パッケージ mosaic を用いたリサンプリングに基づく統計的推測

Daniel Kaplan* Nicholas J. Horton†
Randall Pruim‡

2012 年 9 月 3 日版

日本語訳 荒木 孝治

2012 年 10 月 19 日

目次

1 はじめに

mosaic パッケージは，現代的な計算技術により可能となった方法を用いて，統計学とモデリング（のみならず，代数学）を教えることを支援するために開発された．開発の目標は，入門レベルの大学生に利用可能な，効率的な計算を提供することにある．低価格で高速な計算環境と，R のような強力でフリーなソフトウェアが広く利用可能となったため，アクセスしやすさを制約する因子は，アイデアが明確

* dtkaplan@gmail.com
† nhorton@smith.edu
‡ rpruim@calvin.edu

に表現され，理解される記法を与える知性に関する因子である．

本稿の目的は，mosaic パッケージを用いることによりランダム化に基づく統計的推測の方法を説明することにある．このパッケージは，学生が学び，容易に一般化できるアプローチを支援する．このため，

- 学生は，できるだけ少ないコマンドで統計を実行できるようにする．
- 基本的なコマンドは，プログラミングのオペレーションではなく，統計のオペレーションに関連すべきである．ループや計数，加算といったプログラミングのオーバーヘッドをできるだけ避ける．
- ブラックボックスは，計算において簡単なオペレーションを実行すべきである．その出力は検証可能でなければならない．
- コマンドは，高レベルのプロシージャの単なる名前ではなく，統計の論理的構造を明示すべきである．
- 平均，計数，比率といった簡単な記述統計から一般化線形モデルといったより複雑なモデルへと拡張する簡単な道筋を示すべきである．

といった信条のもとに，リサンプリングに基づく統計的推測を行うための比較的直接的な方法を説明する．このパッケージは，Robin Lock と同僚が USCOTS (United States Conference on Teaching Statistics) 2011 で提起した一連の問題により導かれた (`http://www.causeweb.org/uscots/breakout/breakout3_6.php`)．mosaic パッケージの目的の 1 つは，プログラミングの深遠な局面をマスターしなくても素人でも容易に利用することのできる基本的なコマンドを提供することである．パッケージおよび本イニシアティブのさらなる情報に関しては，プロジェクト MOSAIC のウェブサイト（www.mosaic-web.org）を参照のこと．

このパッケージを用いると，学生は，George Cobb が統計的推測の “3R”（ランダム化，繰り返し，棄却）（Cobb，2007）と呼ぶオペレーションを実行することができる．様々な方法で 3R を組み合わすことにより，学生は，無批判に公式を適用するだけではなく，推測の論理を一般化し，自分のものにする方法を学ぶことができる．

Terry Speed（2001）は，シミュレーションの役割に関する興味深い議論を行っている．そこで彼は，シミュレーションの役割の変化を記している．シミュレーションは，従来は，数学ができない人が利用するものであった．しかし今や，全ての統計分析をシミュレーションにより実行することができる時代となった．

統計学における最も重要なオペレーションは，母集団から標本を理想的にはランダムに抽出するというサンプリングであることに間違いはない．しかし残念なことに，サンプリングは手間がかかるため，講師は理論的な説明を好み，サンプリングの実際の役割に重きを置かない．さらに，統計学の大部分のテキストが用いている代数での記法は，サンプリングを明確に説明する目的には適していない．

しかし，コンピュータを用いると，これらの効率と記法といった障害を克服することができる．統計学のコースにおける統計的概念の中で中心的な役割を持つものとして，サンプリングを正しく位置づけることができる．並べ替え検定とブートストラッピングを用いるリサンプリングに基づく推測は，入門的統計学およびそれ以上においてますます重要となりつつある技術である．

並べ替え検定とブートストラッピングを用いるランダム化に基づく推測は，入門あるいはそれ以上の統計においてますます重要が技術となってきている．異なるソフトウェアのアプローチを比較するのに有益にすることにより教育者にとって役立たせるために, Robin Lock と同僚は, USCOTS 2011 (United States Conference on Teaching Statistics) においてブートストラッピングと並べ替え検定に関連した一連の問題を提示した．これは，`http://www.causeweb.org/uscots/breakout/breakout3_6.php`

で参照できる．

ブートストラッピングと並べ替え検定は，推定と検定に対する強力で洗練されたアプローチであり，漸近的な結果を見つけることが難しかったり満足できなかったりする多くの状況においても実行することができる（Efron and Tibshirani, 1993; Hesterberg et al 2005）．ブートストラッピングでは，母集団からの復元サンプリングを行い，関心のある統計量の計算を繰り返すことにより標本分布を経験的に構成する．2 群の比較のための並べ替え検定は，グループのラベルを並べ替え，それに対する標本統計量（例えば，新しいラベルに対応する 2 群の差）を計算することにより帰無仮説における分布を経験的に構成する．

