The Grammar of Graphics

Data / Variables

제 3장 데이터(Data)

개념들

- 데이터(Data)
- 플랫 파일(flat files)
- 데이터 소스(Data Source)
- 뷰(View)
- 스트리밍(Streaming)

데이터(Data)

- 라틴어 어원: 주어진 것

데이터(Data)

We ordinarily think of data as derived from measurements from a machine, survey, census, test, rat- ing, or questionnaire – most frequently numerical.

In a more general sense, however, data are symbolic representations of observations or thoughts about the world.

– The Grammar of Graphics 41p

플랫 파일(flat files)

- 과거의 그래픽 시스템에서 사용하던 형식
- 행열로 이루어진 숫자 변수들을 저장한 파일
- 데이터의 형식에 의해 차트 형태가 결정
  - 꺾은 선 그래프(line charts)
  - 막대 그래프(bar charts)
  - 파이 차트(pie charts)
  - 산포도(scatterplots)

데이터 소스(Data Source)

- 객체 지향 그래픽 시스템에서 사용
- 데이터는 Data Source의 일부
- 기저의 데이터 형식에 대해 가정하지 않음

뷰(View)

- 그래프와의 맵핑하는 데 사용
- 데이터 소스와 어떤 관계를 가지고 있음
- **스트리밍(Streaming)**되는 데이터 소스에 대응
    - 뷰는 시시각각 변화할 수 있어야한다
    - 기저 데이터의 이해를 위해 고정된 뷰는 피한다

세가지 데이터 형식

- 실증적 데이터(empirical)
- 추상적 데이터(abstract)
- 메타 데이터(meta)

데이터 함수(Data Functions)

데이터 함수(Data Functions)

- 데이터셋으로부터 변수(variables)를 만드는 함수
  - `function(dataset) -> variables`
- 객체지향 데이터베이스와 함께 사용할 수 있음
- 참조하는 스킴과 실제 데이터와는 무관하다

데이터 함수(Data Functions)

Table 3.1 **Data Functions**

Empirical Data	*Abstract Data*	*Metadata*
col(*source*(), *name*(), *unit*(), *weight*()) *map*(*source*(), *id*()) *stream*(*source*(), *id*()) *image*(*source*()) *sample*(x, *n*) *reshape*(x_1,...,x_n, "<*index*>")	*iter*(*from*,*to*,*step*) *mesh*(*min*, *max*, *n*) *count*(*n*) *proportion*(*n*) *percent*(*n*) *constant*(*c*, *n*) *string*("<*string*>", *n*) *rand*(*n*)	*meta*(*source*(), *name*())

실증적(Empirical) 함수

- 관찰된 데이터의 열을 조작하는 하는 함수

`col(source(), name(), unit(), weight())`

- 데이터 소스의 열을 변수와 연결한다
- 그래프 스펙(GPL 표현) 상에서 이 함수가 사용되지 않았다면 열이름과 변수명이 같다고 추정
- *unit()*: 단위
- *weight()*: 통계적 계산을 위한 가중치 지정

map, stream, image

- `map(source(), id(()))`
  - 지도
- `stream(source(), id(()))`
  - 데이터 스트리밍
- `image(source())`
  - 이미지 소스

sample(x,n)

- 샘플링 기법들을 구현
  - `sample.srs`(simple random)
  - `sample.jackknife`(Tukey, 1958)
  - `sample.boot`(Efrom & Tibshirani, 1993)

$$reshape(x_1, \ldots x_n, \text{“}< index >\text{”}) \textbf{(1)}$$

- 행렬이나 표를 하나의 변수(컬럼)으로 변환
  - $\mathbf{X}_{m \times n}$ ($x_1 \ldots x_n$ columns)
  - $i, j = row\ and\ column\ indices$
  - $\mathbf{X}(i = 1, \ldots m\ and\ j = 1, \ldots n)$

$reshape(x_1, ... x_n, \text{“}< index >\text{”})$ **(2)**

- Let k be the row index of the variable × output by the reshape() function
- $reshape.rect() : k = n \cdot (i-1) + j$
- $reshape.tri() : k = i \cdot (i-1)/2 + j : (i \geq j)$
- $reshape.low() :$
  - $k = (i-1) \cdot (i-2)/2 + j : (i > j)$
- $reshape.diag() : k = i : (i = j)$

$reshape.rect()$

- $reshape.rect() : k = n \cdot (i - 1) + j$
- $\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- $t(\times) = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9]$

$reshape.tri()$

- $reshape.tri() : k = i \cdot (i-1)/2 + j : (i \geq j)$
- $\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- $t(\times) = [1 \quad 4 \quad 5 \quad 7 \quad 8 \quad 9]$

$reshape.low()$

- $reshape.low()$
  - $k = (i-1) \cdot (i-2)/2 + j : (i > j)$
  - $\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
  - $t(\times) = \begin{bmatrix} 4 & 7 & 8 \end{bmatrix}$

$reshape.diag()$

- $reshape.diag() : k = i : (i = j)$
- $\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
- $t(\times) = \begin{bmatrix} 1 & 5 & 9 \end{bmatrix}$

$reshape(x_1, ... x_n, \text{“}< index >\text{”})$ **(3)**

- $< index >$에 들어갈 수 있는 값
  - value: the value of the entries
  - rowindex: the row index
  - colindex: the column index
  - rowname: the row name
  - colname: the column name

추상적 함수(Abstrcat Functions)

- 열(columns)를 생성하는 함수
- 행수(n)을 인자로 받는 경우가 많다
  - n이 없으면 데이터셋의 행수를 n으로 사용

iter(from,to,step)

- from에서 to까지 step 간격을 가진 seqence를 생성
- `count(10) = iter(1, 10, 1)`
- `proportion(10) = iter(.1, 1.0, .1)`
- `percent(n):`

`mesh(min, max, n)`

- `mesh(min, max, n)`:
  - 1D, 2D, 3D mesh 계산

constant, string

- constant(c, n):
  - n 개의 c 생성
- string("<string>", n):
  - n개의 문자열 생성
- DATA: s = string("Hello world")
- DATA: s = "Hello world" (축약표현)

rand(n)

- rand(n):
  - 독립적 무작위 숫자 생성
  - rand.uniform()
  - rand.normal()

메타데이터 함수

- `meta(source(), name())`:
  - 데이터셋의 행과 source()를 연결시키는 함수
  - source(): 비디오, 이미지, 웹주소 등등

실증적 데이터(Empirical Data)

Empirical

- 그리스어 어원: 경험, 무엇과 가까워지다

실증적 데이터를 바라보는 두 견해

- Realist(실재론자)
  - 데이터란 잠재적 현상들의 현시(manifestations)
  - 데이터는 보편적이며 기본적인 사실을 가리킴
- Nominalist(유명론자)
  - 데이터란 데이터가 보여주는 것
  - 루트비히 포이어바흐 - 당신은 당신이 먹는 것

