ を順位に変換しての積率相関係数	$-1.0 \sim 1.0$
kendall	ケンドールの順位相関係数	類似度	$\frac{c-d}{\frac{1}{2}n(n-1)}$ 注 1,2)	$-1.0 \sim 1.0$
euclid	ユークリッド距離 (数値)	距離	$\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$	$0 \sim \infty$
cosine	コサイン	類似度	$\frac{\mathbf{x}\cdot\mathbf{y}}{\lvert\mathbf{x}\rvert\lvert\mathbf{y}\rvert}=\frac{\sum_{i=1}^{n}x_i\ y_i}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}}$	$-1.0 \sim 1.0$
cityblock	都市ブロック距離	距離	$\sum_{i=1}^{n} \mid x_i-y_i \mid$	$-\infty \sim \infty$
hamming	ハミング距離	距離	$\mid \{i \mid x_i \neq y_i, i=1,2,\cdots,n\} \mid$	$0 \sim n$

注 1) $c=|\{(i,j)|(x_i>x_j \text{ and } y_i>y_j) \text{ or } (x_i<x_j \text{ and } y_i<y_j), i>j, i=1,2,\cdots,n, j=1,2,\cdots,n\}|$

注 2) $d=|\{(i,j)|(x_i>x_j \text{ and } y_i<y_j) \text{ or } (x_i<x_j \text{ and } y_i>y_j), i>j, i=1,2,\cdots,n, j=1,2,\cdots,n\}|$

表 4.24 2 変数の値の組みわせによる 2×2 分割表

	$b_i=1$	$b_i=0$	計
$a_i=1$	f_{11}	f_{10}	$f_{1.}$
$a_i=0$	f_{01}	f_{00}	$f_{0.}$
計	$f_{.1}$	$f_{.0}$	$f_{..}$

$P(a)=f_{1.}/f_{..}$
$P(b)=f_{.1}/f_{..}$
$P(\bar{a})=f_{0.}/f_{..}$
$P(a,b)=f_{11}/f_{..}$
$P(a\|b)=f_{11}/f_{.1}$

```
11,0.2,-1
32,0.15,-2
13,0.33,-2
$ msim c=pearson,cosine f=x,y,z i=dat1.csv o=rsl1.csv
#END# kgsim c=pearson,cosine f=x,y,z i=dat1.csv o=rsl1.csv
$ more rsl1.csv
fld1,fld2,pearson,cosine
x,y,-0.5088704666,0.7860308044
x,z,0.1963041929,-0.5338153343
y,z,0.3311001423,-0.5524409416
```

例 2: **対角行列、上三角行列を出力**

x、y、z 項目の 2 項目間の組み合わせについてピアソンの積率相関係数とコサインを計算する。(d オプションあり)

```
$ msim c=pearson,cosine f=x,y,z -d i=dat1.csv o=rsl2.csv
#END# kgsim -d c=pearson,cosine f=x,y,z i=dat1.csv o=rsl2.csv
```

表 4.25　0-1 ベクトルの類似度一覧

パラメータ値	内容	距離/類似度	定義式	範囲
chi	カイ 2 乗値	類似度	$\sum_{i=0}^{1}\sum_{j=0}^{1}\frac{f_{ij}-e_{ij}}{e_{ij}}$ 注 1)	$0 \sim \infty$
phi	ファイ係数	類似度	$\frac{f_{11}f_{00}-f_{10}f_{01}}{\sqrt{f_{1.}f_{0.}f_{.1}f_{.0}}}$	$-1.0 \sim 1.0$
jaccard	ジャックカード係数	類似度	$\frac{P(a,b)}{P(a)+P(b)-P(a,b)}$	$0.0 \sim 1.0$
support	支持度	類似度	$P(a,b)$	$0.0 \sim 1.0$
lift	リフト値	類似度	$\frac{P(a,b)}{P(a)P(b)}$	$0 \sim \infty$
confMax	最大確信度	類似度	$\max(P(a\|b), P(b\|a))$	$0.0 \sim 1.0$
confMin	最小確信度	類似度	$\min((P(a\|b), P(b\|a))$	$0.0 \sim 1.0$
yuleQ	yule の連関係数 (Q)	類似度	$\frac{\alpha-1}{\alpha+1}$ 注 2)	$-1.0 \sim 1.0$
yuleY	yule の連関係数 (Y)	類似度	$\frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$ 注 2)	$-1.0 \sim 1.0$
kappa	kappa	類似度	$\frac{P(a,b)+P(\bar{a},\bar{b})-P(a)P(b)-P(\bar{a})P(\bar{b})}{1-P(a)P(b)-P(\bar{a})P(\bar{b})}$	$-1.0 \sim 1.0$
oddsRatio	oddsRatio	類似度	$\frac{P(a,b)P(\bar{a},\bar{b})}{P(a,\bar{b})P(\bar{a},b)}$	$0 \sim \infty$
convMax	最大 conviction	類似度	$\max(\frac{P(a)P(\bar{b})}{P(a,\bar{b})}, \frac{P(\bar{a})P(b)}{P(\bar{a},b)})$	$0.5 \sim \infty$
convMin	最小 conviction	類似度	$\min(\frac{P(a)P(\bar{b})}{P(a,\bar{b})}, \frac{P(\bar{a})P(b)}{P(\bar{a},b)})$	$0.5 \sim \infty$

注 1) $e_{ij} = \frac{f_{i.}f_{.j}}{f_{..}}$ 注 2) $\alpha = \frac{f_{11}f_{00}}{f_{10}f_{01}}$

```
$ more rsl2.csv
fld1,fld2,pearson,cosine
x,x,1,1
x,y,-0.5088704666,0.7860308044
x,z,0.1963041929,-0.5338153343
y,x,-0.5088704666,0.7860308044
y,y,1,1
y,z,0.3311001423,-0.5524409416
z,x,0.1963041929,-0.5338153343
z,y,0.3311001423,-0.5524409416
z,z,1,1
```

例 3: キー単位での計算

key 項目を単位にして計算する。

```
$ more dat2.csv
key,x,y,z
A,14,0.17,-14
A,11,0.2,-1
A,32,0.15,-2
B,13,0.33,-2
B,10,0.8,-5
B,15,0.45,-9
$ msim k=key c=pearson,cosine f=x,y,z i=dat2.csv o=rsl3.csv
```

```
#END# kgsim c=pearson,cosine f=x,y,z i=dat2.csv k=key o=rsl3.csv
$ more rsl3.csv
key%0,fld1,fld2,pearson,cosine
A,x,y,-0.8746392857,0.8472573627
A,x,z,0.3164384831,-0.521983618
A,y,z,0.1830936883,-0.6719258683
B,x,y,-0.7919009884,0.8782575583
B,x,z,-0.471446429,-0.9051543403
B,y,z,-0.1651896746,-0.8514129252
```

例 4: 類似度名の指定

01 値のデータに付いての計算。ハミング距離と phi 係数を計算する。

```
$ more dat3.csv
x,y,z
1,1,0
1,0,1
1,0,1
0,1,1
$ msim c=hamming,phi f=x,y,z i=dat3.csv o=rsl4.csv
#END# kgsim c=hamming,phi f=x,y,z i=dat3.csv o=rsl4.csv
$ more rsl4.csv
fld1,fld2,hamming,phi
x,y,0.75,-0.5773502692
x,z,0.5,-0.3333333333
y,z,0.75,-0.5773502692
```

例 5: 類似度名の変更

01 値のデータに付いての計算。ハミング距離と phi 係数を計算し、出力項目名を変更する。

```
$ msim c=hamming:ハミング距離,phi:ファイ係数 a=変数 1, 変数 2 f=x,y,z i=dat3.csv o=rsl5.csv
#END# kgsim a=変数 1, 変数 2 c=hamming:ハミング距離,phi:ファイ係数 f=x,y,z i=dat3.csv o=rsl5.csv
$ more rsl5.csv
変数 1, 変数 2, ハミング距離, ファイ係数
x,y,0.75,-0.5773502692
x,z,0.5,-0.3333333333
y,z,0.75,-0.5773502692
```

関連コマンド

mstats：1 変量の統計量を計算するときはこちら。

mmvsim：移動窓を設定した類似度計算。

4.63　mslide 行ずらし

指定した項目の値を指定した行数ずらして出力する。例えば、日別の株価データにおいて収益率 (当日の株価/前日の株価) を計算するなどレコード間の演算を行いたい場合に利用する。

典型的な例を表 4.26〜4.29 に示す。

表 4.26　入力データ

date	val
4/6	1
4/7	2
4/8	3
4/9	4

表 4.27　f=val:nextVal

date	val	nextVal
4/6	1	2
4/7	2	3
4/8	3	4

表 4.28　f=val:nextVal -r

date	val	nextVal
4/7	2	1
4/8	3	2
4/9	4	3

表 4.29　f=val t=2

date	val	val1	val2
4/6	1	2	3
4/7	2	3	4

表 4.26 に示される入力データは日別の集計値が 4 日分示されており、スーパーの売上推移や株価推移と考えればよい。この入力データについて、日々の増加率 (ここでは簡単のために「増加率=翌日の値/当日の値」とする) を計算することを考える。入力データに示される日付 4/6〜4/9 について、それぞれの日の値 (val) を 1 行上にずらし、新しい項目 (newVal) として出力した結果が表 4.27 に示されている。この出力結果に対して mcal コマンドで nextVal/val を計算すれば増加率が求められる。ちなみに、4/9 の行が消えているのは、4/9 の行の次の行が存在しないからである。存在しない時も-n オプションを指定することで NULL 値を出力することができる。

表 4.27 は、下の行の値を上にずらしたが、-r オプションを指定することで、逆に (上の行の値を下に) ずらすことも可能である (表 4.28)。さらに、t=を指定することで、スライドの回数を指定することもできる。t=2 で実行した結果を表 4.29 に示している。これは ”`mslide f=val:val1 | mslide f=val1:val2`”のように、mslide を複数回実行するのと同じ効果がある。なお、`t=`を指定した場合、新たに出力される項目名は、`f=`で指定した項目名に、1 から始まる連番が付与されたものとなる。また `t=`と`-l` を併用することで、最後にずらした結果のみを出力することも可能である。

mslide の機能は mwindow によく似ている。mslide はレコード間の演算を項目演算として実現し、一方で、mwindow はレコード間演算を行集計として実現している。よって、mslide した後の演算は mcal や msel が主役となり、一方で mwindow したのちは msum や mavg などのデータ集約のコマンドが主役となる。

書式

`mslide f= [s=] [k=key] [t=] [-r] [-n] [-l] [i=] [o=]` [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] rverb—[–help]— rverb—[–helpl]— `[--version]`

`f=`　ずらす対象となる項目名を指定する。複数項目指定可能。
　`t=`を指定しないときは、コロンに続いて窓キーの項目名を指定しなければならない。
`s=`　ここで指定した項目 (複数項目指定可) で並べ替えられた後、行をずらす。
　`-q` オプションを指定しないとき、`s=`パラメータは必須。
`k=`　ここで指定された項目の値を単位に処理する。
`t=`　ずらす回数を指定する。省略すれば `t=1` が設定される。
`-r`　逆方向に (上の値を下に) ずらす。
`-n`　次 (前) の行がなくても NULL 値を出力する。
`-l`　最後にずらした結果のみを出力する。

利用例

例 1: **基本例**

```
$ more dat1.csv
date,val
20130406,1
20130407,2
20130408,3
20130409,4
$ mslide s=date f=val:newVal i=dat1.csv o=rsl1.csv
#END# kgslide f=val:newVal i=dat1.csv o=rsl1.csv s=date
$ more rsl1.csv
date%0,val,newVal
20130406,1,2
20130407,2,3
20130408,3,4
```

例 2: **逆にずらした例**

```
$ mslide s=date f=val:newVal -r i=dat1.csv o=rsl2.csv
#END# kgslide -r f=val:newVal i=dat1.csv o=rsl2.csv s=date
$ more rsl2.csv
date%0,val,newVal
20130407,2,1
20130408,3,2
20130409,4,3
```

例 3: **複数回指定した場合**

```
$ mslide s=date f=val t=2 i=dat1.csv o=rsl3.csv
#END# kgslide f=val i=dat1.csv o=rsl3.csv s=date t=2
$ more rsl3.csv
date%0,val,val1,val2
20130406,1,2,3
20130407,2,3,4
```

例 4: **最後にずらした項目だけを出力する例**

```
$ mslide s=date f=val t=2 -l i=dat1.csv o=rsl4.csv
#END# kgslide -l f=val i=dat1.csv o=rsl4.csv s=date t=2
$ more rsl4.csv
date%0,val,val2
20130406,1,3
20130407,2,4
```

例 5: **複数項目名を変更して出力する例**

```
$ mslide s=date f=date:d_,val:v_ t=2 i=dat1.csv o=rsl5.csv
#END# kgslide f=date:d_,val:v_ i=dat1.csv o=rsl5.csv s=date t=2
$ more rsl5.csv
date%0,val,d_1,d_2,v_1,v_2
20130406,1,20130407,20130408,2,3
20130407,2,20130408,20130409,3,4
```

関連コマンド

4.64 msortf レコードの並べ換え

f=パラメータで指定した項目を基準にして、レコードを並べ換える。
ソーティングアルゴリズムは quick sort を利用しており、安定ソート (キーの値が同じ行については元の順序を保存する) にはならないことに注意する。
出力 CSV の項目名の後ろには、並び順情報として % で始まる文字列が付加される。書式は **% 優先順位［並び順］**で、詳細は以下、f=パラメータを参照のこと。

書式

msortf f= [pways=] [maxlines=] [blocks=] [threadCnt=] [-noflg] [i=] [o=] [-assert_diffSize] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=	レコードを並べ換える基準となる項目名リストを指定する。 並び順は、数値/文字列、昇順/降順の組み合せで 4 通り指定できる。 指定方法は % に続けて n と r を以下の通り組み合わせる。 文字列昇順:**項目名** (% 指定なし)、文字列逆順:f=**項目名** %r、数値昇順:f=**項目名** %n、数値降順:f=**項目名** %nr。
-noflg	出力 CSV のヘッダーにソーティングの印 (%0,%0n など) を付けない。
pways=	同時併合ファイル数 ([2-100]:デフォルト 32)【任意】 分割ソートされた複数のファイルを同時に何個併合するかを指定する。
blocks=	バッファブロック数 ([1-1000]:デフォルト 10)【任意】 メモリ内でソートする際のメモリサイズ上限をブロックサイズで指定する。 1 ブロックは入力バッファサイズ× 4 で、デフォルトは 4MB。
maxlines=	メモリソートレコード件数上限 ([100-1000 万]:デフォルト 50 万)【任意】 メモリ内でソートする際の件数の上限を指定する。 データの一行あたりの平均サイズに応じて、blocks=制限と maxlines=制限のいずれかが使われる。
threadCnt=	メモリ内でソートを実行する thread 数 ([1-50]:デフォルト 8)【任意】 分割ソートする際に、マルチスレッドの機能を用いて同時にソートする数を指定する。

備考

1. f=で、文字列項目に対して %n を指定した場合の動作は不定である。
2. f=を省略した場合は i=で指定したファイルを順番に併合する (mcat と同様)。
3. キー項目にに NULL 値が含まれる場合、NULL 値はどのような値よりも小さい値として扱われる。
4. f=a,a のように同じ項目を 2 つ以上指定した場合、最初に指定した項目のみが有効となり、2 番目以降に指定した項目は無視される。msortf f=a,a は msortf f=a と見なされ、msortf f=a,b,a,b は msortf f=a,b と見なされる。

利用例

例 1: 基本例

item、date 順に並べ替える。

```
$ more dat1.csv
item,date,quantity,price
B,20081201,4,40
```

```
A,20081201,10,200
A,20081201,10,100
B,20081203,5,50
B,20081201,2,500
A,20081201,3,300
$ msortf f=item,date i=dat1.csv o=rsl1.csv
#END# kgsortf f=item,date i=dat1.csv o=rsl1.csv
$ more rsl1.csv
item%0,date%1,quantity,price
A,20081201,10,200
A,20081201,10,100
A,20081201,3,300
B,20081201,4,40
B,20081201,2,500
B,20081203,5,50
```

例 2: **数量** (quantity) **降順，金額** (price) **昇順に並べ替える例**

```
$ msortf f=quantity%nr,price%n i=dat1.csv o=rsl2.csv
#END# kgsortf f=quantity%nr,price%n i=dat1.csv o=rsl2.csv
$ more rsl2.csv
item,date,quantity%0nr,price%1n
A,20081201,10,100
A,20081201,10,200
B,20081203,5,50
B,20081201,4,40
A,20081201,3,300
B,20081201,2,500
```

CSV 特殊文字と並び順についての注意

msortf では CSV の特殊文字を解釈して並べ替えを行う為に、CSV の特殊文字であるカンマとダブルクォーツを含むデータについては UNIX の sort コマンドの実行結果と異なることがある。例えば、以下に示すような第一項目に a(0x61)、NULL 値、スペース文字 (0x20)、+(0x2b)、-(0x2d)、カンマ (0x2c)、ダブルクォーツ (0x22) を持つデータについて見てみよう。カンマとダブルクォーツは CSV の特殊文字のため特別な表記となっている。また表示上の分かりやすさのために第二項目 f2 として全行に”x”を付加している。

```
f1,f2
a,x
,x
 ,x
+,x
-,x
",",x
"""",x
```

このデータを「msortf f=f1」で並べ替えた結果は以下の通りで、CSV フォーマットにおける特殊記号を考慮した上での本来の並び順 (NULL, スペース, ダブルクォーツ,+, カンマ,-,a) になっていることがわかる。

```
f1,f2
,x
 ,x
"""",x
+,x
",",x
-,x
a,x
```

ベンチマークテスト

msortf の速度についてのベンチマークテストの結果を示す。入力データは以下に示すような 6 項目からなるデータである。全ての項目は一様乱数により生成されている。

```
key,fld1,fld2,fld3,fld4,fldn
95547922,162,159,192,118,74
81438069,138,157,155,122,58
26885062,129,199,133,198,75
32651684,180,107,123,170,-14
10245631,164,103,159,154,-63
15145156,182,191,175,107,-60
29254245,188,185,129,124,5
85423170,116,164,175,113,57
55155879,105,163,195,167,25
66997216,195,139,195,113,39
.
.
```

キー項目の値の種類数および状態に関する比較

レコード数を 100 万件とし、キー (key 項目) の値の種類を 2、10、100、1000、10000 とした場合の結果を表??に示す。また、表中の (1),(2) をグラフ化したものが図 4.3 に、また (4),(5) をグラフ化したものが図 4.4 に示されている。「乱数」項目は乱数の上限を最大とした場合の乱数値をキーとしている。また「乱数昇順 (降順)」は、「乱数」データを事前に昇順 (降順) に並べ替えたデータである。msortf 以外に、比較の為に MUSASHI の xtsort コマンド、および UNIX の sort コマンドの結果も示す。「sort -k1」は、第 1 番目の項目で並べ替えた結果である。また上三行は msortf、xtsort、sort の各コマンドで第 1 番目の項目を文字列として並べ替えた時の結果であり、下三行は数値として並べ替えたときの結果である。実行環境は、MacBookPro, Mac OS X 10.9.1, 2.6GHz Intel Core i7, 16GB メモリ、である。

表 4.30 キー項目の値の種類数および状態に関する比較

No.	コマンド	2 種	10 種	100 種	1000 種	10000 種	乱数	乱数昇順	乱数降順
(1)	msortf f=key	0.29	0.33	0.37	0.40	0.43	0.50	0.29	0.28
(2)	xtsort -k key	1.25	1.24	1.22	1.20	1.19	1.12	0.85	1.00
(3)	sort -k1	16.96	16.63	16.05	15.56	15.08	13.68	6.85	7.13
(4)	msortf f=key%n	0.46	0.56	0.65	0.72	0.79	1.02	0.59	0.59
(5)	xtsort -k key%n	2.52	2.72	2.96	3.16	3.21	3.22	2.31	2.32
(6)	sort -k1 -n	16.65	14.52	11.54	8.56	5.71	0.95	0.33	0.36

msortf は xtsort より 2 倍〜5 倍高速である。sort については、条件にもよるが数十倍高速である。MUSASHI の sort とで利用している quick sort のアルゴリズムは全く同じであるが、MCMD においては分割ソートにおいてマルチスレッドを使い並列処理している。その影響の差が出ているということであろう。次に、キーの種類数を 100 および最大に固定し、データ件数を 100 万件から 1000 万件まで変化させた時の文字ソートの速度について実験した。ここでは msortf、xtsort の 2 つのコマンドの比較を図 4.5,4.6 に示す。

関連コマンド

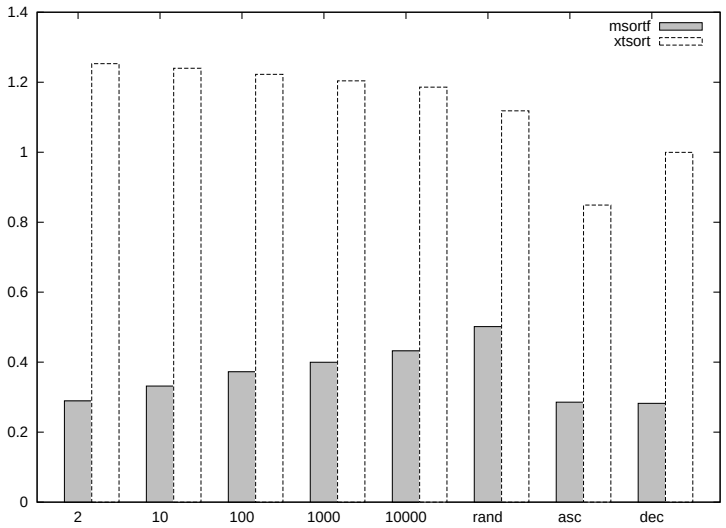


図 4.3　キーの種類数別に見る msortf,xtsort の文字列ソートの比較 (横軸がキーの種類数，縦軸が秒数)

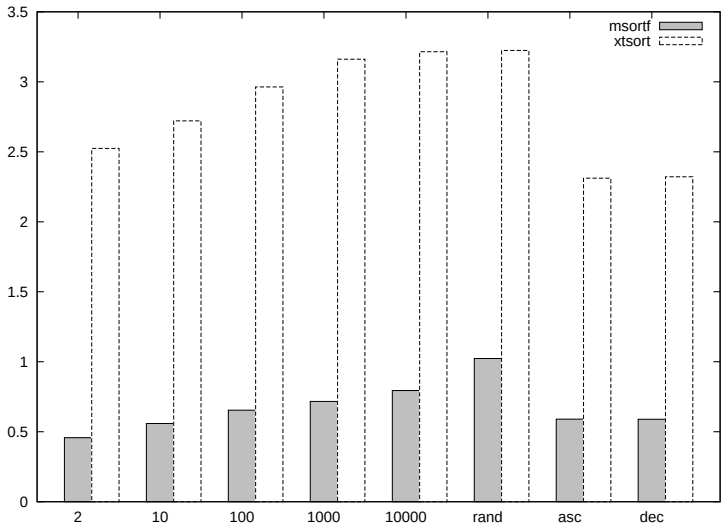


図 4.4　キーの種類数別に見る msortf,xtsort の数値ソートの比較 (横軸がキーの種類数，縦軸が秒数)

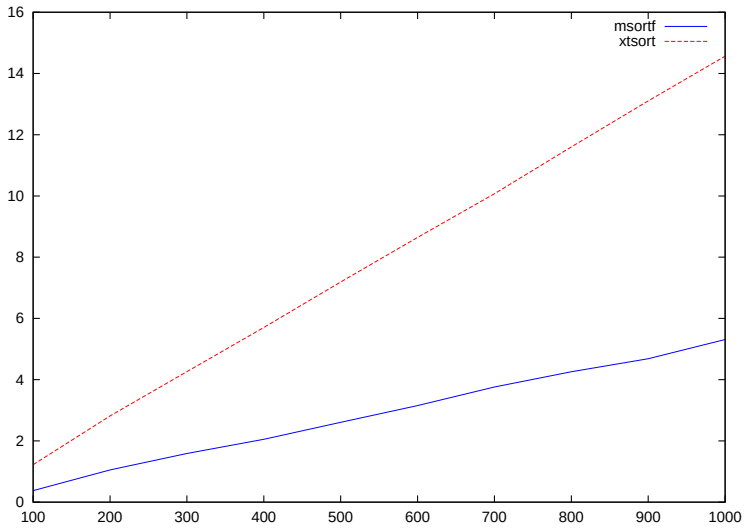


図 4.5 キーの種類数を 100 とした場合の結果 (横軸がデータ件数，縦軸が秒数)

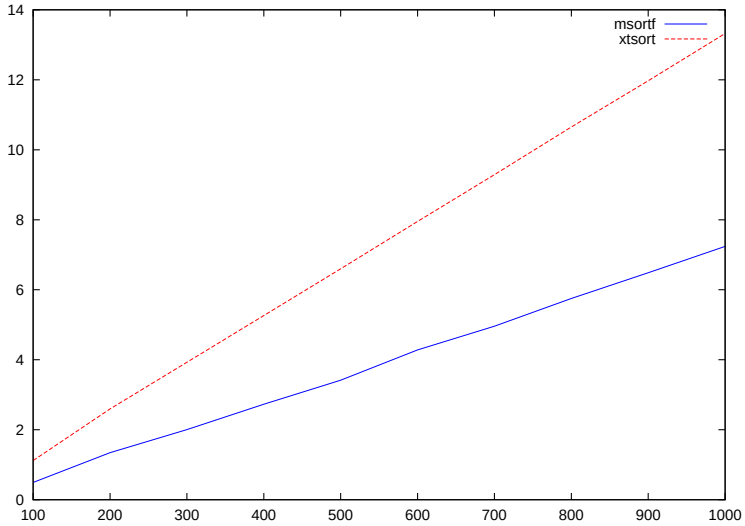


図 4.6 キーの種類数を乱数 (最大) とした場合の結果 (横軸がデータ件数，縦軸が秒数)

4.65　msplit 区切り文字による項目分割

区切り文字によって項目を分割する。

書式

```
msplit f= a= [-r] [delim=] [i=] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

f=	区切り文字で分割するデータの項目名を指定する
a=	新項目を指定する ここで指定した項目名分分割される。足りない分は NULL となる
delim=	新しい区切り文字を指定する。デフォルト値は半角スペース。
-r	f=で指定した項目を取り除く

利用例

例 1: **基本例**

半角スペースで分割