本稿では，Lock の無作為化の問題を用いて mosaic パッケージの利用法を例示する．さらに，Lock の問題の簡単な設定を超えて，mosaic のオペレーションが無作為化の論理をより複雑なモデルへと簡単に一般化することができることを示す．

2 背景と設定

2.1 R と RStudio

R はオープンソースの統計環境であり，入門統計を教えるために多くの組織で利用されている．いくつかある特長のなかで特に，R を用いると，初心者に対してより専門的な高度な統計学へ進むための合理的な道を提供しながら，ランダム化を通じて統計的推測の概念を示すのが簡単になる．RStudio（`http://www.rstudio.org`）は，システムの利用を容易にする R の統合開発環境である．

2.2 設定

パッケージ *mosaic* はインターネット上で利用可能であり，システムの標準的な特徴を利用することにより，R にインストールすることができる（これは一度行うだけでよい）．

```
install.packages("mosaic")
```

一旦インストールしておくと，パッケージを利用するときに，ロードする（これはセッション毎に必要である）．パッケージをロードすることに加えて，数値を表示する桁数も設定する．

```
require(mosaic)
options(digits = 3)
```

上記のようなコマンドは，R のセッションを開始するたびに自動的に実行されるように設定ファイルとして学生に渡しておくこともできる．

mosaic パッケージは R の中で機能するので，学生にデータセットを提供するのに R が持つデータを扱う機能を全て利用することができる．特に，read.csv() といった R の関数は，ウェブサイトの URL を通じてデータファイルにアクセスすることをサポートしている．そのため，学生個人のコンピュータにデータファイルをダウンロードする必要はない．URL は長く，わかりにくいことが多いので，mosaic は，簡単な名前によりデータファイルを参照することができる fetchData() 関数を提供している．fetchData() により自動的にチェックされるウェブサイトに Lock の問題に関連したデータファイルを置いている．（講師は，スタートアップファイルコマンドにより自分自身のサイトを指定す

ることができる．これにより，教師は，自分自身のサイトにファイルを置くと，直ちに学生が利用可能となる.）

では，Lock のデータセットの 1 つをインターネットからダウンロードしてみよう．

```
> mustangs <- fetchData("MustangPrice.csv")
Complete file name given. No searching necessary.
```

3　Lock の問題

Lock の問題は，Robin Lock 他により 2011 年に提案された統計的推測に関する一連の簡単な問題であり，講師がランダム化ソフトウェアの有用性を比較することをできるようにするものである．

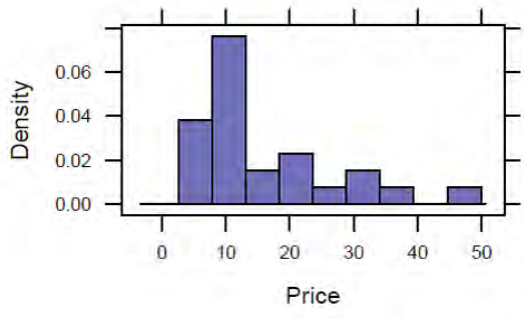
Lock の問題 1．平均のブートストラッピング（中古ムスタング）

> インターネットのウェブサイトで売り出されているムスタング（*Mustang*）の中古車価格を学生が集めた．ファイル *MustangPrice.csv* に，価格（単位：*1000* ドル），年，走行距離（単位：*1000* マイル）のデータがある．このデータを用いて中古ムスタングの平均価格の *90%* 信頼区間を作れ．

学生は，計算する前にデータを吟味することを学ぶ必要がある．このために，R の標準的なツールを利用することができる．2 つの理由から，lattice スタイルのグラフィックス関数の利用を推奨する．

1. mosaic で用いられる構文と一貫性のある構文を利用している．
2. より複雑な設定に自然と拡張することができる．

```
xhistogram(~Price, data=mustangs)
```



基本的な lattice の記法では，~ を含む “式” および `data=`により関連するデータセットを指定し，この記法は mosaic 全体でも利用されている．例えば，標本平均は次のようにして求める．

```
> mean(~Price, data=mustangs)
[1] 16
```

R をよく知っている人は，なぜ次のように書かないのか疑問に思うだろう．

```
mean(mustangs$Price)
```

もちろん，次のようにして同じ計算を行うことができる．

```
mean(~Price, data=mustangs)
```

上記のようにすることで，次のステップを期待している．つまり，ランダム化やモデリングといったさらなるオペレーションへの入門である．

リサンプリングは，復元ランダムサンプリングともいうが，ランダム化における重要なオペレーションである．resample() 関数は，これを実行する．小さなデータに対して非常に基本的な方法で例示する．

```
simple=c(1,2,3,4,5)
resample(simple)
[1] 4 4 5 5 5

resample(simple)
[1] 1 1 3 5 1

resample(simple)
[1] 4 1 1 2 5
```