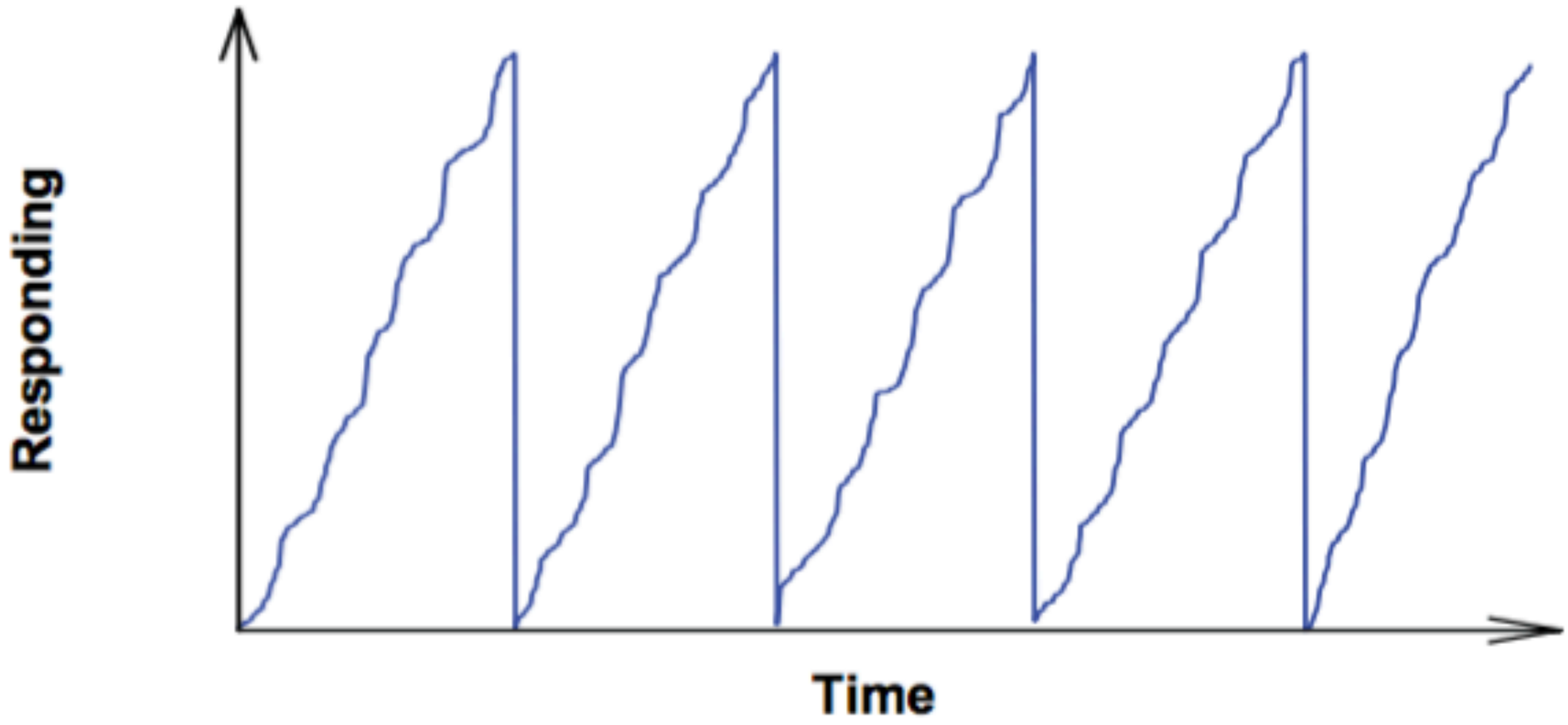

Figure 3.1 *Cumulative record*

Reshaping Data

Correlation Matrix

ex)

```
       mpg   cyl  disp    hp  drat    wt
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78
disp -0.85  0.90  1.00  0.79 -0.71  0.89
hp   -0.78  0.83  0.79  1.00 -0.45  0.66
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71
wt   -0.87  0.78  0.89  0.66 -0.71  1.00
```

```
reshape.low(mpg,cyl,disp,hp,drat,wt)
```

```
-0.85
-0.85
0.90
-0.78
0.83
0.79
0.68
-0.70
-0.71
-0.45
-0.87
0.78
0.89
0.66
-0.71
```

DATA: r = *reshape.low*(pounding, sinking, shaking, nauseous, stiff,
faint, vomit, bowels, urine, "value")
ELEMENT: *point.dodge.asymmetric*(*position.bin.dot*(r))

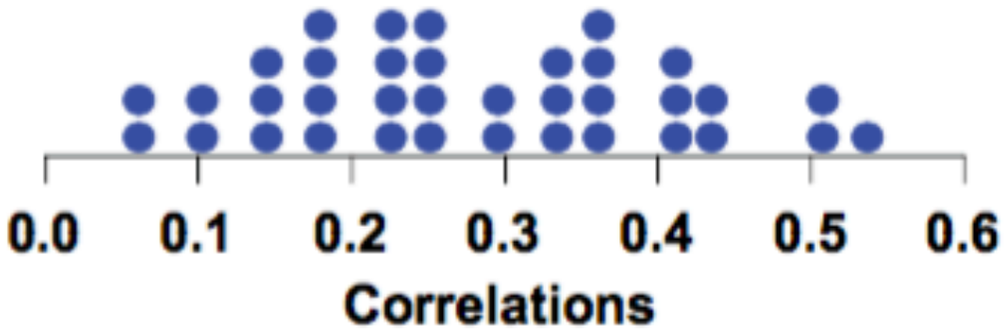

Figure 3.2 *Dot plot of correlations from correlation matrix*

```
reshape.tri(mpg,cyl,disp,hp,drat,wt)

1.00
-0.85
1.00
-0.85
0.90
1.00
-0.78
0.83
0.79
1.00
0.68
-0.70
-0.71
-0.45
1.00
-0.87
0.78
0.89
0.66
-0.71
```

```
DATA: row = reshape.tri(pounding, sinking, shaking, nauseous,
        stiff, faint, vomit, bowels, urine, "rowname")
DATA: col = reshape.tri(pounding, sinking, shaking, nauseous,
        stiff, faint, vomit, bowels, urine, "colname")
DATA: r = reshape.tri(pounding, sinking, shaking, nauseous, stiff,
        faint, vomit, bowels, urine, "value")
ELEMENT: polygon(position(bin.rect(col*row)), color.hue(r))
```