```
$ more dat1.csv
id,data
A,1 10 2
A,2 20 3
B,1 15 5
B,3 10 4
B,1 20 6
$ msplit f=data a=d1,d2,d3 i=dat1.csv o=rsl1.csv
#END# kgsplit a=d1,d2,d3 f=data i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,data,d1,d2,d3
A,1 10 2,1,10,2
A,2 20 3,2,20,3
B,1 15 5,1,15,5
B,3 10 4,3,10,4
B,1 20 6,1,20,6
```

例 2: -r **利用**

-r を指定することで、f=で項目を削除できる。

```
$ msplit f=data a=d1,d2,d3 -r i=dat1.csv o=rsl2.csv
#END# kgsplit -r a=d1,d2,d3 f=data i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,d1,d2,d3
A,1,10,2
A,2,20,3
B,1,15,5
B,3,10,4
B,1,20,6
```

例 3: **分割数不一致**

a=で指定した項目数よりも分割できる項目数が少ない場合は、NULL が追加され、多い場合、先頭から指定した分割数まで出力する

```
$ more dat2.csv
id,data
A,1 10 2
A,2 20 3
B,1 15 5
B,3 4
B,1
$ msplit f=data a=d1,d2 i=dat2.csv o=rsl3.csv
#END# kgsplit a=d1,d2 f=data i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,data,d1,d2
A,1 10 2,1,10
A,2 20 3,2,20
B,1 15 5,1,15
B,3 4,3,4
B,1,1,
```

例 4: delim **指定**

delim=を使用して半角スペース以外の文字で分割する

```
$ more dat3.csv
id,data
A,1_10_3
A,2_20_5
B,1_15_6
B,3_10_7
B,1_20_8
$ msplit f=data a=d1,d2,d3 delim=_ i=dat3.csv o=rsl4.csv
#END# kgsplit a=d1,d2,d3 delim=_ f=data i=dat3.csv o=rsl4.csv
$ more rsl4.csv
id,data,d1,d2,d3
A,1_10_3,1,10,3
A,2_20_5,2,20,5
B,1_15_6,1,15,6
B,3_10_7,3,10,7
B,1_20_8,1,20,8
```

関連コマンド

4.66　mstats 一変数の統計量算出

f=パラメータで指定した数値項目について c=パラメータで指定した統計量の計算をする。k=を指定することで、キー単位で集計することができる。f=で指定した項目の NULL 値は無視される。ただし、全行が NULL 値であれば NULL 値が出力される。(**注**)k=と f=パラメータで指定した項目以外については、どの行が出力されるか¿は不定であることに注意してください。

書式

mstats c= f= [k=] [-n]　[i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

k=	ここで指定された項目 (複数項目指定可) を単位として集計する。
f=	ここで指定された項目 (複数項目指定可) の値が集計される。
c=	統計量 (以下のリストから一つだけ指定可) sum|mean|count|ucount|devsq|var|uvar|sd|usd|USD|cv|min|qtile1| median|qtile3|max|range|qrange|mode|skew|uskew|kurt|ukurt
-n	Null データの場合は NULL にするフラグ

統計量リスト

利用例

例 1: **基本例**

「顧客」項目を単位に「数量」と「金額」項目の各統計量合計値を計算する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
B,5,20
B,2,10
C,1,15
C,3,10
C,1,21
$ mstats k=顧客 f=数量,金額 c=sum i=dat1.csv o=rsl1.csv
#END# kgstats c=sum f=数量,金額 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客%0,数量,金額
A,1,10
B,7,30
C,5,46
```

例 2: **基本例** 2

各統計量最大値を計算する。

```
$ mstats k=顧客 f=数量,金額 c=max i=dat1.csv o=rsl2.csv
#END# kgstats c=max f=数量,金額 i=dat1.csv k=顧客 o=rsl2.csv
```

c=の値	内容	式	備考
count	件数 (NULL 値以外)	n: NULL 値以外の件数	文字列項目に対しては適用できない。
ucount	ユニーク件数	un: 重複値を省いた件数	文字列項目に対しては適用できない。
sum	合計	$sum = \sum_{i=1}^{n} x_i$	
mean	算術平均	$m = \frac{1}{n}\sum_{i=1}^{n} x_i$	
devsq	偏差平方和	$S = \sum_{i=1}^{n}(x_i - m)^2$	
var	分散	$s^2 = \frac{1}{n}S$	
uvar	分散 (不偏推定値)	$u^2 = \frac{1}{n-1}S$	
sd	標準偏差	$s = \sqrt{s^2}$	
usd	標準偏差 (不偏分散の sqrt)	$u = \sqrt{u^2}$	一般的によく使われる標準偏差
USD	不偏標準偏差	省略	正確な不偏推定
cv	変動係数	$cv = s/m \times 100\%$	
mode	最頻値	$mode$: 最頻出の値	全ての値が異なる場合は NULL を、同頻度の場合はより小さい方の値を出力する。
min	最小値	$min = \min_i x_i$	
max	最大値	$max = \max_i x_i$	
range	範囲	$r = max - min$	
median	中央値	$Q2$ = 昇順に並べた時の第 2 四文位点	
qtile1	第 1 四分位点	$Q1$ = 昇順に並べた時の第 1 四文位点	
qtile3	第 1 四分位点	$Q3$ = 昇順に並べた時の第 3 四文位点	
qrange	四分位範囲	$rq = Q3 - Q1$	
skew	歪度	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-m)^3}{s^3}$	
uskew	歪度 (不偏推定値)	省略	
kurt	尖度	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-m)^4}{s^4} - 3.0$	
ukurt	尖度 (不偏推定値)	省略	

```
$ more rsl2.csv
顧客 %0, 数量, 金額
A,1,10
B,5,20
C,3,21
```

関連コマンド

msim：2 変量の統計量を求める。
mavg：c=avg に特化したコマンド。
msum：c=sum に特化したコマンド。
mcount：c=count と異なり、集計キーの行数をカウントする。

4.67　msum 項目値の合計

k=パラメータで指定した項目の値が同じ行について、f=パラメータで指定した集計項目の項目値を合計する。
(**注**)k=と f=パラメータで指定した項目以外については、どの行が出力されるかは不定であることに注意してください。

書式

msum f= [k=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

k=　集計の単位となる項目名リスト（複数項目指定可）を指定する。
f=　ここで指定された項目（複数項目指定可）の値が集計される。NULL 値は無視される。
-n　f=で指定した項目に NULL 値が入っていると計算結果も NULL とする。

利用例

例 1: **基本例**
「顧客」項目を単位に「数量」と「金額」項目の合計値を計算し、「数量合計」と「金額合計」という項目名で出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
B,1,15
A,2,20
B,3,10
B,1,20
$ msum k=顧客 f=数量:数量合計,金額:金額合計 i=dat1.csv o=rsl1.csv
#END# kgsum f=数量:数量合計,金額:金額合計 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客 %0,数量合計,金額合計
A,3,30
B,5,45
```

関連コマンド

mhashsum：集計キーを事前に並べ替えなくても計算できる。
mavg：平均バージョン。
mstats：その他の多様な統計量を求めるのであればこれ。

4.68 msummary 1 変数の統計量の計算

f=パラメータで指定した集計項目で c=パラメータで指定した統計量の計算をする。

書式

msummary c= f= [a=] [k=] [-n] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

k= キー項目名リスト (複数項目指定可)【集計キーブレイク処理】
ここで指定された項目を単位として集計する。
指定する場合は事前に指定する集計の単位となる項目順に並べ替えておく必要がある。

f= 集計項目名リスト (複数項目指定可)
ここで指定された項目の値が集計される。
-x,-nfn オプション使用時は、項目番号 (0〜) で指定。

c= 統計量リスト (複数項目指定可)
出力する統計量をコンマで区切って指定する。
統計量リスト:
sum/mean/count/ucount/devsq/var/uvar/sd/usd/cv/min/qtile1/median/qtile3/max/
range/qrange/mode/skew/uskew/kurt/ukurt

-a 新項目名
f=パラメータで指定した項目名をデータとして出力する際の項目名 (省略時は fld) を指定する。

統計量リスト

c=パラメータで指定できる統計量と定義を表 4.31 に示す。

利用例

例 1: 基本例

「顧客」項目を単位に「数量」と「金額」項目の中央値・平均値を求める。統計量を求めた項目名は「変数」という項目に出力する。

```
$ more dat1.csv
顧客,数量,金額
A,1,10
A,2,20
B,1,15
B,3,10
B,1,20
$ msummary k=顧客 f=数量,金額 c=median:中央値,mean:平均値 a=変数 i=dat1.csv o=rsl1.csv
#END# kgsummary a=変数 c=median:中央値,mean:平均値 f=数量,金額 i=dat1.csv k=顧客 o=rsl1.csv
$ more rsl1.csv
顧客%0,変数,中央値,平均値
A,数量,1.5,1.5
A,金額,15,15
B,数量,1,1.666666667
B,金額,15,15
```

表 4.31　統計量リスト

c=の値	内容	式	備考
count	件数 (NULL 値以外)	n: NULL 値以外の件数	文字列項目に対しては適用できない。
ucount	ユニーク件数	un: 重複値を省いた件数	文字列項目に対しては適用できない。
sum	合計	$sum = \sum_{i=1}^{n} x_i$	
mean	算術平均	$m = \frac{1}{n} \sum_{i=1}^{n} x_i$	
devsq	偏差平方和	$S = \sum_{i=1}^{n} (x_i - m)^2$	
var	分散	$s^2 = \frac{1}{n} S$	
uvar	分散 (不偏推定値)	$u^2 = \frac{1}{n-1} S$	
sd	標準偏差	$s = \sqrt{s^2}$	
usd	標準偏差 (不偏分散の sqrt)	$u = \sqrt{u^2}$	一般的によく使われる標準偏差
cv	変動係数	$cv = s/m \times 100\%$	
mode	最頻値	$mode$: 最頻出の値	全ての値が異なる場合は NULL を、同頻度の場合はより小さい方の値を出力する。
min	最小値	$min = \min_i x_i$	
max	最大値	$max = \max_i x_i$	
range	範囲	$r = max - min$	
median	中央値	$Q2$ = 昇順に並べた時の第 2 四文位点	
qtile1	第 1 四分位点	$Q1$ = 昇順に並べた時の第 1 四文位点	
qtile3	第 1 四分位点	$Q3$ = 昇順に並べた時の第 3 四文位点	
qrange	四分位範囲	$rq = Q3 - Q1$	
skew	歪度	$\frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - m)^3}{s^3}$	
uskew	歪度 (不偏推定値)	省略	
kurt	尖度	$\frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - m)^4}{s^4} - 3.0$	
ukurt	尖度 (不偏推定値)	省略	

関連コマンド

mstats：求める統計量が 1 つのとき用いる。

4.69 mtab2csv TSV から CSV データへの変換

タブ区切りデータを CSV データへ変換する。d=で区切り文字を指定することで、タブ以外の区切り文字のテキストファイルも変換することが可能である。変換後の項目数に違いある場合には、その直前行まで出力され、その後エラー終了する。

書式

```
mtab2csv [d=] [-r]  [i=] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

d= 区切り文字の指定 (1 バイト文字のみ指定可)。
-r 改行コード (CR:\r) を取り除く。
MCMD が扱う CSV は改行コードが LF(\n) 固定であるため、
Windows のテキスト改行 CR+LF(\r\n) や Mac のテキスト改行 CR(\r) があれば、
単なる文字列として扱ってしまい、変換後に不具合が生じる。
この問題を回避するためのオプションである。

利用例

例 1: **基本例**

タブ区切りデータを csv へ変換

```
$ more dat1.tab
id        data        data2
A         1102        a
A         2203        aaa
B         1155        bbbb
B         3104        c
B         1206        de
$ mtab2csv i=dat1.tab o=rsl1.csv
#END# kgtab2csv i=dat1.tab o=rsl1.csv
$ more rsl1.csv
id,data,data2
A,1102,a
A,2203,aaa
B,1155,bbbb
B,3104,c
B,1206,de
```

例 2: d=**指定**

d=を使用して tab 以外の区切り文字を使う

```
$ more dat2.bar
id-data-data2
A-1102-a
A-2203-aaa
B-1155-bbbb
B-3104-c
B-1206-de
$ mtab2csv d=- i=dat2.bar o=rsl2.csv
```

```
#END# kgtab2csv d=- i=dat2.bar o=rsl2.csv
$ more rsl2.csv
id,data,data2
A,1102,a
A,2203,aaa
B,1155,bbbb
B,3104,c
B,1206,de
```

関連コマンド

mxml2csv:XML データを CSV データへ変換する
msplit:項目値の区切り文字によるデータ分割

4.70 mtee 複数ファイルへのコピー

入力データの内容を標準出力および複数ファイルへそのままコピー出力する。

書式

mtee [o=] [-nostdout] [i=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

o=	ここで指定された複数のファイルに入力ファイルと同一内容が出力される。 またこのパラメータが省略された時には標準出力にのみ出力される。
-nostdout	標準出力には出力しない場合に指定する。

利用例

例 1: **基本例**

dat1.csv ファイルを rsl1.csv と rsl2.csv という 2 つのファイルにコピーする。また、標準出力に出力されるので、画面上に内容が出力される。

```
$ more dat1.csv
customer,quantity,price
A,1,10
A,2,20
B,1,15
$ mtee i=dat1.csv o=rsl1.csv,rsl2.csv
customer,quantity,price
A,1,10
A,2,20
B,1,15
#END# kgtee i=dat1.csv o=rsl1.csv,rsl2.csv
$ more rsl1.csv
customer,quantity,price
A,1,10
A,2,20
B,1,15
$ more rsl2.csv
customer,quantity,price
A,1,10
A,2,20
B,1,15
```

例 2: **標準出力なし**

-nostdout を指定すると、rsl1.csv と rsl2.csv という 2 つのファイルにコピーのみ行い、標準出力には出力しない。

```
$ mtee i=dat1.csv o=rsl1.csv,rsl2,csv -nostdout
#END# kgtee -nostdout i=dat1.csv o=rsl1.csv,rsl2,csv
```

関連コマンド

4.71 mtonull NULL 値へ置換

f=パラメータで指定した項目を対象に、v=パラメータで指定した値にマッチした項目データを NULL 値に置換する。マッチの方法としては完全一致 (デフォルト) と部分文字列マッチ (-sub オプション) を選択できる。

書式

mtonull f= v= [-sub] [-W] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f= 置換対象の項目名リスト (複数項目指定可) を指定する。
v= f=パラメータで指定した項目の値が、ここで指定した文字列リスト (複数項目指定可) のいずれかにマッチすれば NULL 値に置換する。

オプション

-sub 検索を完全一致ではなく部分文字列マッチで比較
f= パラメータで指定した項目の値に、
v= パラメータで指定の文字列が部分文字列として一つでも含まれていれば
その項目値を NULL 値に置換する。
-W -sub オプションが指定されているときにワイド文字として部分文字列マッチをおこなう。

利用例

例 1: 基本例

quantity と price 項目が 0 を NULL 値に置換する。

```
$ more dat1.csv
item,quantity,price
A,0,1
B,1,0
C,2,200
D,3,0
E,0,298
$ mtonull f=quantity,price v=0 i=dat1.csv o=rsl1.csv
#END# kgtonull f=quantity,price i=dat1.csv o=rsl1.csv v=0
$ more rsl1.csv
item,quantity,price
A,,1
B,1,
C,2,200
D,3,
E,,298
```

例 2: NULL 値に置換する数字の指定

quantity と price 項目が 0 もしくは 1 を NULL 値に置換する。

```
$ mtonull f=quantity,price v=0,1 i=dat1.csv o=rsl2.csv
#END# kgtonull f=quantity,price i=dat1.csv o=rsl2.csv v=0,1
$ more rsl2.csv
item,quantity,price
A,,
B,,
C,2,200
D,3,
E,,298
```

例 3: **部分文字列マッチでの置換**

quantity と price 項目が 0 を含めば NULL 値に置換する。

```
$ mtonull -sub f=quantity,price v=0 i=dat1.csv o=rsl3.csv
#END# kgtonull -sub f=quantity,price i=dat1.csv o=rsl3.csv v=0
$ more rsl3.csv
item,quantity,price
A,,1
B,1,
C,2,
D,3,
E,,298
```

例 4: **指定の文字列の置換**

item 項目に apple、orange、pineapple を含む値を NULL 値に置換する。

```
$ more dat2.csv
item,price
fruit:apple,100
fruit:peach,250
fruit:grape,300
fruit:pineapple,450
fruit:orange,500
$ mtonull f=item v=apple,orange,pineapple -sub i=dat2.csv o=rsl4.csv
#END# kgtonull -sub f=item i=dat2.csv o=rsl4.csv v=apple,orange,pineapple
$ more rsl4.csv
item,price
,100
fruit:peach,250
fruit:grape,300
,450
,500
```

関連コマンド

mnullto：逆に NULL 値を指定の文字列に置換する。

4.72 mtra 縦型データをベクトル項目に変換

f=パラメータで指定した項目値をアイテムとし、それらのアイテムを連結し新しいベクトル項目 (トランザクション項目とも呼ぶ) として出力する。アイテムの区切り文字は delim=パラメータで指定する。

書式

mtra f= [s=] [k=] [delim=] [-r] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

f=	ここで指定した項目 (複数項目指定可) の値がアイテムとして連結されトランザクション項目となる。 NULL 値は無視される。
s=	ここで指定した項目 (複数項目指定可) で並べ替えられた後、変換が行われる。
k=	文字列パターンの単位となる項目名 (複数項目指定可) リスト。 -r オプションが指定された時は指定できない。
delim=	ここで指定した文字を区切り文字とする（省略時はスペース)。
-r	条件反転 トランザクション項目を縦型データに変換する。

利用例

例 1: 基本例

customer を単位に item をスペース区切りで結合し、transaction という項目名で出力する。

```
$ more dat1.csv
customer,item
A,a
A,b
B,c
B,d
B,e
$ mtra k=customer f=item:transaction i=dat1.csv o=rsl1.csv
#END# kgtra f=item:transaction i=dat1.csv k=customer o=rsl1.csv
$ more rsl1.csv
customer%0,transaction
A,a b
B,c d e
```

例 2: アイテムの区切り文字を-(ハイフン）で実行

```
$ mtra k=customer f=item:transaction delim=- i=dat1.csv o=rsl2.csv
#END# kgtra delim=- f=item:transaction i=dat1.csv k=customer o=rsl2.csv
$ more rsl2.csv
customer%0,transaction
A,a-b
B,c-d-e
```

例 3: アイテムを降順に並べ替えてから変換

```
$ mtra k=customer s=item%r f=item:transaction i=dat1.csv o=rsl3.csv
#END# kgtra f=item:transaction i=dat1.csv k=customer o=rsl3.csv s=item%r
$ more rsl3.csv
customer%0,transaction
A,b a
B,e d c
```

関連コマンド

mvsort：トランザクションデータは、ベクトル型データを処理する一連の処理コマンド (mv から始まる) によって加工できる。

mcross：トランザクションデータとしてではなく、個々のアイテムを独立した項目として出力し、その出現件数を出力する。

mtrafld：「項目名=値」の形式でトランザクションデータを作成する。

mtraflg：項目名をアイテムとしてトランザクションデータを作成する。

4.73 mtrafld クロス表をトランザクション項目に変換

f=で指定した項目値とその値のペアのアイテムを作成し、それらのアイテムを連結し新しいベクトル項目 (トランザクション項目とも呼ぶ) として出力する。

書式

mtrafld a= [f=] [delim=] [delim2=] [-r] [-valOnly] [i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a=	トランザクション項目名を指定する。
f=	項目名リスト (複数項目指定可)【-r 指定時必須、それ以外は任意】 ここで指定された項目名と値とを連結したアイテムを作成し トランザクション項目として出力される。 -r オプションの指定がある時は トランザクションデータから抜き出す項目名を指定する。 -r オプションが指定されたとき，このパラメータは省略可能である。 省略すると、全ての項目名と値ペアを処理対象とする。 ただし、f=パラメータを省略すると標準入力 (パイプ入力) は利用できない。
delim=	トランザクション項目のアイテムを区切る文字を指定する (省略時はスペース)。
delim2=	項目名と値ペアとを区切る文字を指定する (省略時は=)。
-r	条件反転 トランザクション項目をクロス表に変換する。
-valOnly	このオプションが指定されると、アイテムとして「項目名=」は出力しない。

利用例

例 1: 基本例

price と quantity 項目を 1 つの文字列として連結し、transaction という項目名で出力する。

```
$ more dat1.csv
customer,price,quantity
A,198,1
B,325,2
C,200,3
D,450,2
E,100,1
$ mtrafld a=transaction f=price,quantity i=dat1.csv o=rsl1.csv
#END# kgtrafld a=transaction f=price,quantity i=dat1.csv o=rsl1.csv
$ more rsl1.csv
customer,transaction
A,price=198 quantity=1
B,price=325 quantity=2
C,price=200 quantity=3
D,price=450 quantity=2
E,price=100 quantity=1
```

例 2: **基本例** 2

出力された結果を-r をつけて再実行し元に戻す。

```
$ mtrafld -r a=transaction f=price,quantity i=rsl1.csv o=rsl2.csv
#END# kgtrafld -r a=transaction f=price,quantity i=rsl1.csv o=rsl2.csv
$ more rsl2.csv
customer,price,quantity
A,198,1
B,325,2
C,200,3
D,450,2
E,100,1
```

例 3: **区切り文字の指定**

price と数量 quantity 項目を_(アンダーバー）で区切り 1 つの文字列として連結し、項目名とデータは：(コロン）で区切り transaction という項目名で出力する。

```
$ mtrafld a=transaction f=price,quantity delim=_ delim2=':' i=dat1.csv o=rsl3.csv
#END# kgtrafld a=transaction delim2=: delim=_ f=price,quantity i=dat1.csv o=rsl3.csv
$ more rsl3.csv
customer,transaction
A,price:198_quantity:1
B,price:325_quantity:2
C,price:200_quantity:3
D,price:450_quantity:2
E,price:100_quantity:1
```

例 4: NULL **値を含む場合**

```
$ more dat2.csv
customer,price,quantity
A,198,1
B,,2
C,200,
D,450,2
E,,
$ mtrafld a=transaction f=price,quantity i=dat2.csv o=rsl4.csv
#END# kgtrafld a=transaction f=price,quantity i=dat2.csv o=rsl4.csv
$ more rsl4.csv
customer,transaction
A,price=198 quantity=1
B,quantity=2
C,price=200
D,price=450 quantity=2
E,
```

例 5: NULL **値を含む場合** 2

出力された結果を-r をつけて再実行し元に戻す。

```
$ mtrafld -r a=transaction f=price,quantity i=rsl4.csv o=rsl5.csv
#END# kgtrafld -r a=transaction f=price,quantity i=rsl4.csv o=rsl5.csv
$ more rsl5.csv
customer,price,quantity
A,198,1
B,,2
C,200,
```

```
D,450,2
E,,
```

例 6: -valOnly **の指定**

```
$ mtrafld -valOnly f=price,quantity a=transaction i=dat2.csv o=rsl6.csv
#END# kgtrafld -valOnly a=transaction f=price,quantity i=dat2.csv o=rsl6.csv
$ more rsl6.csv
customer,transaction
A,198 1
B,2
C,200
D,450 2
E,
```

関連コマンド

mvsort：トランザクションデータはベクトル型データを処理する一連の処理コマンド (mv から始まる) によって加工できる。

mcross：トランザクションデータとしてではなく、個々のアイテムを独立した項目として出力し、その出現件数を出力する。

mtra：項目の値をアイテムとしてトランザクションデータを作成する。

mtraflg：項目名をアイテムとしてトランザクションデータを作成する。

4.74　mtraflg クロス表をトランザクション項目に変換

f=パラメータで指定した項目値が NULL 値かどうかをチェックし、NULL 値以外であれば，それらの項目名を 1 つのアイテムとして連結し新しいベクトル項目 (トランザクション項目とも呼ぶ) として出力する。

書式

mtraflg a= f= [delim=] [-r]　[i=] [o=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

a=	トランザクション項目名を指定する。
f=	ここで指定された項目値 (複数項目指定可) をチェックし、トランザクションデータを作成する。 (-r オプションの指定がある時はトランザクションデータから項目名として抜き出す値のリスト)
delim=	ここで指定した文字をトランザクション項目のアイテム間の区切りとする（省略時はスペース)。 文字列の指定はできない。1 バイト文字のみ指定可能。
-r	条件反転 トランザクション型から縦型へデータを変換する。

利用例

例 1: **基本例**

egg と milk 項目の値が NULL 値以外なら、それら項目名を要素としたベクトルを作成する。

```
$ more dat1.csv
customer,egg,milk
A,1,1
B,,1
C,1,
D,1,1
$ mtraflg f=egg,milk a=transaction i=dat1.csv o=rsl1.csv
#END# kgtraflg a=transaction f=egg,milk i=dat1.csv o=rsl1.csv
$ more rsl1.csv
customer,transaction
A,egg milk
B,milk
C,egg
D,egg milk
```

例 2: **基本例** 2

出力された結果を-r をつけて再実行し元に戻す。

```
$ mtraflg -r f=egg,milk a=transaction i=rsl1.csv o=rsl2.csv
#END# kgtraflg -r a=transaction f=egg,milk i=rsl1.csv o=rsl2.csv
$ more rsl2.csv
customer,egg,milk
A,1,1
B,,1
C,1,
D,1,1
```

例 3: 区切り文字の指定

区切り文字を-(ハイフン）で連結し、transaction という項目名で出力する。

```
$ mtraflg f=egg,milk a=transaction delim=- i=dat1.csv o=rsl3.csv
#END# kgtraflg a=transaction delim=- f=egg,milk i=dat1.csv o=rsl3.csv
$ more rsl3.csv
customer,transaction
A,egg-milk
B,milk
C,egg
D,egg-milk
```