データフレームに適用するとき，resample() はランダムに行全体を選択する．次を用いてこれを例示する．

```
resample(mustangs)

    Age Miles Price
10    1   1.1  37.9
20   14 102.0   8.2
19   12 117.4   7.0
25   14 115.1   4.9
19.1 12 117.4   7.0
 6   15 111.0  10.0

... and so on
```

左の列におけるケース番号を見ると，ケース 19 が 2 回選択されていることがわかる．

1 回のリサンプリング試行による平均の計算は，次により実行できる．

```
mean(~Price, data=resample(mustangs))
```

```
[1] 20.5
```

1 回の試行ではほとんど意味が無いが，リサンプリング無しの場合と異なる結果を通常得ることを示すために学生に計算させるのは良いことである．

コマンドを繰り返すことにより別の試行を実行できる．

```
mean(~Price, data=resample(mustangs))

[1] 12.3
```

さらに 5 回，1 つのコマンドで試行しよう．

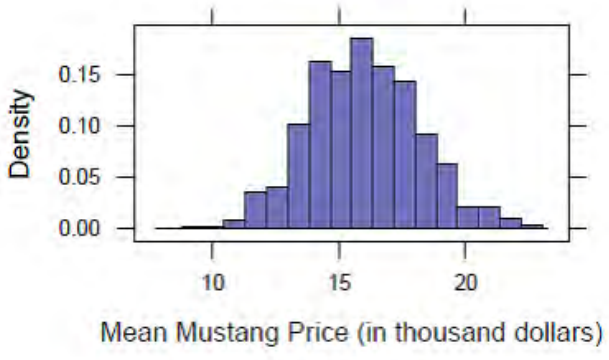
```
do(5) * mean(~Price, data=resample(mustangs))

  result
1   14.6
2   15.5
3   16.1
4   17.9
5   14.5
```

次に，1000 回のリサンプリング試行を実行し，`trials` というオブジェクトに結果を保存する．

```
trials <- do(1000) * mean(~Price, data=resample(mustangs))
```

このリサンプリング分布をプロットするのは簡単である．

```
xhistogram(~ result, data=trials, xlab="Mean Mustang Price (in thousand dollars)")
```



この分布を信頼区間に変換する最も簡単な方法は，その目的に合った演算を適用することである．

```
confint(trials, level=0.90, method="quantile")

    name  5 % 95 %
1 result 12.5 19.5

confint(trials, level=0.90, method="stderr")

    name lower upper
1 result  12.4  19.5
```

confint() 関数はブラックボックスである．それを紹介するとき，講師は，一般目的の演算を用いてある程度詳細に信頼区間の背後にある計算を示したくなるだろう．confint() は 2 つの基本的なアプローチ，つまり，分布のパーセント点に基づくものと正規理論に基づくものを実装している．

- パーセント点を用いた 90% 信頼区間の計算

```
qdata(c(.05, .95), result, data=trials)

  5%  95%
12.5 19.5
```

- 正規理論と標準誤差を用いて．先ず適切な自由度に対応する t_* の限界値を計算する．または，簡単のため，あるいは t_* がどれくらい違っているかを明確にするために z_* を用いる．信頼率 90% は裾の確率 0.95 に対応する．

```
tstar <- qt(.95, df=24)
zstar <- qnorm(0.95)
```

誤差限界は次のようになる．

```
tstar * sd(~result,data=trials)
[1] 3.68

zstar * sd(~result,data=trials)
[1] 3.54
```

信頼区間を求めたいとき，このような一連のコマンドを利用することに意味はほとんどない．そのために confint() がある．演算を抽象化し，利用者が信頼水準と方法（“quantile” または “stderr”）を指定するだけでよく，信頼水準を裾の確率に変換する必要はない．誤差限界の計算の詳細な手順を繰り返すべき範囲は，環境と講師の目標に依存するので，講師が決定する必要がある．R と mosaic は両方のアプローチをサポートする．

Lock の問題 2： 比率の検定（NFL の延長時間）

> National Football League（NFL）は，規定時間の終了時に同点のゲームの勝利者を決定するのに延長時間を利用する．延長時間に得点をあげたチームがゲームの勝者となる．コイン投げにより，どちらのチームが最初にボールを持つかを決定する．コイン投げに勝つことは有利なのか？ 1974 年から 2009 年のシーズンのデータでは，コイン投げの勝者は 428 回のゲームのうち 240 回のゲームの勝者となっている．これを NFL のサンプルとして，延長ゲームでコイン投げの勝者が半数以上のゲームで勝つかどうかを検定する．

全体として，入門的な学生でも，最初にボールを持つことはゲームの結果と関係がないとするとき，428 回のゲームの内の約半数でコイン投げに勝ったチームが勝利することを理解する．問題は，240 回の勝利が 428 回のうちの “約” 半数と判断できるかどうかということである．統計的推測は，サンプリングの変動という文脈の中で，“428/2=214” と 240 を比較する．