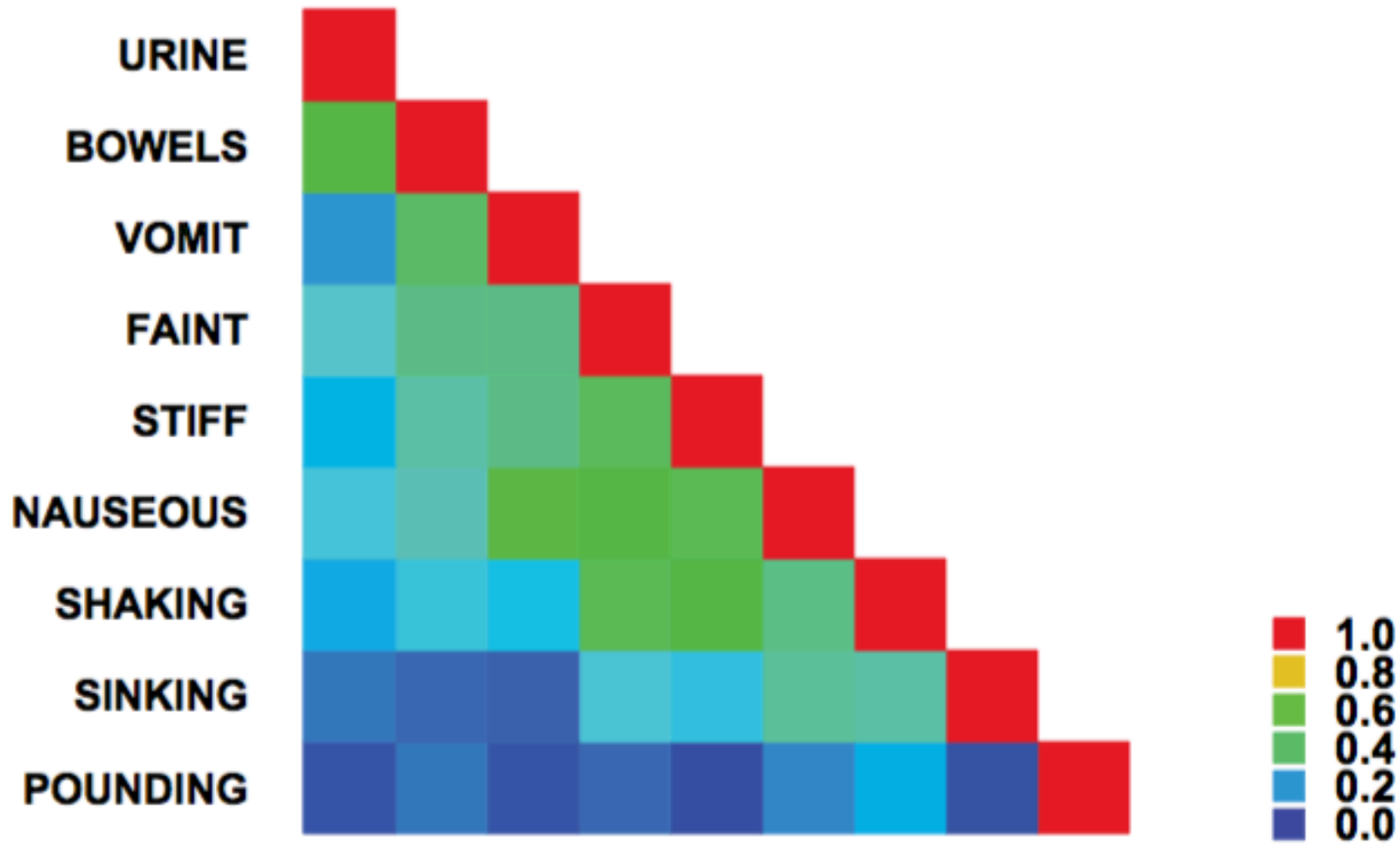

Figure 3.3 *Correlation matrix of combat symptoms*

ELEMENT: *interval*(*position*(*summary.count*(*bin.rect*(military, *dim*(1)))))

DATA: mean = *sample.boot*(military, 1000, "mean")
ELEMENT: *interval*(*position*(*summary.count*(*bin.rect*(mean, *dim*(1)))))

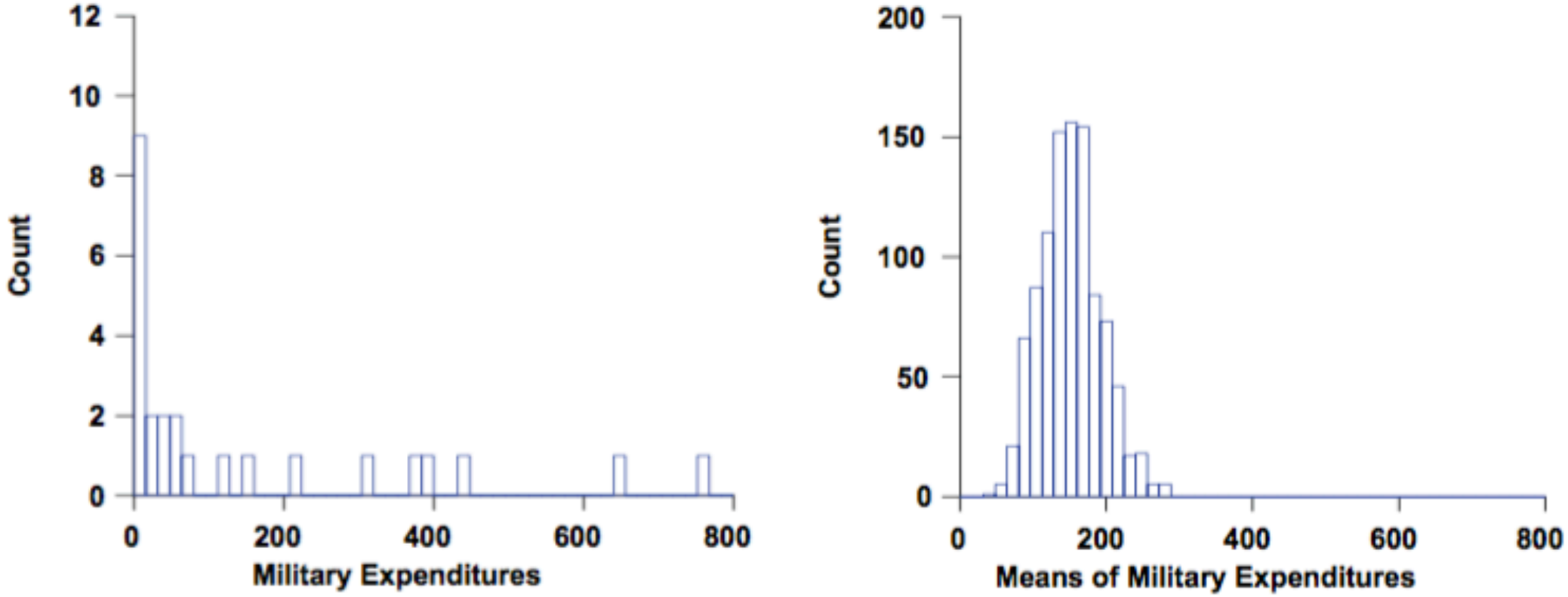

Figure 3.4 *Bootstrapped means*

DATA: **case** = *iter*(1, 256, 1)
ELEMENT: *line*(*position*(**case***rate**))

