The Grammar of Graphics

Algebra

정보시각화 고전 독서 모임 5번째

2015. 8. 18.

@nacyo_t

가 각 간 갓

나 낙 난 낫

다 닥 단 닷

ㄱ ㄴ ㅅ

(ㄱ,ㄴ,ㅅ)×(가,나,다)

낚옷

2.1.8 Variable

a variable X is $X = [O, V, f]$

- the domain O is a set of objects,
- the codomain V is a set of values,
- the function f assigns to each element of O an element in V

2.1.8 변수(Variable)

변수 X는 $X = [O, V, f]$

- 정의역 O은 객체들의 집합
- 공역 V은 값들의 집합
- 함수 f는 O의 각 값들을 V의 요소에 대응시킨다.

O

a set of objects

$$\{1, 2, 3, 4, 5\}$$

$$V$$

a set of values

$$\{ant, bee, drone, moth, locust\}$$

$$f$$

$$f(O_i) = V_i$$

Variable X

$$[\{1, 2, 3, 4\}, \{ant, bee, drone, moth\}, f]$$

- 테이블 표현과 비슷
  - 1 -> ant
  - 2 -> bee
  - 3 -> drone
  - 4 -> moth

2.1.9 Varsets

a varset X is $\mathsf{X} = [V, \tilde{O}, f]$

- the domain V is a set of values,
- the codomain $\tilde{O}$ is a set of all possible bags of objects,
- the function f assigns to each element of V an element in $\tilde{O}$

2.1.9 Varsets

varset X는 $\mathsf{X} = [V, \tilde{O}, f]$

- 도메인 V는 값들의 집합
- 공역 $\tilde{O}$는 모든 가능한 백(bags)들의 집합
- 함수 f는 V의 각 값들을 $\tilde{O}$에 대응시킨다

$$V$$

a set of Values

$$\{red, blue\}$$

$$\tilde{O}$$

set of all possible bags of objects,

$$\{<\cdot>, <\cdot, \cdot>, \cdots\}$$

$$f$$

$$red \rightarrow < 1, 2 >$$

$$blue \rightarrow < 3 >$$

다른 Variable Y 예제

$$[\{1, 2, 3\}, \{red, blue\}, f]$$

- 1 -> red
- 2 -> red
- 3 -> blue

Varset Y

$$[\{red, blue\}, \{<1>, <2>, <3>, <1,2>, <2,3>, <3,1>, <1,2,3>\}, f]$$

- red -> <1,2>
- blue -> <3>

Varset은

Variable의 값들로 인덱스된 형태

Table

KEY	*VALUE*
1	*red*
2	*red*
3	*blue*

$$Variable \approx Varset \approx Table(Column)$$

$$Y = [\{1, 2, 3\}, \{red, blue\}, f_1]$$

$$\mathbf{Y} = [\{red, blue\}, \{< \cdot >, < \cdot, \cdot >, \ \ldots\}, f_2]$$

$$Table\ Y = \begin{array}{ll} \underline{KEY} & \underline{VALUE} \\ 1 & red \\ 2 & red \\ 3 & blue \end{array}$$

2. How To Make a Pie

2.2 Recipe

Variables(32p)

Table 2.1 **ACLS Database Table**

CaseID	Gender	Bias in favor of males	...

We can extract the data we need with the following queries:

```
Response = loadFromSQL("ACLS", "bias_toward_males", "case")
Gender = loadFromSQL("ACLS", "Gender", "case")
```

Variables 재정의

Data Definition Statements

```
SOURCE: <source name> = <fn>(<args>)
DATA: <variable name> = <fn>(<args>)
TRANS: <variable name> = <fn>(<args>)
```

SOURCE

```
SOURCE: acls_data = csvSource(path("C:\\acls.csv"))
```

csv file(이 존재한다고 가정)

```
CaseID, Gender, Response, ...
1, Male, Frequently, ...
2, Female, Not Sure, ...
3, Male, Frequently, ...
...
3834, Male, Rarely, ...
3835, Female, Infrequently, ...
```

Source Table

CaseID	*Gender*	*Response*	...
1	*Male*	*Frequently*	...
2	*Female*	*NotSure*	...
3	*Male*	*Frequently*	...
...	...	...	...
3834	*Male*	*Rarely*	...
3835	*Female*	*Infrequently*	...

DATA

```
SOURCE: acls_data = csvSource(path("C:\\acls.csv"))
DATA: gender = col(source(acls_data), name("Gender"))
DATA: response = col(source(acls_data), name("Response"))
```

col 함수를 통해서 gender, responce 변수 할당

gender (1)

$$Variable\ gender = [\{1, 2, \ldots, 3835\}, \{Male, Female\}, f_1]$$

$$Varset\ \textsf{gender} = [\{Male, Female\}, \{\ldots\}, f_2]$$

gender (2)

```
select CaseID, Gender from acls_data
```

$$Table\ gender = \begin{array}{ll} CaseID & Gender \\ \hline 1 & Male \\ 2 & Female \\ 3 & Male \\ \dots & \dots \\ 3834 & Male \\ 3835 & Female \end{array}$$

response (1)

$$Variable\ responese = [\{1, 2, \ldots, 3835\}, \{R, I, O, F, N\}, f_1]$$

$$Varset\ \textsf{response} = [\{R, I, O, F, N\}, \{\ldots\}, f_2]$$

response (2)

```
select CaseID, Response from acls_data
```

$$Table\ response = \begin{array}{ll} CaseID & Response \\ \hline 1 & Frequently \\ 2 & NotSure \\ 3 & Frequently \\ \dots & \dots \\ 3834 & Rarely \\ 3835 & Infrequently \\ \hline \end{array}$$

Algebra

Varset Algebra

Algebra와 Frames (1)

Graphics algebra provides a method for specifying F(which we call a frame) when we wish to construct agraphic based on some function of a set of data.

– nVIZn 3p

Algebra와 Frames (2)

This chapter deals with restoring and balancing sets of variables in order to create the specification for the frames in which graphs are embedded.
– GoG Ch5. Algebra 63p

Frame이란?

A frame is a set of tuple $(x_1, \ldots, x_p)$ *ranging over all possible values in the domain of a p-dimensional varset.*

– GoG 30p

그래프가 그려지는 공간을 정의. (2개 varset이 있을 때는 일반적으로 xy평면과 같은 2차원 평면을 정의)

Frame Specification

- 1판에서는 frame spcification이 있었음(40p)

```
FRAME: x * y
GRAPH: point()
```

- 2판의 표현법(39p)

```
ELEMENT: point(position(x * y))
```

Symbol

A symbol is used to represent an entity operated on by an algebra. The symbols in varset algebra are varsets.

심볼은 대수식에서 계산되어지는 항목을 나타낸다.
varset 대수식에서 심볼은 varset이다.

Operators

An operator is a method for relating symbols in an algebra.

연산자는 대수식에서 심볼들을 관계짓는 메서드이다.

- Cross(*), Nest(/), Blend(+)
- Varsets(**A** and **B**)
- Value Domain of Varsets(V_{A} and V_{B})

Cross: *

Cross(*) 수학적 정의

Varset 정의

$$\mathrm{X} = [V, \tilde{O}, f]$$

Cross 정의

$$A * B : (s, t) \rightarrow A(s) \cap B(t)$$

$$V_{A*B} = \{(s, t), \forall s \in V_A, \forall t \in V_B\}$$

테이블(Variable) 표현으로 이해하기

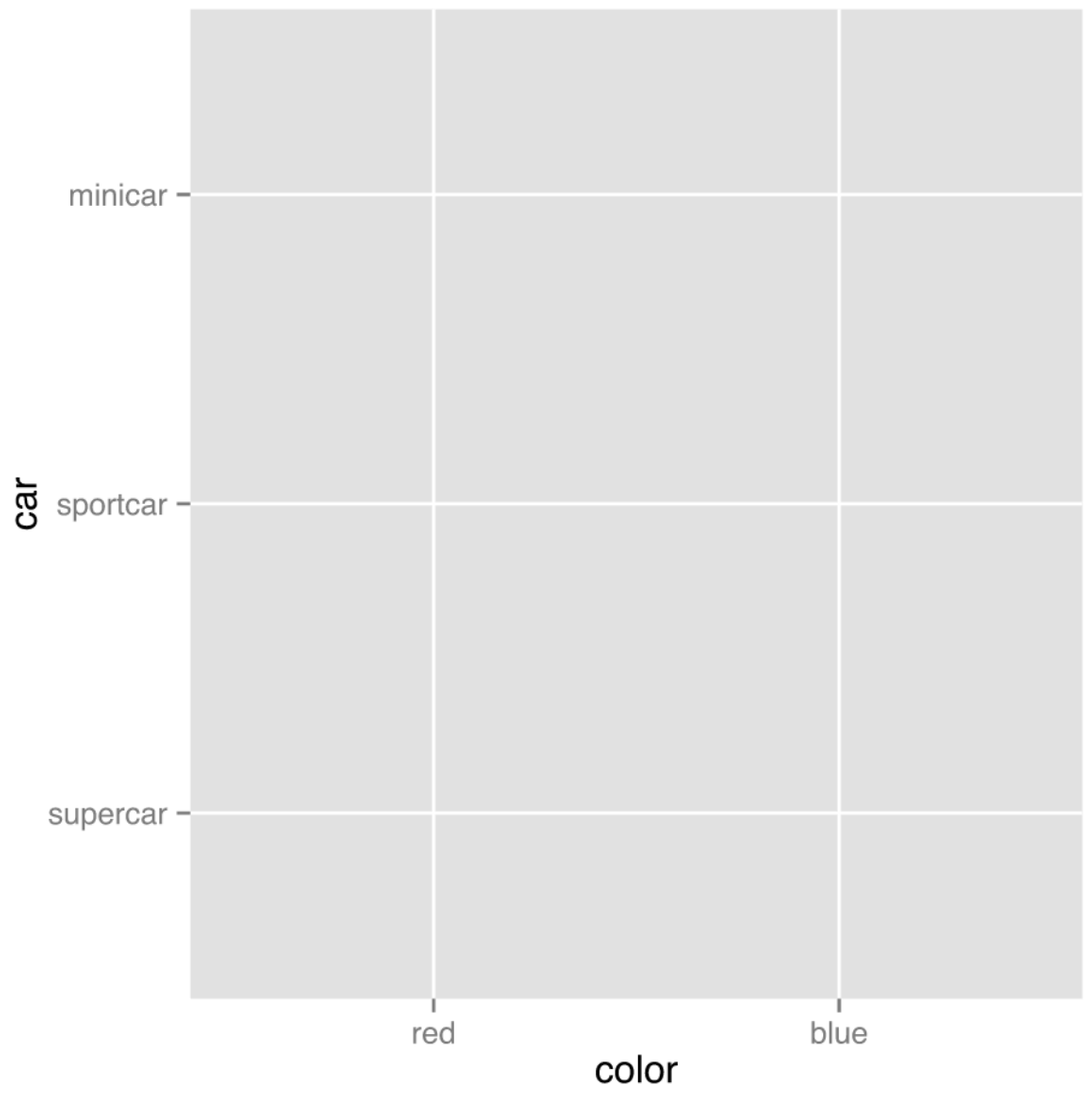
$$color = \begin{array}{cc} key & color \\ \hline 1 & red \\ 2 & blue \\ 3 & blue \\ 4 & red \end{array} \quad , \quad car = \begin{array}{cc} key & response \\ \hline 1 & Supercar \\ 2 & Sportcar \\ 3 & Minicar \\ 4 & Supercar \end{array}$$

$$V_{\text{color*car}} \approx Frame$$

```
SELECT DISTINCT color.color, car.car
FROM color CROSS JOIN car;
```

```
color       car
----------  ----------
red         supercar
red         sportcar
red         minicar
blue        supercar
blue        sportcar
blue        minicar
```