スタイル 1　 2 項分布の組み込み関数の利用

帰無仮説が成立する状況で 428 回のゲームのランダムサンプルを抽出するシミュレーションを行い，240 回以上勝という結果を与える試行の比率を見る．

```
prop(rbinom(100000, prob=0.5, size=428) >= 240)

   TRUE
0.00699
```

結果より，もし帰無仮説が正しいとき，コイン投げの勝者が 240 回以上勝つことはありそうにないことがわかる．別のシミュレーションでは結果が少し異なるが，試行回数を増やすことによりこの差を小さくすることができる．

```
prop(rbinom(100000, prob=0.5, size=428) >= 240)

   TRUE
0.00655
```

決定論的な確率計算を行うことによりシミュレーションが持つ変動を避けたい場合ももある．しかし，この正確な再生可能性は，“240 以上” ということに対応する分布の裾を反映する限界点をカスタマイズする必要性が生じる．右側の場合，観測数から 1 を引くことを意味する．

```
pbinom(239, prob=0.5, size=428)

[1] 0.993
```

決定論的な確率計算を行いたい場合でも，乱数発生の論理を解説したくなるだろう．

スタイル 2　コイン投げをシミュレートする.

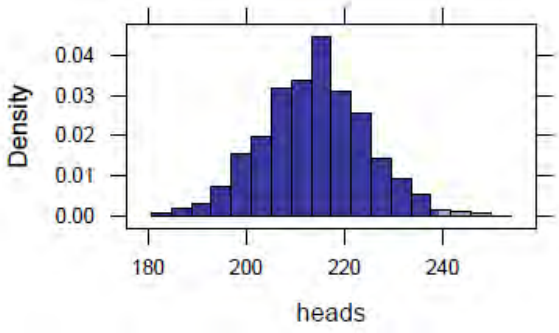
コイン投げは統計学のコースの要であるので，mosaic パッケージはランダムなコイン投げの結果を表にまとめる演算を提供している.

```
do(1) * rflip(428)

    n heads tails
1 428   206   222
```

1000 回試行し，コイン投げの勝者が 428 回のうち 240 回以上勝つ比率を数える.

```
trials <- do(1000) * rflip(428)
prop(trials$heads >= 240)

 TRUE
0.009
```

```
xhistogram(~ heads, groups=(heads >= 240), data=trials)
```



観測された 240 勝のパターンは，帰無仮説の元での結果とは判断しにくい．`groups =` オプションを用いて網掛けした領域は，結果を視覚的に強化するのに役立つ.

Lock の問題 3：2 群からの平均の並べ替え検定（睡眠と記憶）

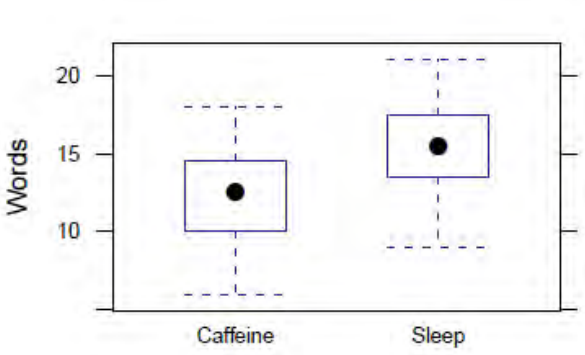
記憶に関する実験（Mednicj et al, 2008）において，学生は 24 語のリストを記憶するように指示される．学生は，語を聞いた後，ランダムに 2 つのグループに分けられる．12 人から構成される 1 つのグループは，1.5 時間睡眠し，もう 1 つのグループは，カフェインの錠剤が与えられ，眠らない．次に示す結果は，休憩後，参加者が記憶していた語の数を示す．2 つの処理の間で語の平均記憶数に違いがあるかどうかを検定せよ.

```
sleep <- fetchData("SleepCaffeine.csv")
Complete file name given. No searching necessary.
```

睡眠グループは，平均的により多くの単語を覚えているようである．

```
mean(Words ~ Group, data=sleep)

Caffeine   Sleep
    12.2    15.2

obs <- diff(mean(Words ~ Group, data=sleep))
obs
Sleep
3
```

```
bwplot(Words ~ Group, data=sleep)
```



帰無仮説の状態を作るために，Words の説明変数である Group をスクランブルする．

```
diff(mean(Words ~ shuffle(Group), data=sleep))

Sleep
-1.17
```

これは 1 回の試行にすぎない．再度試行する．

```
diff(mean(Words ~ shuffle(Group), data=sleep))

Sleep
0.333
```