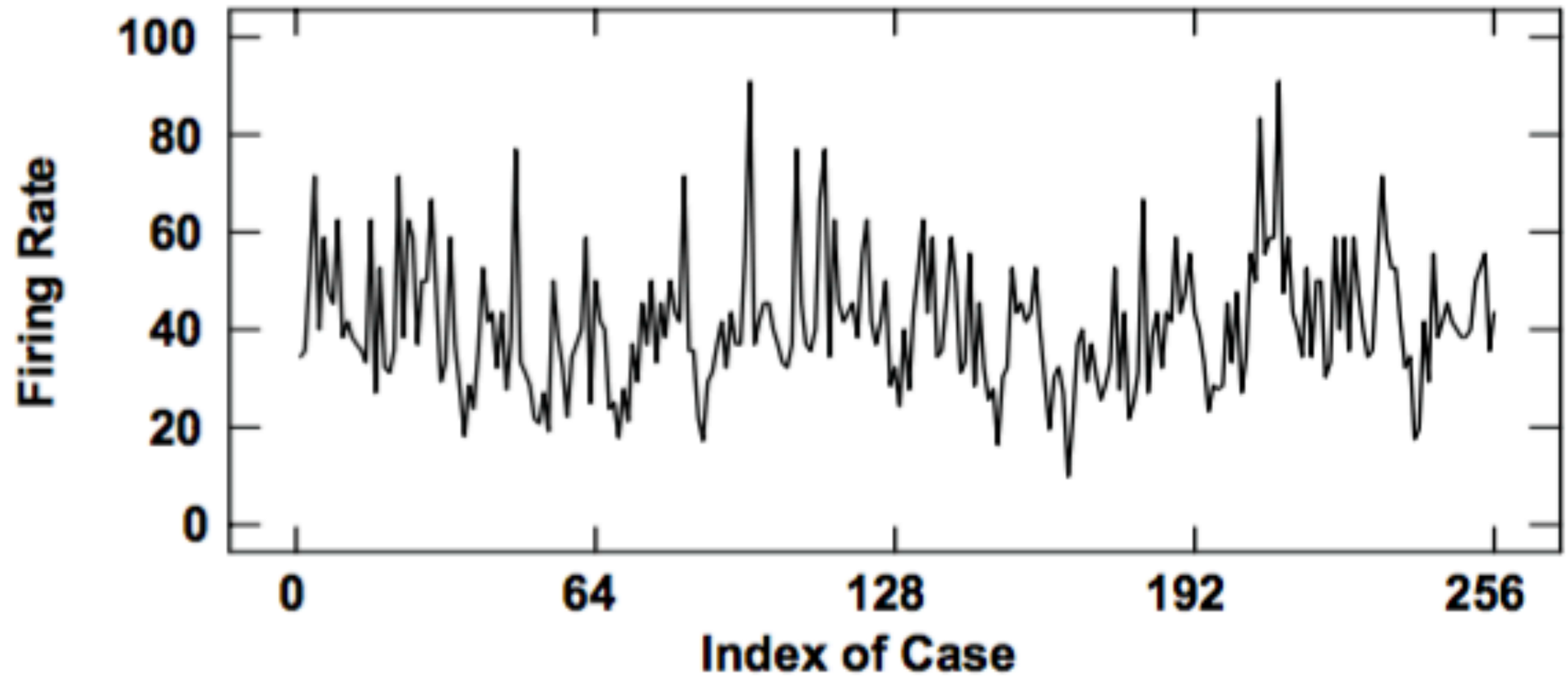

Figure 3.5 *Firing rate of cat retinal cell ganglion*

DATA: **z** = *constant*(1)
COORD: *rect*(*dim*(1, 2, 3))
ELEMENT: *interval*(*position*(*summary.count*(gov*urban*z)))