$Frame\ of$ (color $*$ car)



$f : V \rightarrow \tilde{O}$ (1) sql

```
SELECT
  ('(' || color.color || ', ' || car.car || ')')
    AS "Value Domain",
  GROUP_CONCAT(
    CASE WHEN color.key = car.key THEN color.key ELSE NULL END
  ) AS "Set of Cases"
FROM color CROSS JOIN car
GROUP BY "Value Domain";
```

$f : V \rightarrow \tilde{O}$ (2) result

Value Domain	Set of Cases
(blue, minicar)	3
(blue, sportcar)	2
(blue, supercar)	
(red, minicar)	
(red, sportcar)	
(red, supercar)	1,4

Cross의 테이블(Variable) 표현

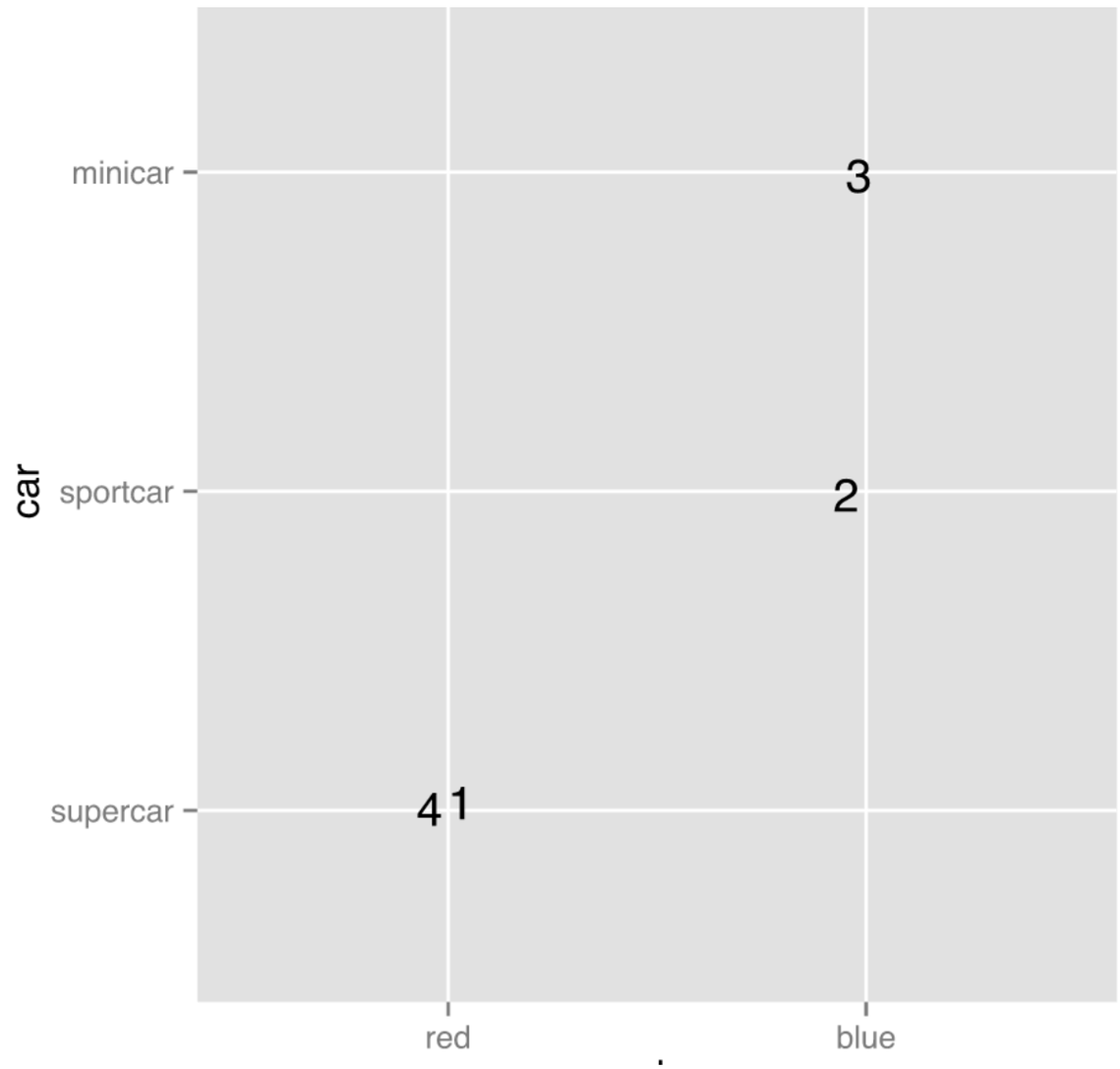
SELECT color.key, color.color, car.car
FROM color, car WHERE color.key = car.key;

$$Table\ color * key = \begin{array}{|lll|}\hline key & color & car \\ \hline 1 & red & Supercar \\ 2 & blue & Sportcar \\ 3 & blue & Minicar \\ 4 & red & Supercar \\ \hline \end{array}$$

단, 정의역(V_{A*B}) 전체가 표현된다는 보장은 없음.

Cross의 그래프 표현

```
ELEMENT: point(position(color*car), text=key)
```



Nest: /

Cross와 Nest의 수학적 정의

Cross(*)

$$A * B : (s,t) \rightarrow A(s) \cap B(t)$$

$$V_{A*B} = \{(s,t), \forall s \in V_A, \forall t \in V_B\}$$

Nest(/)

$$A/B : (s,t) \rightarrow A(s) \cap B(t)$$

$$V_{A/B} = \{(s,t), \forall s \in V_A, \forall t \in V_B : A(s) \cap B(t) \neq \varnothing\}$$

앞선 예제의 테이블(Variable) 표현

$$color = \begin{array}{cc} key & color \\ \hline 1 & red \\ 2 & blue \\ 3 & blue \\ 4 & red \\ \hline \end{array} \quad , \quad car = \begin{array}{cc} key & response \\ \hline 1 & Supercar \\ 2 & Sportcar \\ 3 & Minicar \\ 4 & Supercar \\ \hline \end{array}$$

$V_{\text{color/car}} \neq Frame$

```
SELECT DISTINCT color.color, car.car
FROM color INNER JOIN car
WHERE color.key = car.key;
```

```
color       car
----------  ----------
red         supercar
blue        sportcar
blue        minicar
```

$$V_{\text{color/car}} \neq V_{\text{color*car}}$$

```
# cross
color       car
----------  ----------
red         supercar
red         sportcar
red         minicar
blue        supercar
blue        sportcar
blue        minicar
```

```
# nest
color       car
----------  ----------
red         supercar
blue        sportcar
blue        minicar
```

$$V_{\text{color/car}} \neq V_{\text{color*car}} \rightarrow$$

$$\text{color/car} \neq \text{color} * \text{car}$$

$$f : V \rightarrow \tilde{O}\ of\ \textbf{color} * \textbf{car}$$

Value Domain	Set of Cases
(blue, minicar)	3
(blue, sportcar)	2
(blue, supercar)	
(red, minicar)	
(red, sportcar)	
(red, supercar)	1,4

$f : V \rightarrow \tilde{O}\ of$ color/car

```
Value Domain      Set of Cases
----------------  ------------
(blue, minicar)   3
(blue, sportcar)  2
(red, supercar)   1,4
```

```
SELECT
  ('('||color.color||','||car.car||')') AS "Value Domain",
  GROUP_CONCAT(car.key) as "Set of Cases"
FROM color INNER JOIN car WHERE color.key = car.key
GROUP BY "Value Domain";
```

Nest연산은 Frame을 만들지 않는다

Nest 연산의 역할 (1)

Wilkinson presents three algebraic operators called cross (*), nest(/), and blend (+), together with the rules for their use. Theyare derived from the set operators product (*), discreteunion (⊔), and union (∪), respectively.

– nVIZn 3p

Nest 연산의 역할 (2)

The disjoint union of two sets A and B, denoted by A ⊔ B, produces a set whose members are tagged elements. A tagged element is one of the form x:$, where x ∈X is the element and the symbol $ is the tag. A tag is some- times called an identifier or a color; it may be a string, a numerical value, or another piece of information.

– GoG 26p

Nest 연산의 역할 (3)

The nesting variable tags the nested variable such that (s, t) implies (s : t). We will not use this notation below, but we will assume an algebra system can identify the nesting elements through a tagging index.

– GoG 65p

Nest 연산의 용도

- Tagging
  - Data values
  - Metadata
  - Data organization

테이블로 직관적으로 이해하기

$$Table\ color/key = \begin{array}{ll} key & color/car \\ \hline 1 & red : Supercar \\ 2 & blue : Sportcar \\ 3 & blue : Minicar \\ 4 & red : Supercar \\ \hline \end{array}$$

d3.js의 nest 함수로 이해하기 (1)

```
data = [
   {color: 'red', car: 'supercar'},
   {color: 'blue', car: 'sportcar'},
   {color: 'blue', car: 'minicar'},
   {color: 'red', car: 'supercar'}
]
```

d3.js의 nest 함수로 이해하기 (2)

```
var nestedData = d3.nest()
   .key(function(d) { return d.car; })
   .map(dataSet);
```

```
{
'supercar':[
    {color: 'red', car: 'supercar'},
    {color: 'red', car: 'supercar'},
  ],
  'sportcar':[
    {color: 'blue', car: 'sportcar'},
  ],
  'minicar':[
    {color: 'blue', car: minicar'},
  ]
}
```

$$/ \approx d3.nest \approx group\,by$$

Blend: +

Blend의 수학적 정의

$$\mathsf{A} + \mathsf{B} : s \rightarrow \mathsf{A}(s) \cup \mathsf{B}(s)$$

$$V_{\mathsf{A}+\mathsf{B}} = V_{\mathsf{A}} \cup V_{\mathsf{B}}$$

항등원(Identity emelemnt)

- +의 항등원: 공집합
- cross와 nest의 항등원 varset

$$1 : unity \rightarrow \Omega$$

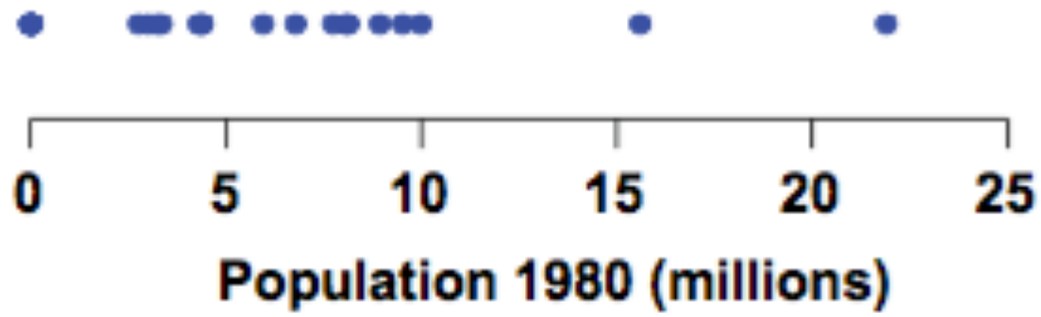
$$V_1 = unity$$

- 차수(order)를 맞추기 위해서도 사용

cross 항등원(같은 표현)

```
ELEMENT: point(position(pop1980))

ELEMENT: point(position(pop1980*1))
```



Expression

- term(expression with no + operator)
- factor(expression with no * operator)
- monomial(one term)
- polynomial(more than one term)
- order(number of factors in term)
- k(largest order among the monomials)

Algebraic Form

An algebraic form is a monomial or a polynomial whose terms all have the same number of factors.