帰無仮説の元での分布を得るために，多くの試行を行う．

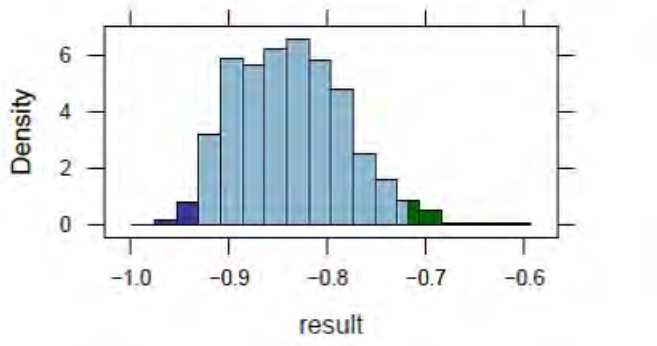
Lock の問題 4：相関のブートストラッピング

問題 1 におけるムスタングの中古価格データは，走行距離（単位：1000 マイル）に関するデータを含んでいる．価格と走行距離との相関係数の 95% 信頼区間を求めよ．

```
with(mustangs, cor(Price, Miles))

[1] -0.825
```

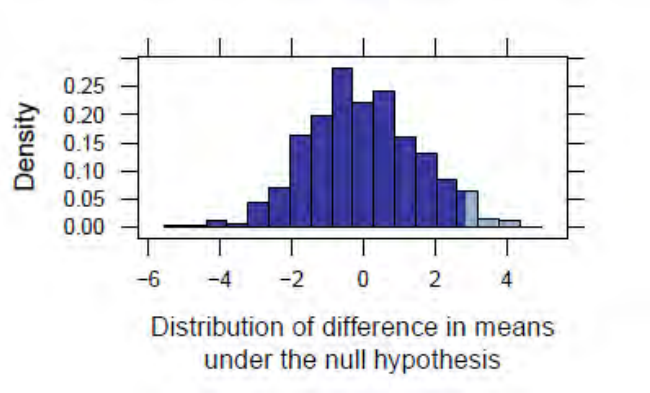
```
trials <- do(1000) * with(resample(mustangs), cor(Price, Miles))
quantiles <- qdata(c(.025, .975), trials$result); quantiles

  2.5%  97.5%
-0.928 -0.720
```

```
xhistogram(~ result, data=trials,
  groups=cut(result, c(-Inf, quantiles, Inf)),
  nbin=30)
```




```
trials <- do(1000) * diff(mean(Words ~ shuffle(Group), data=sleep))
xhistogram(~ Sleep, groups=(Sleep >= obs), data=trials,
  xlab="Distribution of difference in means\nunder the null hypothesis")
```

この検定に対する片側 p 値は，観測された差の値以上の結果を生み出す試行の比率である．この検定に対する片側 p 値は，観測された差以上の結果を生み出す試行数を合計することにより計算することができる．ここでは，1000 試行の内 35 回のみが条件を満たすので，p 値は 0.035 である．よって，2 つのグループが母集団として，同じ平均記憶数を持つことはあり得ないと結論づけることができる．

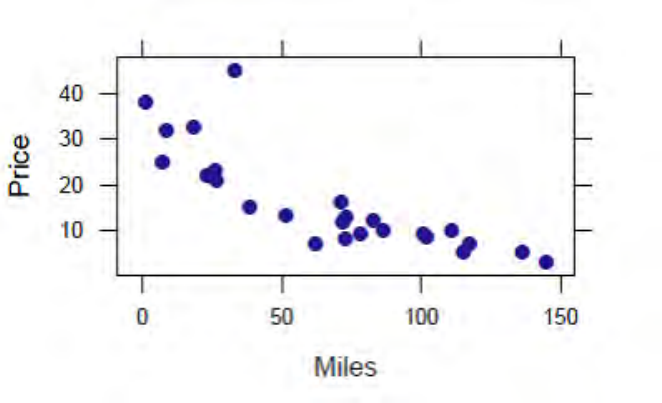


4　平均と比率を超えて

Lock の問題は，多くの入門応用統計のコースで扱う平均，比率，平均の差，比率の差といった記述統計に関するものである．しかし，そのような基本統計量をより一般的なフレームワーク，つまり，線形モデルの文脈で紹介することは意味がある．

通常，線形モデルは，回帰の文脈で解釈される．例えば，Lock のムスタングの価格データセットでは，車の価格と走行距離との間には関係があると思われる．グラフを描いてみる．

```
xyplot( Price ~ Miles, data=mustangs)
```



グラフより明確な関係を見ることができるが，その関係の強さを相関係数を用いて数値化できる．

回帰では，変数間の直線的な関数関係を見る．R の組み込み関数である lm() は，この計算を行う．

```
lm(Price ~ Miles, data=mustangs)

Call:
lm(formula = Price ~ Miles, data = mustangs)
Coefficients:
(Intercept) Miles
    30.495 -0.219
```

ここで利用されている記法は，散布図を作成するものと同じである．結果より，1000 マイル走る毎に中古車価格は 219 ドル安くなる（あるいは，データの単位で表現すると，0.2188 千ドル）．言い換えると，ムスタングの価格は，1 マイル当たり約 22 セント安くなる．

1 マイル当たり 22 セントというのは，走行距離の “効果の大きさ” と考えることができる．これは，学生（および車の購入者）にとって，相関係数が -0.82 であるというよりはるかにわかりやすい．相関係数と比べて，回帰を用いて効果の大きさで記述する方に利点がある．回帰の利点は他にもあるが，それは後に説明する．しかし，今強調しておきたいことは，平均や比率，平均の差，比率の差といった伝統的な計算に代わるフレームワークとして回帰を利用できるということである．