関連コマンド

mvsort：トランザクションデータはベクトル型データを処理する一連の処理コマンド (mv から始まる) によって加工できる。

mcross：トランザクションデータとしてではなく、個々のアイテムを独立した項目として出力し、その出現件数を出力する。

mtra：項目の値をアイテムとしてトランザクションデータを作成する。

mtrafld：「項目名=値」の形式でトランザクションデータを作成する。

4.75　muniq レコードの単一化

値が重複した行を単一化する。

書式

muniq [k=] [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

k=　行を単一化する単位となる項目名リストを指定する。

利用例

例 1: 基本例

date 項目を単位に重複行を削除し単一にする。

```
$ more dat1.csv
date,customer
20081201,A
20081202,A
20081202,B
20081202,B
20081203,C
$ muniq k=date i=dat1.csv o=rsl1.csv
#END# kguniq i=dat1.csv k=date o=rsl1.csv
$ more rsl1.csv
date%0,customer
20081201,A
20081202,B
20081203,C
```

例 2: 複数の項目での重複行の削除

date と customer 項目を単位に重複行を削除し単一にする。

```
$ muniq k=date,customer i=dat1.csv o=rsl2.csv
#END# kguniq i=dat1.csv k=date,customer o=rsl2.csv
$ more rsl2.csv
date%0,customer%1
20081201,A
20081202,A
20081202,B
20081203,C
```

関連コマンド

mbest：同一キーの中で何番目の行を選択するかを指定したい場合は mbest コマンドを使う。

4.76 mvcat ベクトルの併合

複数のベクトルを併合して新しいベクトルを生成する。
典型的な例を表 4.32〜4.34 に示す。

表 4.32 入力データ

in.csv

no	items1,items2
1	a c,b
2	a d,a e
3	b f,
4	e,e

表 4.33 基本例

```
mvcat vf=item1,items2 a=catItems i=in.csv
```

no	catItems
1	a c b
2	a d a e
3	b f
4	e e

表 4.34 併合前のベクトルを残す例

```
mvcat vf=item1,items2 -A i=in.csv
```

no	items1,items2,new
1	a c,b,a c b
2	a d,a e,a d a e
3	b f,,b f
4	e,e,e e

書式

mvcat vf= a= [-A] [i=] [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf= 併合する複数のベクトル項目名 (i=ファイル上) を指定する。
項目名にワイルドカードを使うことができる。
a= 併合後の項目名を指定する。
-A 新しい項目として追加する。このオプションを指定しなければ、併合元の項目 (vf=) は削除される。

利用例

例 1: ワイルドカードを利用した例

```
$ more dat1.csv
items1,items2,items3,items4
b a c,b,x,y
c c,,x,y
e a a,a a a,x,y
$ mvcat vf=items* a=items i=dat1.csv o=rsl1.csv
#END# kgvcat a=items i=dat1.csv o=rsl1.csv vf=items*
$ more rsl1.csv
items
b a c b x y
```

```
c c x y
e a a a a a x y
```

関連コマンド

4.77　mvcommon ベクトル要素の参照選択

ベクトルから、参照ファイルで指定された要素を選択する。
典型的な例を表 4.35〜4.38 に示す。

表 4.35　入力データ

in.csv

no	items
1	a b c
2	a d
3	b f e f
4	f c d

表 4.36　参照ファイル

ref.csv

item
a
c
e

表 4.37　基本例

vf=items m=ref.csv K=item

no	items
1	a c
2	a
3	e
4	c

表 4.38　アンマッチアイテムの選択例

vf=items m=ref.csv K=item -r

no	items
1	b
2	d
3	b f f
4	f d

mvcommon コマンドは、参照ファイルデータを一旦全てメモリにセットするので、巨大な参照ファイルを指定した場合はメモリを使い果たす可能性があることに注意する。

書式

mvcommon vf= [-A] K= [-r] m=| i= [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=　結合キーとなるアイテム集合の項目名 (i=ファイル上) を指定する。
　　複数項目指定可能。アイテムはソーティングされている必要はない。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。
　　例) f=数量:置換後の項目名
-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。
m=　参照ファイルを指定する。
　　このパラメータが省略された時には標準入力が用いられる。(i=指定ありの場合)
K=　参照ファイル (m=) 上の結合キーとなるアイテムの項目名を指定する。
-r　vf=と K=の要素がマッチしない要素を選択する。

利用例

例 1: **複数項目に対して結合する例**

```
$ more dat1.csv
items1,items2
b a c,b b
c c,a d
e a a,a a
$ more ref1.csv
item
a
c
e
$ mvcommon vf=items1,items2 K=item m=ref1.csv i=dat1.csv o=rsl1.csv
#END# kgvcommon K=item i=dat1.csv m=ref1.csv o=rsl1.csv vf=items1,items2
$ more rsl1.csv
items1,items2
a c,
c c,a
e a a,a a
```

例 2: **項目名を変更する例**

item2 に新項目名 new2 を指定しているので、項目名が変更され出力される。

```
$ mvcommon vf=items1,items2:new2 K=item m=ref1.csv i=dat1.csv o=rsl2.csv
#END# kgvcommon K=item i=dat1.csv m=ref1.csv o=rsl2.csv vf=items1,items2:new2
$ more rsl2.csv
items1,new2
a c,
c c,a
e a a,a a
```

例 3: **項目を追加する例**

item1 に新項目名 new1 を、item2 に新項目名 new2 を指定し、-A オプションを付けているので新項目 new1 と new2 が追加され出力される。

```
$ mvcommon vf=items1:new1,items2:new2 -A K=item m=ref1.csv i=dat1.csv o=rsl3.csv
#END# kgvcommon -A K=item i=dat1.csv m=ref1.csv o=rsl3.csv vf=items1:new1,items2:new2
$ more rsl3.csv
items1,items2,new1,new2
b a c,b b,a c,
c c,a d,c c,a
e a a,a a,e a a,a a
```

関連コマンド

mvjoin：選択でなくベクトル要素を結合する。

4.78 mvcount ベクトルサイズの計算

ベクトルのサイズ (ベクトルの要素数) を計算する。
典型的な例を表 4.39〜??に示す。

表 4.39 入力データ

in.csv

no	items
1	a b c
2	a d
3	b f e f
4	

表 4.40 基本例

mvcount vf=items:size i=in.csv

no	items	size
1	a b c	3
2	a d	2
3	b f e f	4
4		0

書式

mvcount vf= [i=] [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf= 要素数をカウントするベクトルの項目名 (i=ファイル上) を指定する。
結果項目を:に続けて複数項目指定可能。
複数項目指定可能。

利用例

例 1: 複数項目に対して適用する例

```
$ more dat1.csv
items1,items2
b a c,b
c c,
e a a,a a a
$ mvcount vf=items1:size1,items2:size2 i=dat1.csv o=rsl1.csv
#END# kgvcount i=dat1.csv o=rsl1.csv vf=items1:size1,items2:size2
$ more rsl1.csv
items1,items2,size1,size2
b a c,b,3,1
c c,,2,0
e a a,a a a,3,3
```

関連コマンド

4.79　mvdelim ベクトル要素の区切り文字変更

ベクトル型の要素を区切る文字列を変更する。ただし、区切り文字に空文字を指定すれば区切り文字を削除することになる。

典型的な例を表 4.41〜4.45 に示す。カンマも指定することはできるが、ダブルクォーテーションで括られ 一つの項目となる (表 4.43)。v=のように値を指定しなければ空文字と見なされ、結果として区切り文字が消去される (表 4.44)。文字列や漢字も指定することは可能であるが (表 4.45)、ベクトルを扱うコマンド (mvsort など) で指定できる区切り文字 (delim=によって指定) はアルファベット 1 文字であるため、もはやベクトルとして利用することはできなくなることに注意する。

表 4.41　入力データ

in.csv

no	items
1	b a a
2	a a b b
3	a b b a
4	a b c

表 4.42　基本例:区切りをマイナス記号に置換

vf=items v=- i=in.csv

no	items
1	b-a-a
2	a-a-b-b
3	a-b-b-a
4	a-b-c

表 4.43　区切り文字にカンマを指定すると。。

vf=items v=, i=in.csv

no	items
1	”b,a,a”
2	”a,a,b,b”
3	”a,b,b,a”
4	”a,b,c”

表 4.44　空文字を指定すると区切りが消える

vf=items v= i=in.csv

no	items
1	baa
2	aabb
3	abba
4	abc

表 4.45　複数文字も指定できるが。。

vf=items v=xx i=in.csv

no	items
1	bxxaxxa
2	axxaxxbxxb
3	axxbxxbxxa
4	axxbxxc

書式

mvdelim vf= v= [-A]　[i=] [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=　区切り文字を変更するベクトル項目名を指定する。複数項目指定可能。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。
v=　新しい区切り文字を指定する。何も指定しなければ空文字として扱われる。
-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。

利用例

例 1: 基本例

ベクトル型要素のデフォルトの区切り文字である半角スペースを_(アンダーバー) に置換する。

```
$ more dat1.csv
item1
b a c
c c
e a a
$ mvdelim vf=item1 v=_ i=dat1.csv o=rsl1.csv
#END# kgvdelim i=dat1.csv o=rsl1.csv v=_ vf=item1
$ more rsl1.csv
item1
b_a_c
c_c
e_a_a
```

例 2: **カンマ**

CSV の区切り文字であるカンマに置換すると、CSV の区切り文字との区別を付けるために、ベクトル全体がダブルクオーツで囲われる。

```
$ mvdelim vf=item1 v=, i=dat1.csv o=rsl2.csv
#END# kgvdelim i=dat1.csv o=rsl2.csv v=, vf=item1
$ more rsl2.csv
item1
"b,a,c"
"c,c"
"e,a,a"
```

関連コマンド

4.80　mvdelnull ベクトルの NULL 要素の削除

ベクトル要素で NULL の要素を全て削除する。ベクトル要素が NULL であれば、要素の区切り文字が連続する。以下に示したベクトルは全て NULL を含む。ただし、わかりやすさのためにベクトルの末尾に‘\n’を記している。上から順番に、3 番目、1 番目、4 番目の要素が NULL である。

```
a b  c\n
 a b\n
a b c \n
```

書式

mvdelnull vf= [-A] i= [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=　NULL 要素を削除する対象となる項目名 (i=ファイル上) を指定する。
　　複数項目指定可能。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。
-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。

利用例

例 1: null を削除する基本例

```
$ more dat1.csv
items
b a  c
 c c
e a   b
$ mvdelnull vf=items i=dat1.csv o=rsl1.csv
#END# kgvdelnull i=dat1.csv o=rsl1.csv vf=items
$ more rsl1.csv
items
b a c
c c
e a b
```

例 2: 分かりやすく区切り文字を.(ドット) にした例

```
$ more dat2.csv
items
b.a..c
.c.c
e.a...b.
$ mvdelnull vf=items delim=. i=dat2.csv o=rsl2.csv
#END# kgvdelnull delim=. i=dat2.csv o=rsl2.csv vf=items
$ more rsl2.csv
items
```

```
b.a.c
c.c
e.a.b
```

例 3: 項目名を変更して出力

```
$ mvdelnull vf=items:new i=dat1.csv o=rsl3.csv
#END# kgvdelnull i=dat1.csv o=rsl3.csv vf=items:new
$ more rsl3.csv
new
b a c
c c
e a b
```

例 4: -A を指定することで追加項目として出力

```
$ mvdelnull vf=items:new -A i=dat1.csv o=rsl4.csv
#END# kgvdelnull -A i=dat1.csv o=rsl4.csv vf=items:new
$ more rsl4.csv
items,new
b a  c,b a c
 c c,c c
e a   b ,e a b
```

関連コマンド

mvnullto：NULL 要素を任意の値に置換する。

4.81　mvjoin ベクトル要素の参照結合

ベクトル要素をキーにして参照ファイル上のベクトル型データを結合する。ベクトル型の項目とは、表 4.46 の items 項目のように、スペースで区切られた複数の文字列を値として持つ項目である。

典型的な例を表 4.46〜4.49 に示す。

表 4.46　入力データ

in.csv

no	items
1	a b c
2	a d
3	b f e f
4	f c d

表 4.47　参照ファイル

ref.csv

item	taxo
a	X
b	Y
c	Z
e	X
f	Z

表 4.48　基本例

`vf=items m=ref.csv K=item f=taxo`

no	items
1	a b c X Y Z
2	a d X
3	b f e f Y Z X Z
4	f c d Z Z

表 4.49　アンマッチ要素の指定例

`vf=items m=ref.csv K=item f=taxo n=*`

no	items
1	a b c X Y Z
2	a d X *
3	b f e f Y Z X Z
4	f c d Z Z *

mvjoin コマンドは、参照ファイルデータを一旦全てメモリにセットするので、巨大な参照ファイルを指定した場合はメモリを使い果たす可能性があることに注意する。

書式

mvjoin vf= [-A] K= f= [n=] m=| i= [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=　結合キーとなるベクトルの項目名 (i=ファイル上) を指定する。
　　複数項目指定可能。ベクトル要素はソーティングされている必要はない。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。

-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。

m=　参照ファイルを指定する。

K=　参照ファイル (m=) 上の結合キーとなるベクトル要素の項目名を指定する。
　　並べ変わっている必要はないが、ベクトル要素は単一化されていなければならない。
　　単一化されていない時の動作は不定である。

f=　結合するベクトル (要素) 項目名を指定する。

n=　vf=と K=のベクトル要素がマッチしなかった場合に結合する文字列を指定する。
　　省略した場合、対象のベクトル (要素) の結合は行われない。

利用例

例 1: **ベクトルを結合する例**

```
$ more dat1.csv
items
b a c
c c
e a a
$ more ref1.csv
item,taxo
a,X Y
b,X
c,Z Z
$ mvjoin vf=items K=item m=ref1.csv f=taxo i=dat1.csv o=rsl1.csv
#END# kgVjoin K=item f=taxo i=dat1.csv m=ref1.csv o=rsl1.csv vf=items
$ more rsl1.csv
items
b a c X X Y Z Z
c c Z Z Z Z
e a a X Y X Y
```

例 2: **複数項目に対して適用する例**

```
$ more dat2.csv
items1,items2
b a c,b b
c c,a d
e a a,a a
$ more ref2.csv
item,taxo
a,X
b,X
c,Y
d,Y
$ mvjoin vf=items1,items2 K=item m=ref2.csv f=taxo i=dat2.csv o=rsl2.csv
#END# kgVjoin K=item f=taxo i=dat2.csv m=ref2.csv o=rsl2.csv vf=items1,items2
$ more rsl2.csv
items1,items2
b a c X X Y,b b X X
c c Y Y,a d X Y
e a a X X,a a X X
```

関連コマンド

mvcommon：結合ではなく、要素を選択するだけならこのコマンドを利用する。

4.82　mvnullto ベクトル要素の NULL 置換

ベクトル要素で NULL の要素を任意の値に置換する。ベクトル要素が NULL であれば、要素の区切り文字が連続する。以下に示したベクトルは全て NULL を含む。ただし、わかりやすさのためにベクトルの末尾に'\n' を記している。上から順番に、3 番目、1 番目、4 番目の要素が NULL である。

```
a b  c\n
 a b\n
a b c \n
```

書式

mvnullto vf= [v=|-p] [O=] [-A] i= [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=	NULL 置換の対象となる項目名 (i=ファイル上) を指定する。 複数項目指定可能。 結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。
-A	vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。 なお-A オプションを指定した場合、vf=パラメータで指定するすべての 項目に新項目名を指定しなければならない。
v=	置換文字列を指定する。
-p	直前の要素で置換する。v=と同時に指定はできない。
O=	NULL 値以外の要素を全て、ここで指定した文字列に置換する。 指定しなければ NULL 値以外は置換しない。

利用例

例 1: null を文字列 'null' に置換する例

```
$ more dat1.csv
items
b a  c
 c c
e a   b
$ mvnullto vf=items v=null i=dat1.csv o=rsl1.csv
#END# kgvnullto i=dat1.csv o=rsl1.csv v=null vf=items
$ more rsl1.csv
items
b a null c
null c c
e a null null b null
```

例 2: 分かりやすく区切り文字を.(ドット) にした例

```
$ more dat2.csv
items
b.a..c
.c.c
e.a...b.
```

```
$ mvnullto vf=items v=null delim=. i=dat2.csv o=rsl2.csv
#END# kgvnullto delim=. i=dat2.csv o=rsl2.csv v=null vf=items
$ more rsl2.csv
items
b.a.null.c
null.c.c
e.a.null.null.b.null
```

例 3: null を直前の値に置換する例

```
$ mvnullto vf=items -p i=dat1.csv o=rsl3.csv
#END# kgvnullto -p i=dat1.csv o=rsl3.csv vf=items
$ more rsl3.csv
items
b a a c
 c c
e a a a b b
```

例 4: O=を指定することで、null 以外は全て指定の値に置換される

```
$ mvnullto vf=items v=null O=X i=dat1.csv o=rsl4.csv
#END# kgvnullto O=X i=dat1.csv o=rsl4.csv v=null vf=items
$ more rsl4.csv
items
X X null X
null X X
X X null null X null
```

関連コマンド

mvdelnull：NULL 要素を削除する。

4.83　mvreplace ベクトル要素の参照置換

ベクトル要素をキーにして参照ファイル上のベクトル型データで置換する。ベクトル型の項目とは、表 4.50 の items 項目のように、スペースで区切られた複数の文字列を値として持つ項目である。

典型的な例を表 4.50〜4.53 に示す。

表 4.50　入力データ

in.csv

no	items
1	a b c
2	a d
3	b f e f
4	f c d

表 4.51　参照ファイル

ref.csv

item	taxo
a	X
b	Y
c	Z
e	X
f	Z

表 4.52　基本例

```
vf=items m=ref.csv K=item f=taxo
```

no	items
1	X Y Z
2	X d
3	Y Z X Z
4	Z Z d

表 4.53　アンマッチ要素の指定例

```
vf=items m=ref.csv K=item f=taxo n=*
```

no	items
1	X Y Z
2	X *
3	Y Z X Z
4	Z Z *

mvreplace コマンドは、参照ファイルデータを一旦全てメモリにセットするので、巨大な参照ファイルを指定した場合はメモリを使い果たす可能性があることに注意する。

書式

mvreplace vf= K= f= [n=] m=| [-A]　i= [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf=　結合キーとなるベクトルの項目名 (i=ファイル上) を指定する。
　　複数項目指定可能。ベクトル要素はソーティングされている必要はない。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。

-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。

m=　参照ファイルを指定する。

K=　参照ファイル (m=) 上の結合キーとなるベクトル要素の項目名を指定する。
　　並べ変わっている必要はないが、ベクトル要素は単一化されていなければならない。
　　単一化されていない時の動作は不定である。

f=　結合するベクトル (要素) 項目名を指定する。

n=　vf=と K=のベクトル要素がマッチしなかった場合に結合する文字列を指定する。
　　省略した場合、対象のベクトル (要素) の結合は行われない。

利用例

例 1: **ベクトルで置換する例**

```
$ more dat1.csv
items
b a c
c c
e a a
$ more ref1.csv
item,taxo
a,X Y
b,X
c,Z Z
$ mvreplace vf=items K=item m=ref1.csv f=taxo i=dat1.csv o=rsl1.csv
#END# kgvreplace K=item f=taxo i=dat1.csv m=ref1.csv o=rsl1.csv vf=items
$ more rsl1.csv
items
X X Y Z Z
Z Z Z Z
e X Y X Y
```

例 2: **複数項目に対して適用する例**

```
$ more dat2.csv
items1,items2
b a c,b b
c c,a d
e a a,a a
$ more ref2.csv
item,taxo
a,X
b,X
c,Y
d,Y
$ mvreplace vf=items1,items2 K=item m=ref2.csv f=taxo i=dat2.csv o=rsl2.csv
#END# kgvreplace K=item f=taxo i=dat2.csv m=ref2.csv o=rsl2.csv vf=items1,items2
$ more rsl2.csv
items1,items2
X X Y,X X
Y Y,X Y
e X X,X X
```

関連コマンド

mvjoin：要素の置換ではなく、結合であれば `mvjoin` を使う。

4.84 mvsort ベクトル要素のソート

ベクトル型の項目をソートする。ベクトル型の項目とは、表 4.54 の items 項目のように、スペースで区切られた複数の文字列を値として持つ項目である。

典型的な例を表 4.54〜4.57 に示す。オプションに何も指定しなければ文字列昇順に並べられ (表 4.55)、項目名の後ろに % に続けて n を付けることで数値順に並べられる (表 4.56)。ただし、ベクトルにアルファベットや漢字が含まれる場合の動作は不定である。また項目名の後ろに r を指定することで逆順に並べることもできる (表 4.57)。

表 4.54 入力データ

in.csv

no	items
1	2 1 13
2	4 5 2 5
3	112 14
4	5 31

表 4.55 基本的な利用:ベクトルの要素を文字列昇順に並べる

vf=items

no	items
1	1 13 2
2	2 4 5 5
3	112 14
4	31 5

表 4.56 数値昇順で並べる

vf=items%n

no	items
1	1 2 13
2	2 4 5 5
3	14 112
4	5 31

表 4.57 数値降順として並べる

vf=items%nr

no	items
1	13 2 1
2	5 5 4 2
3	112 14
4	31 5

書式

mvsort vf= [-A] [i=] [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

パラメータ

vf= ソーティングするベクトル項目名を指定する。複数項目指定可能。
% に続けて n を指定すれば数値ソートに、
% に続けて r を指定すれば逆順ソートに、
また、n と r の両方を指定すれば数値逆順ソートとなる。
結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。

-A vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
なお-A オプションを指定した場合、vf=パラメータで指定するすべての
項目に新項目名を指定しなければならない。

利用例

例 1: 複数項目を並べる例

item1 項目を文字列降順に並べ、item2 項目を数値昇順に並べる。

```
$ more dat1.csv
items1,items2
b a c,10 2
c c,2 5 3
e a a,1
$ mvsort vf=items1%r,items2%n i=dat1.csv o=rsl1.csv
#END# kgvsort i=dat1.csv o=rsl1.csv vf=items1%r,items2%n
$ more rsl1.csv
items1,items2
c b a,2 10
```

```
c c,2 3 5
e a a,1
```

関連コマンド

4.85　mvuniq ベクトル要素の単一化

ベクトル型の項目の要素を単一化する。内部的には tree 構造を利用して単一化をしているので、出力される要素の順序は文字列昇順に並び変わる。

一方で、-n オプションを指定すると、ベクトルを系列として考え、要素を先頭から順番に走査し、互いに隣接した要素のみを単一化し出力する。

典型的な例を表 4.59,4.60 に示す。表 4.59 では、全ての要素が単一化されているのが分かる。一方で、-n オプションを指定して実行すると、表 4.60 の 3 行目に見られるように、互いに隣接する b のみが単一化される。

表 4.58　入力データ

in.csv

no	items
1	b a a
2	a a b b b
3	a b b a
4	a b c

表 4.59　基本的な例

vf=items i=in.csv

no	items
1	a b
2	a b
3	a b
4	a b c

表 4.60　ベクトルを系列と考え、互いに隣り合う同じ要素のみを単一化する例

vf=items -n i=in.csv

no	items
1	b a
2	a b
3	a b a
4	a b c

書式

mvuniq vf= [-A] [-n] [i=] [o=] [delim=] [-assert_diffSize] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]

vf=　単一化する対象項目名を指定する。複数項目指定可能。
　　結果の項目名を変更したいときは、:(コロン) に続けて新項目名を指定する。
-A　vf=で:(コロン) に続けて指定した項目名で、新たな項目が追加される。
　　なお-A オプションを指定した場合、vf=パラメータで指定するすべての
　　項目に新項目名を指定しなければならない。
-n　ベクトルを系列と考え隣接する同一要素のみ単一化する

利用例

例 1: 複数項目を単一化する例

```
$ more dat1.csv
items1,items2
b a c,1 1
c c,2 2 3
e a a,3 1
$ mvuniq vf=items1,items2 i=dat1.csv o=rsl1.csv
#END# kgvuniq i=dat1.csv o=rsl1.csv vf=items1,items2
$ more rsl1.csv
items1,items2
a b c,1
c,2 3
a e,1 3
```

関連コマンド

4.86 mwindow スライド窓の生成

複数行をずらしながら複製していく。移動平均の計算など、時系列データにおいて一定幅の窓を設定し、その窓をずらしながらその窓を単位に何らかの処理 (例えば平均) をする目的に利用する。このような窓をスライド窓 (sliding window) と呼ぶ。

典型的な例を表 4.61〜4.63 に示す。

表 4.61 入力データ

date	val
4/6	1
4/7	2
4/8	3
4/9	4

表 4.62 wk=date:win t=2

win	date	val
4/7	4/6	1
4/7	4/7	2
4/8	4/7	2
4/8	4/8	3
4/9	4/8	3
4/9	4/9	4

表 4.63 wk=date:win t=2 -r

win	date	val
4/6	4/6	1
4/6	4/7	2
4/7	4/7	2
4/7	4/8	3
4/8	4/8	3
4/8	4/9	4

表 4.61 に示される入力データは日別の集計値が 4 日分示されており、スーパーの売上推移や株価推移と考えればよい。この入力データについて、2 日間を窓サイズとして移動平均を計算することを考える。入力データに示される日付 4/6〜4/9 についてサイズ 2 の窓を作成すると [(4/6,1),(4/7,2)], [(4/7,2),(4/8,3)], [(4/8,3),(4/9,4)] の 3 つの窓が作成される。ここで '[]' は一つの窓を示し、'()' は行を示すものとする。そしてこれらの窓のユニークキー (以下「窓キー」と呼ぶ) として、各窓の日付の最大値 (`wk=`で指定した項目の最終行の値) を win という項目名で出力する (表 4.62)。窓キーを各窓の最小値 (先頭行) とするには`-r` オプションを用いればよい (表 4.63)。あとは、出力結果 (表 4.62) に対して `mavg` を実行することで移動平均が計算される。ちなみに `mmvavg` コマンドは、上述の一連の処理 (`mwindow`+`mavg`) と同様の処理を行うが、`mmvavg` の方が高速である (約 3.5 倍速:200MB,1000 万件データで窓サイズを 10 で実験した結果)。

書式

`mwindow wk= t= [k=key] [-r] [-n] [i=] [o=]` [-assert_diffSize] [-assert_nullkey] [-nfn] [-nfno] [-x] [-q] [tmpPath=] `[--help] [--helpl] [--version]`

パラメータ

`wk=` 出力データにおいて、窓をユニークに識別する値となる入力データ上の項目を指定する。
ここで指定した項目で並べ替えられたのちスライド窓を生成していくが、
降順で並べ替えるときは %r、数値として並べ替えるときは %n と追加する。
数値の降順で並べ替えるときは %nr と追加すればよい。
またコロンに続いて窓キーの項目名を指定しなければならない。複数項目を指定することもできる。