- $\mathsf{G} * 1 + \mathsf{A} * \mathsf{C}/\mathsf{D} + \mathsf{B} * \mathsf{C}/\mathsf{D}$
- $factors = \mathsf{G}, 1, \mathsf{A}, \mathsf{C}/\mathsf{D}, \mathsf{B}$
- $terms = \mathsf{G} * 1, \mathsf{A} * \mathsf{C}/\mathsf{D}, \mathsf{B} * \mathsf{C}/\mathsf{D}$
- $k = 2$

왜 차수를 맞춰야할까?

- 차수는 그래프가 그려지는 공간의 차원을 결정
- 차수가 같아야 Blend 연산을 할 때 의미를 가짐

Operator Precedence

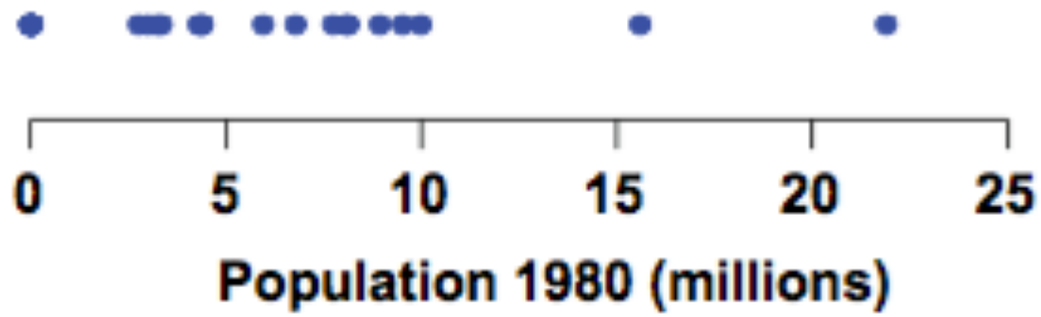
$$/ \rightarrow * \rightarrow +$$

그래프의 세계로

1차원 그래프

pop1980

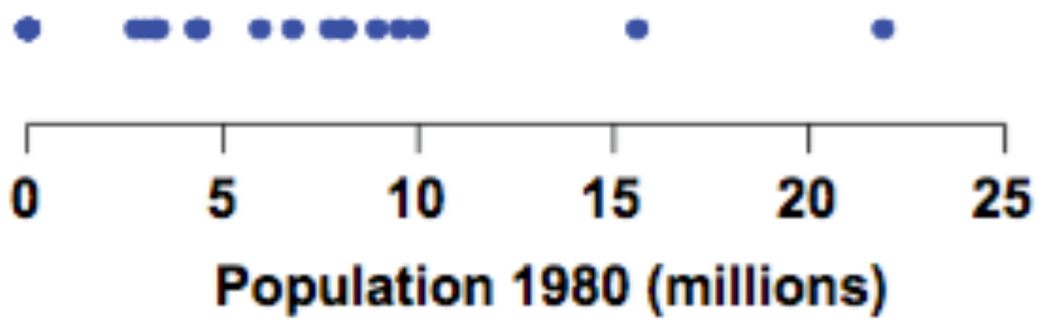
```
COORD: rect(dim(1))
ELEMENT: point(position(pop1980))
```



2차원 그래프

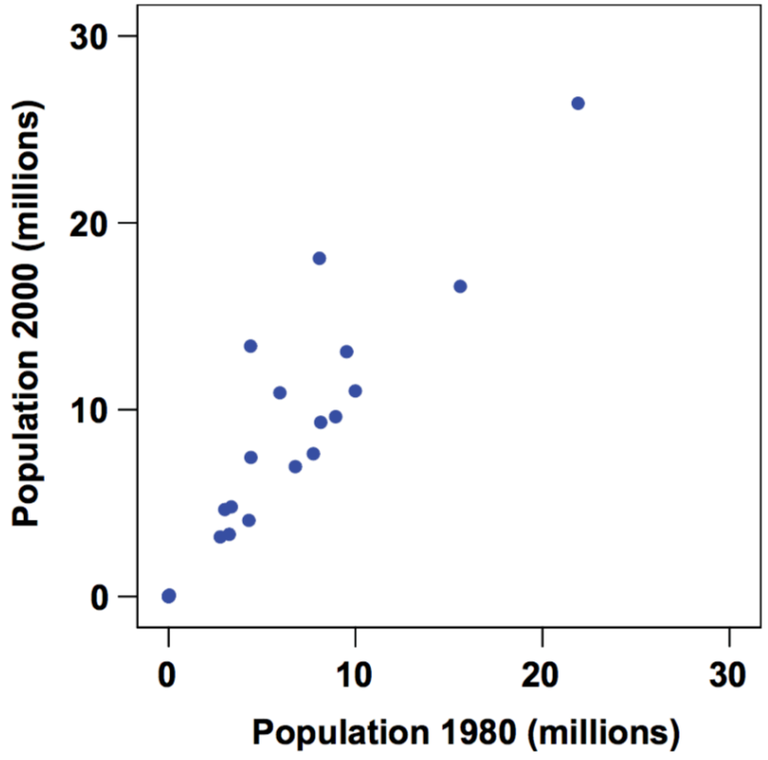
pop1980 * 1

```
COORD: rect(dim(1, 2))
ELEMENT: point(position(pop1980 * 1))
# FRAME = [0, 30] x unity
```



pop1980 * pop2000

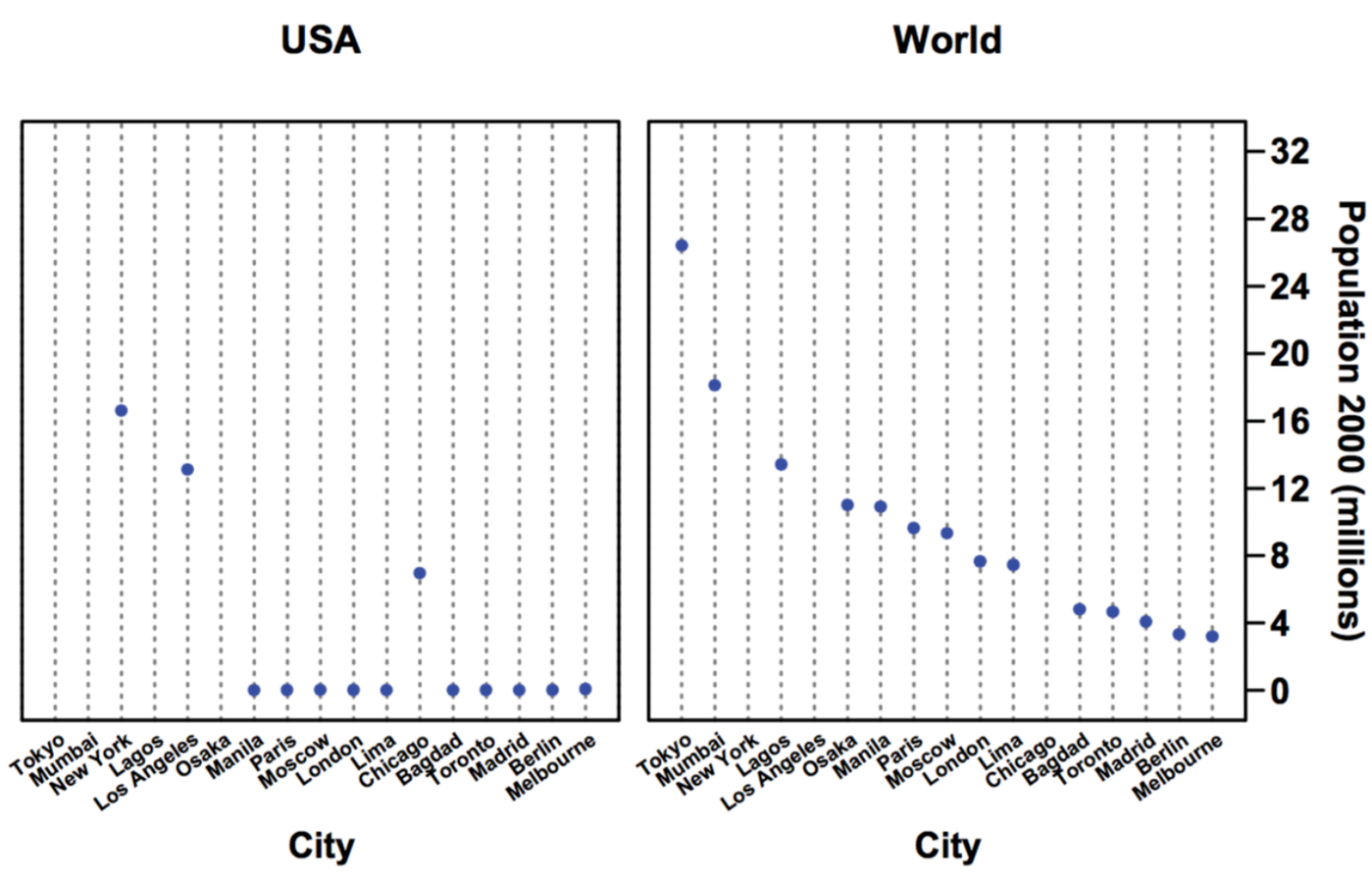
```
COORD: rect(dim(1,2))
ELEMENT: point(position(pop1980*pop2000))
# FRAME = [0, 30] x [0, 30]
```



3차원 그래프?