ムスタングの平均価格を，次のように通常の方法で計算することもできる．

```
mean( Price, data=mustangs )

[1] 16
```

線形モデルを用いて，同じ結果を得ることができる．

```
lm( Price ~ 1, data=mustangs)

Call:
lm(formula = Price ~ 1, data = mustangs)

Coefficients:
(Intercept)
         16
```

これはわかりにくいかもしれない．lm() において，記号 ~ は目的変数と説明変数を指定するために用いられる．`Price ~ Miles` において，目的変数は Price，説明変数は Miles である[*1]．走行距離は車によって異なる．走行距離のこの変動は，価格の変動の説明に用いられる．

`Price ~ 1` において，`1` は説明変数がないこと，あるいは少し異なる視点からいうと，車は全て同じであることを意味する．関数 mean() においても，これと同じ記法を利用することができる．

```
mean( Price ~ 1, data=mustangs)
1
16
```

なぜか？これにより，異なるグループに対する平均を計算することができるように記法を拡張することができる．例として，睡眠とカフェインのどちらが単語を記憶する能力を高めるかを比較するかという Lock の睡眠データを取り上げる．

グループ（睡眠とカフェインのグループ）を無視するとき，記憶された単語の平均は次のようになる，

```
mean( Words ~ 1, data=sleep )
```

[*1] （訳注）~の左の変数が目的変数，右の変数が説明変数となる

```
1
13.8
```

グループで層別して計算するには，説明変数を利用する．

```
mean( Words ~ Group, data=sleep )

Caffeine    Sleep
    12.2     15.2
```

慣習として，回帰は，ムスタングデータにおける Miles のような数値変数を扱う手法として導入された．これに対して層別の平均は，睡眠データにおける Group のような質的変数に対して利用される．しかし，質的変数を回帰に適用することは可能である．

```
lm( Words ~ Group, data=sleep )

Call:
lm(formula = Words ~ Group, data = sleep)

Coefficients:
(Intercept)   GroupSleep
       12.2          3.0
```

mean() で報告される情報と lm() の結果との違いは，結果を表示する形式である．lm() の Group-Sleep の係数は，2 つのグループの間の差を与える．

```
diff( mean( Words ~ Group, data=sleep ))

Sleep
    3
```

関数 lm() を用いて，比率または比率の差を計算することができる．the Health Evaluation and Linkage to Primary Care 無作為化臨床試験における患者に関する情報を含む HELPrct データセットを用いて説明する．ホームレスであると報告された人の比率を考える．

```
prop( homeless ~ 1, data=HELPrct)

homeless:
0.461
```

ホームレスの患者の比率は，性別で異なる，つまり，男性は少しホームレスになりりやすいようである．

```
prop( homeless ~ sex, data=HELPrct )

homeless:female   homeless:male
          0.374           0.488
```

これらの 2 つの比率の差は，次のようになる．

```
diff(prop( homeless ~ sex, data=HELPrct ))

homeless:male
        0.115
```

lm 関数を用いて同じ結果を得ることができる．(“homeless” または “housed”，いずれの水準の比率を求めたいかを指定する必要がある．)

```
lm( homeless=="homeless" ~ 1, data=HELPrct )

Call:
lm(formula = homeless == "homeless" ~ 1, data = HELPrct)

Coefficients:
(Intercept)
      0.461
```

mean() や prop()，diff() が同じ結果を与えるのに，なぜ lm() を用いるのか．それは，lm() の場合，一般化できるからである．lm() は，質的または量的な説明変数に対して適用でき，非常に重要なことに，複数の説明変数を利用できる．

mean() や prop() を用いるときでさえ，記法~ 1 を用いる方が良い理由がある．それは，応答変数の変動を説明することを試みているわけではないことを強調する．利用する説明変数は存在しない．

```
lm(homeless=="homeless" ~ sex, data=HELPrct)

Call:
lm(formula = homeless == "homeless" ~ sex, data = HELPrct)

Coefficients:
(Intercept) sexmale
      0.374   0.115
```

4.1 線形モデルでのランダム化

lm() でランダム化を実行するには，mean() や prop() と同じスタイルで行えばよい．ここでは例として，ムスタングデータにおいて，走行距離に応じた価格の変化の信頼区間をリサンプリングを用いて求める．

```
trials <- do(1000) * lm(Price ~ Miles, data=resample(mustangs))
confint(trials)

       name  lower  upper
1 Intercept 24.889 36.034
2     Miles -0.277 -0.163
3     sigma  2.969  9.033
4 r.squared  0.518  0.884
```

または，HELPrct データを用いて男性と女性の間のホームレス率の違いを並べ替え検定することができる．

```
nulldist <- do(1000) * lm(homeless=="homeless" ~ shuffle(sex), data=HELPrct)

prop(~ abs(sexmale) > 0.1146, data=nulldist)
 TRUE
0.059
```