`t=` 窓のサイズ (行数) を指定する。

`k=` ここで指定された項目の値を単位に窓の生成を行う。

`-r` 窓における基準行を先頭行とする。指定がなければ最終行となる。

`-n` 窓のサイズが `t=`で指定した行数に満たなくても出力する。

`i=` 入力ファイル名

`-nfn` 入力データの 1 行目を項目名行とみなさない。

利用例

例 1: **基本例**

```
$ more dat1.csv
date,val
20130406,1
20130407,2
20130408,3
20130409,4
$ mwindow wk=date:win t=2 i=dat1.csv o=rsl1.csv
#END# kgwindow i=dat1.csv o=rsl1.csv t=2 wk=date:win
$ more rsl1.csv
win%0,date,val
20130407,20130406,1
20130407,20130407,2
20130408,20130407,2
20130408,20130408,3
20130409,20130408,3
20130409,20130409,4
```

例 2: **基準行を先頭にした例**

```
$ mwindow wk=date:win t=3 -r i=dat1.csv o=rsl2.csv
#END# kgwindow -r i=dat1.csv o=rsl2.csv t=3 wk=date:win
$ more rsl2.csv
win%0,date,val
20130406,20130406,1
20130406,20130407,2
20130406,20130408,3
20130407,20130407,2
20130407,20130408,3
20130407,20130409,4
```

例 3: **指定行数に満たない窓も出力する例**

```
$ mwindow wk=date:win t=3 -r -n i=dat1.csv o=rsl3.csv
#END# kgwindow -n -r i=dat1.csv o=rsl3.csv t=3 wk=date:win
$ more rsl3.csv
win%0,date,val
20130406,20130406,1
20130406,20130407,2
20130406,20130408,3
20130407,20130407,2
20130407,20130408,3
20130407,20130409,4
20130408,20130408,3
20130408,20130409,4
20130409,20130409,4
```

例 4: **キー項目を指定した例**

```
$ more dat2.csv
store,date,val
a,20130406,1
a,20130407,2
a,20130408,3
a,20130409,4
b,20130406,11
b,20130407,12
```

```
b,20130408,13
b,20130409,14
$ mwindow k=store wk=date:win t=2 i=dat2.csv o=rsl4.csv
#END# kgwindow i=dat2.csv k=store o=rsl4.csv t=2 wk=date:win
$ more rsl4.csv
win%1,store%0,date,val
20130407,a,20130406,1
20130407,a,20130407,2
20130408,a,20130407,2
20130408,a,20130408,3
20130409,a,20130408,3
20130409,a,20130409,4
20130407,b,20130406,11
20130407,b,20130407,12
20130408,b,20130407,12
20130408,b,20130408,13
20130409,b,20130408,13
20130409,b,20130409,14
```

例 5: **前日までの移動平均を求める**

冒頭に示した移動平均の例では、窓における最終日を基準として平均を計算している。時に、基準日を前日として移動平均を計算したいケースがある。そういった場合は mslide で 1 日日付をずらしてから本コマンドを使えばよい。その例を以下に示す。

```
$ mslide f=date:date2 s=date i=dat1.csv o=rsl5.csv
#END# kgslide f=date:date2 i=dat1.csv o=rsl5.csv s=date
$ more rsl5.csv
date%0,val,date2
20130406,1,20130407
20130407,2,20130408
20130408,3,20130409
$ mwindow wk=date2:win t=2 i=rsl5.csv o=rsl6.csv
#END# kgwindow i=rsl5.csv o=rsl6.csv t=2 wk=date2:win
$ more rsl6.csv
win%0,date,val,date2
20130408,20130406,1,20130407
20130408,20130407,2,20130408
20130409,20130407,2,20130408
20130409,20130408,3,20130409
```

関連コマンド

mmvavg：移動窓の平均 (移動平均) に特化した計算コマンド。
mmvstats：移動窓の各種統計量を計算する。

4.87　mxml2csv xml から csv への変換

xml 形式のデータを csv 形式のデータに変換する。基本的な変換規則は、一行の単位となる要素 (XML タグ) と、項目に対応させる要素 (もしくは属性) を、それぞれ k=、f=によって指定する。項目の値としては、要素で囲われたテキスト、要素が出現するかどうか、属性の値、属性が出現するかどうか、の 4 通りの方法を指定できる。

XML のパーサーとしては SAX を用いているので、XML のサイズ制約はない。XML のエンコーディングが UTF-8 以外であれば、UTF-8 に変換されて CSV に出力される。XML データは、整形式 XML 文書 (well-formed XML document) であることを前提としている。そうでない場合の処理結果は不定である。

典型例を、表 4.64 に示された XML データを用いて示す。より詳細な解説は、次節で行うとして、ここでは概略について解説する。表 4.65 は、行の単位を要素とし (このような要素のことを「キー要素」と呼ぶ)、項目として、要素 b の属性 att の値 (CSV 項目名 b_att)、要素 c の属性 p の値 (b_p) とフラグ (b_p_f)、そして要素 d と a 内のテキスト (d と a) を出力した結果である。ここで、フラグとは、指定した要素もしくは属性が、出現するかどうかを 0-1 で出力したものである。要素のテキスト出力は、指定した要素で囲われた範囲に出現する文字列を全て連結した文字列である。ただし、スペースと制御文字は出力されないことに注意する。

表 4.64　入力 XML データ

```
<a att="aa">
  <b att="bb1">
    <c p="pp1" q="qq1"/>
    <d>text1</d>
  </b>
  <b att="bb2">
    <c q="qq2"></c>
    <d>text2</d>
  </b>
  <b>
    <c p="pp3"/>
    <d>text3</d>
  </b>
</a>
```

表 4.65　キー:/a/b、項目:b@att,c@p,d,/a

b_att	c_p	c_p_f	d	a
bb1	pp1	1	text1	text1
bb2			text2	text1text2
	pp3	1	text3	text1text2text3

キー要素の指定方法

一行の単位となるキー要素 (k=パラメータで指定) は絶対パスで指定する。絶対パスは、ルート記号 ('/') から始めて、要素の階層を'/' 記号で区切って表される。本コマンドにおけるキー要素の役割は、キー要素の終了タグが出現した時に、以下の 2 点を実行することにある。

- 項目データを一行出力する。上記の例では、キー要素の終了タグが 3 回出現しており、その度に CSV データ一行が出力され改行される。
- 項目データを初期化する。ただし、キー要素の外側にある項目要素の出力データは初期化しない。表 4.65 の要素 a の出力において、テキストが連結されていくのは、キー要素の終了タグが出現した時であっても、要素/a がキー要素/a/b より外側にあり、出力データが初期化されないためである。

要素の出力項目指定

f=で要素を CSV 項目として出力する場合、以下に示すフォーマットに従う。

要素パス [% フラグ]:CSV 項目名

「項目名」は、出力される CSV の項目名であり、必ず指定しなければならない。

要素を項目として出力する方法は 2 通りある。一つは、指定の要素の開始タグと終了タグで囲われたテキストを出力する方法で、他方は、指定した要素が存在するかどうかを 0-1 値で出力する方法である。前者は、対象となる要素パスを指定し、後者は、加えてフラグ %f を付与する。また要素パスの指定方法は、絶対パスと相対パスの 2 通りある。相対パスは、k=で指定された要素からのパスを指定する。以下、表 4.64 の XML について、いくつかの要素パスの指定例を示す。

- k=/a/b において、f=:B と指定すると、相対パスは空でキー要素と同じ要素となる。CSV の項目名は B である。
- k=/a/b において、f=c:C と f=/a/b/c:C は同じ意味である。前者は相対パスによる指定、後者は絶対パスによる指定である。両者とも CSV 項目名は C である。
- f=d:D は要素 d で囲われたテキストを出力し、f=d%f:D は要素 d が存在するかどうかを出力する。CSV 項目名は D である。
- k=/a/b において f=/a:A とした場合、項目要素はキー要素の外側にあるため、a 要素内に含まれるテキストが次々と結合される。なぜならば、キー要素の終了タグが出現しても、項目要素の終了タグが出現しないために、その時点でデータがクリアされることがないためである。

属性の出力項目指定

f=で属性を CSV 項目として出力する場合、以下に示すフォーマットに従う。

要素パス@要素名 [% フラグ]:CSV 項目名

「項目名」は、出力される CSV の項目名であり、必ず指定しなければならない。

要素パスの指定は、要素の出力項目指定と同様である。そして、属性名は、要素パスに続けて@で連結して指定する。要素名の後に %f を付けることで、その要素が存在するかどうかを 0-1 値で出力する。

書式

```
mxml2csv k= f= [i=] [o=] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [--help] [--helpl] [--version]
```

パラメータ

k= 一行の単位となる要素をルートからのパス名として指定する。
パスはルート記号'/' から始まり、要素を'/' でつなげることで指定する。
例: /article/sentence/chunk

f= 項目として出力する要素もしくは属性をカンマで区切って指定する。
フォーマットは以下のとおり。
要素パス [% フラグ]:CSV 項目名
要素パス@要素名 [% フラグ]:CSV 項目名

i= xml データファイル名を指定する。省略時は標準入力から読み込まれる。

利用例

例 1: 基本例

概要にて解説した例。/a/b をキー要素として、5 つの CSV 項目を出力する。

```
$ more dat1.xml
<a att="aa">
  <b att="bb1">
    <c p="pp1" q="qq1"/>
    <d>text1</d>
  </b>
  <b att="bb2">
    <c q="qq2"></c>
    <d>text2</d>
  </b>
  <b>
    <c p="pp3"/>
    <d>text3</d>
  </b>
</a>
$ mxml2csv k=/a/b f=@att:b_att,c@p:c_p,c@p%f:c_p_f,d:d,/a:a i=dat1.xml  o=rsl1.csv
#END# kgxml2csv f=@att:b_att,c@p:c_p,c@p%f:c_p_f,d:d,/a:a i=dat1.xml k=/a/b o=rsl1.csv
$ more rsl1.csv
b_att,c_p,c_p_f,d,a
bb1,pp1,1,text1,text1
bb2,,,text2,text1text2
,pp3,1,text3,text1text2text3
```

例 2: **絶対パス**

基本例と同じ要素を絶対パスで指定する例。/a/b をキー要素として、5 つの CSV 項目を出力する。

```
$ mxml2csv k=/a/b f=/a/b@att:b_att,/a/b/c@p:c_p,/a/b/c@p%f:c_p_f,/a/b/d:d,/a:a i=dat1.xml  o=rsl2.csv
#END# kgxml2csv f=/a/b@att:b_att,/a/b/c@p:c_p,/a/b/c@p%f:c_p_f,/a/b/d:d,/a:a i=dat1.xml k=/a/b o=rsl2.csv
$ more rsl2.csv
b_att,c_p,c_p_f,d,a
bb1,pp1,1,text1,text1
bb2,,,text2,text1text2
,pp3,1,text3,text1text2text3
```

例 3: **キー要素の変更**

絶対パスの例でキー要素を a に変更した例。a の終了タグは一つしかないので、一行だけ出力されている。f=で指定した/a/b@att は、2 回出現しているが、最後に出現した値が出力されている。

```
$ mxml2csv k=/a f=/a/b@att:b_att,/a/b/c@p:c_p,/a/b/c@p%f:c_p_f,/a/b/d:d,/a:a i=dat1.xml o=rsl3.csv
#END# kgxml2csv f=/a/b@att:b_att,/a/b/c@p:c_p,/a/b/c@p%f:c_p_f,/a/b/d:d,/a:a i=dat1.xml k=/a o=rsl3.csv
$ more rsl3.csv
b_att,c_p,c_p_f,d,a
bb2,pp3,1,text3,text1text2text3
```

関連コマンド

第 5 章

mcal

mcal コマンドは、項目間の演算を行うことを主目的に開発されたコマンドである。mcal 以外のコマンドは行志向の処理を行い、mcal が項目志向の処理を担当すると考えればよい。mcal には 100 以上の関数/演算子が実装されており、それらを組み合わせることで多様な処理を実現できるようになっている。

5.1 mcal 項目間演算

c=パラメータで指定した計算式で計算をおこない、a=パラメータで指定した項目名に出力する。mcal が出力する項目は、プログラムの単純化のため、例外なく 1 つに限定している。計算式の詳細は後述の「式の構成要素」を参照のこと。

書式

mcal a= c= [i=] [o=] [-assert_diffSize] [-assert_nullkey] [-assert_nullin] [-assert_nullout] [-nfn] [-nfno] [-x] [-q] [tmpPath=] [precision=] [--help] [--helpl] [--version]

パラメータ

a= 新たに計算結果の出力として追加される項目の名前を指定する。
c= 用意された関数を組み合わせて計算する式を指定する。

利用例

以下に mcal を使った簡単な例を示す。個々の関数や演算子についての詳細は、それぞれ個別に解説を用意しているので、そちらを参照されたい。

```
# 入力ファイル (dat1.csv)
顧客,数量,単価
A,3,10
B,1,15
C,2,20

$ mcal c='${数量}*${単価}' a=金額 i=dat1.csv
顧客,数量,単価,金額
A,3,10,30
B,1,15,15
C,2,20,40

$ mcal c='${数量}*${単価}<=30' a=金額30以下 i=dat1.csv
顧客,数量,単価,金額30以下
A,3,10,1
B,1,15,1
C,2,20,0

$ mcal c='if(top(),${単価},#{}+${単価})' a=単価累計 i=dat1.csv
顧客,数量,単価,単価累計
A,3,10,10
B,1,15,25
C,2,20,45
```

シェルでの利用時の注意事項

UNIX 系の OS で bash などのシェルを利用する場合、演算子などの記号はシェルにとっても特殊な意味を持つことが多い。例えば$記号に続く単語列はシェル変数を意味する。一方で mcmd では項目値を参照するために$を使っている。そこでシェルにそれらの記号を解釈させないために、以下のようにシングルクオーツで囲む必要がある。

```
$ mcal c='${date}-10'
```

エラーメッセージ

```
#ERROR# unknown function or operator
```

このエラーメッセージが出るということは、関数もしくは演算子の指定に誤りがある。例えば、以下のような文字列の結合関数 cat についてのエラーメッセージについて考える。

```
$ mcal c='cat("-",1,2)'
ERROR : unknown function or operator: cat_SNN(cat_SN) (kgcal)
```

「cat_SNN」のアンダーバーの前の cat は関数名を示し、その後の SNN は引数の型を示している。S は文字列型、N は数字型、D は日付型、T は時刻型、B は真偽型である。3 つの引数を指定しているので 3 文字 (SNN) となっている。すなわち、このエラーメッセージは「引数として SNN をとる cat という名前の関数」は登録されていないということを意味する。以下のように 2,3 番目の引数を文字列型にすればよい。

```
$ mcal c='cat("-","1","2")'
```

ここで、エラーメッセージの括弧の中は 2 文字 (SN) となっているが、それは 2 番目以降のパラメータが可変個指定可能であるため、それらの代表として 1 つだけ表記しているためである。

関連コマンド

msel：演算の結果を用いて行選択するのであればこのコマンドを使う。

5.2 式の構成要素

mcal で記述する式を構成する要素は、定数、項目値、演算子、関数の 4 つに大きく分類できる。そして、それぞれの要素において、ユーザはデータ型を意識して利用する必要がある。mcal が扱うデータは CSV テキストなので、値は全て文字列として与えらており、それら文字列を mcal でどのようなデータ型として扱うかはユーザに委ねられているからである。mcal が規定するデータ型は文字列型 (*str*), 数値型 (*num*)、日付型 (*date*)、時刻型 (*time*)、論理型 (*bool*) の 5 つである。以下では、式を構成するそれぞれの要素で、5 つのデータ型をどのように扱うかを示す。

5.3 定数

表 5.1 定数の書式一覧

データ型	書式	内容	例
数値 (*num*)	整数、実数の文字列表記	内部的には全て倍精度実数を利用	`20, 0.55, 1.5*e10`
文字列 (*str*)	”文字列”	ダブルクオーテーションで括った文字列	`"abc" "日本語"`
日付 (*date*)	0dyyyymmdd	先頭に”0d”を付けた年月日固定長	`0d20080923`
時刻 (*time*)	0tyyyymmddHHMMSS	先頭に”0t”を付けた年月日時分秒固定長	`0t20080923121115`
	0tHHMMSS	先頭に”0t”を付けた時分秒固定長 (内部的に本日の日付が補完される)	`0t121115`
論理 (*bool*)	0b1, 0b0	先頭に”0b”を付けた”1”(true) もしくは”0”(false)	`0b1, 0b0`

5.4 項目値

CSV データ上の項目名を指定するには表 5.2 に示される通り、CSV データをどのデータ型として扱うかによって異なってくる。CSV データは全て文字列データであるために、それらをどのような型のデータとして扱うかはユーザの

判断に任されている。

表 5.2 項目値の書式一覧

データ型	書式	CSV データ内容	例
数値 (*num*)	${ 項目名 }	整数、実数 (指数表現含む) の文字列表記	`${amount}, ${株価}`
文字列 (*str*)	$s{ 項目名 }	文字列	`$s{gender}, $s{性別}`
日付 (*date*)	$d{ 項目名 }	年月日固定長 (yyyymmdd)	`$d{date}, $d{発注日}`
時刻 (*time*)	$t{ 項目名 }	年月日時分秒の固定長 (yyyymmddHHMMSS) 時分秒 (HHMMSS) の固定長 (内部的に本日の日付が補完される)	`$d{time}, $d{出発時刻}`
論理 (*bool*)	$b{ 項目名 }	項目値の一文字目が”1”の時に true、”0”の時に false、 その他の場合には NULL と解釈される。	`$b{condition}, $b{条件}`

5.5 ワイルドカード

項目名にはワイルドカードを指定することができる。例えば sum 関数は複数の数値項目の合計を計算する関数であるが、ワイルドカードを指定することで、多数の項目を一つのワイルドカードで指定することも可能となる。例えば、入力データとして `A1,A2,A3` の 3 つの数値項目名があったとすると、`sum(${A*})` とすれば、`A1,A2,A3` の合計値を計算してくれる。もちろん `sum(${A*},${B*})` のように複数のワイルドカードを指定することも可能である。

5.6 前行の項目値

項目値の指定に`$`の代わりに`#`を指定すると、前行の項目値となる。ただし、先頭行は前行がないので NULL となる。各データ型における指定方法を 5.3 に示す。

表 5.3 前行の項目値の書式一覧

データ型	書式	例
数値 (*num*)	#{ 項目名 }	`#{amount}, #{株価}`
文字列 (*str*)	#s{ 項目名 }	`#s{gender}, #s{性別}`
日付 (*date*)	#d{ 項目名 }	`#d{date}, #d{発注日}`
時刻 (*time*)	#t{ 項目名 }	`#d{time}, #d{出発時刻}`
論理 (*bool*)	#b{ 項目名 }	`#b{condition}, #b{条件}`

5.7 前行の結果項目値

前行項目の指定において項目名を省略すると前行の計算結果項目の値となる。各データ型における指定方法を 5.4 に示す。

if 関数と top() 関数とを組み合わせる事で、累計計算などが可能となる。以下に、金額項目の累計計算例を示す。

```
$ mcal c='if(top(),${金額},${金額}+#{})' a=累計金額
```

表 5.4 前行の結果項目値の書式一覧

データ型	書式	例
数値 (*num*)	#{}	`#{}`
文字列 (*str*)	#s{}	`#s{}`
日付 (*date*)	#d{}	`#d{}`
時刻 (*time*)	#t{}	`#d{}`
論理 (*bool*)	#b{}	`#b{}`

5.8 算術演算子

+ や-などの算術演算子は数値型だけでなく、文字列型や日付型のデータに対しても定義されている。それらの一覧を表 5.5 に示す。

表 5.5 算術演算子一覧

演算子	書式	内容	例
加算 (+)	$num_1 + num_2$	数値の足し算	`1.5+2.3 (3.8)`
	$str_1 + str_2$	文字列の結合	`"150"+"円" ("150 円")`
	$date + num$	num 日後の日付	`0d20121130+2 (0d20121202)`
	$time + num$	num 秒後の時刻	`0t095959+2 (0t100001)`
減算 (-)	$num_1 - num_2$	数値の引き算	`1.5-2.3 (-1.8)`
	$str_1 - str_2$	部分文字列の削除	`"aababa"-"a" ("bb")`
		(貪欲マッチによる)	`"aababa"-"aba" ("aba")`
	$date - num$	num 日前の日付	`0d20121202-2 (0d20121130)`
	$time - num$	num 秒前の時刻	`0t100001-2 (0t095959)`
	$date_1 - date_2$	日付差	`0d20121202-0d20121130 (2)`
	$time_1 - time_2$	時刻差	`0t095959-0t100001 (-2)`
乗算 (*)	$num_1 * num_2$	掛け算	`10*2 (20)`
除算 (/)	num_1/num_2	割り算	`10/2 (5)`
剰余 (%)	$num_1\%num_2$	剰余	`10%3 (1)`
累乗 (^)	$num_1\hat{}num_2$	累乗	`10^3 (1000)`

例の括弧内は結果を表す (内容は定数の表記で示している)。

5.9 比較演算子

比較演算は同一のデータ型の値同士でのみ適用可能である。全てのデータ型に共通した書式であり、以下では数値型についてのみ (例では文字型につても) 表 5.6 に示す。文字型、日付型、時刻型においても同様に利用できる。

表 5.6 比較演算子一覧

比較内容	書式	例
等しい	$num_1 == num_2$	`1.5==1.5(0b1), "abc"=="abcd" (0b0)`
等しくない	$num_1! = num_2$	`1.5!=1.5(0b0), "abc"=="abcd" (0b1)`
より大きい	$num_1 > num_2$	`10>5(0b1), "abc">"abcd" (0b0)`
より小さい	$num_1 < num_2$	`10<5(0b0), "abc"<"abcd" (0b1)`
以上	$num_1 >= num_2$	`10>=10(0b1), "a">"" (0b1)`
以下	$num_1 <= num_2$	`8<=9(0b1), "a"<="a" (0b1)`

例の括弧内は結果を表す (内容は定数の表記で示している)。

5.10 論理演算子

3 つの論理演算子 (論理積、論理和、排他的論理和) が利用でき、それぞれの書式を表 5.7 に示す。また、それぞれの演算における真偽 (1:真,0:偽) の組み合せとその結果を表 5.8, 表 5.9, 表 5.10 に示す。

5.11 演算子の優先順位

演算子 (後述) の優先順位は表 5.11 に示すとおりである。同一の演算子間の優先順位は出現順序による。優先順位を変更するときは括弧を利用すれば良い。

5.12 関数

以下では、数値関連 (5.12)、三角関数関連 (5.13)、文字列関連 (5.14)、正規表現関連 (5.15)、日付時間関連 (5.16)、論理関数 (5.17)、行/項目情報関連 (5.18)、NULL 値関連 (5.19)、そして型変換関連 (5.20) の 9 つに分けて解説する。

5.13 日付型と時刻型について

mcal では日付時刻について 2 つの型を用意している。一つは日付型で他方は時刻型である。時刻型は時刻だけでなく日付とセットで表現する。内部的にはグレゴリオ暦に基づいた boost C++ ライブラリの date_time ライブラリを利用しており、日付型には boost::gregorian::date クラスを、時刻型には boost::posix_time::ptime クラスを使っている。詳細は boost.org のドキュメントを参照されたい。

date クラスは 32 ビット整数で管理されており、1400 年 1 月 1 日から 9999 年 12 月 31 日の範囲をサポートしている。日付の演算は全てグレゴリオ暦に基づいたものとなっている。不正な日付 (例えば、2013/2/29 や 1399/12/31) が与えられたときは NULL 値が出力される。

一方で ptime クラスは、64 ビットで管理されており、ミリ秒まで扱える時刻システムであるが、mcal コマンドにおいてはミリ秒を扱うインターフェースは備えていない。また ptime クラスは date クラスも内部で参照しており、日付をまたいだ時間計算を可能としている。不正な時刻 (例えば、18:62:11) が与えられたときは NULL 値が出力される。

MCMD は CSV テキストを扱うので、日付/時刻は、データ上は文字列で表現される必要がある。それらの文字列を日付型および時刻型に内部で変換して各種演算を行い、最終結果を再度文字列に戻して出力している。文字列のフォーマットは、日付型は 8 桁固定長文字列 (例えば、”20130911”)、時刻型は 14 桁固定長文字列 (例え

表 5.7 論理演算子一覧

内容	書式	例
論理積	$bool_1$&&$bool_2$	`"abc"=="abc" && "xyz"=="abc" (0b0)`
論理和	$bool_1$||$bool_2$	`"abc"=="abc" || "xyz"=="abc" (0b1)`
排他的論理和	$bool_1$ ^^ $bool_2$	`"abc"=="abc" ^^ "xyz"=="abc" (0b1)`

例の括弧内は結果を表す (内容は定数の表記で示している)。

表 5.8 論理積

$bool_1$	$bool_2$	結果
1	1	1
1	0	0
0	1	0
0	0	0
null	1	null
null	0	0
null	null	null

表 5.9 論理和

$bool_1$	$bool_2$	結果
1	1	1
1	0	1
0	1	1
0	0	0
null	1	1
null	0	null
null	null	null

表 5.10 排他的論理和

$bool_1$	$bool_2$	結果
1	1	0
1	0	1
0	1	1
0	0	0
null	1	null
null	0	null
null	null	null