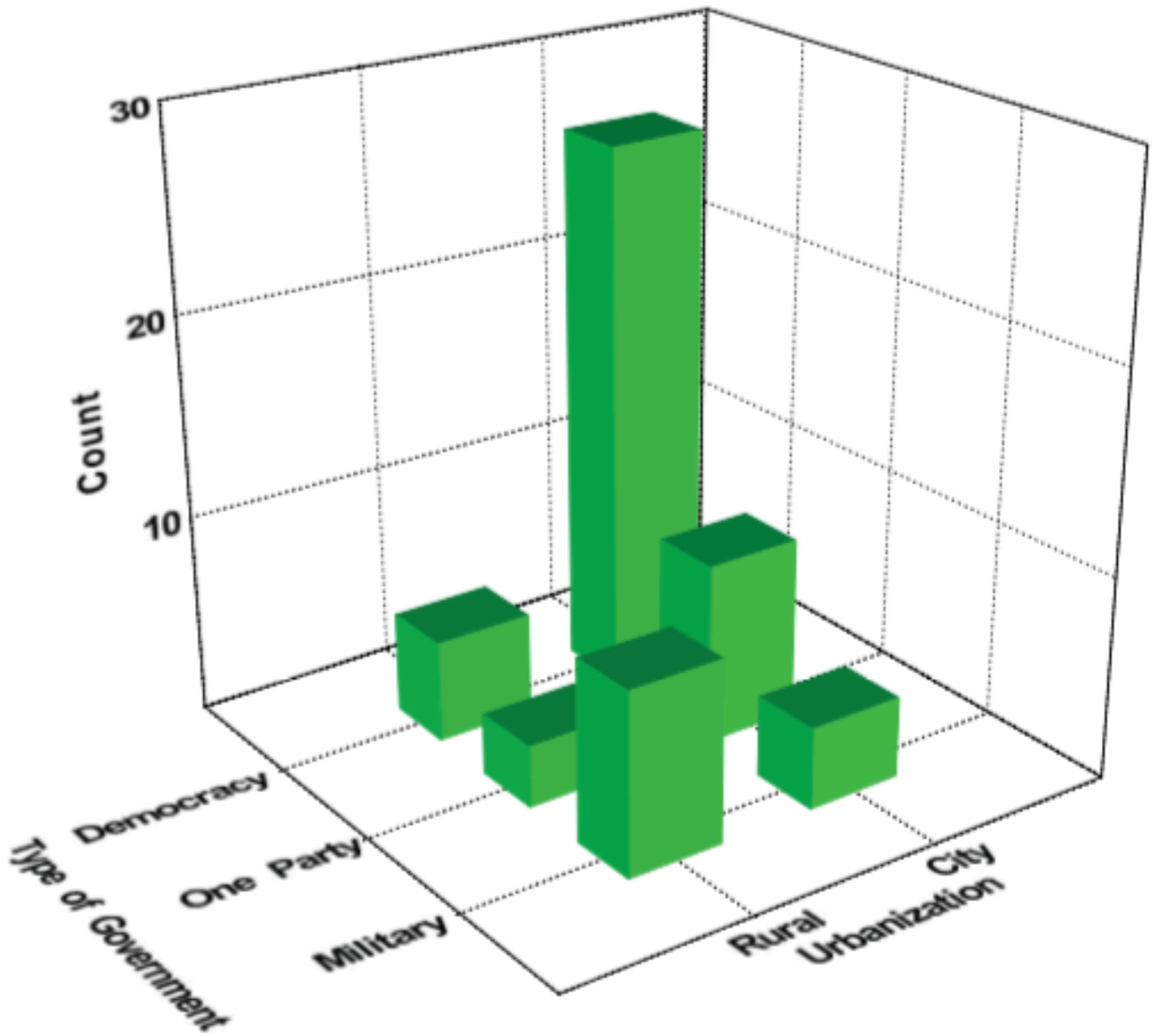

Figure 3.6 *Count bar chart*

DATA: **x, y** = *mesh*(*min*(–5, –5), *max*(5, 5))
TRANS: **z** = (**x**^2+**y**^2)/*sqrt*(**x**^2+**y**^2
COORD: *rect*(*dim*(1, 2, 3))
ELEMENT: *surface*(*position*(**x***y***z**))

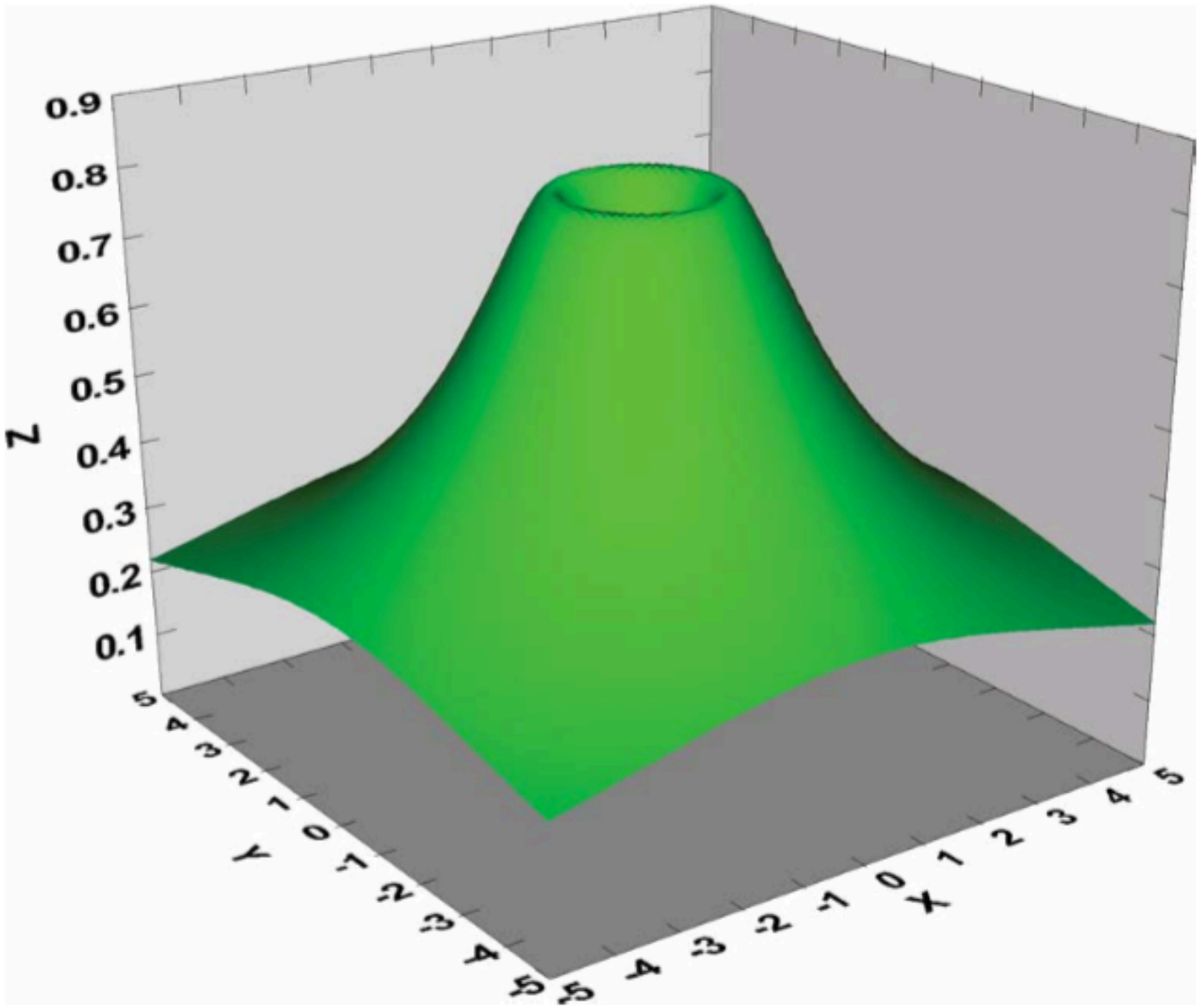

Figure 3.7 *Automated function plot*

추상적 데이터(Abstract Data)

메타데이터(Metadata)

데이터 마이닝(Data Mining)

데이터 마이닝(Data Mining)

- MOLAP
- ROLAP
- Visual Query of Databases

제 4장 변수(Variables)

Variable (1)

- 라틴어 어원: variare, 다양하게 만들다
- 개념과 데이터를 연결 (`income`)
- 변수들 간의 연산 가능 (`death * birth`)

Variable (2)

- 기존 통계 그래픽 시스템
  - 케이스(행) * 변수(열)
  - 행은 변수의 인스턴스나 샘플
- 새로운 정의
  - 행을 나타내건, 열을 나타내건 상관없음
  - 단, 변수 맵핑 함수는 모든 인덱스에 대해 하나의 값을 반환해야한다

변환(Transforms)

- 변수를 변환하는 함수
- 변수를 적절하고 의미있게 만들기 위한 통계적인 조작
- 또는 새로운 변수 생성, 집계 합수, 요약

Table 4.1 **Variable Transforms**

Mathematical	*Statistical*	*Multivariate*
log(**x**)	*mean*(**x**)	*sum*($\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_n$)
exp(**x**)	*median*(**x**)	*diff*($\mathbf{x}_1$,$\mathbf{x}_2$)
sin(**x**)	*mode*(**x**)	*prod*($\mathbf{x}_1$,$\mathbf{x}_2$)
cos(**x**)	*residual*(**x**,**y**)	*quotient*($\mathbf{x}_1$,$\mathbf{x}_2$)
tan(**x**)	*sort*(**x**)	*influence*($\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_n$)
asin(**x**)	*rank*(**x**)	*miss*($\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_n$,"<*f*>")
acos(**x**)	*prank*(**x**)	
atan(**x**)	*cut*(**x**,*k*)	
atanh(**x**)	*zinv*(**x**)	
sign(**x**)	*lag*(**x**)	
pow(**x**, *p*)	*grpfun*(**x**,**g**,"<*f*>")	