4.2 複数の説明変数

回帰モデルでは，複数の説明変数を利用することができる．例えば，ムスタングの Price に Age と Miles がどれくらい関係しているかを調べることができる．

```
trialsmod1 <- do(1000) * lm( Price ~ Age, data=resample(mustangs))
trialsmod2 <- do(1000) * lm( Price ~ Miles, data=resample(mustangs))
trialsmod3 <- do(1000) * lm( Price ~ Miles + Age, data=resample(mustangs))
```

第 1 のモデルは，Price（価格）が 1 年当たり 1000-2000 ドル減少することを示す．

```
confint(trialsmod1)

       name lower upper
1 Intercept 23.78 37.150
2       Age -2.29 -1.181
3     sigma  4.01 11.395
4 r.squared  0.27  0.756
```

第 2 のモデルは，1 マイル当たり 16-28 セント減少することを意味する．

```
confint(trialsmod2)
       name lower upper
1 Intercept 24.576 36.325
2     Miles -0.279 -0.159
```

```
3     sigma  2.809  9.059
4 r.squared  0.520  0.891
```

第 3 のモデルは，各説明変数を互いの文脈の中に置き，Age は Miles の proxy であることを示す.

```
confint(trialsmod3)
       name  lower   upper
1 Intercept 25.453 36.0534
2     Miles -0.323 -0.0948
3       Age -0.973  0.7159
4     sigma  2.684  9.1336
5 r.squared  0.520  0.9100
```

この結果を解釈する 1 つの方法は，Miles で調整するとき，Age の効果はほとんどないということである．Miles と Age に対する信頼区間のインフレは，これらの説明変数の共線性の結果である．

4.3 シミュレーションと分散分析

分散分析を考える 1 つの方法は，モデル項が他のモデル項の文脈において，ランダムな “ゴミ” よりも良いということを数量化することである．例えば，ムスタングの価格の Miles を含むモデルにおける Age の役割を考えよ．分散分析の 1 つの結果は次のようになる．

```
anova( lm( Price ~ Miles + Age, data=mustangs))

Analysis of Variance Table

Response: Price
          Df Sum Sq Mean Sq F value Pr(>F)
Miles      1 2016      2016   46.94  7e-07  ***
Age        1    4         4    0.09   0.77
Residuals 22  945 43
---
Signif. codes: 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

Age の p 値は，Age がモデルに寄与していないことを示唆する．しかし，異なる結論を示唆する異なる分散分析結果もある．

```
anova( lm( Price ~ Age + Miles, data=mustangs))

Analysis of Variance Table

Response: Price
          Df Sum Sq Mean Sq F value Pr(>F)
Age        1 1454      1454    33.9 7.4e-06 ***
Miles      1  565       565    13.2  0.0015 **
Residuals 22  945        43
```

```
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

分散分析の 2 つの結果の違いは，いくつかの方法で説明することができる．例えば，分散分析をモデルの入れ子になった逐次的な結果と考えることにより，モデルの項の記載の順序が違いを生み出す．例えば，モデルの列からの R^2 は，Age は Miles ほどは寄与していないことを示唆する．

```
do(1) * lm( Price ~ Miles, data=mustangs)

  Intercept   Miles   sigma r.squared
1      30.5  -0.219    6.42      0.68

do(1) * lm( Price ~ Miles + Age, data=mustangs)

  Intercept   Miles     Age sigma r.squared
1      30.9  -0.205  -0.155  6.55     0.681
```

ここで，do() はランダム化なしに，単に，それらを直ちに比較できるように結果をフォーマットするだけのために利用している．説明変数として Age を追加することは，R^2 の値を少し増大させ（0.680 から 0.681 へ）ていることに注意.

Age をシャフルすることは，帰無仮説の元で期待されるものと比較して R^2 の実際の変化を生み出すことができる.

```
trials1 <- do(1000) * lm( Price ~ Miles + shuffle(Age), data=mustangs )
confint(trials1)

       name  lower  upper
1 Intercept 25.704 35.451
2     Miles -0.232 -0.206
3       Age -0.581  0.565
4     sigma  6.059  6.787
5 r.squared  0.660  0.727
```

`Price ~ Miles + Age` より得られる R^2 の値は，Age をシャフルしたとき観測された値の範囲の右側に入っている.