表 5.11 演算子の優先順位

優先順位	演算子
1	`*,/,%,^`
2	`+,-`
3	`>,<,>=,<=`
4	`== ,!=`
5	`&&`
6	`||,^^`

ば、”20130911110528”)、もしくは 6 桁固定長文字列 (例えば、”110528”) を標準としている。

日付型と時刻型と各種関数の関係を図 5.1 に示す。

表 5.12　数値関連関数一覧

節	関数名	機能	出力型
5.98	sum($num_1, num_2, \cdots$)	合計	*num*
5.22	avg($num_1, num_2, \cdots$)	平均	*num*
5.97	sqsum($num_1, num_2, \cdots$)	平方和	*num*
5.68	min($num_1, num_2, \cdots$)	最小値	*num*
5.66	max($num_1, num_2, \cdots$)	最大値	*num*
5.78	product($num_1, num_2, \cdots$)	積	*num*
5.42	factorial(*num*)	階乗	*num*
5.48	gcd(num_1,num_2)	最大公約数	*num*
5.57	lcm(num_1,num_2)	最小公倍数	*num*
5.96	sqrt(*num*)	平方根	*num*
5.14	abs(*num*)	絶対値	*num*
5.93	sign(*num*)	符号	*num*
5.53	int(*num*)	整数部	*num*
5.47	fract(*num*)	小数部	*num*
5.91	round(*num*, 基準値)	四捨五入	*num*
5.45	floor(*num*, 基準値)	切り捨て	*num*
5.28	ceil(*num*, 基準値)	切り上げ	*num*
5.77	power(*num*, 指数)	累乗	*num*
5.41	exp(*num*)	指数関数	*num*
5.63	log(*num*, 底)	対数	*num*
5.62	ln(*num*)	自然対数	*num*
5.65	log2(*num*)	底が 2 の対数	*num*
5.64	log10(*num*)	常用対数	*num*
5.37	dist(タイプ,$num_1, num_2, \cdots$)	距離	*num*
5.38	distgps(緯度 1, 経度 1, 緯度 2, 経度 2)	GPS 距離	*num*
5.50	heron($num_1, num_2, \cdots$)	三角形の面積	*num*
5.80	rand([乱数の種])	一様乱数	*num*
5.81	randi(最小値, 最大値 [, 乱数の種])	整数一様乱数	*num*
5.74	nrand(平均, 標準偏差 [, 乱数の種])	正規乱数	*num*
5.76	pi()	円周率	*num*
5.40	e()	ネイピア数	*num*
5.46	format()	書式付き出力	*str*

表 5.13　三角関数関連関数一覧

節	関数名	機能	出力範囲
5.15	acos(*num*)	コサインの逆関数	$0 \sim \pi$
5.19	asin(*num*)	サインの逆関数	$-\pi \sim \pi$
5.20	atan(*num*)	タンジェントの逆関数	$-\pi \sim \pi$
5.21	atan2(num_1,num_2)	座標 (num_1, num_2) の角度	$-\pi \sim \pi$
5.29	cos(r)	コサイン	$-1.0 \sim 1.0$
5.94	sin(r)	サイン	$-1.0 \sim 1.0$
5.100	tan(r)	タンジェント	$-\infty \sim \infty$
5.35	degree(r)	角度	$-\pi \sim \pi$
5.79	radian(角度)	度数を入力したときのラジアンを出力	$-\pi \sim \pi$
5.30	cosh(r)	双曲線余弦	$0 \sim \infty$
5.95	sinh(r)	双曲線正弦	$-\infty \sim \infty$
5.101	tanh(r)	双曲線逆正接	$-1.0 \sim 1.0$

r はラジアンを表した数値。

またユーザは日付/時刻として固定長文字列を標準とせずに、ユリウス通日 (紀元前 4713 年 1 月 1 日正午からの日数) や UNIX 時刻 (1970 年 1 月 1 日 00 時 00 分 00 秒 (GMT) からの経過秒数) などの整数を標準の日時の表記として利用してもよいであろう。ユリウス通日や UNIX 時刻と、日付型/時刻型との変換関数も備えており、十分に運用可能である。ただし、mcal が提供する日付/時刻関数を使う限りにおいては、内部的にはグレゴリオ暦によって管理されて

表 5.14 文字列関連関数一覧

節	関数名	機能	出力型
5.27	cat(*token*, $str_1, str_2, \cdots$)	文字列併合	*str*
5.60	length(*str*)	文字列長	*num*
5.43	fixlen(*str*, 長さ, 位置, padding 文字)	固定長変換	*str*
5.90	right(*str*, 長さ)	末尾切り出し	*str*
5.59	left(*str*, 長さ)	先頭切り出し	*str*
5.67	mid(*str*, 開始位置, 長さ)	部分文字列切り出し	*str*
5.106	toupper(*str*)	小文字大文字変更	*str*
5.104	tolower(*str*)	大文字小文字変更	*str*
5.26	capitalize(*str*)	先頭文字大文字変換	*str*
5.54	match(検索文字列,$str_1, str_2, \cdots$)	検索	*bool*
5.49	hasspace(*str*)	空白類文字検索	*bool*

表 5.15 正規表現関連関数一覧

節	関数名	機能	出力型
5.83	regexm(*str*, 正規表現)	全体マッチ	*bool*
5.87	regexs(*str*, 正規表現)	マッチ	*bool*
5.86	regexrep(*str*, 正規表現, 置換文字列)	マッチ文字列の置換	*str*
5.82	regexlen(*str*, 正規表現)	マッチ文字数	*num*
5.85	regexpos(*str*, 正規表現)	開始位置	*num*
5.89	regexstr(*str*, 正規表現)	マッチ文字列	*str*
5.84	regexpfx(*str*, 正規表現)	マッチ文字列のプレフィックス	*str*
5.88	regexsfx(*str*, 正規表現)	マッチ文字列のサフィックス	*str*

表 5.16 日付時間関連関数一覧

節	関数名	機能	出力型
5.103	today()	本日の日付	*date*
5.72	now()	現在時刻	*time*
5.107	tseconds(*time*)	経過秒数	*num*
5.58	leapyear(*dt*)	閏年判定	*bool*
5.111	year(*dt*)	西暦年	*num*
5.70	month(*dt*)	月	*num*
5.33	day(*dt*)	日	*num*
5.109	week(*dt*)	週番号	*num*
5.39	dow(*dt*)	曜日	*num*
5.102	time(*time*)	時分秒	*str*
5.34	date(*time*)	年月日	*str*
5.51	hour(*time*)	時	*num*
5.69	minute(*time*)	分	*num*
5.92	second(*time*)	秒	*num*
5.16	age(dt_1, dt_2)	年令	*num*
5.36	diff(dt_1, dt_2)	期間	*num*
5.108	uxt(*dt*)	UNIX 時変換	*num*(UNIX 時刻)
5.56	julian(*dt*)	ユリウス通日変換	*num*(ユリウス通日)

dt は、*date*,*time* の何れかを表す。

おり、その範囲は、1400 年 1 月 1 日から 9999 年 12 月 31 日に限定されることに注意する。また UNIX 時刻は 32 ビット整数で管理されているため、2038 年 1 月 19 日 3 時 14 分 7 秒を超えると正しく計算できないことに注意する。ただユリウス通日や UNIX 時刻を利用する欠点は、その数字を見ただけでは実際にいつの日付時刻なのか理解出来ない点

表 5.17　論理関連関数一覧

節	関数名	機能	出力型
5.17	and($bool_1, bool_2, \cdots$)	論理積	*bool*
5.75	or($bool_1, bool_2, \cdots$)	論理和	*bool*
5.71	not($bool$)	否定	*bool*
5.52	if($bool, num_1, num_2$)	条件選択	*num*
5.52	if($bool, str_1, str_2$)		*str*
5.52	if($bool, date_1, date_2$)		*date*
5.52	if($bool, time_1, time_2$)		*time*

表 5.18　行/項目情報関連関数一覧

節	関数名	機能	出力型
5.61	line()	現在処理中の行番号を返す	*num*
5.105	top()	先頭行	*bool*
5.25	bottom()	終端行	*bool*
5.44	fldsize()	項目数	*num*
5.18	argsize($str_1, str_2, \cdots$)	引数の数	*num*

表 5.19　NULL 値関連関数一覧

節	関数名	機能	出力型
5.73	nulln()	NULL 値	*num*
5.73	nulls()		*str*
5.73	nulld()		*date*
5.73	nullt()		*time*
5.73	nullb()		*bool*
5.55	isnull(num)	NULL 値判定	*bool*
5.55	isnull(str)		*bool*
5.55	isnull($date$)		*bool*
5.55	isnull($time$)		*bool*
5.55	isnull($bool$)		*bool*
5.31	countnull($num_1, num_2, \cdots$)	NULL 値の数	*num*
5.31	countnull($str_1, str_2, \cdots$)		*num*
5.31	countnull($date_1, date_2, \cdots$)		*num*
5.31	countnull($time_1, time_2, \cdots$)		*num*
5.31	countnull($bool_1, bool_2, \cdots$)		*num*

表 5.20　型変換関連関数一覧

5.112	*num*	*str*	*date*	*time*	*bool*
num		n2s(num)			n2b(num)
str	s2n(str)		s2d(str)	s2t(str)	s2b(str)
date		d2s($date$)		d2t($date$)	
time		t2s($time$)	t2d($time$)		
bool	b2n($bool$)	b2s($bool$)			

各セルの関数は、「行ラベル → 列ラベル」の変換関数を示している。

空白セルは、そのような変換関数が定義されていないことを意味する。

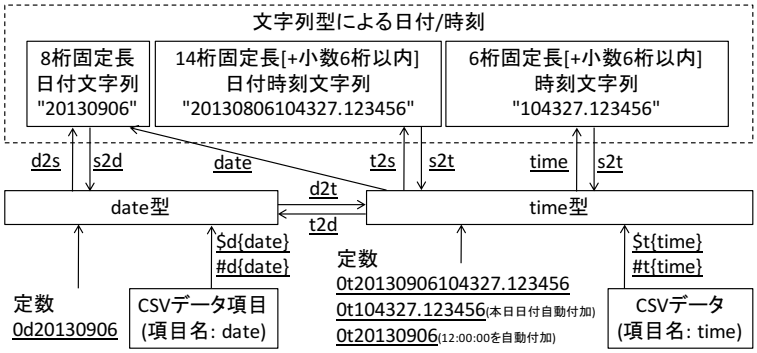


図 5.1 2013 年 9 月 6 日 10 時 43 分 27 秒を例に、date 型と time 型と各種関数の関係を図示している。実線で囲われたボックスは実データを表し、アンダーラインを付したものは関数等を表している。

にあろう。

5.14 abs 絶対値

書式: abs(num)

num の絶対値を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,1.0
2,-2.5
3,
4,0
$ mcal c='abs(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=abs(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,1.0,1
2,-2.5,2.5
3,,
4,0,0
```

5.15 acos コサインの逆関数

書式: acos(num)

アークコサイン (コサインの逆関数) を計算する。パラメータの範囲は $-1.0 \sim 1.0$ で戻り値の範囲は $0.0 \sim \pi$ である。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,-1.0
2,0
3,1.0
4,
5,2
$ mcal c='acos(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=acos(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,-1.0,3.141592654
2,0,1.570796327
3,1.0,0
4,,
5,2,
```

5.16 age 年令

書式 1: age(生年月日,*date*)

書式 2: age(生年月日,*time*)

日付 *date*(書式 1) もしくは時刻 *time*(書式 2) における年令を返す。生年月日は書式 1,2 によって日付型/時刻型を指定する。

利用例

例 1: **基本例**

2013 年 9 月 1 日における年令を求める。

```
$ more dat1.csv
id,dob
1,19641010
2,20000101
3,
4,19770812
$ mcal c='age($d{dob},0d20130901)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=age($d{dob},0d20130901) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,dob,rsl
1,19641010,48
2,20000101,13
3,,
4,19770812,36
```

5.17 and 論理積

書式: and($bool_1, bool_2, \cdots$)

$bool_i$ で与えられた真偽値全ての論理積を計算する。NULL 値を含めた真偽値表は表 5.8 を参照のこと。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,b1,b2,b3
1,1,0,1
2,1,1,1
3,1,,1
4,1,1,1
$ mcal c='and($b{b1},$b{b2},$b{b3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=and($b{b1},$b{b2},$b{b3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,b1,b2,b3,rsl
1,1,0,1,0
2,1,1,1,1
3,1,,1,
4,1,1,1,1
```

例 2: **ワイルドカードを利用した例**

b から始まる項目 (b1,b2,b3) をワイルドカード「b*」によって指定している。

```
$ mcal c='and($b{b*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=and($b{b*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,b1,b2,b3,rsl
1,1,0,1,0
2,1,1,1,1
3,1,,1,
4,1,1,1,1
```

5.18 argsize 引数の数

書式 1: argsize($str_1, str_2, \cdots$)

str_i で与えられた文字列の数を返す。ワイルドカードをで項目名を指定することで意味が出てくる。すなわちワイルドカード条件を満たす多数の項目の数をカウントすることができる。データ型としては文字列型のみに対応していることに注意する。

利用例

例 1: **基本例**

"v"で始まる項目名の数を数える。

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='argsize($s{v*})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=argsize($s{v*}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,3
2,-5,2,1,3
3,1,,3,3
4,,,,3
```

5.19 asin サインの逆関数

書式: asin(num)

アークサイン (サインの逆関数) を計算する。パラメータの範囲は $-1.0 \sim 1.0$ で戻り値の範囲は $-\pi/2 \sim \pi/2$ である。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,-1.0
2,0
3,1.0
4,
5,2
$ mcal c='asin(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=asin(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,-1.0,-1.570796327
2,0,0
3,1.0,1.570796327
4,,
5,2,
```

5.20 atan タンジェントの逆関数

書式: atan(num)

アークタンジェント (タンジェントの逆関数) を計算する。パラメータの範囲は $-\infty \sim \infty$ で戻り値の範囲は $-\pi/2 \sim \pi/2$ である。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,-1.0
2,0
3,1.0
4,
5,1.0e+10
$ mcal c='atan(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=atan(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,-1.0,-0.7853981634
2,0,0
3,1.0,0.7853981634
4,,
5,1.0e+10,1.570796327
```

5.21 atan2 座標の角度

書式: atan2(num_1, num_2)

x, y 座標 (num_1, num_2) と原点を結ぶ線分と x 軸とが作る角度をラジアンで返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,x,y
1,5,10
2,10,20
3,-1,0
4,0,0
5,,
$ mcal c='atan2(${x},${y})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=atan2(${x},${y}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,x,y,rsl
1,5,10,1.107148718
2,10,20,1.107148718
3,-1,0,3.141592654
4,0,0,0
5,,,
```

5.22 avg 平均

書式 1: avg($num_1, num_2, \cdots$)

書式 2: avg($num_1, num_2, \cdots, str$)

num_i で与えられた数値の平均を計算する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: 基本例

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='avg(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=avg(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,2
2,-5,2,1,-0.6666666667
3,1,,3,2
4,,,,
```

例 2: ワイルドカードを利用した例

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='avg(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=avg(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,2
2,-5,2,1,-0.6666666667
3,1,,3,2
4,,,,
```

例 3: -n を利用した例

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='avg(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=avg(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,2
2,-5,2,1,-0.6666666667
3,1,,3,
4,,,,
```

5.23 binomdist 二項分布の累積確率

書式: binomdist(成功回数 k, 試行回数 n, 成功確率 p)

試行回数 n、成功確率 p の二項分布 $B(n,p)$ に従う確率変数 X について、累積確率 $P[X \leq k]$ を計算する。k,n,p は、定数としても項目として与えることができる。なお、$k > n$ や $p < 0, p > 1$ の場合の計算結果は null となる。

利用例

例 1: **基本例**

```
$ mcal c='binomdist(${k},${n},${p})' a=prob i=dat1.csv o=rsl1.csv
#ERROR# calculation error  (kgcal)
$ more rsl1.csv
k,n,p,prob
0,2,0.5,0.25
1,2,0.5,0.75
2,2,0.5,1
4,10,0.2,0.9672065024
40,100,0.2,0.9999987078
```

5.24 berrand ベルヌーイ乱数

書式: berrand(確率 [, 乱数の種])

ベルヌーイ分布 $f(x) = p^x(1-p)^{1-x} x = 0, 1$ について、指定された確率 p における x の乱数を生成する。本関数では、$x = 0$ を false、$x = 1$ を true とした論理型として値を返す。また確率 p は定数としてのみ指定可能で、項目名を指定することはできない。

同じ乱数の種を指定すれば、同じ乱数系列となる。指定可能な乱数の種の範囲は-2147483648〜2147483647 である。乱数の種を省略すると、時刻 (1/1000 秒単位) に応じた異なる乱数の種が利用される。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

利用例

例 1: **基本例**

確率 $p = 0.2$ のベルヌーイ乱数を生成する。乱数の種を指定しているので、何度実行しても同じ乱数系列が生成される。

```
$ mnewnumber l=10 a=num |
$ mcal c='berrand(0.2,111)' a=rand o=rsl1.csv
#END# kgNewnumber a=num l=10
#END# kgcal a=rand c=berrand(0.2,111) o=rsl1.csv
$ more rsl1.csv
num,rand
0,0
1,1
2,1
3,0
4,0
5,0
6,0
7,0
8,0
9,0
```

5.25 bottom 終端行

書式: bottom()

終端行であれば真を、そうでなければ偽を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
val
1
2
3
4
$ mcal c='bottom()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=bottom() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
val,rsl
1,0
2,0
3,0
4,1
```

5.26 capitalize 先頭文字大文字変換

書式: capitalize(*str*)

文字列の先頭文字のみ大文字に変更する。アルファベット 26 文字以外の文字については何の影響もない。

利用例

例 1: **基本例**

str 項目の先頭文字を大文字に変換する。

```
$ more dat1.csv
id,str
1,abc
2,aBd
3,
4,#abc
$ mcal c='capitalize($s{str})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=capitalize($s{str}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abc,Abc
2,aBd,ABd
3,,
4,#abc,#abc
```

5.27 cat 文字列併合

書式: cat($token, str_1, str_2, \cdots$)

$token$ を区切り文字として、指定された str_i をその順番に併合して一つの文字列を生成する。$token$ として空文字""を指定すれば、文字列の単純な結合となる。

利用例

例 1: 基本例

3 つの項目 str1,str2,str3 を"#"の区切り文字を挿入して併合する。

```
$ more dat1.csv
id,str1,str2,str3
1,abc,def,ghi
2,A,,CDE
3,,,
4,,,XY
$ mcal c='cat("#",$s{str1},$s{str2},$s{str3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=cat("#",$s{str1},$s{str2},$s{str3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str1,str2,str3,rsl
1,abc,def,ghi,abc#def#ghi
2,A,,CDE,A##CDE
3,,,,##
4,,,XY,##XY
```

例 2: token が空文字の場合

```
$ mcal c='cat("",$s{str1},$s{str2},$s{str3})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=cat("",$s{str1},$s{str2},$s{str3}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,str1,str2,str3,rsl
1,abc,def,ghi,abcdefghi
2,A,,CDE,ACDE
3,,,,
4,,,XY,XY
```

例 3: ワイルドカードを利用した例

str から始まる項目 (str1,str2,str3) をワイルドカード「str*」によって指定している。

```
$ mcal c='cat("",$s{str*})' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=cat("",$s{str*}) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,str1,str2,str3,rsl
1,abc,def,ghi,abcdefghi
2,A,,CDE,ACDE
3,,,,
4,,,XY,XY
```

5.28 ceil 切り上げ

書式: ceil(*num*, 基数)

num を切り上げにより丸める。この時、基数の整数倍の値集合のうち、*num* より大きい最小の目盛点にまるめられる。例えば、ceil(3.42,0.5) の場合、0.5 とびに目盛がうたれた数直線上で、3.42 より大きい最小の目盛点、すなわち 3.5 が基数 0.5 における切り上げ点となる。基数を省略すると 1.0 がデフォルト値として用いられる。これは小数点以下 1 桁目を切り上げて整数値にまるめることに等しい。

利用例

例 1: **基本例**

小数点以下一桁目を切り上げる。

```
$ more dat1.csv
id,val
1,3.28
2,3.82
3,
4,-0.6
$ mcal c='ceil(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=ceil(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.28,4
2,3.82,4
3,,
4,-0.6,-0
```

例 2: **基本例**

小数点以下二桁目を切り上げる。

```
$ mcal c='ceil(${val},0.1)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=ceil(${val},0.1) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,3.28,3.3
2,3.82,3.9
3,,
4,-0.6,-0.5
```

例 3: **基数** 0.5 **の例**

0.5 を基数として切り上げる。

```
$ mcal c='ceil(${val},0.5)' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=ceil(${val},0.5) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,val,rsl
1,3.28,3.5
2,3.82,4
3,,
4,-0.6,-0.5
```

例 4: 基数 10 の例

一桁目を切り上げる。

```
$ more dat2.csv
id,val
1,1341.28
2,188
3,1.235E+3
4,-1.235E+3
$ mcal c='ceil(${val},10)' a=rsl i=dat2.csv o=rsl4.csv
#END# kgcal a=rsl c=ceil(${val},10) i=dat2.csv o=rsl4.csv
$ more rsl4.csv
id,val,rsl
1,1341.28,1350
2,188,190
3,1.235E+3,1240
4,-1.235E+3,-1230
```

5.29 cos コサイン

書式: $\cos(r)$

ラジアン r に対するコサインを計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,1.047197
3,
4,6.283185
$ mcal c='cos(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=cos(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,-1
2,1.047197,0.5000004774
3,,
4,6.283185,1
```

5.30 cosh 双曲線余弦

書式: cosh(r)

双曲線余弦 (ハイパーボリック コサイン) を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,-1.047197
3,
4,6.283185
$ mcal c='cosh(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=cosh(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,11.59194573
2,-1.047197,1.600286169
3,,
4,6.283185,267.7466792
```

5.31 countnull 合計

書式 1: countnull($num_1, num_2, \cdots$)

書式 2: countnull($str_1, str_2, \cdots$)

書式 3: countnull($date_1, date_2, \cdots$)

書式 4: countnull($time_1, time_2, \cdots$)

書式 5: countnull($bool_1, bool_2, \cdots$)

num_i(その他の型も同様) で与えられた数値の中で NULL 値の数を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
a,b,c,d
1,,3,4
1,,,
,,,
$ mcal c='countnull(${a},${b},${c},${d})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=countnull(${a},${b},${c},${d}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
a,b,c,d,rsl
1,,3,4,1
1,,,,3
,,,,4
```

例 2: **他の項目型として項目名をワイルドカードで指定**

```
$ mcal c='countnull($s{*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=countnull($s{*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
a,b,c,d,rsl
1,,3,4,1
1,,,,3
,,,,4
```

5.32 d2julian: date-ユリウス通日変更

書式: d2julian(*date*)
日付からユリウス通日を求める

ユリウス通日は紀元前 4713 年 1 月 1 日正午（世界標準時による）からの日数である。
ただしグリゴリオ歴は 1400-1-1〜9999-12-31 が有効範囲のため、
それに対応するユリウス通日の有効値範囲は 2232300〜5373484 となる。
その範囲外では NULL 出力となる。

利用方法

```
c=d2julian(0d20080822)
```

利用例

表 5.21 入力データ

日付	時間	数
20020824	20020824145408	10660
20020622	20020622173449	22740
20020824	20020824145408	14800
20021009	20021009095743	54510
20020121	20020121173449	18750

上記のデータを用いて、それぞれの場合の利用例を以下に示す。

実行例 1)
”日付”項目を入力としてユリウス通日に変換し、新たに”ユリウス通日”をいう項目を作成して出力している。

```
------------------------------------------------
mcal c='d2julian($d{日付})' a="ユリウス通日" i=date.csv o=od2julian.csv
------------------------------------------------
```

表 5.22 出力ファイル (od2julian.csv)

日付	時間	数	ユリウス通日
20020824	20020824145408	10660	2452511
20020622	20020622173449	22740	2452448
20020824	20020824145408	14800	2452511
20021009	20021009095743	54510	2452557
20020121	20020121173449	18750	2452296

mcal 日付時間関連へ戻る

5.33 day 日

書式 1: day(*date*)

書式 2: day(*time*)

書式 3: days(*date*)

書式 4: days(*time*)

日付 *date* もしくは時刻 *time* から日を取り出す。書式 1,2 では数値として、書式 3,4 では 2 桁固定長文字列として返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19770812
$ mcal c='day($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=day($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,1
2,20121021,21
3,,
4,19770812,12
```

例 2: **時刻型でも可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='day($t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=day($t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,1
2,20121021111213,21
3,,
4,19770812122212,12
```

5.34 date 年月日

書式: date(*time*)

時刻 *time* から年月日を 8 桁固定長文字列として取り出す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='date($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=date($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,20000101000000,20000101
2,20121021111213,20121021
3,,
4,19770812122212,19770812
```

5.35 degree 角度

書式: degree(r)

ラジアン r に対応する角度を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,1.047197
3,
4,6.283185
$ mcal c='degree(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=degree(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,179.9999626
2,1.047197,59.99996842
3,,
4,6.283185,359.9999824
```

5.36 diff 期間

書式 1: diffyear($date_1, date_2$)
書式 2: diffyear($time_1, time_2$)
書式 3: diffmonth($date_1, date_2$)
書式 4: diffmonth($time_1, time_2$)
書式 5: diffday($date_1, date_2$)
書式 6: diffday($time_1, time_2$)
書式 7: diffhour($time_1, time_2$)
書式 8: diffminute($time_1, time_2$)
書式 9: diffsecond($time_1, time_2$)
書式 10: diffusecond($time_1, time_2$)

二つの引数が日付型の場合、$date_1$ と $date_2$ の差 ($date_1 - date_2$) を計算し、その年数 (difyear)、月数 (diffmonth)、日数 (diffday) で計算する。二つの引数が時刻型の場合、$time_1$ と $time_2$ の差 ($time_1 - time_2$) を計算し、その年数 (difyear)、月数 (diffmonth)、日数 (diffday)、時間数 (diffhour)、分数 (diffminute)、秒数 (diffsecond)、マイクロ秒数 (diffusecond) で計算する。端数は切り捨てられる。

利用例

例 1: 月単位での期間

date 項目から 2013 年 9 月 1 日までの経過期間を日数で計算する。

```
$ more dat1.csv
id,date
1,19641010
2,20000101
3,20130831
4,20130901
5,20130902
$ mcal c='diffday(0d20130901,$d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=diffday(0d20130901,$d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,19641010,17858
2,20000101,4992
3,20130831,1
4,20130901,0
5,20130902,-1
```

例 2: 分単位での期間

time 項目から 2012 年 1 月 1 日 00 時 00 分 00 秒までの経過期間を分単位で計算する。

```
$ more dat2.csv
id,time
1,20120101000000
2,20120101011112
3,
4,20111231235000
5,20111231235000.123456
$ mcal c='diffmonth(0t20120101000000,$t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=diffmonth(0t20120101000000,$t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
```

```
1,20120101000000,0
2,20120101011112,-1
3,,
4,20111231235000,0
5,20111231235000.123456,0
```