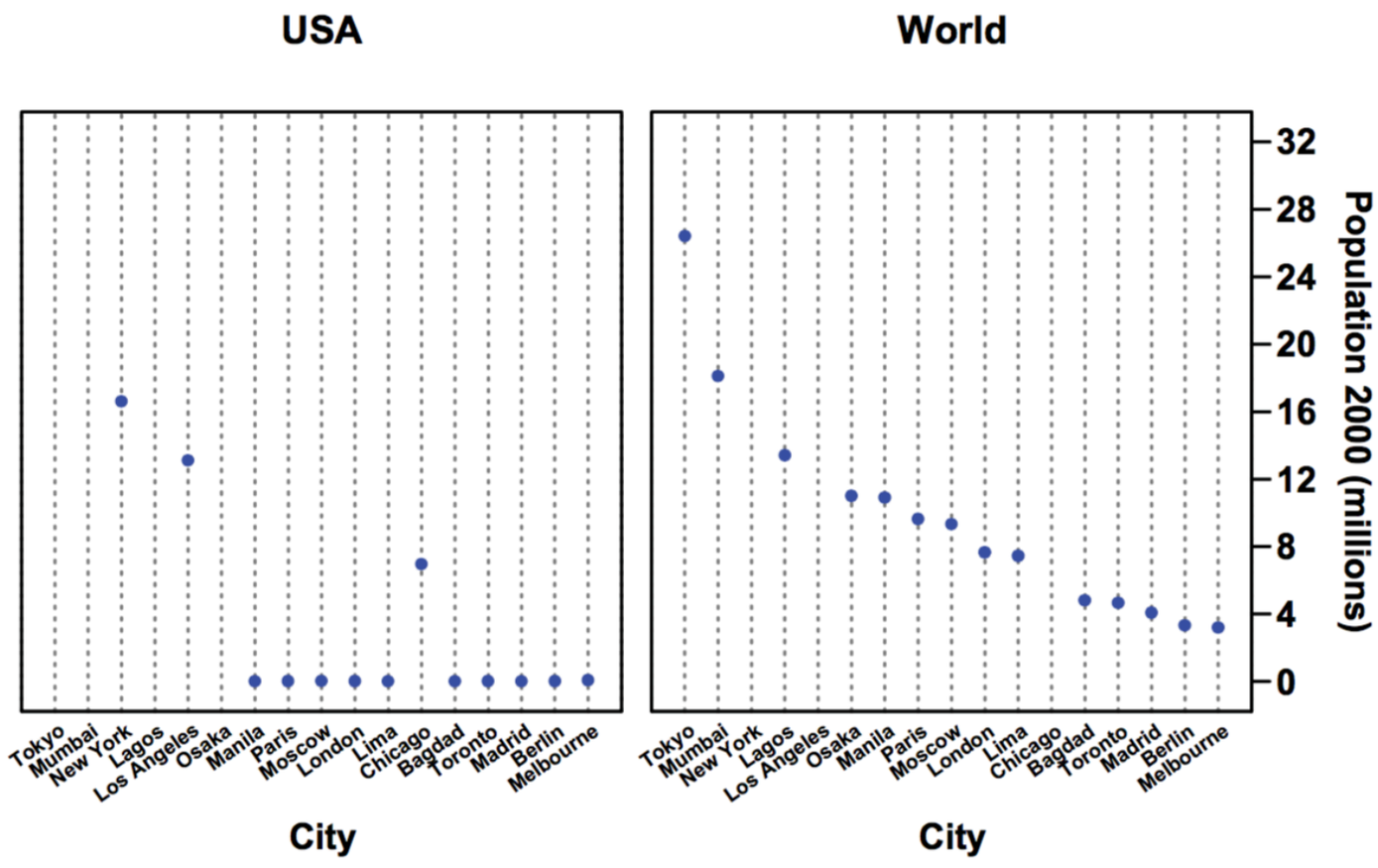
책의 예제 (1)

```
ELEMENT: point(position(city*pop1980*group))
# FRAME = (Tokyo, ...) x [0, 30] x (World, USA)
```



책의 예제(2) - 좀 더 정확한 표현

```
COORD: rect(dim(3), rect(dim(1,2)))
ELEMENT: point(position(city*pop1980*group))
```



3차원 그래프!

COORD: *rect*(*dim*(1, 2, 3)
ELEMENT: *point*(*position*(*summary.mean*(urban*gov*birth)),
shape(*shape.cube*))

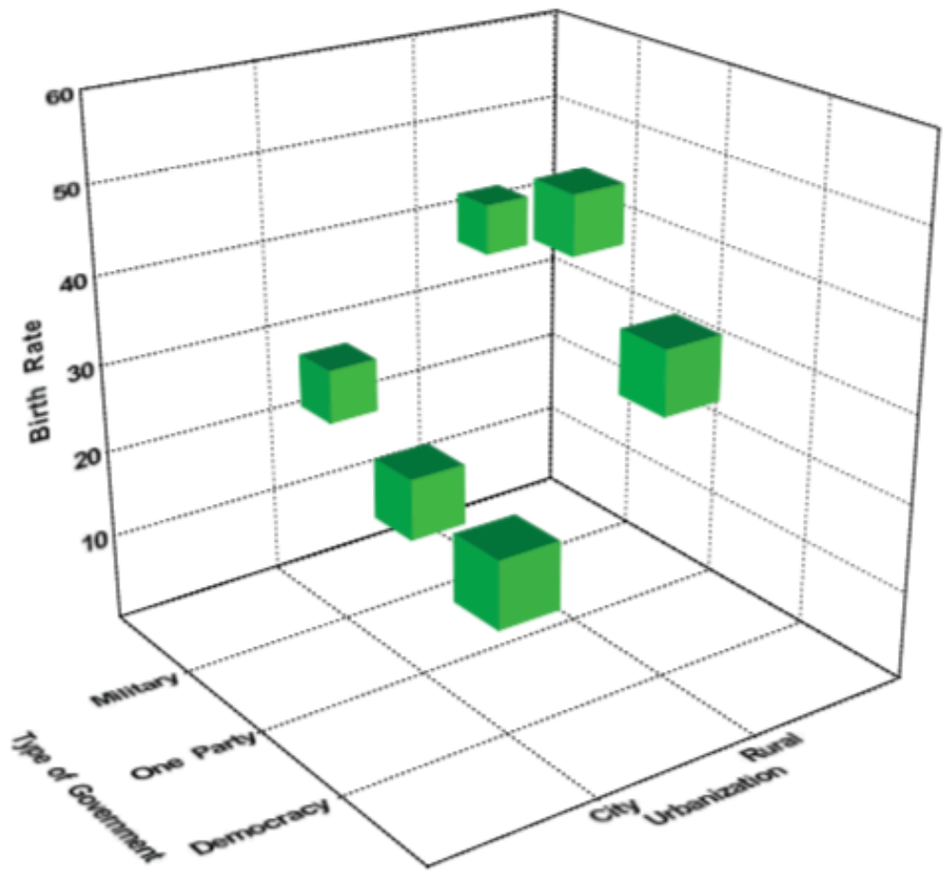
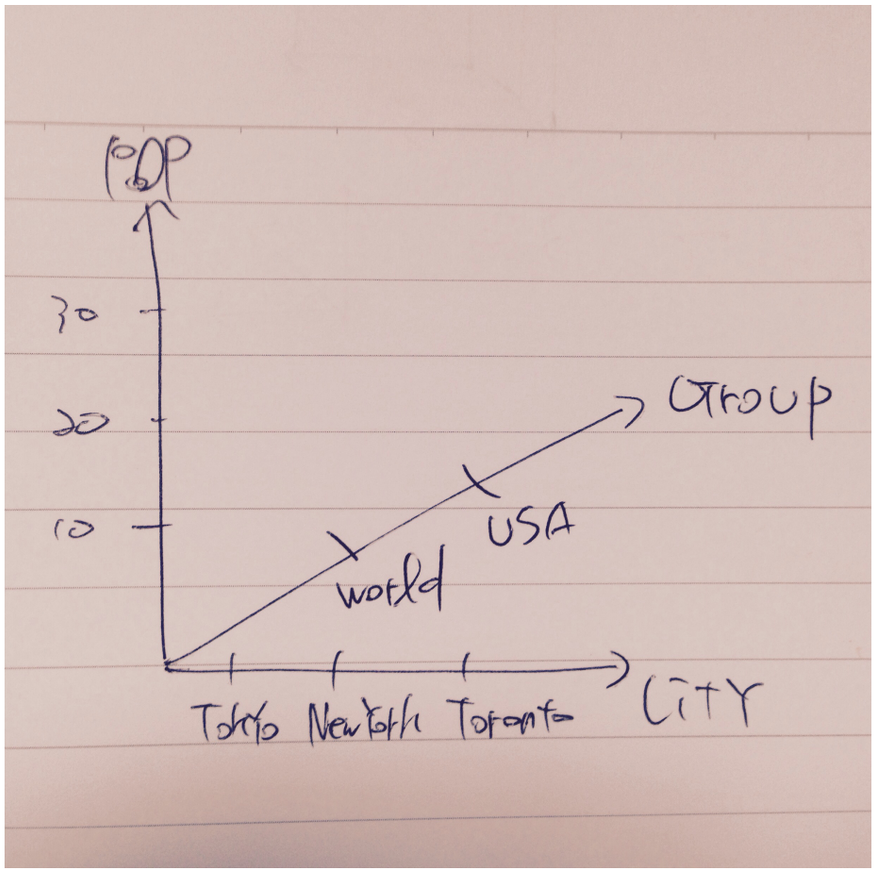


Figure 7.4 *3D point plot on categorical domain*

앞의 예제를 3차원으로

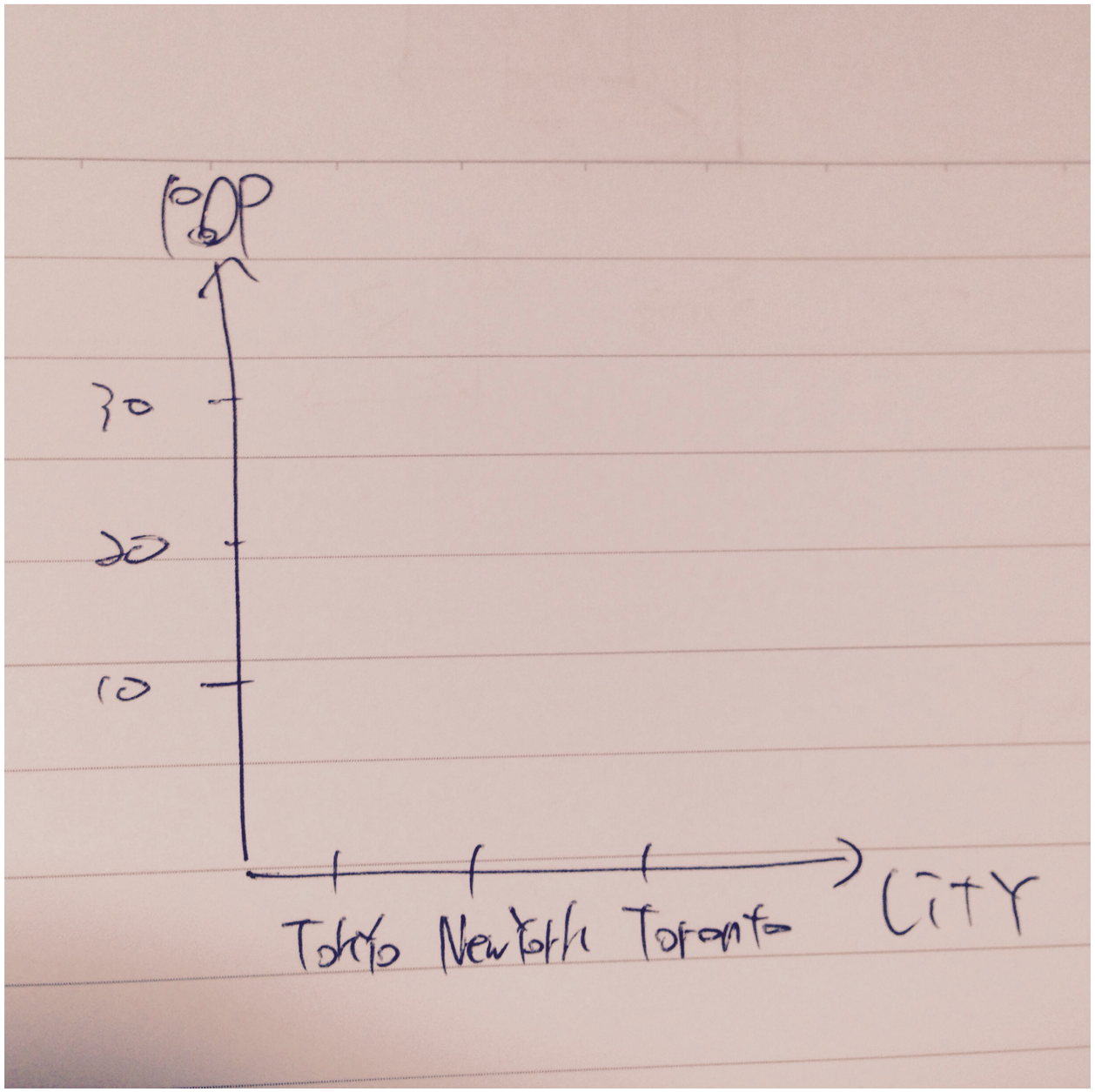
```
COORD: rect(dim(1,2,3))
ELEMENT: point(position(city*pop1980*group))
# FRAME = (Tokyo, ...) x [0, 30] x (World, USA)
```