Miles をシャフルしたとき，結果はかなり異なる.

```
trials2 <- do(1000) * lm( Price ~ shuffle(Miles) + Age, data=mustangs )
confint(trials2)

       name   lower  upper
1 Intercept 24.7016 35.631
2     Miles -0.0772  0.081
3       Age -1.8770 -1.563
4     sigma  7.6286  8.571
```

```
5 r.squared  0.4580  0.567
```

$R^2 = 0.681$ は信頼区間の外側にあるので，Miles に関する帰無仮説を棄却できる．

5　他の方法でシミュレーションを利用する

リサンプリングとシャフリングという基本的な手法は，信頼区間と p 値の生成のみではなく，統計学の他の多くの概念を例示するために利用できる．例えば，t 分布や F 分布といった分布の起源を示すことは有益である．

例として，HELPrct データにおけるホームレスと性別の独立性の χ^2 検定を取り上げる．帰無仮説の元での簡単な検定の p 値の分布を構成するシミュレーションを次に示す．

1. 検定を実行する．例えば，

```
chisq.test( tally( ~ homeless + sex,
   data=HELPrct, margins=FALSE))

Pearson's Chi-squared test with Yates' continuity correction
data: tally(~homeless + sex, data = HELPrct, margins = FALSE)
X-squared = 3.87, df = 1, p-value = 0.04913
```

2. 関心のある統計量の値を取得するためにコマンドを修正する．今の場合，次のようにして p 値を抽出する．

```
chisq.test( tally( ~ homeless + sex,
                   data=HELPrct, margins=FALSE))$p.value

[1] 0.0491
```

3. 帰無仮説の状況を実現するために，ランダム化を導入する．

```
chisq.test( tally( ~ shuffle(homeless) + sex,
                   data=HELPrct, margins=FALSE))$p.value

[1] 0.976
```

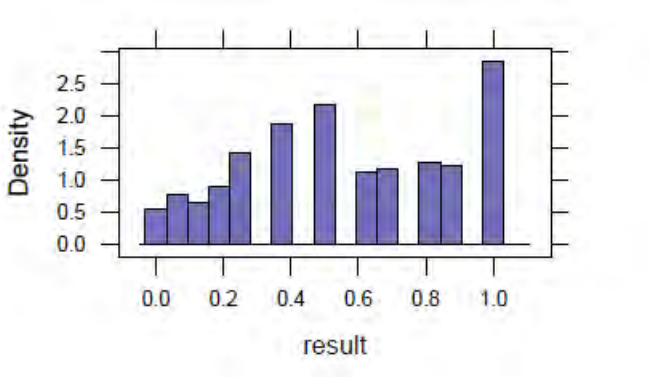
4. 繰り返すことにより帰無仮説の元での分布を得る．

```
trials = do(1000)* chisq.test( tally( ~ shuffle(homeless) + sex,
                     data=HELPrct, margins=FALSE))$p.value
```

厳密に言うと，必要なのは最後のステップのみである．他は単にコマンドの構成の方法を説明するために示しただけである．

学生は，帰無仮説の元では，p 値は 0.05 より少し大きいだけであると考えがちである．帰無仮説の元での p 値の分布は 0 から 1 の一様分布であり，または，帰無仮説が真のときでさえ，帰無仮説を棄却する（$p < 0.05$）確率は約 5% 程度であることがわかると，学生は驚く．

```
prop(~result < 0.05, data=trials)

 TRUE
0.052

xhistogram( ~result, data=trials )
```



おそらくより重要なのは，シミュレーションアプローチにより，より発展的で適切なモデルや検定に接続できることである．より適切なのは，よく利用されている χ^2 検定ではなく，年齢を共変量とするロジスティック回帰といったおそらくより現代的で柔軟な手法である．

```
trials = do(1000) *
   glm( homeless=="homeless" ~ age + sex,
     data=resample(HELPrct), family="binomial")
confint(trials)

       name     lower   upper
1 Intercept -2.363803 -0.3915
2       age -0.000953  0.0482
3   sexmale  0.035213  0.9432
```

6 謝辞

初期のドラフトに対するコメントをもらった Sarah Anoke，および，USCOTS のセッションをオーガナイズした St. Lawrence 大学の Robin Lock に感謝する．

MOSAIC プロジェクトは，US National Science Foundation より支援を受けている（DUE-

0920350)．本パッケージとイニシアティブに関するさらなる情報については，MOSAIC プロジェクトのウェブサイト（www.mosaic-web.org）を参照.

参考文献

1. G. W. Cobb, The introductory statistics course: a Ptolemaic curriculum?, *Technology Innovations in Statistics Education*, 2007, 1(1).
2. B. Efron & R. J. Tibshirani, *An Introduction to the Bootstrap*, 1993, Chapman & Hall, New York.
3. T. Hesterberg, D. S. Moore, S. Monaghan, A. Clipson & R. Epstein. *Bootstrap Methods and Permutation Tests (2nd edition)*, 2005, W.H. Freeman, New York.
4. D.T. Kaplan, *Statistical Modeling: A Fresh Approach, 2nd edition*, `http://www.mosaic-web.org/StatisticalModeling`.
5. S.C. Mednicj, D. J. Cai, J. Kanady, S. P. Drummond. “Comparing the benefits of caffeine, naps and placebo on verbal, motor and perceptual memory”, *Behavioural Brain Research*, 2008, 193(1):79-86.
6. T. Speed, “Simulation”, *IMS Bulletin*, 2011, 40(3):18.