例 3: **マイクロ秒単位での期間**

time 項目から 2012 年 1 月 1 日 00 時 00 分 00 秒までの経過期間を秒単位およびマイクロ秒単位で計算する。

```
$ mcal c='diffsecond(0t20120101000000,$t{time})' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=diffsecond(0t20120101000000,$t{time}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,time,rsl
1,20120101000000,0
2,20120101011112,-4272
3,,
4,20111231235000,600
5,20111231235000.123456,599
$ mcal c='diffusecond(0t20120101000000,$t{time})' a=rsl i=dat2.csv o=rsl4.csv
#END# kgcal a=rsl c=diffusecond(0t20120101000000,$t{time}) i=dat2.csv o=rsl4.csv
$ more rsl4.csv
id,time,rsl
1,20120101000000,0
2,20120101011112,-4272
3,,
4,20111231235000,600
5,20111231235000.123456,599.876544
```

5.37 dist 距離

書式: dist(タイプ,$num_1, num_2, \cdots, n_k, num_{k+1}, num_{k+2}, \cdots, num_{2k}$)

2 つの k 次元ベクトル $(num_1, num_2, \cdots, n_k), (num_{k+1}, num_{k+2}, \cdots, num_{2k})$ の距離を計算する。距離としては以下のものが利用できる。詳細な定義は msim を参照のこと。

- euclid: ユークリッド距離
- cityblock: 都市ブロック距離
- hamming: ハミング距離

ハミング距離については、値は文字型として指定しなければならない (以下の例を参考のこと)。

利用例

例 1: **ユークリッド距離**

```
$ more dat1.csv
id,x1,y1,x2,y2
1,0,0,1,1
2,0,1,2,0
3,,,,
$ mcal c='dist("euclid",${x1},${y1},${x2},${y2})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=dist("euclid",${x1},${y1},${x2},${y2}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,x1,y1,x2,y2,rsl
1,0,0,1,1,1.414213562
2,0,1,2,0,2.236067977
3,,,,,
```

例 2: **都市ブロック距離**

```
$ mcal c='dist("cityblock",${x1},${y1},${x2},${y2})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=dist("cityblock",${x1},${y1},${x2},${y2}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,x1,y1,x2,y2,rsl
1,0,0,1,1,2
2,0,1,2,0,3
3,,,,,
```

例 3: **ハミング距離**

ハミング距離の計算では、値を文字列として指定していることに注意する。

```
$ more dat2.csv
id,x1,y1,x2,y2
1,a,b,a,c
2,0,1,0,1
3,,,,
$ mcal c='dist("hamming",$s{x1},$s{y1},$s{x2},$s{y2})' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=dist("hamming",$s{x1},$s{y1},$s{x2},$s{y2}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,x1,y1,x2,y2,rsl
1,a,b,a,c,1
2,0,1,0,1,2
3,,,,,
```

5.38 distgps GPS 距離

書式: distgps(緯度 1, 経度 1, 緯度 2, 経度 2[, 方位])

2 地点の緯度経度座標からその直線距離 (km 単位) を求める。緯度経度は 10 進表記で与える。地球の球状を考慮した計算を行っている。緯度経度の指定は、60 進表記には対応しておらず 10 進表記に変換しておくことが必要である。60 進法で表記された度分秒をそれぞれ d, m, s とすると、10 進の座標は $d + m/60 + s/60/60$ で計算できる。

また方位については、北緯と東経はプラスで表記し、南緯と西経はマイナスで表記する。

例えば、大阪駅 (北緯 34.702398, 東経 135.495188), 東京駅間 (北緯 35.681391, 東経 139.766103) の距離を求めるには以下のとおり指定すればよい。

```
distgps(34.702398,135.495188,35.681391,139.766103)
```

またエベレスト (北緯 32.655556, 東経 79.015833), からアコンカグア (南緯 27.987778, 西経 86.944444) の距離は、以下のとおり指定すればよい。

```
distgps(32.655556,79.015833,-27.987778,-86.944444)
```

利用例

例 1: **基本例**

```
$ more dat1.csv
point1,point2,lat1,lon1,lat2,lon2
osaka,tenma,34.702398,135.495188,34.704923,135.512233
osaka,tokyo,34.702398,135.495188,35.681391,139.766103
osaka,kobe,34.702398,135.495188,34.679453,135.178221
osaka,Fuji,34.702398,135.495188,35.360556,138.727500
Evelest,Aconcagua,32.655556,79.015833,-27.987778,-86.944444
Denali,Kilimanjaro,63.069444,-151.007222,-3.075833,37.353333
$ mcal c='distgps(${lat1},${lon1},${lat2},${lon2})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=distgps(${lat1},${lon1},${lat2},${lon2}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
point1,point2,lat1,lon1,lat2,lon2,rsl
osaka,tenma,34.702398,135.495188,34.704923,135.512233,1.585046048
osaka,tokyo,34.702398,135.495188,35.681391,139.766103,405.774306
osaka,kobe,34.702398,135.495188,34.679453,135.178221,29.12042213
osaka,Fuji,34.702398,135.495188,35.360556,138.727500,304.7527532
Evelest,Aconcagua,32.655556,79.015833,-27.987778,-86.944444,16956.12242
Denali,Kilimanjaro,63.069444,-151.007222,-3.075833,37.353333,11362.37758
```

5.39 dow 曜日

書式 1: dow(*date*) 曜日番号 (1〜7)

書式 2: dow(*time*) 曜日番号 (1〜7)

書式 3: dowj(*date*) 日本語曜日

書式 4: dowj(*time*) 日本語曜日

書式 5: dowe(*date*) 英語曜日

書式 6: dowe(*time*) 英語曜日

書式 7: dowes(*date*) 英語短縮曜日

書式 8: dowes(*time*) 英語短縮曜日

日付 *date* もしくは時刻 *time* から曜日を返す。曜日の表記によって書式 1〜8 を使い分ける。曜日番号は、ISO8601 の規定に従い、1 が月曜日で 7 が日曜日に対応する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19770812
$ mcal c='dow($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=dow($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,6
2,20121021,7
3,,
4,19770812,5
```

例 2: **日本語表記**

```
$ mcal c='dowj($d{date})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=dowj($d{date}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,date,rsl
1,20000101,土
2,20121021,日
3,,
4,19770812,金
```

例 3: **英語表記**

```
$ mcal c='dowe($d{date})' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=dowe($d{date}) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,date,rsl
1,20000101,Saturday
2,20121021,Sunday
3,,
4,19770812,Friday
```

例 4: **英語短縮表記**

```
$ mcal c='dowes($d{date})' a=rsl i=dat1.csv o=rsl4.csv
#END# kgcal a=rsl c=dowes($d{date}) i=dat1.csv o=rsl4.csv
$ more rsl4.csv
id,date,rsl
1,20000101,Sat
2,20121021,Sun
3,,
4,19770812,Fri
```

例 5: **時刻型でも可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='dow($t{time})' a=rsl i=dat2.csv o=rsl5.csv
#END# kgcal a=rsl c=dow($t{time}) i=dat2.csv o=rsl5.csv
$ more rsl5.csv
id,time,rsl
1,20000101000000,6
2,20121021111213,7
3,,
4,19770812122212,5
```

5.40 e ネイピア数

書式: e()

ネイピア数 (e) を出力する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id
1
2
$ mcal c='e()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=e() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,2.718281828
2,2.718281828
```

5.41 exp 指数関数

書式: exp(num)

power 関数の特殊形で、e(ネイピア数) を底とする num 乗を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,exponent
1,1
2,-1
3,
4,0
5,0.5
$ mcal c='exp(${exponent})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=exp(${exponent}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,exponent,rsl
1,1,2.718281828
2,-1,0.3678794412
3,,
4,0,1
5,0.5,1.648721271
```

5.42 factorial 階乗

書式: factorial(num)

num の階乗を計算する。結果が実数の最大値を超えると、NULL 値が出力される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,1
2,5
3,
4,10000
$ mcal c='factorial(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=factorial(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,1,1
2,5,120
3,,
4,10000,
```

例 2: **定数を与える例**

5 の階乗を計算する。定数を与えているので、全行同じ結果が出力される。

```
$ mcal c='factorial(5)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=factorial(5) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,1,120
2,5,120
3,,120
4,10000,120
```

5.43 fixlen 固定長変換

書式 1: fixlen(*str*, 長さ, 位置,padding 文字)

書式 2: fixlenw(*str*, 長さ, 位置,padding 文字)

str を固定の長さの文字列に変換する。*str* が指定の長さに満たない場合は、右詰もしくは左詰めで指定した文字を埋め込む。左右の選択では、「位置」パラメータに"L"もしくは"R"を指定する。埋め込む文字は「padding 文字」パラメータに任意の文字を指定する。また、*str* の長さが、指定した固定長の長さを超えた場合、右詰の場合は先頭側が、左詰めの場合は末尾側の文字列が削られることに注意する。

マルチバイト文字を対象とした固定長変換については fixlenw 関数を利用する。

利用例

例 1: **基本例**

str 項目を 5 文字の固定長文字列に変換する。5 文字に満たない文字列は右詰 ("R") で"#"を埋める。

```
$ more dat1.csv
id,str
1,abc
2,123
3,
4,1234567
$ mcal c='fixlen($s{str},5,"R","#")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=fixlen($s{str},5,"R","#") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abc,##abc
2,123,##123
3,,#####
4,1234567,34567
```

例 2: **左詰め例**

左詰 ("L") で"#"を埋める。

```
$ mcal c='fixlen($s{str},5,"L","#")' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=fixlen($s{str},5,"L","#") i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,abc,abc##
2,123,123##
3,,#####
4,1234567,12345
```

例 3: **マルチバイト文字を含む場合**

```
$ more dat2.csv
id,str
1,こんにちは
2,大阪
$ mcal c='fixlenw($s{str},4,"R","埋")' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=fixlenw($s{str},4,"R","埋") i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,str,rsl
1,こんにちは,んにちは
2,大阪,埋埋大阪
```

5.44 fldsize 項目数

書式: fldsize()

入力ファイルの項目数を返す。mcmd では全行同じ項目数を想定しているので、この関数の結果は全行同じ値になる。

利用例

例 1: **基本例**

```
$ more dat1.csv
a,b,c,d
1,2,3,4
2,3,4,5
3,,,
4,x,y,z
$ mcal c='fldsize()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=fldsize() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
a,b,c,d,rsl
1,2,3,4,4
2,3,4,5,4
3,,,,4
4,x,y,z,4
```

5.45 floor 切り捨て

書式: floor(num, 基数)

num を切り捨てにより丸める。この時、基数の整数倍の値集合のうち num より小さい最大の目盛点にまるめられる。例えば、floor(3.82,0.5) の場合、0.5 とびに目盛がうたれた数直線上で、3.82 より小さい最大の目盛点、すなわち 3.5 が基数 0.5 における切り捨て点となる。基数を省略すると 1.0 がデフォルト値として用いられる。これは小数点以下 1 桁目を切り捨てて整数値にまるめることに等しい。

利用例

例 1: **基本例**

小数点以下一桁目を切り捨てる。

```
$ more dat1.csv
id,val
1,3.28
2,3.82
3,
4,-0.6
$ mcal c='floor(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=floor(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.28,3
2,3.82,3
3,,
4,-0.6,-1
```

例 2: **基本例**

小数点以下二桁目を切り捨てる。

```
$ mcal c='floor(${val},0.1)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=floor(${val},0.1) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,3.28,3.2
2,3.82,3.8
3,,
4,-0.6,-0.6
```

例 3: **基数** 0.5 **の例**

0.5 を基数として切り捨てる。

```
$ mcal c='floor(${val},0.5)' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=floor(${val},0.5) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,val,rsl
1,3.28,3
2,3.82,3.5
3,,
4,-0.6,-1
```

例 4: 基数 10 の例

一桁目を切り捨てる。

```
$ more dat2.csv
id,val
1,1341.28
2,188
3,1.235E+3
4,-1.235E+3
$ mcal c='floor(${val},10)' a=rsl i=dat2.csv o=rsl4.csv
#END# kgcal a=rsl c=floor(${val},10) i=dat2.csv o=rsl4.csv
$ more rsl4.csv
id,val,rsl
1,1341.28,1340
2,188,180
3,1.235E+3,1230
4,-1.235E+3,-1240
```

5.46 format 書式付き出力

書式: format(*num*, 書式)

mcal では数値を内部的には浮動小数点として管理している。それを C 言語の printf 関数で利用できる出力書式を指定する事で，数値の出力形式を変更する事ができる。利用できる書式は以下の 3 つである。

- %f: 小数形式表記
- %e,%E: 指数形式表記 (大文字 %E で指数記号を大文字表記)
- %g,%G: f,e の自動判断 (大文字 %G で指数記号を大文字表記)

例えば、小数形式表記の 0.00726 は、指数形式表記では 7.260000e-03 となり、1265 は、1.265000e+03 となる。

さらに、% の直後に数字もしくはプラス記号を指定することで、表示桁数と符号表記を制御できる。数字は「**全体の幅**. **小数点の幅**」によって指定する。例えば、123.456789 は、%5.2f の書式によって 123.46 に、%8.3f の書式によって 123.457 となる。

プラス符号を常に表記したければ数字の前にプラス記号を付加すればよい。例えば、123.456789 は、%+5.2f の書式によって +123.46 となる。

書式の中には上記で説明した記号以外にも、任意の文字列を含めることが可能である。例えば、250 は、書式"**合計** %g **円**"によって、"**合計** 250 **円**"となる。% という文字を表示したければ”%%”と記すれば良い。15 は、書式"%g%%"によって、15% となる。

利用例

例 1: **基本例**

val を実数として小数点以下 2 桁に変換する。

```
$ more dat1.csv
id,val
1,0.00726
2,123.456789
3,
4,-0.335
$ mcal c='format(${val},"%8.3f")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=format(${val},"%8.3f") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,0.00726,   0.007
2,123.456789, 123.457
3,,
4,-0.335,  -0.335
```

例 2: **指数表現**

val を指数表現で出力。

```
$ mcal c='format(${val},"%e")' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=format(${val},"%e") i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,0.00726,7.260000e-03
2,123.456789,1.234568e+02
3,,
4,-0.335,-3.350000e-01
```

5.47 fract 小数部

書式: fract(num)

num の小数部を返す。符号はそのまま出力される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.14
2,3
3,
4,-12.56789
5,1.2345e+2
6,1.2345e-10
$ mcal c='fract(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=fract(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.14,0.14
2,3,0
3,,
4,-12.56789,-0.56789
5,1.2345e+2,0.45
6,1.2345e-10,1.2345e-10
```

5.48 gcd 最大公約数

書式: gcd(num_1, num_2)

num_1 と num_2 の最大公約数を求める。実数は整数に変換して (切り下げ) 実行される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val1,val2
1,12,36
2,6,5
3,,
4,12.1,36.2
$ mcal c='gcd(${val1},${val2})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=gcd(${val1},${val2}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val1,val2,rsl
1,12,36,12
2,6,5,1
3,,,
4,12.1,36.2,12
```

例 2: **定数を与える例**

val1 項目と 36 の最大公約数を求める。

```
$ mcal c='gcd(${val1},36)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=gcd(${val1},36) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val1,val2,rsl
1,12,36,12
2,6,5,6
3,,,
4,12.1,36.2,12
```

5.49 hasspace 空白類文字検索

書式 1: hasspace(*str*, 長さ)

書式 2: hasspacew(*str*, 長さ)

文字列 *str* に空白類文字列が含まれていれば真を、含まれていなければ偽を返す。空白類文字とは、ASCII コードの 0x20 および 0x09〜0x0d までの事をいう。それぞれ、半角スペース (0x20)、水平 tab(0x09)、改行 (0x0a)、垂直タブ (0x0b)、改ページ (0x0c)、復帰 (0x0d) である。マルチバイト文字における空白類文字を検索したければ hasspacew を使うこと。

利用例

例 1: **基本例**

`str` 項目に空白類文字列が含まれていれば真を返す。`id=1` の行は半角スペース文字が含まれ、`id=2` の行は tab 文字が含まれ、そして `id=4` の行は改行文字が含まれているために真となっている。ここで、`id=3` の行は全角スペースのため、検知できていない。

```
$ more dat1.csv
id,str
1,a b
2,ab        c
3,ab   c
4,
5,"aa
bb"
$ mcal c='hasspace($s{str})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=hasspace($s{str}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,a b,1
2,ab        c,1
3,ab   c,0
4,,0
5,"aa
bb",1
```

例 2: **マルチバイト文字**

hasspacew 関数を使えば全角スペースも正しく検知できる。

```
$ mcal c='hasspacew($s{str})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=hasspacew($s{str}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,a b,1
2,ab        c,1
3,ab   c,1
4,,0
5,"aa
bb",1
```

5.50 heron 三角形の面積

書式: heron(タイプ,$num_1, num_2, \cdots, n_k, num_{k+1}, num_{k+2}, \cdots, num_{2k}, num_{2k+1}, num_{2k+2}, \cdots, num_{3k}$)

k 次元空間の 3 点の座標 $(num_1, num_2, \cdots, n_k), (num_{k+1}, num_{k+2}, \cdots, num_{2k}), (num_{2k+1}, num_{2k+2}, \cdots, num_{3k})$ によって作られる 3 角形の面積を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,x1,y1,x2,y2,x3,y3
1,0,0,1,0,0,1
2,0,0,0,2,2,0
4,,,,,,
3,0,0,1,1,2,2
$ mcal c='heron(${x1},${y1},${x2},${y2},${x3},${y3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=heron(${x1},${y1},${x2},${y2},${x3},${y3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,x1,y1,x2,y2,x3,y3,rsl
1,0,0,1,0,0,1,0.5
2,0,0,0,2,2,0,2
4,,,,,,,
3,0,0,1,1,2,2,0
```

5.51 hour 時

書式 1: hour(*time*) 数値

書式 2: hours(*time*) 2 桁固定長文字列

時刻 *time* から時刻を取り出す。返す型によって、書式 1,2 を使い分ければよい。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='hour($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=hour($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,20000101000000,0
2,20121021111213,11
3,,
4,19770812122212,12
```

例 2: **文字列とし出力**

```
$ mcal c='hours($t{time})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=hours($t{time}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,00
2,20121021111213,11
3,,
4,19770812122212,12
```

5.52 if 条件選択

書式: if($bool, num_1, num_2$), if($bool, str_1, str_2$), if($bool, date_1, date_2$),if($bool, time_1, time_2$),if($bool, bool_1, bool_2$)

第 1 パラメータの値が真であれば第 2 パラメータを、偽であれば第 3 パラメータを返す。第 1 パラメータが NULL 値であれば NULL 値を返す。第 2 パラメータと第 3 パラメータは同じ型でなくてはならないことに注意する。

利用例

例 1: **基本例**

time 項目が数値として 120000 以下であれば”AM”、そうでなければ”PM”を出力する。

```
$ more dat1.csv
id,time
1,101215
2,210110
3,
4,120001
$ mcal c='if(${time}<=120000,"AM","PM")' a=ampm i=dat1.csv o=rsl1.csv
#END# kgcal a=ampm c=if(${time}<=120000,"AM","PM") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,ampm
1,101215,AM
2,210110,PM
3,,
4,120001,PM
```

例 2: **異なる型を指定した場合**

第 2 パラメータと第 3 パラメータに異なる型を指定するとエラーとなる。

```
$ mcal c='if(${time}<=120000,"am",1)' a=ampm i=dat1.csv o=rsl2.csv
#ERROR# unknown function or operator: if_BSN (kgcal)
$ more rsl2.csv
```

例 3: **真偽値による条件選択**

$b{項目名}によって、データ上の値”1”は真、”0”は偽、そしてその他の値は NULL として解釈される。

```
$ more dat2.csv
id,val
1,1
2,0
3,
4,-2
$ mcal c='if($b{val},"true","false")' a=bool i=dat2.csv o=rsl3.csv
#END# kgcal a=bool c=if($b{val},"true","false") i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,val,bool
1,1,true
2,0,false
3,,
4,-2,
```

例 4: **時刻型として比較**

```
$ mcal c='if($t{time}<=0t120000,"am","pm")' a=ampm i=dat1.csv o=rsl4.csv
#END# kgcal a=ampm c=if($t{time}<=0t120000,"am","pm") i=dat1.csv o=rsl4.csv
$ more rsl4.csv
id,time,ampm
1,101215,am
2,210110,pm
3,,
4,120001,pm
```

例 5: if **関数のネスト**

```
$ more dat3.csv
id,val
1,10
2,0
3,-5
4,0
$ mcal c='if(${val}>0,"plus",if(${val}<0,"minus","zero"))' a=sign i=dat3.csv o=rsl5.csv
#END# kgcal a=sign c=if(${val}>0,"plus",if(${val}<0,"minus","zero")) i=dat3.csv o=rsl5.csv
$ more rsl5.csv
id,val,sign
1,10,plus
2,0,zero
3,-5,minus
4,0,zero
```

5.53 int 整数部

書式: int(num)

num の整数部を返す。符号はそのまま出力される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.14
2,3
3,
4,-12.56789
5,1.2345e+2
6,1.2345e-10
$ mcal c='int(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=int(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.14,3
2,3,3
3,,
4,-12.56789,-12
5,1.2345e+2,123
6,1.2345e-10,0
```

5.54 match 検索

書式 1: match(検索文字列,$str_1, str_2, \cdots$)
書式 2: matcha(検索文字列,$str_1, str_2, \cdots$)
書式 3: matchs(検索文字列,$str_1, str_2, \cdots$)
書式 4: matchas(検索文字列,$str_1, str_2, \cdots$)
$str_1, str_2, \cdots$ から指定した検索文字列を検索し、マッチすれば真をマッチしなければ偽を返す。
OR 検索か AND 検索か、そして完全一致か部分一致かにより、表 5.23 に示すとおり異なる関数名を指定する。

表 5.23 入力データ

	OR 検索	AND 検索
完全一致	match	matcha
部分一致	matchs	matchas

match 関数は、指定した検索文字列が、$str_1, str_2, \cdots$ のいずれかに完全一致すれば真を返す。matcha 関数は、指定した検索文字列が、$str_1, str_2, \cdots$ の全てに完全一致すれば真を返す。matchs 関数は、指定した検索文字列が、$str_1, str_2, \cdots$ のいずれかに部分一致すれば真を返す。matchas 関数は、指定した検索文字列が、$str_1, str_2, \cdots$ の全てに部分一致すれば真を返す。NULL 値を含めた OR/AND 論理演算の真偽値表は表 5.8 を参照のこと。

利用例

例 1: OR 完全一致

f1,f2,f3 項目のいずれかが 1 であれば真を返す。

```
$ more dat1.csv
id,f1,f2,f3
1,1,1,1
2,1,0,1
3,,,
4,1,,1
$ mcal c='match("1",$s{f1},$s{f2},$s{f3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=match("1",$s{f1},$s{f2},$s{f3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,f1,f2,f3,rsl
1,1,1,1,1
2,1,0,1,1
3,,,,0
4,1,,1,1
```

例 2: AND 完全一致

f1,f2,f3 項目の全てが文字列"1"であれば真を返す。

```
$ mcal c='matcha("1",$s{f1},$s{f2},$s{f3})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=matcha("1",$s{f1},$s{f2},$s{f3}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,f1,f2,f3,rsl
1,1,1,1,1
2,1,0,1,0
3,,,,0
4,1,,1,0
```

例 3: OR **部分一致**

s1,s2,s3 項目のいずれかが、文字列 ab を含んでいれば真を返す。

```
$ more dat2.csv
id,s1,s2,s3
1,ab,abx,x
2,abc,xaby,xxab
3,,,
4,#ac,x,x
$ mcal c='matchs("ab",$s{s1},$s{s2},$s{s3})' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=matchs("ab",$s{s1},$s{s2},$s{s3}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,s1,s2,s3,rsl
1,ab,abx,x,1
2,abc,xaby,xxab,1
3,,,,0
4,#ac,x,x,0
```

例 4: AND **部分一致**

文字列 ab が s1,s2,s3 項目の全てに、文字列 ab が含まれて入れば真を返す。

```
$ mcal c='matchas("ab",$s{s1},$s{s2},$s{s3})' a=rsl i=dat2.csv o=rsl4.csv
#END# kgcal a=rsl c=matchas("ab",$s{s1},$s{s2},$s{s3}) i=dat2.csv o=rsl4.csv
$ more rsl4.csv
id,s1,s2,s3,rsl
1,ab,abx,x,0
2,abc,xaby,xxab,1
3,,,,0
4,#ac,x,x,0
```

例 5: NULL **値の検索**

str 項目が NULL 値であれば真を返す。

```
$ mcal c='match(nulls(),$s{s1},$s{s2},$s{s3})' a=rsl i=dat2.csv o=rsl5.csv
#END# kgcal a=rsl c=match(nulls(),$s{s1},$s{s2},$s{s3}) i=dat2.csv o=rsl5.csv
$ more rsl5.csv
id,s1,s2,s3,rsl
1,ab,abx,x,0
2,abc,xaby,xxab,0
3,,,,1
4,#ac,x,x,0
```

5.55 isnull NULL 値判定

書式: isnull(*num*), isnull(*str*), isnull(*date*), isnull(*time*), isnull(*bool*)

num(他のデータ型も同様) で与えられた値が NULL 値であるかどうかを判定する。NULL 値であれば 0b1(真) を、そうでなければ 0b0(偽) を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,a
2,
3,b
$ mcal c='isnull(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=isnull(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,a,0
2,,1
3,b,0
```

例 2: **他の項目型も指定可能**

```
$ mcal c='isnull($s{val})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=isnull($s{val}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,a,0
2,,1
3,b,0
```

例 3: **空文字を定数で与えた場合**

```
$ mcal c='isnull("")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=isnull("") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,val,rsl
1,a,1
2,,1
3,b,1
```

5.56 julian ユリウス暦変換

書式 1: julian(*date*)

書式 2: julian(*time*)

書式 3: julian2d(*num*)

書式 4: julian2t(*num*)

書式 1,2 では、日付 *date* もしくは時刻 *time* をユリウス通日に変換する。逆に書式 3,4 では、ユリウス通日を日付型もしくは時刻型に変換する。ここで、日付型が与えられたときは、その日の最初の時刻である 00:00:00 として計算される。

利用例

例 1: **基本例**

日付型の date 項目を julian 関数でユリウス通日に変換し、julian2d 関数でまたもとに戻す。

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19700101
$ mcal c='julian($d{date})' a=julian i=dat1.csv o=rsl1.csv
#END# kgcal a=julian c=julian($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,julian
1,20000101,2451545
2,20121021,2456222
3,,
4,19700101,2440588
$ mcal c='julian2d(${julian})' a=date2 i=rsl1.csv o=rsl2.csv
#END# kgcal a=date2 c=julian2d(${julian}) i=rsl1.csv o=rsl2.csv
$ more rsl2.csv
id,date,julian,date2
1,20000101,2451545,20000101
2,20121021,2456222,20121021
3,,,
4,19700101,2440588,19700101
```