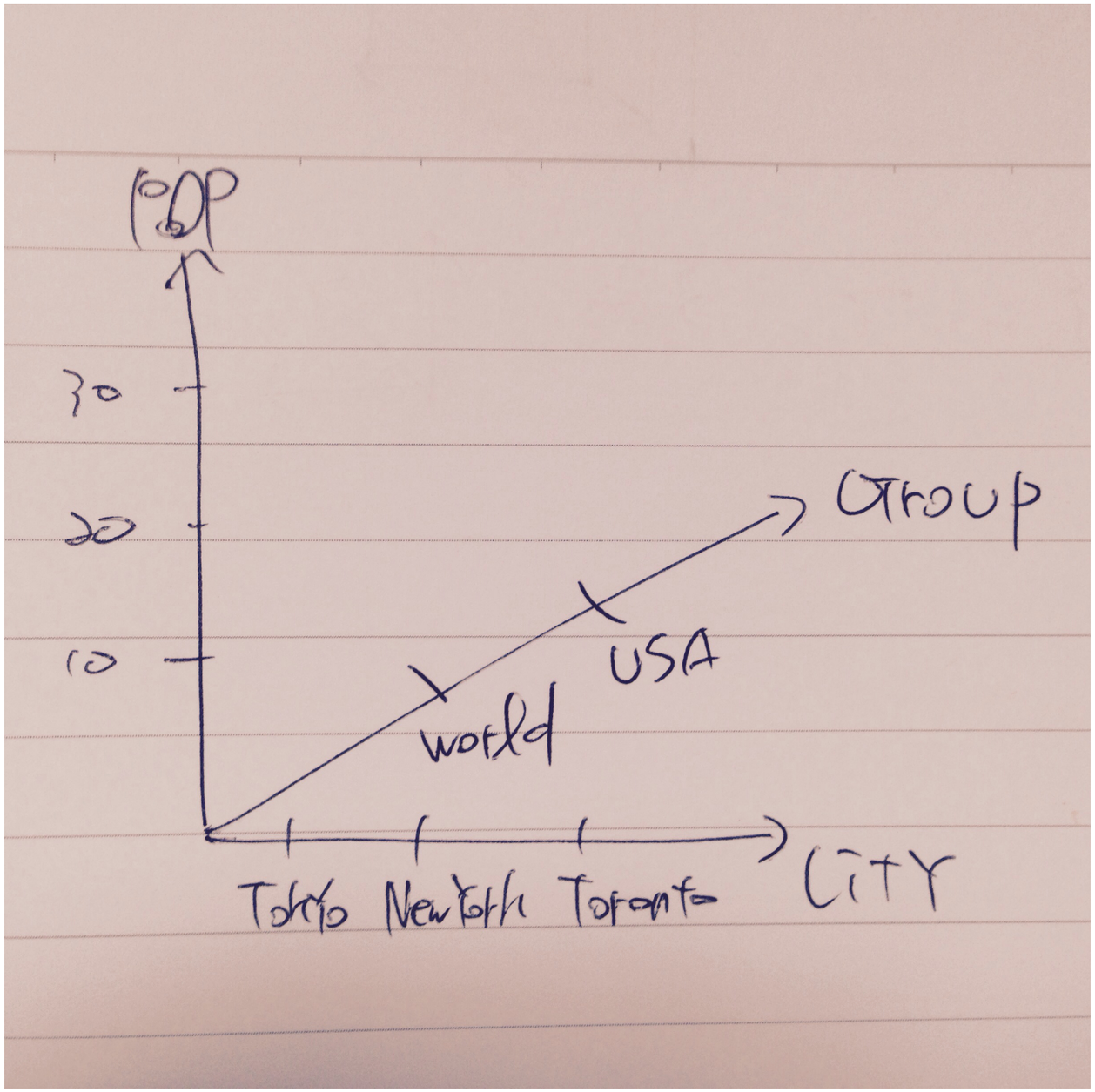
Step by Step

```
rect(dim(1,2,3)) to rect(dim(3), rect(dim(1,2)))
```