수학 함수(Mathematical Functions)

- `log(x)`
- `exp(x)`
- `sin(x), asin(x)`
- `cos(x), acos(x)`
- `tan(x), atan(x)`
- `atanh(x)`
- `sign(x)`

sin(x)

```
> sin(seq(1, 30, 1))
 [1]  0.841470985  0.909297427  0.141120008 -0.756802495 -0.958924275
 [6] -0.279415498  0.656986599  0.989358247  0.412118485 -0.544021111
[11] -0.999990207 -0.536572918  0.420167037  0.990607356  0.650287840
[16] -0.287903317 -0.961397492 -0.750987247  0.149877210  0.912945251
[21]  0.836655639 -0.008851309 -0.846220404 -0.905578362 -0.132351750
[26]  0.762558450  0.956375928  0.270905788 -0.663633884 -0.988031624
```

통계 함수(Statistical Functions)

- `mean(x), median(x), residual(x,y)`
- `residual(x, y)`
- `sort(x)`
- `rank(x), prank(x)`
- `cut(x,k)`
- `zinv(x)`
- `lag(x)`

rank(x)

```
> rank(sample(1:10, 10, replace=T))
[1]  2.5  4.5  4.5  6.5  1.0  8.5  2.5  8.5 10.0  6.5
```

prank(x)

```
> (seq(1, 100) - 0.5) / 100
  [1] 0.005 0.015 0.025 0.035 0.045 0.055 0.065 0.075 0.085 0.095 0.105 0.115
 [13] 0.125 0.135 0.145 0.155 0.165 0.175 0.185 0.195 0.205 0.215 0.225 0.235
 [25] 0.245 0.255 0.265 0.275 0.285 0.295 0.305 0.315 0.325 0.335 0.345 0.355
 [37] 0.365 0.375 0.385 0.395 0.405 0.415 0.425 0.435 0.445 0.455 0.465 0.475
 [49] 0.485 0.495 0.505 0.515 0.525 0.535 0.545 0.555 0.565 0.575 0.585 0.595
 [61] 0.605 0.615 0.625 0.635 0.645 0.655 0.665 0.675 0.685 0.695 0.705 0.715
 [73] 0.725 0.735 0.745 0.755 0.765 0.775 0.785 0.795 0.805 0.815 0.825 0.835
 [85] 0.845 0.855 0.865 0.875 0.885 0.895 0.905 0.915 0.925 0.935 0.945 0.955
 [97] 0.965 0.975 0.985 0.995
```

zinv(x)

```
> qnorm((seq(1, 50) - 0.5) / 100)
 [1] -2.57582930 -2.17009038 -1.95996398 -1.81191067 -1.69539771 -1.59819314
 [7] -1.51410189 -1.43953147 -1.37220381 -1.31057911 -1.25356544 -1.20035886
[13] -1.15034938 -1.10306256 -1.05812162 -1.01522203 -0.97411388 -0.93458929
[19] -0.89647336 -0.85961736 -0.82389363 -0.78919165 -0.75541503 -0.72247905
[25] -0.69030882 -0.65883769 -0.62800601 -0.59776013 -0.56805150 -0.53883603
[31] -0.51007346 -0.48172685 -0.45376219 -0.42614801 -0.39885507 -0.37185609
[37] -0.34512553 -0.31863936 -0.29237490 -0.26631061 -0.24042603 -0.21470157
[43] -0.18911843 -0.16365849 -0.13830421 -0.11303854 -0.08784484 -0.06270678
[49] -0.03760829 -0.01253347
```

다변량 함수(Multivariate Functions)

- $sum(x_1, x_2, \ldots, x_n)$
- $diff(x_1, x_2)$
- $prod(x_1, x_2)$
- $quotient(x_1, x_2)$
- $influence(x_1, x_2, \ldots, x_n)$
- $miss(x_1, x_2, \ldots, x_n, "< f >")$

정렬

- 시카고의 투표 조작을 밝혀내는 방법
  - 모든 투표자의 테입을 정렬해놓고
  - 중복된 이름과 주소를 찾음

- 정렬
  - one-to-one transformation
  - 변수의 패턴을 드러냄
  - 서브셋들의 비교를 쉽게 해줌

범죄자 성별 비율 그래프

TRANS: total = *sum*(male, female)
TRANS: m = *quotient*(male, total)
TRANS: f = *quotient*(female, total)
TRANS: mf = *diff*(m, f)
TRANS: mf = *sort*(mf)
ELEMENT: *point*(*position*(mf*crime))