例 2: **時刻型も同様**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19700101000100
$ mcal c='julian($t{time})' a=julian i=dat2.csv o=rsl3.csv
#END# kgcal a=julian c=julian($t{time}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,time,julian
1,20000101000000,2451544.5
2,20121021111213,2456221.967
3,,
4,19700101000100,2440587.501
$ mcal c='julian2t(${julian})' a=time2 i=rsl3.csv o=rsl4.csv
#END# kgcal a=time2 c=julian2t(${julian}) i=rsl3.csv o=rsl4.csv
$ more rsl4.csv
```

```
id,time,julian,time2
1,20000101000000,2451544.5,20000101000000
2,20121021111213,2456221.967,20121021111228.800015
3,,,
4,19700101000100,2440587.501,19700101000126.400014
```

5.57 lcm 最小公倍数

書式: $\mathrm{lcm}(num_1, num_2)$

num_1 と num_2 の最小公倍数を求める。実数は整数に変換して (切り下げ) 実行される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val1,val2
1,5,3
2,12,4
3,,
4,5.1,3.1
$ mcal c='lcm(${val1},${val2})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=lcm(${val1},${val2}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val1,val2,rsl
1,5,3,15
2,12,4,12
3,,,
4,5.1,3.1,15
```

例 2: **定数を与える例**

val1 項目と 15 の最小公倍数を求める。

```
$ mcal c='lcm(${val1},15)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=lcm(${val1},15) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val1,val2,rsl
1,5,3,15
2,12,4,60
3,,,
4,5.1,3.1,15
```

5.58 leapyear 閏年判定

書式 1: leapyear(*date*)

書式 2: leapyear(*time*)

日付 *date* もしくは時刻 *time* が閏年かどうか判定する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19770812
$ mcal c='leapyear($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=leapyear($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,1
2,20121021,1
3,,
4,19770812,0
```

例 2: **時刻型でも判定可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='leapyear($t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=leapyear($t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,1
2,20121021111213,1
3,,
4,19770812122212,0
```

5.59 left 先頭切り出し

書式 1: left(*str*, 長さ)

書式 2: leftw(*str*, 長さ)

文字列 *str* について先頭から長さパラメータで指定した文字数を切り出す。マルチバイト文字を含む場合は leftw を使うこと。

利用例

例 1: **基本例**

str 項目の先頭から 3 文字を切り出す。

```
$ more dat1.csv
id,str
1,abcdefg
2,12345678
3,
4,12
$ mcal c='left($s{str},3)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=left($s{str},3) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abcdefg,abc
2,12345678,123
3,,
4,12,12
```

例 2: **マルチバイト文字を含む例**

マルチバイト文字を含む場合は leftw を使う。

```
$ more dat2.csv
id,str
1,あいうえお
2,1あ2345678
3,1あ
4,ああ
$ mcal c='leftw($s{str},3)' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=leftw($s{str},3) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,あいうえお,あいう
2,1あ2345678,1あ2
3,1あ,1あ
4,ああ,ああ
```

5.60 length 文字列長

書式 1: length(*str*)

書式 2: lengthw(*str*)

文字列の長さを計算する。lengthw 関数を用いると、*str* をワイド文字として扱う。NULL 値の長さは 0 であることに注意する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,str
1,abc
2,3.1415
3,
4,hello world!
$ mcal c='length($s{str})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=length($s{str}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abc,3
2,3.1415,6
3,,0
4,hello world!,12
```

例 2: **マルチバイト文字を含む例**

以下は utf-8 でエンコーディングされた日本語を用いた例である。utf-8 の日本語は 1 文字 3 バイトでエンコーディングされているので、length 関数では日本語としての文字数ではなく、そのバイト数を返す。

```
$ more dat2.csv
id,str
1,こんにちは
2,大阪
$ mcal c='length($s{str})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=length($s{str}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,こんにちは,15
2,大阪,6
```

例 3: **ワイド文字として扱う例**

lengthw を使うと、内部で文字列をワイド文字に変換するので、マルチバイト文字 1 文字を正しく認識して計算する。

```
$ mcal c='lengthw($s{str})' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=lengthw($s{str}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,str,rsl
1,こんにちは,5
2,大阪,2
```

5.61 line 行番号

書式: line()

mcal コマンドが処理中の行番号を返す。mcmd では行番号は全て 0 から始まるように統一されており、line 関数においても、データの先頭行の行番号は 0 であることに注意する。

利用例

例 1: 基本例

0 から始まる番号が出力される。

```
$ more dat1.csv
id
1
2
3
4
$ mcal c='line()' a=no i=dat1.csv o=rsl1.csv
#END# kgcal a=no c=line() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,no
1,0
2,1
3,2
4,3
```

例 2: 1 から始める

1 から始まる番号を出力する。

```
$ mcal c='line()+1' a=no i=dat1.csv o=rsl2.csv
#END# kgcal a=no c=line()+1 i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,no
1,1
2,2
3,3
4,4
```

5.62 ln 自然対数

書式: ln(num, 底)

num の自然対数を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,10
2,2.718281828
3,
4,0
5,1
6,-8
$ mcal c='ln(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=ln(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,10,2.302585093
2,2.718281828,0.9999999998
3,,
4,0,
5,1,0
6,-8,
```

5.63 log 対数

書式: log(*num*, 底)

num についての指定された底の対数を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val,base
1,100,10
2,256,2
3,,
4,2,0
5,0,2
6,1,10
7,-8,2
$ mcal c='log(${val},${base})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=log(${val},${base}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,base,rsl
1,100,10,2
2,256,2,8
3,,,
4,2,0,-0
5,0,2,
6,1,10,0
7,-8,2,
```

5.64 log10 常用対数

書式: log10(num, 底)

num の常用対数を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,10
2,0.1
3,
4,0
5,1
6,-8
$ mcal c='log10(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=log10(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,10,1
2,0.1,-1
3,,
4,0,
5,1,0
6,-8,
```

5.65 log2 底が 2 の対数

書式: log2(num)

num について 2 を底とする対数を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,10
2,256
3,
4,0
5,1
6,-8
$ mcal c='log2(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=log2(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,10,3.321928095
2,256,8
3,,
4,0,
5,1,0
6,-8,
```

5.66 max 最大値

書式 1: $\max(num_1, num_2, \cdots)$

書式 2: $\max(num_1, num_2, \cdots, str)$

num_i で与えられた数値の最大値を計算する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: 基本例

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='max(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=max(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,3
2,-5,2,1,2
3,1,,3,3
4,,,,
```

例 2: ワイルドカードを利用した例

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='max(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=max(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,3
2,-5,2,1,2
3,1,,3,3
4,,,,
```

例 3: -n を利用した例

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='max(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=max(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,3
2,-5,2,1,2
3,1,,3,
4,,,,
```

5.67 mid 部分文字列切り出し

書式 1: mid(*str*, 開始位置, 長さ)

書式 2: midw(*str*, 開始位置, 長さ)

文字列 *str* の指定した開始位置から長さ分を切り出す。開始位置は 0 から始まることに注意する。マルチバイト文字を含む場合は midw を使うこと。

利用例

例 1: **基本例**

str 項目の 2 文字目からから 3 文字を切り出す。

```
$ more dat1.csv
id,str
1,abcdefg
2,12345678
3,
4,12
$ mcal c='mid($s{str},2,3)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=mid($s{str},2,3) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abcdefg,cde
2,12345678,345
3,,
4,12,
```

例 2: **基本例**

マルチバイト文字を含む場合は midw を使う。

```
$ more dat2.csv
id,str
1,あいうえお
2,1234567あ8
3,1あ
4,ああ
$ mcal c='midw($s{str},2,3)' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=midw($s{str},2,3) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,あいうえお,うえお
2,1234567あ8,345
3,1あ,
4,ああ,
```

5.68 min 最小値

書式 1: $\min(num_1, num_2, \cdots)$

書式 2: $\min(num_1, num_2, \cdots, str)$

num_i で与えられた数値の最小値を計算する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: 基本例

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='min(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=min(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,1
2,-5,2,1,-5
3,1,,3,1
4,,,,
```

例 2: ワイルドカードを利用した例

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='min(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=min(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,1
2,-5,2,1,-5
3,1,,3,1
4,,,,
```

例 3: -n を利用した例

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='min(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=min(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,1
2,-5,2,1,-5
3,1,,3,
4,,,,
```

5.69 minute 分

書式 1: minute(*time*) 数値

書式 2: minutes(*time*) 2 桁固定長文字列

時刻 *time* から分を取り出す。返す型によって、書式 1,2 を使い分ければよい。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='minute($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=minute($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,20000101000000,0
2,20121021111213,12
3,,
4,19770812122212,22
```

例 2: **文字列とし出力**

```
$ mcal c='minutes($t{time})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=minutes($t{time}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,00
2,20121021111213,12
3,,
4,19770812122212,22
```

5.70 month 月

書式 1: month(*date*) 数値型月番号
書式 2: month(*time*) 数値型月番号
書式 3: months(*date*) 文字列型 2 桁固定長月番号
書式 4: months(*time*) 文字列型 2 桁固定長月番号
書式 5: monthe(*date*) 英語表記
書式 6: monthe(*time*) 英語表記
書式 7: monthes(*date*) 英語短縮表記
書式 8: monthes(*time*) 英語短縮表記
日付 *date* もしくは時刻 *time* から月を返す。月の表記によって書式 1〜8 を使い分ける。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19770812
$ mcal c='month($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=month($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,1
2,20121021,10
3,,
4,19770812,8
```

例 2: **固定長文字列として**

```
$ mcal c='months($d{date})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=months($d{date}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,date,rsl
1,20000101,01
2,20121021,10
3,,
4,19770812,08
```

例 3: **英語表記**

```
$ mcal c='monthe($d{date})' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=monthe($d{date}) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,date,rsl
1,20000101,January
2,20121021,October
3,,
4,19770812,August
```

例 4: **英語短縮表記**

```
$ mcal c='monthes($d{date})' a=rsl i=dat1.csv o=rsl4.csv
#END# kgcal a=rsl c=monthes($d{date}) i=dat1.csv o=rsl4.csv
$ more rsl4.csv
id,date,rsl
1,20000101,Jan
2,20121021,Oct
3,,
4,19770812,Aug
```

例 5: **時刻型でも可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='month($t{time})' a=rsl i=dat2.csv o=rsl5.csv
#END# kgcal a=rsl c=month($t{time}) i=dat2.csv o=rsl5.csv
$ more rsl5.csv
id,time,rsl
1,20000101000000,1
2,20121021111213,10
3,,
4,19770812122212,8
```

5.71 not 否定

書式: not(*bool*)

bool で与えられた真偽値の否定を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,b
1,1
2,
3,0
$ mcal c='not($b{b})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=not($b{b}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,b,rsl
1,1,0
2,,
3,0,1
```

5.72 now 現在時刻

書式 1: now()

書式 2: unow()

now 関数は時刻型で現在時刻を返す (秒単位)。unow 関数はマイクロ秒単位で現在時刻を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id
1
2
$ mcal c='now()' a=rsl i=dat1.csv o=rsl1-1.csv
#END# kgcal a=rsl c=now() i=dat1.csv o=rsl1-1.csv
$ more rsl1-1.csv
id,rsl
1,20170924070940
2,20170924070940
$ mcal c='unow()' a=rsl i=dat1.csv o=rsl1-2.csv
#END# kgcal a=rsl c=unow() i=dat1.csv o=rsl1-2.csv
$ more rsl1-2.csv
id,rsl
1,20170924070940.281092
2,20170924070940.281092
```

例 2: **時刻のみ切り出し**

```
$ mcal c='time(now())' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=time(now()) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,rsl
1,070940
2,070940
```

例 3: **日付のみ切り出し**

```
$ mcal c='date(now())' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=date(now()) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,rsl
1,20170924
2,20170924
```

5.73 null NULL 値

書式: nulln(),nulls(),nulld(),nullt(),nullb()

型に応じた NULL 値を返す。if 関数と組み合わせて NULL 値を出力したい時に使うことができる。

利用例

例 1: 基本例

rsl という項目の全行に NULL 値を出力する。

```
$ more dat1.csv
id
1
2
3
$ mcal c='nulls()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=nulls() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,
2,
3,
```

例 2: if 文の中での利用

if 文の第二パラメータで数値を指定しているので、それに合わせて nulln() 関数を用いる。

```
$ mcal c='if(${id}==1,1,nulln())' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=if(${id}==1,1,nulln()) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,rsl
1,1
2,
3,
```

例 3: isnull と同等の指定

```
$ mcal c='if(${val}==nulln(),"null","notNull")' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=if(${val}==nulln(),"null","notNull") i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,val,rsl
1,a,
2,,
3,b,
```

5.74 nrand 正規乱数

指定された平均値と標準偏差による正規乱数を発生させる。

書式:nrand ([正規乱数])

同じ乱数の種を指定すれば、同じ乱数系列となる。指定可能な乱数の種の範囲は-2147483648〜2147483647 である。乱数の種を省略すると、時刻 (1/1000 秒単位) に応じた異なる乱数の種が利用される。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

利用例

例 1: 例 1: 基本例

平均 0 標準偏差 1(標準正規分布) に基づく乱数を成する。また、乱数の種は 10 に設定している。

```
$ more dat1.csv
id
1
2
3
4
$ mcal c='nrand(0,1,10)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=nrand(0,1,10) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,1.161957445
2,-0.7429729875
3,-1.10165004
4,1.03149223
```

5.75 or 論理和

書式: $\mathrm{or}(bool_1, bool_2, \cdots)$

$bool_i$ で与えられた真偽値全ての論理和を計算する。NULL 値を含めた真偽値表は表 5.9 を参照のこと。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,b1,b2,b3
1,1,0,0
2,1,,1
3,0,,0
4,0,0,0
$ mcal c='or($b{b1},$b{b2},$b{b3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=or($b{b1},$b{b2},$b{b3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,b1,b2,b3,rsl
1,1,0,0,1
2,1,,1,1
3,0,,0,
4,0,0,0,0
```

例 2: **ワイルドカードを利用した例**

b から始まる項目 (b1,b2,b3) をワイルドカード「b*」によって指定している。

```
$ mcal c='or($b{b*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=or($b{b*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,b1,b2,b3,rsl
1,1,0,0,1
2,1,,1,1
3,0,,0,
4,0,0,0,0
```

5.76 pi 円周率

書式: pi()

円周率 (π) を出力する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id
1
2
$ mcal c='pi()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=pi() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,3.141592654
2,3.141592654
```

5.77 power 累乗

書式: power(*num*, 指数)

num の累乗を計算する。演算子^と等価である。結果が実数の最大値を超えると、NULL 値が出力される。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,base,exponent
1,5,2
2,2,8
3,,
4,0,10
5,10,0
6,2,0.5
7,2,-1
$ mcal c='power(${base},${exponent})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=power(${base},${exponent}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,base,exponent,rsl
1,5,2,25
2,2,8,256
3,,,
4,0,10,0
5,10,0,1
6,2,0.5,1.414213562
7,2,-1,0.5
```

5.78 product 積

書式 1: product($num_1, num_2, \cdots$)

書式 2: product($num_1, num_2, \cdots, str$)

num_i で与えられた数値の積を計算する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: 基本例

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='product(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=product(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-10
3,1,,3,3
4,,,,
```

例 2: ワイルドカードを利用した例

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='product(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=product(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-10
3,1,,3,3
4,,,,
```

例 3: -n を利用した例

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='product(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=product(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-10
3,1,,3,
4,,,,
```

5.79 radian ラジアン

書式: radian(角度)
角度をラジアンに変換する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,x,y
1,5,10
2,10,20
3,-1,0
4,0,0
5,,
$ mcal c='atan2(${x},${y})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=atan2(${x},${y}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,x,y,rsl
1,5,10,1.107148718
2,10,20,1.107148718
3,-1,0,3.141592654
4,0,0,0
5,,,
```

5.80 rand 一様乱数

書式: rand([乱数の種])

0.0〜1.0 の範囲で一様乱数を生成する。単独で利用するとその結果は mrand コマンドを-int オプションなしで実行した結果に等しい。

同じ乱数の種を指定すれば、同じ乱数系列となる。指定可能な乱数の種の範囲は-2147483648〜2147483647 である。乱数の種を省略すると、時刻 (ミリ (1/1000 秒単位) に応じた異なる乱数の種が利用される。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

利用例

例 1: **基本例**

0.0 から 1.0 の一様乱数を生成する。乱数の種を指定しているので、何度実行しても同じ乱数系列が生成される。

```
$ more dat1.csv
id
1
2
3
4
$ mcal c='rand(1)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=rand(1) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,0.4170219984
2,0.9971848081
3,0.7203244893
4,0.9325573612
```

例 2: **乱数の種未指定**

乱数の種を指定していないので、実行の度に異なる乱数系列が生成される。

```
$ mcal c='rand()' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=rand() i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,rsl
1,0.1044888373
2,0.2229405369
3,0.6912147216
4,0.8126586084
```

5.81 randi 整数一様乱数

書式: randi(最小値, 最大値 [, 乱数の種])

最小値〜最大値の範囲で整数乱数を生成する。単独で利用するとその結果は mrand コマンドを-int オプションを指定して実行した結果に等しい。

同じ乱数の種を指定すれば、同じ乱数系列となる。指定可能な乱数の種の範囲は-2147483648〜2147483647 である。乱数の種を省略すると、時刻 (1/1000 秒単位) に応じた異なる乱数の種が利用される。

乱数の生成にはメルセンヌ・ツイスター法を利用している (原作者のページ, boost ライブラリ)。

利用例

例 1: 基本例

100 から 999(3 桁整数 900 種類) の整数乱数を生成する。乱数の種を指定しているので、何度実行しても同じ乱数系列が生成される。

```
$ more dat1.csv
id
1
2
3
4
$ mcal c='randi(100,999,1)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=randi(100,999,1) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,475
2,997
3,748
4,939
```

例 2: 0,1 の整数乱数

0 と 1 の 2 種類の整数乱数を生成する。乱数の種を指定していないので、実行の度に異なる乱数系列が生成される。

```
$ mcal c='randi(0,1)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=randi(0,1) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,rsl
1,0
2,1
3,1
4,1
```

5.82 regexlen マッチ文字数

書式 1: regexlen(*str*, 正規表現)

書式 2: regexlenw(*str*, 正規表現)

指定した正規表現が最長マッチする文字列 *str* の部分文字列の長さを返す。マッチしなければ 0 を返す。すなわち 0 文字にマッチしたと解釈する。*str* もしくは正規表現にマルチバイト文字を含む場合は regexlenw 関数を使うこと。

利用例

例 1: **基本例**

正規表現 c.*a に最も長くマッチする部分文字列の長さを得る。マッチした部分文字列については regexstr と同じ入力データを使っているので、比較すると分かりやすい。

```
$ more dat1.csv
id,str
1,xcbbbayy
2,xxcbaay
3,
4,bacabbca
$ mcal c='regexlen($s{str},"c.*a")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexlen($s{str},"c.*a") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,xcbbbayy,5
2,xxcbaay,4
3,,0
4,bacabbca,6
```

例 2: **マルチバイト文字**

正規表現"**あ**.*****い**"に最も長くマッチする部分文字列の長さを得る。ただし、以下ではマルチバイト文字対応でない regexlen 関数を使っているので、文字数ではなくバイト数を返している。

```
$ more dat2.csv
id,str
1,漢漢あbbbいyy
2,漢あbいいy
3,
4,bあいあbbいあ
$ mcal c='regexlen($s{str},"あ.*い")' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=regexlen($s{str},"あ.*い") i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,漢漢あbbbいyy,9
2,漢あbいいy,10
3,,0
4,bあいあbbいあ,14
```

例 3: **マルチバイト文字** 2

正規表現"**あ**.*****い**"に最も長くマッチする部分文字列の長さを得る。regexlenw 関数を使うと、正しくマルチバイト文字を扱ってくれる。

```
$ mcal c='regexlenw($s{str},"あ.*い")' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=regexlenw($s{str},"あ.*い") i=dat2.csv o=rsl3.csv
$ more rsl3.csv
```

```
id,str,rsl
1, 漢漢あ bbb い yy,5
2, 漢あ b いい y,4
3,,0
4,b あいあ bb いあ,6
```

5.83 regexm 全体マッチ

書式 1: regexm(*str*, 正規表現)

書式 2: regexmw(*str*, 正規表現)

指定した正規表現が、文字列 *str* 全体にマッチすれば真を返す。*str* もしくは正規表現にマルチバイト文字を含み、Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexmw 関数を使うこと。

利用例

例 1: **基本例**

id=1,id=2 共に、正規表現で示された c に続く aa を含んでいるが、id=2 ではマッチする範囲が部分的 (2 文字目の c から最後まで) であるために偽となっている。

```
$ more dat1.csv
id,str
1,caabaa
2,acabaaa
3,
4,bbcbcc
$ mcal c='regexm($s{str},"c.*aa")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexm($s{str},"c.*aa") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,caabaa,1
2,acabaaa,0
3,,0
4,bbcbcc,0
```

例 2: **末尾一致**

正規表現.*c で *str* 項目の全体がカバーされるのは id=4 の行のみである。

```
$ mcal c='regexm($s{str},".*c")' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=regexm($s{str},".*c") i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,caabaa,0
2,acabaaa,0
3,,0
4,bbcbcc,1
```

例 3: **空文字マッチ**

正規表現^$で id=3 の空文字にマッチする。

```
$ mcal c='regexm($s{str},"^$")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=regexm($s{str},"^$") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,str,rsl
1,caabaa,0
2,acabaaa,0
3,,1
4,bbcbcc,0
```

5.84 regexpfx マッチ文字列のプレフィックス

書式 1: regexpfx(*str*, 正規表現)

書式 2: regexpfxw(*str*, 正規表現)

指定した正規表現が最長マッチする文字列 *str* の部分文字列のプレフィックス (部分文字列より先頭側の文字列) を返す。同じ文字列及び正規表現によって regexpfx 関数,regexstr 関数,regexsfx 関数の 3 関数を実行した場合、それぞれで得られた文字列をその順番に結合すると元の文字列が再現できる関係にある。*str* もしくは正規表現にマルチバイト文字を含み、Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexpfxw 関数を使うこと。

利用例

例 1: **基本例**

正規表現 `c.*a` に最も長くマッチする部分文字列のプレフィックスを得る。例えば `id=4` では、*str* 項目の `cabbca` にマッチしており、そのプレフィックスすなわち `ba` を返している。`regexstr,regexpfx` と同じ入力データを使っているので、比較すると分かりやすい。

```
$ more dat1.csv
id,str
1,xcbbbayy
2,xxcbaay
3,
4,bacabbca
$ mcal c='regexpfx($s{str},"c.*a")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexpfx($s{str},"c.*a") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,xcbbbayy,x
2,xxcbaay,xx
3,,
4,bacabbca,ba
```

5.85 regexpos マッチ位置

書式 1: regexpos(*str*, 正規表現)

書式 2: regexposw(*str*, 正規表現)

指定した正規表現が最長マッチする文字列 *str* の部分文字列の開始位置を返す。文字列の先頭位置は 0 であることに注意する。またマッチしなければ NULL 値を返す。*str* もしくは正規表現にマルチバイト文字を含む場合は regexposw 関数を使うこと。

利用例

例 1: **基本例**

正規表現 c.*a に最も長くマッチする部分文字列の位置を得る。先頭文字の位置は 0 であることに注意する。マッチした部分文字列については regexstr と同じ入力データを使っているので、比較すると分かりやすい。

```
$ more dat1.csv
id,str
1,xcbbbayy
2,xxcbaay
3,
4,bacabbca
$ mcal c='regexpos($s{str},"c.*a")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexpos($s{str},"c.*a") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,xcbbbayy,1
2,xxcbaay,2
3,,
4,bacabbca,2
```

例 2: **マルチバイト文字**

正規表現"**い**.*あ"に最も長くマッチする部分文字列の長さを得る。ただし、以下ではマルチバイト文字対応でない regexpos 関数を使っているので、文字数ではなくバイト数でカウントした場合の位置を返している。

```
$ more dat2.csv
id,str
1,漢漢あ bbb い yy
2,漢あ b いい y
3,
4,b あいあ bb いあ
$ mcal c='regexpos($s{str},"あ.*い")' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=regexpos($s{str},"あ.*い") i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,漢漢あ bbb い yy,6
2,漢あ b いい y,3
3,,
4,b あいあ bb いあ,1
```

例 3: **マルチバイト文字** 2

正規表現"**い**.*あ"に最も長くマッチする部分文字列の長さを得る。regexposw 関数を使うと、正しくマルチバイト文字を扱ってくれる。

```
$ mcal c='regexposw($s{str},"あ.*い")' a=rsl i=dat2.csv o=rsl3.csv
#END# kgcal a=rsl c=regexposw($s{str},"あ.*い") i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,str,rsl
1,漢漢あ bbb い yy,2
2,漢あ b いい y,1
3,,
4,b あいあ bb いあ,1
```

5.86 regexrep マッチ文字列の置換

書式 1: regexrep(*str*, 正規表現, 置換文字列)

書式 2: regexrepw(*str*, 正規表現, 置換文字列)

指定した正規表現が最長マッチした文字列 *str* の部分文字列を置換文字列で置換する。*str* もしくは正規表現にマルチバイト文字を含み Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexrepw 関数を使うこと。

利用例

例 1: **基本例**

`id=1,id=2` の *str* 項目にマッチした部分文字列を `MMM` に置換する。