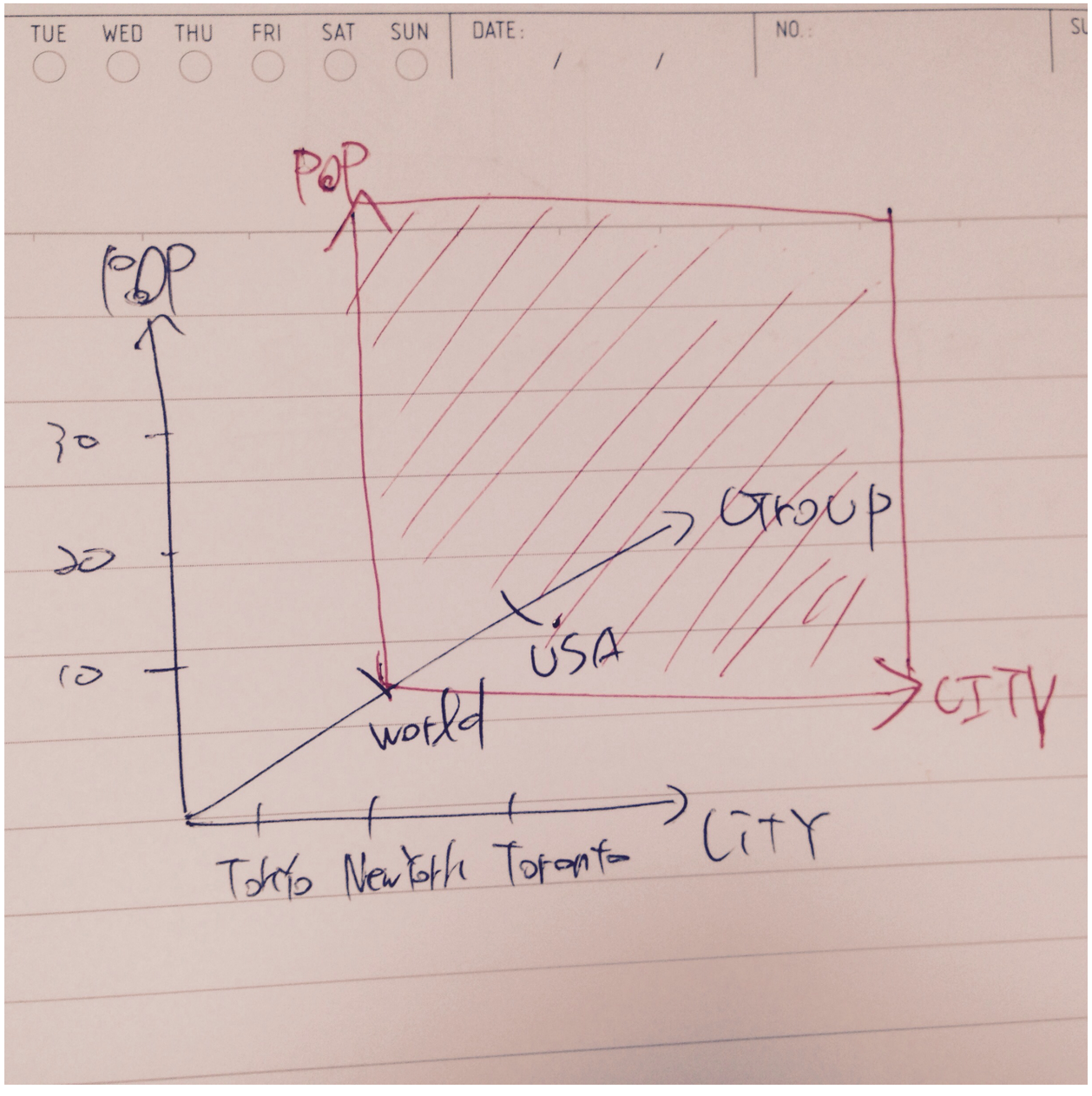
2차원



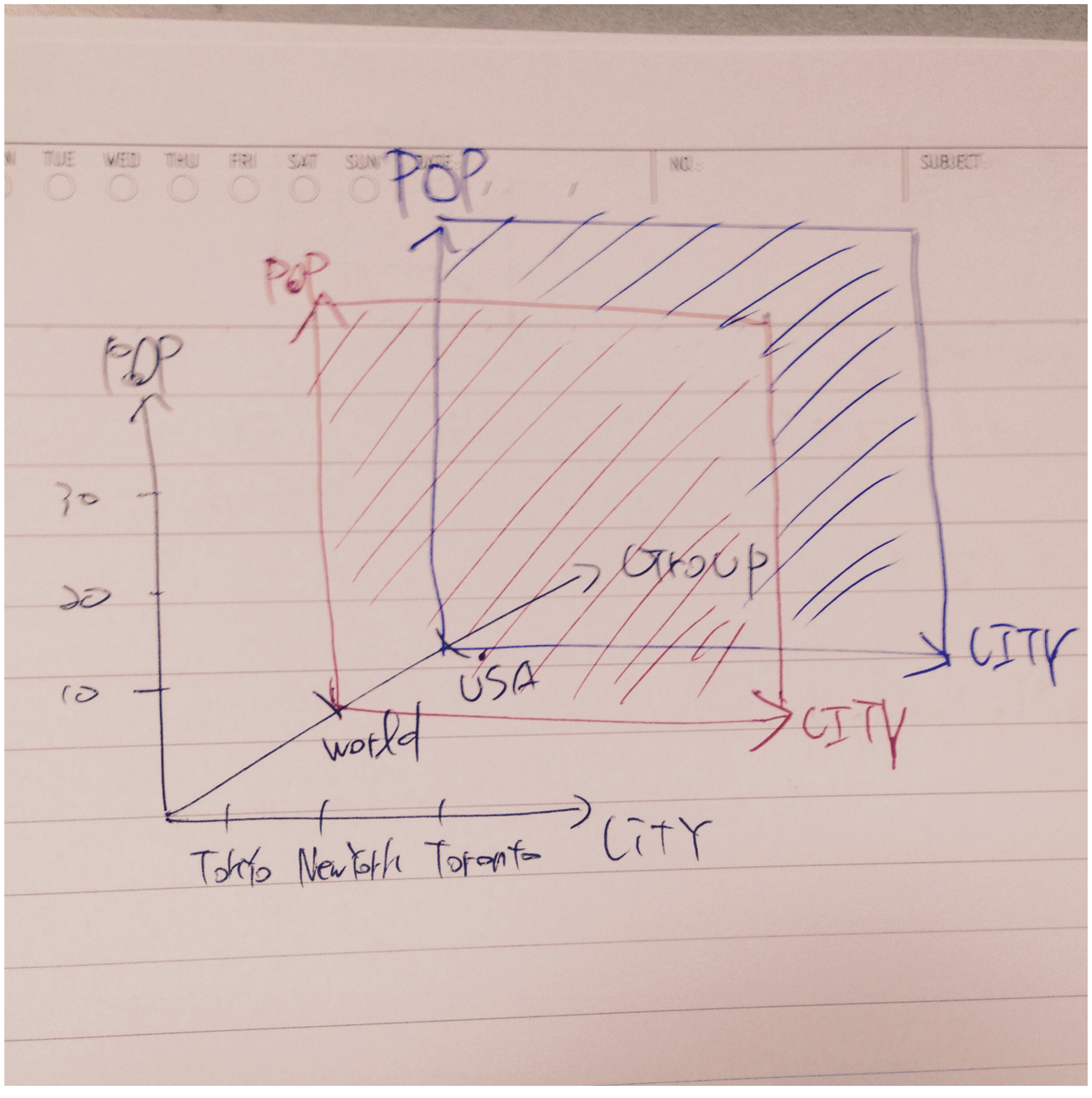
3차원



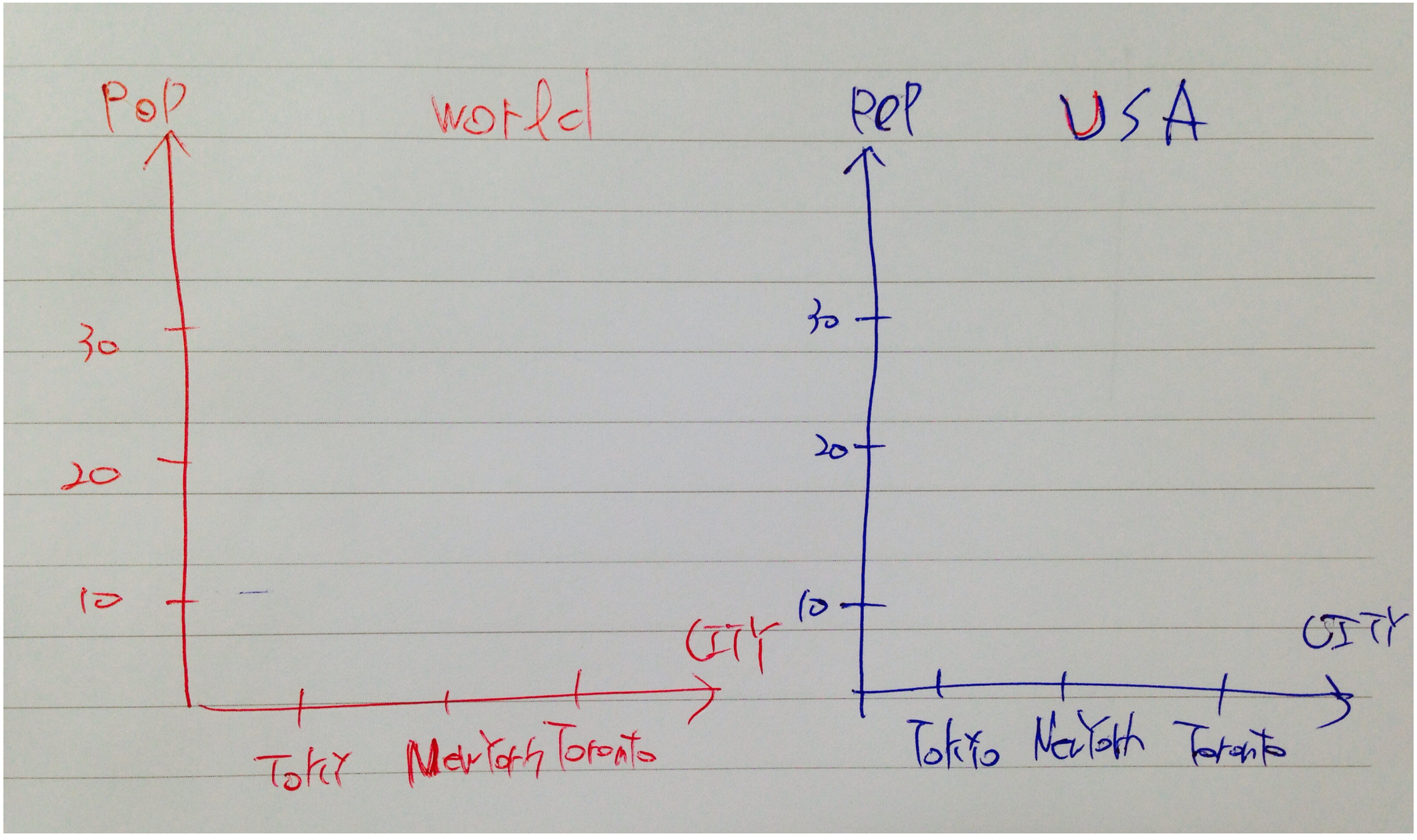
World 평면으로 잘라내기



USA 평면으로 잘라내기



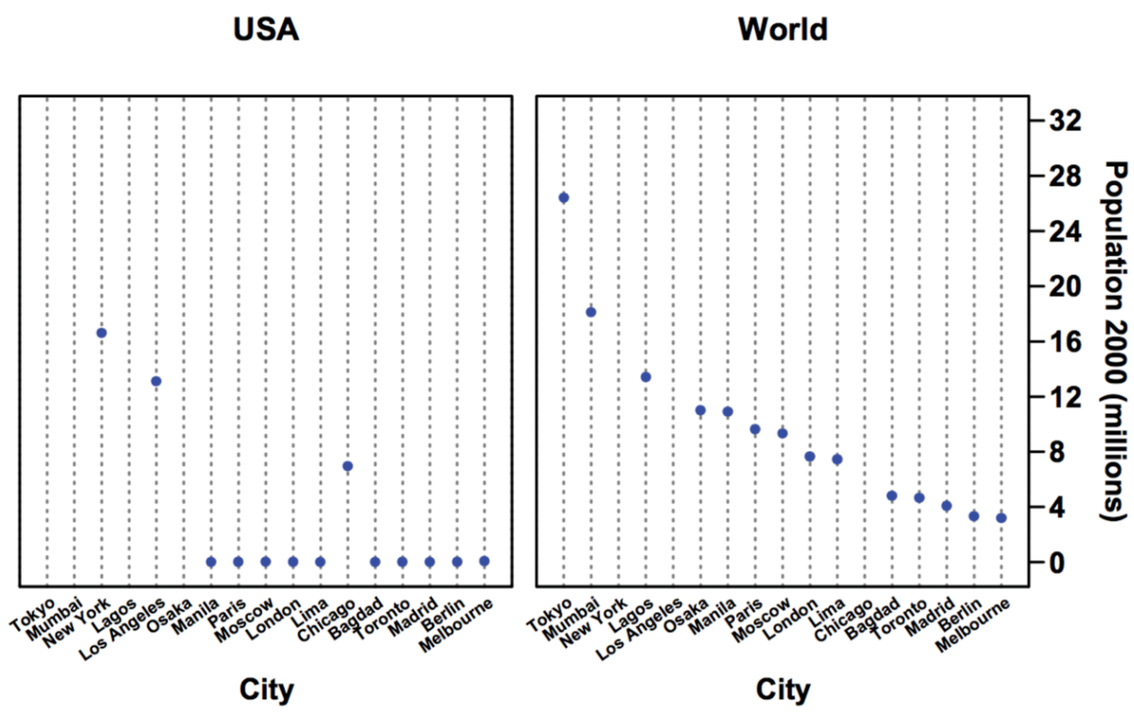
World, USA 평면을 따로 펼쳐놓기



city * pop1980 * group

- 같은 2차원 Frame(city*pop1980)의 평면들

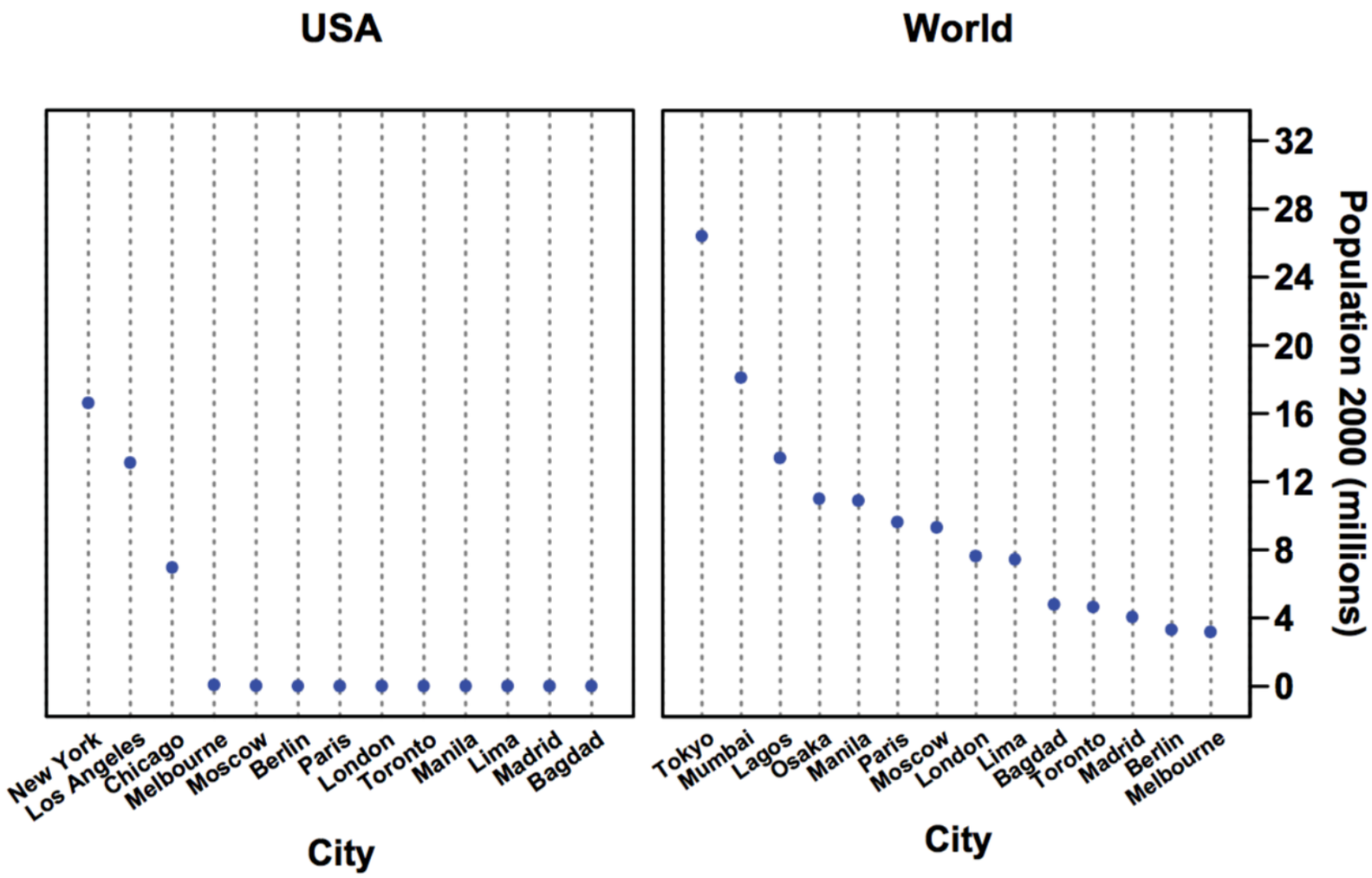
```
COORD: rect(dim(3), rect(dim(1,2)))
ELEMENT: point(position(city*pop1980*group))
```