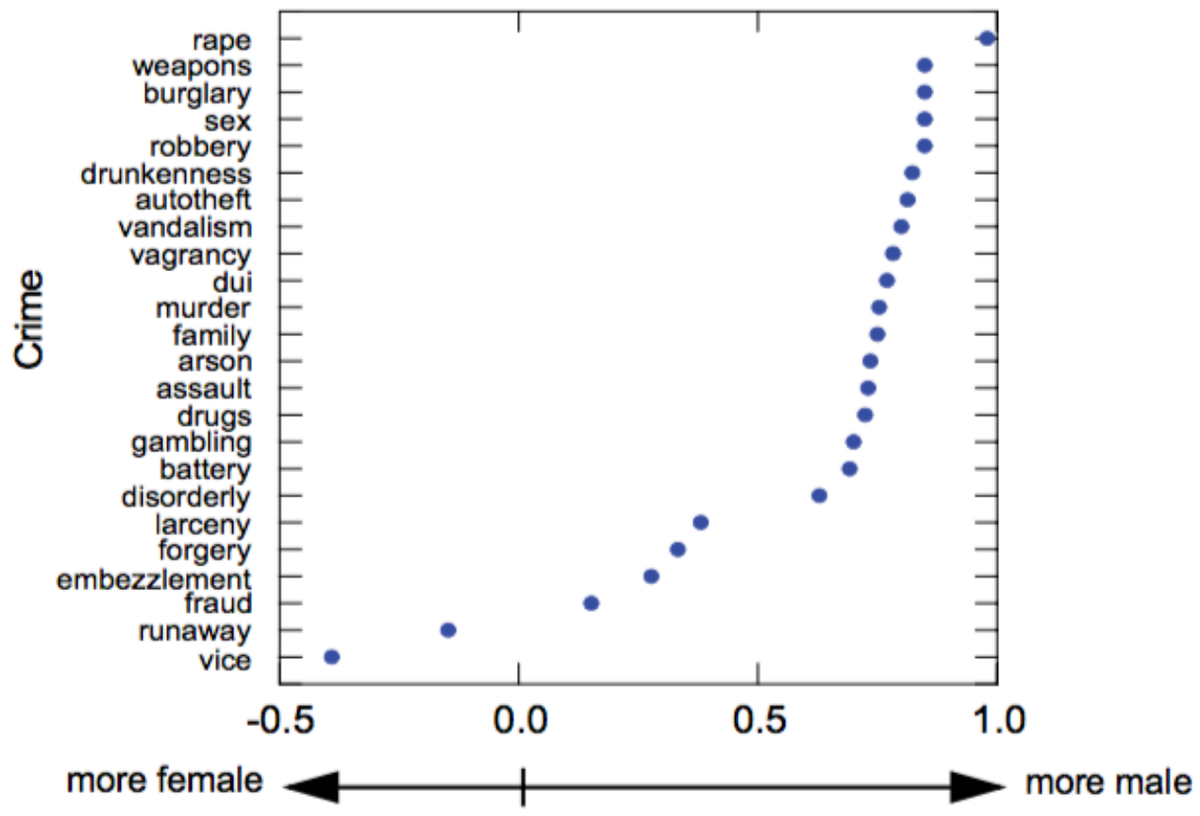

Figure 4.1 *Gender differences in crime patterns*

Demo

Notebook Link

TRANS: alpha = *prank*(military)
TRANS: z = *zinv*(alpha)
ELEMENT: *point*(*position*(military*z))

TRANS: alpha = *prank*(military)
TRANS: z = *zinv*(alpha)
SCALE: *log*(*dim*(1), *base*(10))
ELEMENT: *point*(*position*(military*z))

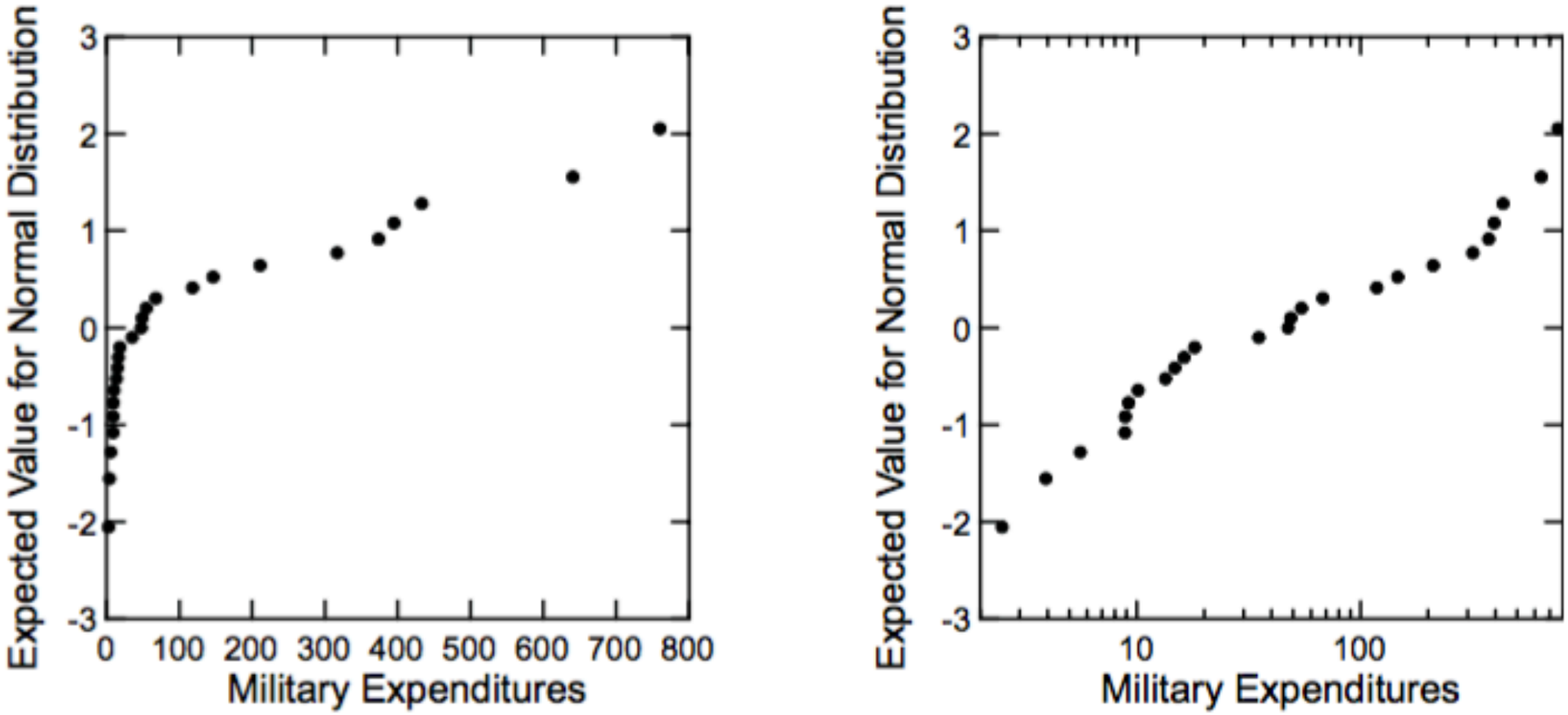

Figure 4.2 *Probability plots of military expenditures*

TRANS: quartile = *cut*(birth, 4)
TRANS: birthquart = *grpfun*(birth, quartile,"median")
ELEMENT: *schema*(*shape*(*shape.box*), *position*(birthquart*birth))