```
$ more dat1.csv
id,str
1,caabaa
2,acabaaa
3,
4,cbcbcc
$ mcal c='regexrep($s{str},"c.*aa","MMM")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexrep($s{str},"c.*aa","MMM") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,caabaa,MMM
2,acabaaa,aMMM
3,,
4,cbcbcc,cbcbcc
```

5.87 regexs マッチ

書式 1: regexs(*str*, 正規表現)

書式 2: regexsw(*str*, 正規表現)

指定した正規表現が、文字列 *str* の一部にでもマッチすれば真を返す。*str* もしくは正規表現にマルチバイト文字を含み、Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexsw 関数を使うこと。

利用例

例 1: **基本例**

id=1,id=2 共に、正規表現で示された c に続く aa を含んでいるので真を返す。

```
$ more dat1.csv
id,str
1,caabaa
2,acabaaa
3,
4,cbcbcc
$ mcal c='regexs($s{str},"c.*aa")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexs($s{str},"c.*aa") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,caabaa,1
2,acabaaa,1
3,,0
4,cbcbcc,0
```

例 2: **先頭一致**

正規表現.*c を *str* 項目が含むのは id=3 以外全ての行である。

```
$ mcal c='regexs($s{str},".*c")' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=regexs($s{str},".*c") i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,caabaa,1
2,acabaaa,1
3,,0
4,cbcbcc,1
```

5.88 regexsfx マッチ文字列のサフィックス

書式 1: regexsfx(*str*, 正規表現)

書式 2: regexsfxw(*str*, 正規表現)

指定した正規表現が最長マッチする文字列 *str* の部分文字列のサフィックス (部分文字列より末尾側の文字列) を返す。同じ文字列及び正規表現によって regexpfx 関数,regexstr 関数,regexsfx 関数の 3 関数を実行した場合、それぞれで得られた文字列をその順番に結合すると元の文字列が再現できる関係にある。*str* もしくは正規表現にマルチバイト文字を含み、Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexsfxw 関数を使うこと。

利用例

例 1: **基本例**

正規表現 `c.*a` に最も長くマッチする部分文字列のサフィックスを得る。例えば `id=4` では、*str* 項目の `cabbca` にマッチしており、そのサフィックスすなわち null 文字を返している。`regexstr,regexpfx` と同じ入力データを使っているので、比較すると分かりやすい。

```
$ more dat1.csv
id,str
1,xcbbbayy
2,xxcbaay
3,
4,bacabbca
$ mcal c='regexsfx($s{str},"c.*a")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexsfx($s{str},"c.*a") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,xcbbbayy,yy
2,xxcbaay,y
3,,
4,bacabbca,
```

5.89 regexstr マッチ文字列

書式 1: regexstr(*str*, 正規表現)

書式 2: regexstrw(*str*, 正規表現)

指定した正規表現が最長マッチする文字列 *str* の部分文字列を返す。*str* もしくは正規表現にマルチバイト文字を含み、Shift_JIS など文字の出現順によっては意に沿わない検索結果となる場合は regexstrw 関数を使うこと。

利用例

例 1: **基本例**

正規表現 `c.*a` に最も長くマッチする部分文字列を得る。`id=2` では、`cba` もしくは `cbaa` いずれの部分文字列にもマッチしたと考えることができるが、本関数では、より長くマッチした文字列を返す。

```
$ more dat1.csv
id,str
1,xcbbbayy
2,xxcbaay
3,
4,bacabbca
$ mcal c='regexstr($s{str},"c.*a")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=regexstr($s{str},"c.*a") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,xcbbbayy,cbbba
2,xxcbaay,cbaa
3,,
4,bacabbca,cabbca
```

5.90 right 末尾切り出し

書式 1: right(*str*, 長さ)

書式 2: rightw(*str*, 長さ)

文字列 *str* について末尾から長さパラメータで指定した文字数を切り出す。マルチバイト文字を含む場合は rightw を使うこと。

利用例

例 1: **基本例**

str 項目の末尾から 3 文字を切り出す。

```
$ more dat1.csv
id,str
1,abcdefg
2,12345678
3,
4,12
$ mcal c='right($s{str},3)' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=right($s{str},3) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abcdefg,efg
2,12345678,678
3,,
4,12,12
```

例 2: **マルチバイト文字を含む例**

マルチバイト文字を含む場合は rightw を使う。

```
$ more dat2.csv
id,str
1,あいうえお
2,1234567あ8
3,1あ
4,ああ
$ mcal c='rightw($s{str},3)' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=rightw($s{str},3) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,str,rsl
1,あいうえお,うえお
2,1234567あ8,7あ8
3,1あ,1あ
4,ああ,ああ
```

5.91 round 四捨五入

書式: round(*num*, 基数)

num を四捨五入によりまるめる。この時、基数の整数倍の値集合のうち最も近い値にまるめられる。例えば、round(3.82,0.5) の場合、0.5 とびに目盛がうたれた数直線上で、3.82 に最も近い目盛点、すなわち 4.0 が基数 0.5 における四捨五入点となる。基数を省略すると 1.0 がデフォルト値として用いられる。これは小数点以下 1 桁目を四捨五入して整数値にまるめることに等しい。

利用例

例 1: **基本例**

小数点以下一桁目を四捨五入する。

```
$ more dat1.csv
id,val
1,3.28
2,3.82
3,
4,-0.6
$ mcal c='round(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=round(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.28,3
2,3.82,4
3,,
4,-0.6,-1
```

例 2: **基本例**

小数点以下二桁目を四捨五入する。

```
$ mcal c='round(${val},0.1)' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=round(${val},0.1) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,val,rsl
1,3.28,3.3
2,3.82,3.8
3,,
4,-0.6,-0.6
```

例 3: **基数** 0.5 **の例**

0.5 を基数として四捨五入する。

```
$ mcal c='round(${val},0.5)' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=round(${val},0.5) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,val,rsl
1,3.28,3.5
2,3.82,4
3,,
4,-0.6,-0.5
```

例 4: **基数** 10 **の例**

一桁目を四捨五入する。

```
$ more dat2.csv
id,val
1,1341.28
2,188
3,1.235E+3
4,-1.235E+3
$ mcal c='round(${val},10)' a=rsl i=dat2.csv o=rsl4.csv
#END# kgcal a=rsl c=round(${val},10) i=dat2.csv o=rsl4.csv
$ more rsl4.csv
id,val,rsl
1,1341.28,1340
2,188,190
3,1.235E+3,1240
4,-1.235E+3,-1230
```

5.92 second 秒

書式 1: second(*time*) 数値

書式 2: seconds(*time*) 2 桁固定長文字列

書式 3: usecond(*time*) 数値

書式 4: useconds(*time*) 2 桁.6 桁固定長文字列

時刻 *time* から秒を取り出す。書式 1 では数値 (整数) として秒を返し、書式 2 では秒の 2 桁を固定長文字列とした返す。書式 3 は、マイクロ秒で返す。書式 4 では、秒の 2 桁固定長、マイクロ秒の小数 6 桁固定長を文字列として返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,20000101121103
2,20121021111209.123
3,211209
4,211209.123
$ mcal c='second($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=second($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,20000101121103,3
2,20121021111209.123,9
3,211209,9
4,211209.123,9
```

例 2: **文字列とし出力**

```
$ mcal c='seconds($t{time})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=seconds($t{time}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101121103,03
2,20121021111209.123,09
3,211209,09
4,211209.123,09
```

例 3: **マイクロ秒を出力**

```
$ mcal c='usecond($t{time})' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=usecond($t{time}) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,time,rsl
1,20000101121103,3
2,20121021111209.123,9.123
3,211209,9
4,211209.123,9.123
$ mcal c='useconds($t{time})' a=rsl i=dat1.csv o=rsl4.csv
#END# kgcal a=rsl c=useconds($t{time}) i=dat1.csv o=rsl4.csv
$ more rsl4.csv
id,time,rsl
1,20000101121103,03.000000
2,20121021111209.123,09.123000
3,211209,09.000000
4,211209.123,09.123000
```

5.93 sign 符号

書式: sign(num)

num の符号を判定する。プラスであれば 1 を、マイナスであれば-1 を、ゼロであれば 0 を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,5
2,-5
3,
4,0
$ mcal c='sign(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sign(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,5,1
2,-5,-1
3,,
4,0,0
```

5.94 sin サイン

書式: $\sin(r)$

ラジアン r に対するサインを計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,0.523599
3,
4,6.283185
$ mcal c='sin(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sin(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,6.535897931e-07
2,0.523599,0.5000001943
3,,
4,6.283185,-3.071795869e-07
```

5.95 sinh 双曲線正弦

書式: sinh(r)

双曲線正弦 (ハイパーボリック サイン) を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,-1.047197
3,
4,6.283185
$ mcal c='sinh(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sinh(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,11.54873178
2,-1.047197,-1.249366168
3,,
4,6.283185,267.7448118
```

5.96 sqrt 平方根

書式: sqrt(num)

num の平方根を求める。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,9
2,2
3,
4,-1
$ mcal c='sqrt(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sqrt(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,9,3
2,2,1.414213562
3,,
4,-1,
```

5.97 sqsum 平方和

書式 1: sqsum($num_1, num_2, \cdots$)

書式 2: sqsum($num_1, num_2, \cdots, str$)

num_i で与えられた数値の平方和を計算する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='sqsum(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sqsum(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,14
2,-5,2,1,30
3,1,,3,10
4,,,,
```

例 2: **ワイルドカードを利用した例**

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='sqsum(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=sqsum(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,14
2,-5,2,1,30
3,1,,3,10
4,,,,
```

例 3: -n **を利用した例**

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='sqsum(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=sqsum(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,14
2,-5,2,1,30
3,1,,3,
4,,,,
```

5.98 sum 合計

書式 1: sum($num_1, num_2, \cdots$)

書式 2: sum($num_1, num_2, \cdots, str$)

num_i で与えられた数値を全て合計する。書式 1 では、NULL 値は無視されるが、全てが NULL 値であれば結果も NULL となる。

書式 2 において最後の引数として"-n"を与えると、NULL 値に対する扱いが変わり、項目値に一つでも NULL 値がある場合は、結果も NULL 値となる。

利用例

例 1: 基本例

```
$ more dat1.csv
id,v1,v2,v3
1,1,2,3
2,-5,2,1
3,1,,3
4,,,
$ mcal c='sum(${v1},${v2},${v3})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=sum(${v1},${v2},${v3}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-2
3,1,,3,4
4,,,,
```

例 2: ワイルドカードを利用した例

v から始まる項目 (v1,v2,v3) をワイルドカード「v*」によって指定している。

```
$ mcal c='sum(${v*})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=sum(${v*}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-2
3,1,,3,4
4,,,,
```

例 3: -n を利用した例

v2 に NULL 値を含む id=3 の行の結果も NULL となる。

```
$ mcal c='sum(${v1},${v2},${v3},"-n")' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=sum(${v1},${v2},${v3},"-n") i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,v1,v2,v3,rsl
1,1,2,3,6
2,-5,2,1,-2
3,1,,3,
4,,,,
```

5.99 t2julian(日時)

日時からユリウス通日を求める

ユリウス通日は紀元前 4713 **年** 1 **月** 1 **日正午（世界標準時による）からの日数である。**
ただしグリゴリオ歴は 1400-1-1〜9999-12-31 **が有効範囲のため、**
それに対応するユリウス通日の有効値範囲は 2232300〜5373484 **となる。**
その範囲外では NULL **値出力となる。**

なお，変換の際に指定した日時の時刻は切り捨てられ，日付のみが変換の対象となる。

利用方法

```
c=t2julian(0t20080822101010)
```

利用例

表 5.24 入力データ

日付	時間	数
20020824	20020824145408	10660
20020622	20020622173449	22740
20020824	20020824145408	14800
20021009	20021009095743	54510
20020121	20020121173449	18750

上記のデータを用いて、それぞれの場合の利用例を以下に示す。

実行例 1)
”時間”項目を入力としてユリウス通日に変換し、新たに”ユリウス通日”をいう項目を作成して出力している。

```
-----------------------------------------------
mcal c='t2julian($t{時間})' a="ユリウス通日" i=date.csv o=ot2julian.csv
-----------------------------------------------
```

表 5.25 出力ファイル (ot2julian.csv)

日付	時間	数	ユリウス通日
20020824	20020824145408	10660	2452511.621
20020622	20020622173449	22740	2452448.733
20020824	20020824145408	14800	2452511.621
20021009	20021009095743	54510	2452557.415
20020121	20020121173449	18750	2452296.733

mcal 日付時間関連へ戻る

5.100 tan タンジェント

書式: $\tan(r)$

ラジアン r に対するタンジェントを計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,0.785398
2,1.047197
3,
4,3.141593
$ mcal c='tan(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=tan(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,0.785398,0.9999996732
2,1.047197,1.732048603
3,,
4,3.141593,3.464102066e-07
```

5.101 tanh 双曲線逆正接

書式: tanh(r)

双曲線逆正接 (ハイパーボリック タンジェント) を計算する。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,val
1,3.141592
2,-1.047197
3,
4,6.283185
$ mcal c='tanh(${val})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=tanh(${val}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,val,rsl
1,3.141592,0.9962720714
2,-1.047197,-0.7807142201
3,,
4,6.283185,0.9999930253
```

5.102 time 時分秒

書式: time(*time*)

時刻 *time* から時分秒を 6 桁固定長文字列として取り出す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='time($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=time($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,20000101000000,000000
2,20121021111213,111213
3,,
4,19770812122212,122212
```

5.103 today 本日の日付

書式: today()

本日の日付を日付型で返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id
1
2
$ mcal c='today()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=today() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,20170924
2,20170924
```

5.104 tolower 小文字変換

文字列をすべて小文字に変更する。アルファベット 26 文字以外の文字については何の影響もない。

利用例

例 1: **基本例**

str 項目のアルファベット大文字を全て小文字に変換する。

```
$ more dat1.csv
id,str
1,ABC
2,aB$12!Cd
3,
4,cBA
$ mcal c='tolower($s{str})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=tolower($s{str}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,ABC,abc
2,aB$12!Cd,ab$12!cd
3,,
4,cBA,cba
```

5.105 top 先頭行

書式: top()

先頭行であれば真を、そうでなければ偽を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
val
1
2
3
4
$ mcal c='top()' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=top() i=dat1.csv o=rsl1.csv
$ more rsl1.csv
val,rsl
1,1
2,0
3,0
4,0
```

例 2: **累計計算の例**

```
$ mcal c='if(top(),${val},${val}+#{})' a=rsl i=dat1.csv o=rsl2.csv
#END# kgcal a=rsl c=if(top(),${val},${val}+#{}) i=dat1.csv o=rsl2.csv
$ more rsl2.csv
val,rsl
1,1
2,3
3,6
4,10
```

5.106 toupper 大文字変換

文字列をすべて大文字に変更する。アルファベット 26 文字以外の文字については何の影響もない。

利用例

例 1: **基本例**

str 項目のアルファベット小文字を全て大文字に変換する。

```
$ more dat1.csv
id,str
1,abc
2,Ab$12!cD
3,
4,Cba
$ mcal c='toupper($s{str})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=toupper($s{str}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,str,rsl
1,abc,ABC
2,Ab$12!cD,AB$12!CD
3,,
4,Cba,CBA
```

5.107 tseconds 経過秒数

書式 1: tseconds(*time*)

書式 2: tuseconds(*time*)

00:00:00 から時刻 *time* までの経過秒数を計算する。tseconds では秒単位で、tuseconds ではマイクロ秒単位で値を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,time
1,000103
2,235959
3,235959.123
4,000000
$ mcal c='tseconds($t{time})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=tseconds($t{time}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,time,rsl
1,000103,63
2,235959,86399
3,235959.123,86399
4,000000,0
```

例 2: **日付が異なっても結果は同じ**

```
$ more dat2.csv
id,time
1,20130901000103
2,20130902000103
3,20130902000103.123
$ mcal c='tseconds($t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=tseconds($t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20130901000103,63
2,20130902000103,63
3,20130902000103.123,63
```

例 3: **マイクロ秒**

```
$ mcal c='tuseconds($t{time})' a=rsl i=dat1.csv o=rsl3.csv
#END# kgcal a=rsl c=tuseconds($t{time}) i=dat1.csv o=rsl3.csv
$ more rsl3.csv
id,time,rsl
1,000103,63
2,235959,86399
3,235959.123,86399.123
4,000000,0
```

5.108 uxt UNIX 時刻変換

書式 1: uxt(*date*)

書式 2: uxt(*time*)

書式 3: uxt2d(*num*)

書式 4: uxt2t(*num*)

書式 1,2 では、日付 *date* もしくは時刻 *time* を UNIX 時刻に変換する。逆に書式 3,4 では、UNIX 時刻を日付型もしくは時刻型に変換する。ここで、日付型が与えられたときは、その日の最初の時刻である 00:00:00 として計算される。

利用例

例 1: **基本例**

日付型の date 項目を uxt 関数で UNIX 時刻に変換し、uxt2d 関数でまたもとに戻す。

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19700101
$ mcal c='uxt($d{date})' a=uxt i=dat1.csv o=rsl1.csv
#END# kgcal a=uxt c=uxt($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,uxt
1,20000101,946684800
2,20121021,1350777600
3,,
4,19700101,0
$ mcal c='uxt2d(${uxt})' a=date2 i=rsl1.csv o=rsl2.csv
#END# kgcal a=date2 c=uxt2d(${uxt}) i=rsl1.csv o=rsl2.csv
$ more rsl2.csv
id,date,uxt,date2
1,20000101,946684800,20000101
2,20121021,1350777600,20121021
3,,,
4,19700101,0,19700101
```

例 2: **時刻型も同様**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19700101000100
$ mcal c='uxt($t{time})' a=uxt i=dat2.csv o=rsl3.csv
#END# kgcal a=uxt c=uxt($t{time}) i=dat2.csv o=rsl3.csv
$ more rsl3.csv
id,time,uxt
1,20000101000000,946684800
2,20121021111213,1350817933
3,,
4,19700101000100,60
$ mcal c='uxt2t(${uxt})' a=time2 i=rsl3.csv o=rsl4.csv
#END# kgcal a=time2 c=uxt2t(${uxt}) i=rsl3.csv o=rsl4.csv
$ more rsl4.csv
```

```
id,time,uxt,time2
1,20000101000000,946684800,20000101000000
2,20121021111213,1350817933,20121021111213
3,,,
4,19700101000100,60,19700101000100
```

5.109 week 週

書式 1: week(*date*)

書式 2: week(*time*)

書式 3: week111(*date*)

書式 4: week111(*time*)

日付 *date* もしくは時刻 *time* の ISO8601 で規定された週番号を返す。ISO8601 で規定された週番号とは、年の最初の木曜日を含む週をその年の第 1 週と規定している。

一方で week111 を利用すると、曜日に関係なく 1/1 を第 1 週の第 1 日目と考えて週番号を返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
1,20000102
1,20000103
1,20000104
1,20000105
1,20000106
1,20000107
1,20000108
1,20000109
2,20121021
3,
4,19770812
$ mcal c='week($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=week($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,52
1,20000102,52
1,20000103,1
1,20000104,1
1,20000105,1
1,20000106,1
1,20000107,1
1,20000108,1
1,20000109,1
2,20121021,42
3,,
4,19770812,32
```

例 2: **時刻型でも可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='week($t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=week($t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,52
```

```
2,20121021111213,42
3,,
4,19770812122212,32
```

5.110 xor 排他的論理和

書式: '*b* 項目名 *db* 項目名'

$bool_i$ で与えられた真偽値全ての排他的論理和を計算する。NULL 値を含めた真偽値表は表 5.10 を参照のこと。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,b1,b2,b3
1,1,0,0
2,1,,1
3,0,,0
4,0,0,0
$ mcal c='$b{b1}^^$b{b2}^^$b{b3}' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=$b{b1}^^$b{b2}^^$b{b3} i=dat1.csv o=rsl1.csv; IN=4 OUT=4
$ more rsl1.csv
id,b1,b2,b3,rsl
1,1,0,0,1
2,1,,1,
3,0,,0,
4,0,0,0,0
```

5.111 year 西暦年

書式 1: year(*date*)

書式 2: year(*time*)

書式 3: years(*date*)

書式 4: years(*time*)

日付 *date* もしくは時刻 *time* から西暦年を取り出す。書式 1,2 は数値として返し、書式 3,4 は文字列として返す。

利用例

例 1: **基本例**

```
$ more dat1.csv
id,date
1,20000101
2,20121021
3,
4,19770812
$ mcal c='year($d{date})' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=year($d{date}) i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,date,rsl
1,20000101,2000
2,20121021,2012
3,,
4,19770812,1977
```

例 2: **時刻型でも可能**

```
$ more dat2.csv
id,time
1,20000101000000
2,20121021111213
3,
4,19770812122212
$ mcal c='year($t{time})' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=year($t{time}) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,time,rsl
1,20000101000000,2000
2,20121021111213,2012
3,,
4,19770812122212,1977
```

5.112 型変換

書式:s2n(*str*), n2s(*num*), n2b(*num*), s2n(*str*), s2d(*str*), s2t(*str*) s2b(*str*), d2s(*date*), d2t(*date*), t2s(*time*), t2d(*time*), b2n(*bool*), b2s(*bool*)

型変換を行う関数群。mcal では型の自動変換は行わないため、必要なときはユーザが明示的に型変換を指定しなければならない。

n2b(s2b) 関数は、1(”1”) を真に 0(”0”) を偽に、それ以外を NULL 値に変換する。b2n(b2s) 関数は、真は 1(”1”) に偽は 0(”0”) に変換する。

d2t 関数は、日付型を時刻型に変換するが、そのとき時刻として 12:00:00 を自動的に補完する。d2s 関数は、8 文字固定長文字列 (”yyyymmdd”) に変換し、t2s 関数は 14 文字固定長文字列 (”yyyymmddHHMMSS”) に変換する。日付/時刻についての詳細は「5.13 日付型と時刻型」を参照のこと。

利用例

例 1: **乱数の固定長化**

1 から 9999 の整数乱数を発生させ、4 桁固定長で出力する。fixlen 関数は整数型のデータ (ここでは randi の結果) には対応していないので、n2s 関数で文字列型に変換する必要がある。

```
$ more dat1.csv
id
1
2
3
4
$ mcal c='fixlen(n2s(randi(1,9999,11)),4,"R","0")' a=rsl i=dat1.csv o=rsl1.csv
#END# kgcal a=rsl c=fixlen(n2s(randi(1,9999,11)),4,"R","0") i=dat1.csv o=rsl1.csv
$ more rsl1.csv
id,rsl
1,1803
2,0684
3,0195
4,6647
```

例 2: **真偽パターン**

項目 v1,v2,v3 が 10 異常かどうかを判定し、01 のパターンを出力する。

```
$ more dat2.csv
id,v1,v2,v3
1,10,5,7
2,5,12,11
3,3,6,2
4,14,16,11
$ mcal c='cat("",b2s(${v1}>=10),b2s(${v2}>=10),b2s(${v3}>=10))' a=rsl i=dat2.csv o=rsl2.csv
#END# kgcal a=rsl c=cat("",b2s(${v1}>=10),b2s(${v2}>=10),b2s(${v3}>=10)) i=dat2.csv o=rsl2.csv
$ more rsl2.csv
id,v1,v2,v3,rsl
1,10,5,7,100
2,5,12,11,011
3,3,6,2,000
4,14,16,11,111
```

索引

abs, 273
acos, 274
age, 275
and, 276
argsize, 277
asin, 278
atan, 279
atan2, 280
avg, 281

berrand, 283
binomdist, 282
bottom, 284

capitalize, 285
cast, 385
cat, 286
ceil, 287
cos, 289
cosh, 290
countnull, 291

d2julian, 292
date, 294
day, 293
degree, 295
diff, 296
dist, 298
distgps, 299
dow, 300

e, 302
exp, 303

factorial, 304
fixlen, 305
fldsize, 306
floor, 307
format, 309
fract, 310

gcd, 311

hasspace, 312
heron, 313
hour, 314

if, 315
int, 317
isnull, 320

julian, 321

lcm, 323
leapyear, 324
left, 325
length, 326
line, 327
ln, 328
log, 329
log10, 330
log2, 331

m2cross, 97
maccum, 63
marff2csv, 65
match, 318
mavg, 67
max, 332
mbest, 69
mbucket, 71
mcal, 264
mcat, 75
mchgnum, 79
mchgstr, 82
mchkcsv, 85
mcombi, 89
mcommon, 91
mcount, 94
mcross, 95
mcsv2arff, 99
mcsv2json, 101
mcsvconv, 103
mcut, 107
mdata, 109
mdelnull, 111
mdformat, 113
mdsp, 115
mduprec, 117
mfldname, 119
mfsort, 121
mhashavg, 122
mhashsum, 124
mid, 333
min, 334
minput, 127
minute, 335
mjoin, 129
mkeybreak, 131
mmbucket, 133
mminput, 138
mmseldsp, 140
mmvavg, 142
mmvsim, 145
mmvstats, 146
mnewnumber, 148
mnewrand, 150
mnewstr, 152
mnjoin, 153
mnormalize, 155
mnrcommon, 157
mnrjoin, 159
mnullto, 161
mnumber, 163
month, 336
mpadding, 167
mpaste, 170
mproduct, 172
mrand, 173
mrjoin, 176
msed, 179
msel, 183
mseldsp, 185
mselnum, 187
mselrand, 189
mselstr, 191
msep, 196
msep2, 198
msetstr, 200
mshare, 202

mshuffle, 203
msim, 206
mslide, 210
msortf, 213
msplit, 218
mstats, 220
msum, 222
msummary, 223
mtab2csv, 225
mtonull, 229
mtra, 231
mtrafld, 233
mtraflg, 236
muniq, 238
mvcat, 239
mvcommon, 241
mvcount, 243
mvdelim, 244
mvjoin, 248
mvreplace, 246, 250, 252
mvsort, 254
mvuniq, 256
mwindow, 257
mxml2csv, 260

not, 338
now, 339
nrand, 341
null, 340

or, 342

pi, 343
power, 344
product, 345

radian, 346
rand, 347
randi, 348
regexlen, 349
regexm, 351
regexpfx, 352
regexpos, 353
regexrep, 355
regexs, 356
regexsfx, 357
regexstr, 358
right, 359
round, 360

second, 362
sign, 363
sin, 364
sinh, 365
sqrt, 366
sqsum, 367
sum, 368

t2julian, 369
tan, 371
tanh, 372
time, 373
today, 374
tolower, 375
top, 376
toupper, 377
tseconds, 378

uxt, 379

week, 381

xor, 383

year, 384

M コマンド, 15
便利ツール, 57