Nest 연산 / 이해하기

책의 예제

```
ELEMENT: point(position(city/group*pop2000))
```



city/group

civy/group의 도메인

$$V_{city/group} = \{Tokyo : world, Mumbai : world, NewYork : usa, \dots\}$$

각 태그(그룹)으로 나눈 도메인

$$V_{city_{usa}} = \{NewYork : usa, LosAngeles : usa, \dots\}$$

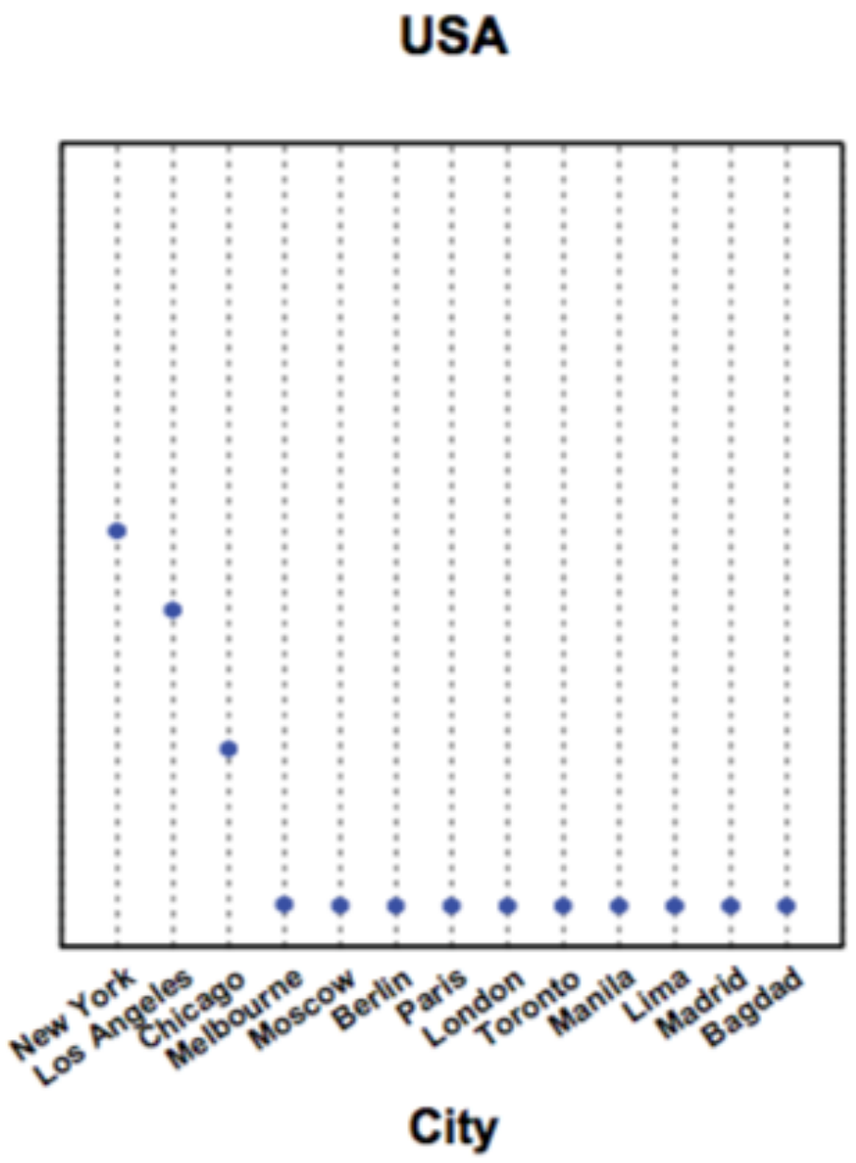
$$V_{city_{world}} = \{Tokyo : world, Mumbai : world, \dots\}$$

서로 다른 2개의 평면!

$$city_{usa} * pop2000$$

$$city_{world} * pop2000$$

$$city_{usa} * pop2000$$



$$city_{world} * pop2000$$

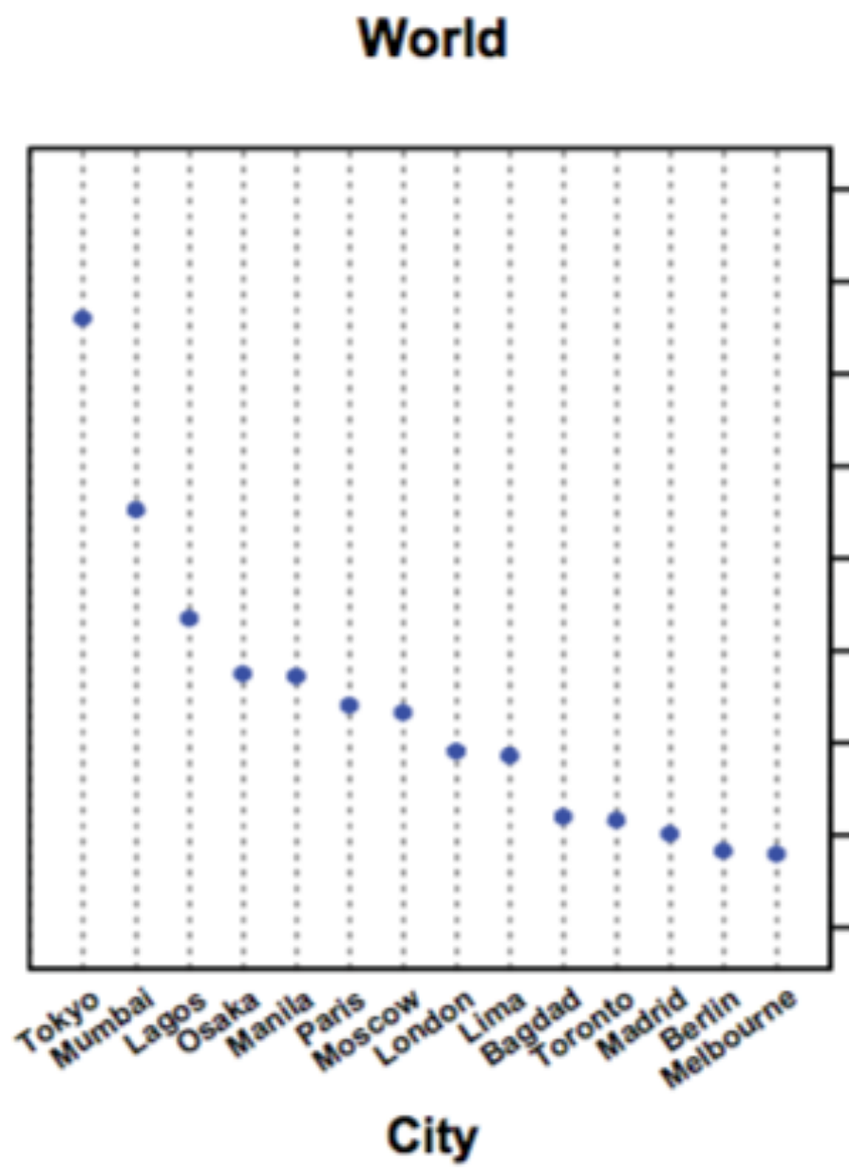
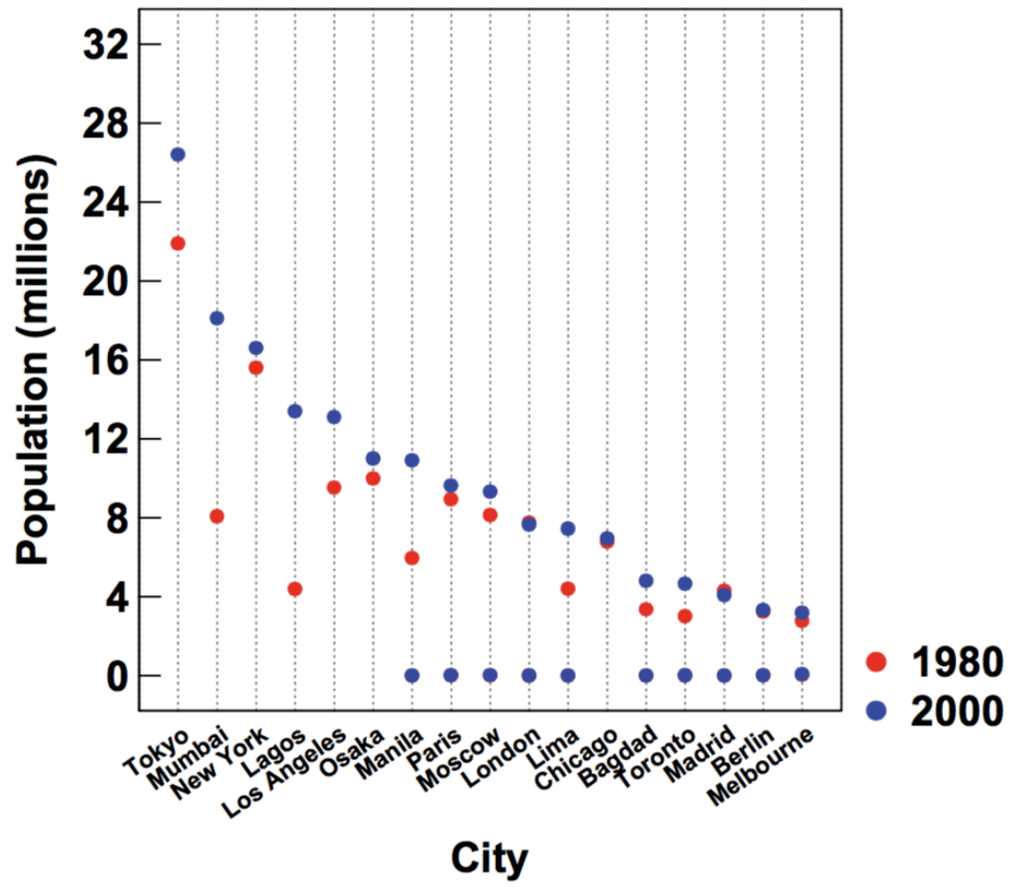