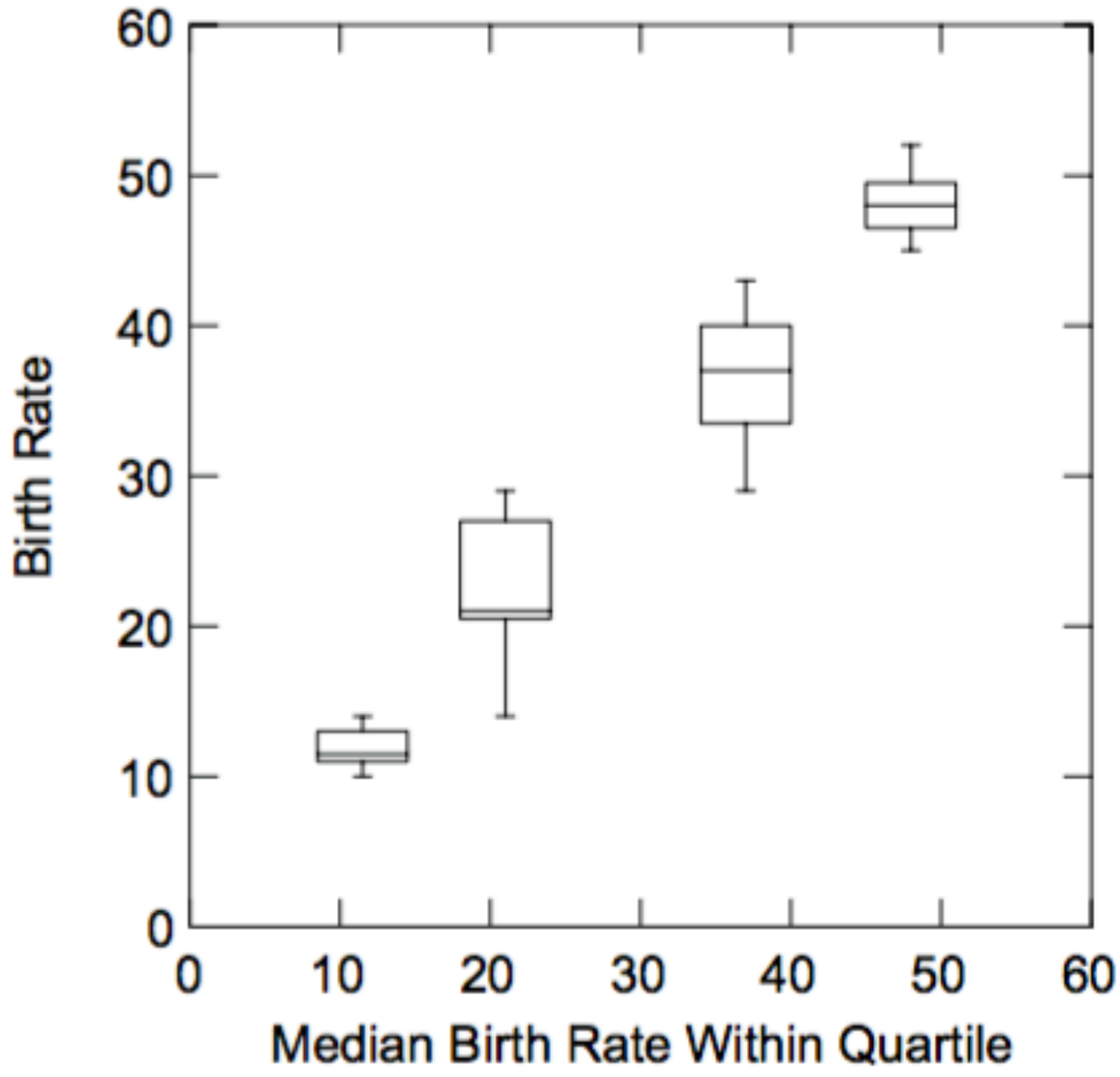

Figure 4.3 *Box plot of a variable against its quartile medians*

TRANS: **residual** = *residual.linear.student*(**birth, death**)
ELEMENT: *point*(*position*(**birth*residual**))

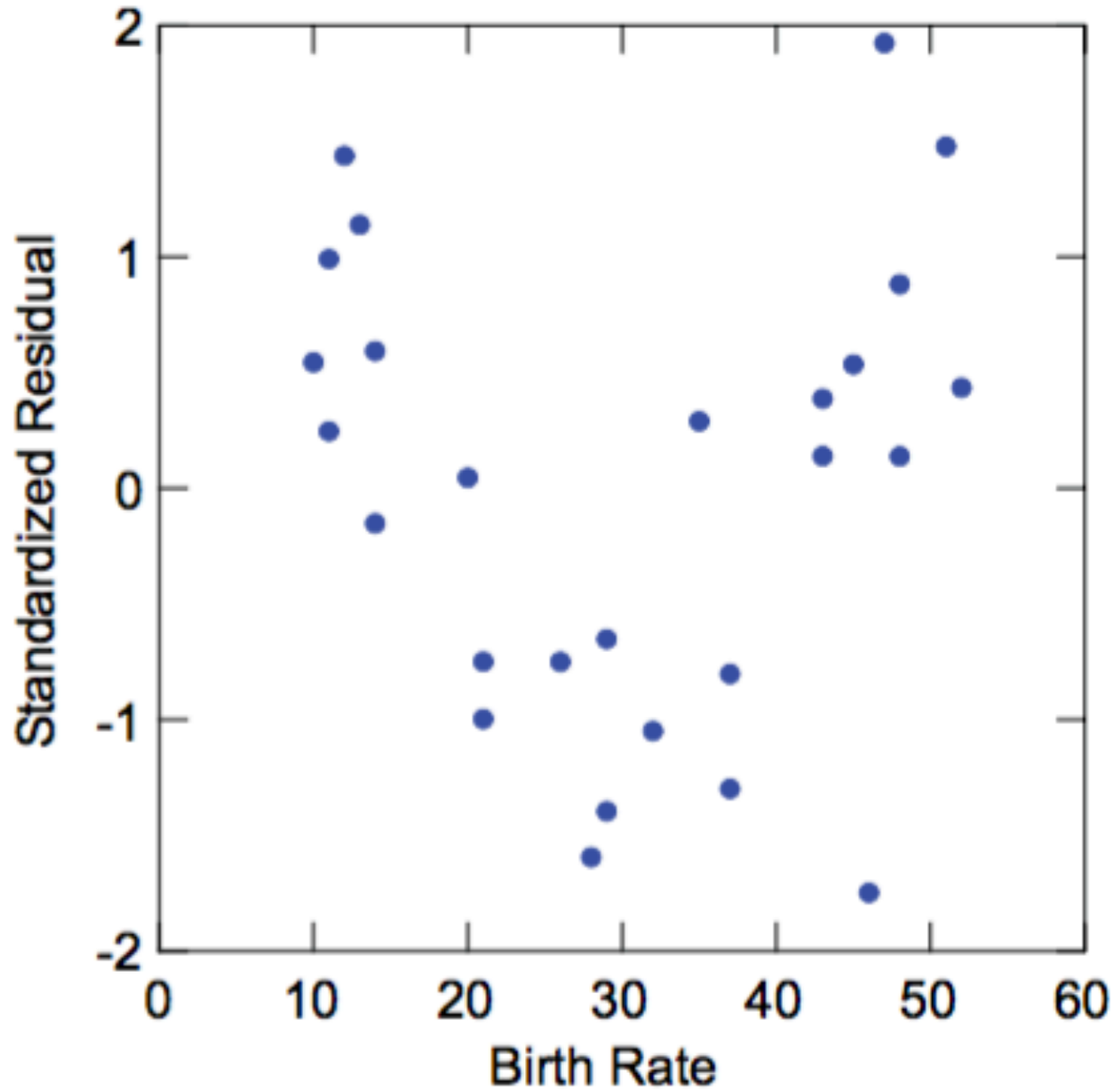

Figure 4.4 *Studentized residual plot*