良いdoファイルとは、どんなコードなのか？

2025/3/4 疫学統計分析・演習2

一度作ったdoファイルを、後々まで利活用するために、「良い」doファイルを作成しておく必要があります。そのためには、どのような基準で工夫をすればよいのか説明します。

# 可読性

Stataのdoファイルに限らず、良いコードとするためには、可読性の高さ（読みやすいこと）が重要になります。読みやすいと、バグやエラーの発生を抑えられますし、誤った解析になっていることをより早く見つけることができます。

可動性を高めるための要素を列挙します。

### 短い

100行のdoファイルよりも、10行のdoファイルの方が読みやすく、全体を把握することが容易です。また、横方向に延々と長いコマンドが伸びているようなdoファイルは、その行のコマンドで何をしているのか分かり難いです。縦方向にも横方向にも短いコードになるようにdoファイルを作成しましょう。

### シンプル

ロジックの複雑性が低いという意味です。とはいえ、行いたい解析が複雑であることもありえます。可能な限り、無駄な複雑さを避けることとシンプルな処理への置き換えを試みます。

### 一貫性

変数名の名前の付け方が一貫していたり、doファイルの構造を統一的なルールで一貫していたりすることを指します。Stataでは命名規則は特に定まっていませんが、個人的には一般的な変数やローカルマクロは小文字のスネークケースで、プログラムやグローバルマクロはキャメルケースを使っています。

|  |  |  |
| --- | --- | --- |
| 命名規則 | 説明 | 具体例 |
| Snake case | 単語をアンダースコア( \_ )で区切る方法 | snake\_case\_variable |
| Camel case | 各単語の最初の文字を大文字で書く方法 | CamelCaseMacro |

### 整理整頓

Stata doファイルのなかで、コマンドの記載がよく整理されているということを意味します。適宜改行するなどして関連する部分をまとめておくことが重要です。さらに、コードのまとまりについてコメントなどで後から分かるようにしておくことも重要です。

## 可動性を高める方法の例

### コード例1

sysuse nlsw88, clear  
gen tmp1 = ln(wage)  
gen tmp2 = hours\*60  
regress tmp1 tmp2  
predict tmp3, xb  
gen tmp4 = ln(wage) - tmp3  
twoway scatter tmp4 tmp2 if union==0, msymbol("o") jitter(5 5) || scatter tmp4 tmp2 if union==1, msymbol("d") jitter(5 5)

### 実行結果



* 長い：最終行のtwowayコマンドが改行なしで書かれています。
* 複雑：残差(tmp4)を求めるために、予測値を計算してから、観察値との差を求めるという操作になっています。
* 一貫性？：新たに作られる変数名が全てtmpから始まっていて、一種の一貫性がありますが、意味を伴っていない命名であるため、分かり難くなっています。
* 整理整頓できていない：改行やコメントを使った整理整頓ができていません。

### コード例1改善

/\* データ読み込み \*/  
sysuse nlsw88, clear  
  
/\* 線形回帰モデルのための変数の作成 \*/  
gen ln\_wage = ln(wage)  
gen minutes = hours\*60  
  
/\* 残差の算出 \*/  
regress ln\_wage minutes  
predict residual, residual  
  
/\* 残差プロット \*/  
twoway ///  
 scatter tmp4 tmp2 if union==0, msymbol("o") jitter(5 5) || ///  
 scatter tmp4 tmp2 if union==1, msymbol("d") jitter(5 5)

実行結果は全く同じですが、改善されたコードの方が読みやすいことがわかるかと思います。

# 安全性

安全性が高いということは、誤った解析や間違ったデータをそのままにして、解析を進めない（進めにくい）という状態を指します。つまり、誤った状態にいち早く気づくことが必要になります。

このためには、Stataでは確認用のコマンドがあります。

|  |  |  |
| --- | --- | --- |
| コマンド | 用途 | 文法 |
| describe | 変数の型などを一覧する。 | describe |
| su | 変数の平均値・標準偏差・最大・最小を表示する。 | su |
| codebook | 変数に関する詳細情報を一覧で表示する。 | codebook |
| misstable sum | 欠損値の状態を一覧で表示する。 | misstable sum |
| assert | 条件が成立していないと、コードを停止する。 | assert 条件 |

### コード例2

sysuse nlsw88, clear  
  
/\* 全ての変数の要約 \*/  
describe  
su  
codebook  
  
/\* 欠損値の状況を確認 \*/  
misstable sum  
  
/\* hoursが想定している範囲（欠損値 or 100時間未満）であるか確認する \*/  
assert hours<100 | hours==.

# 保守性

保守性とは、既存のコードを変更するときに容易に変更することができることを指します。解析では、「交絡要因を変更する」ということは比較的多くあると思います。このためには、変更箇所がまとまっていることや、共通部分をprogramコマンドで簡易コマンド化しておくと楽に変更できます。

### コード例3

sysuse nlsw88, clear  
  
/\* モデル毎の共変量の設定 \*/  
local model\_1 age i.collgrad  
local model\_2 `model\_1’ i.south i.c\_city  
local model\_3 `model\_2’ i.industry i.occupation  
  
/\* 応答変数の対数化 \*/  
gen ln\_wage = ln(wage)

/\* 線形回帰モデル (アウトカム=賃金) \*/  
regress ln\_wage i.union `model\_0’  
regress ln\_wage i.union `model\_1’  
regress ln\_wage i.union `model\_2’  
regress ln\_wage i.union `model\_3’  
  
/\* 線形回帰モデル (アウトカム=労働時間) \*/  
regress hours i.union `model\_0’  
regress hours i.union `model\_1’  
regress hours i.union `model\_2’  
regress hours i.union `model\_3’

この解析では、モデル0からモデル3までの4つの解析を2つのアウトカムで行っています。合計で8回のregressが実行されます。各モデルで交絡候補に入っている変数は、localで定義されています。

モデル1の調整変数を変更したいときには、2箇所のregressをさわるのではなく、localの1箇所を変更するだけで良いというメリットがあります。

解析方針の変更に際して、doファイルを変更箇所する箇所が少ないので、保守性が高いと言えます。

# 効率性

効率性はコンピューターリソースの消費が少ないことや、処理時間が短いことを指します。処理時間が短いことは重要で、解析の試行錯誤回数に直接的に影響します。処理時間が長くなると、1日にできる試行錯誤回数が極端に減ってしまい、適切な解析条件を見つける事が出来ないかも知れません。

より深い効率化のためには、Stataコマンドの特性を知っておく必要があります。最も簡単な効率化は、compressを行うことです。

|  |  |  |
| --- | --- | --- |
| コマンド | 用途 | 文法 |
| compress | 圧縮できる変数をまとめて圧縮する | compress |

また、変数の作成を行う際には、出来るだけegen（処理が遅い）の使用をできるだけ避けると良いです。データセットの読み書きはHDDやSSDへのアクセスを伴い、時間がかかります。これも避け、frameコマンド、preserve-restoreコマンドを使うことを検討します。

### コード例4

sysuse nlsw88, clear  
  
/\* ageの偏差（平均値との差）の算出 \*/  
egen ave\_age = mean(age)  
gen dev\_age1 = age – ave\_age  
  
/\* 上記と同じだが、効率的 \*/  
su age  
gen dev\_age2 = age – r(mean)