Figure 5.6

```
DATA: p1980 = "1980"
DATA: p2000 = "2000"
ELEMENT: point(position(city*(pop1980+pop2000)), color(p1980 + p2000))
```



Frame of Figure 5.6

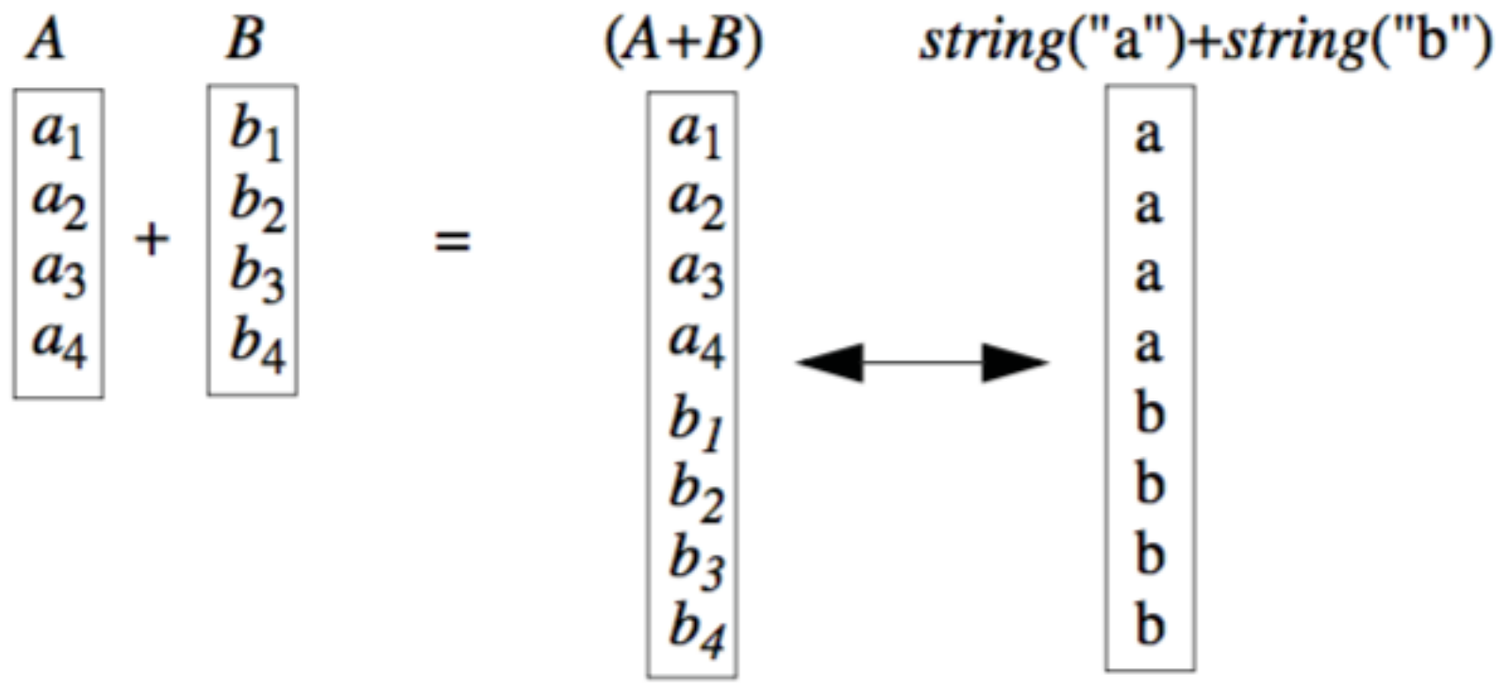
- city*(pop1980+pop2000)

```
element: point(position(city*(pop1980+pop2000)))
# FRAME = {Tokyo, ...} x [0, 30]
```

- city*pop2000

```
element: point(position(city*pop2000))
# FRAME = {Tokyo, ...} x [0, 30]
```

Color Blend Operator



GoG 310p

Figure 5.6, again

```
DATA: p1980 = "1980"
DATA: p2000 = "2000"
ELEMENT: point(position(city*(pop1980+pop2000)), color(p1980 + p2000))
```

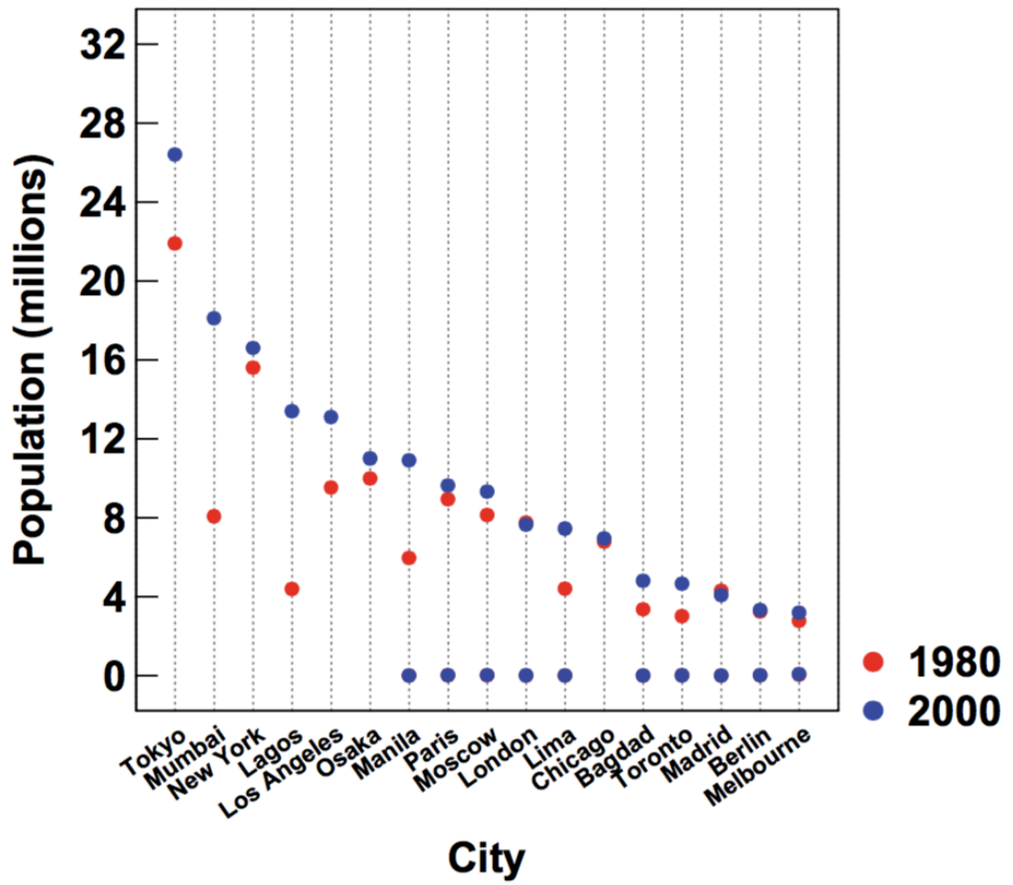


Figure 5.8

```
DATA: p1980 = "1980"
DATA: p2000 = "2000"
ELEMENT: point(position((city/group)*(pop1980+pop2000), color(p1980 + p2000))
```

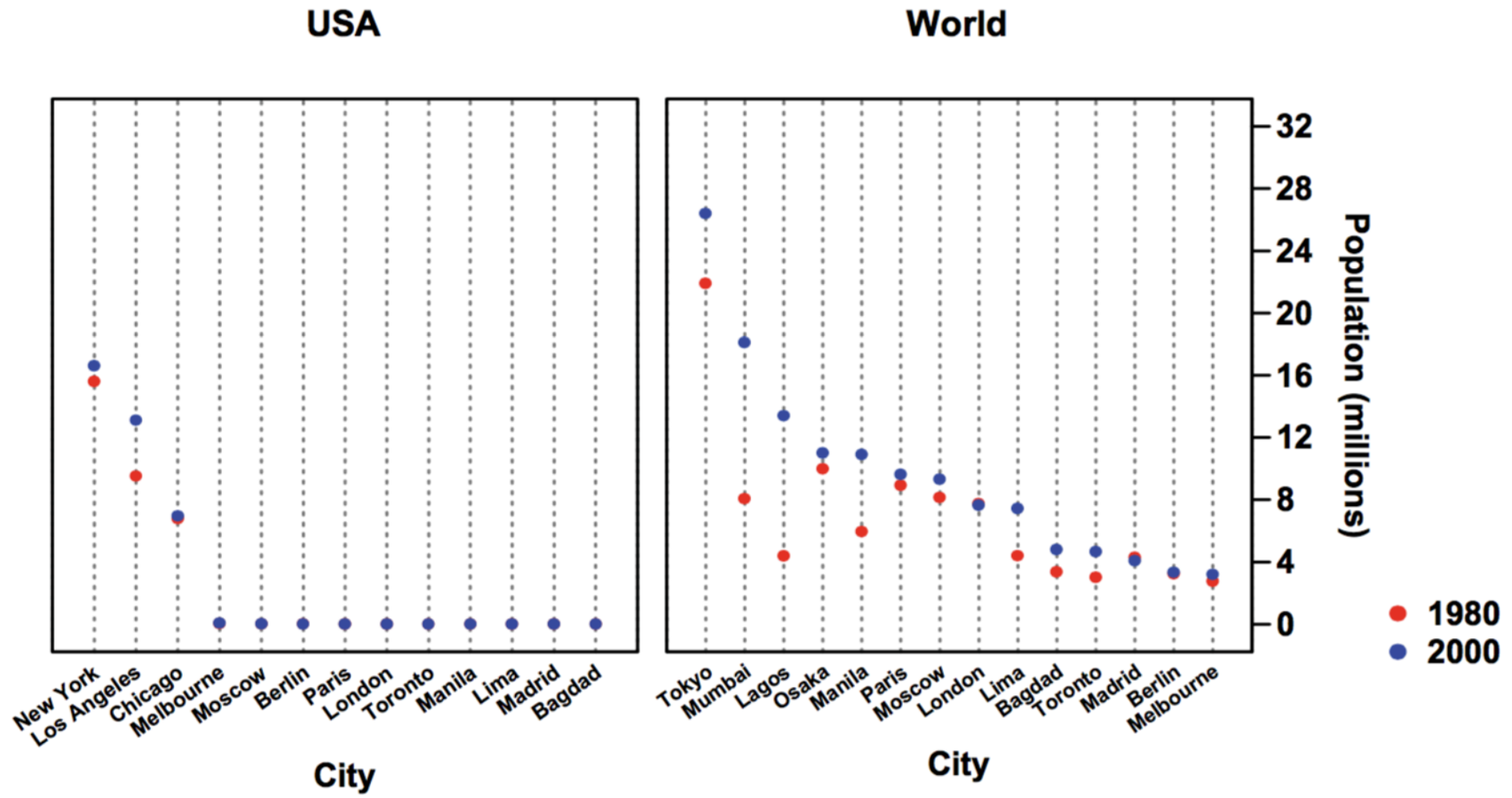
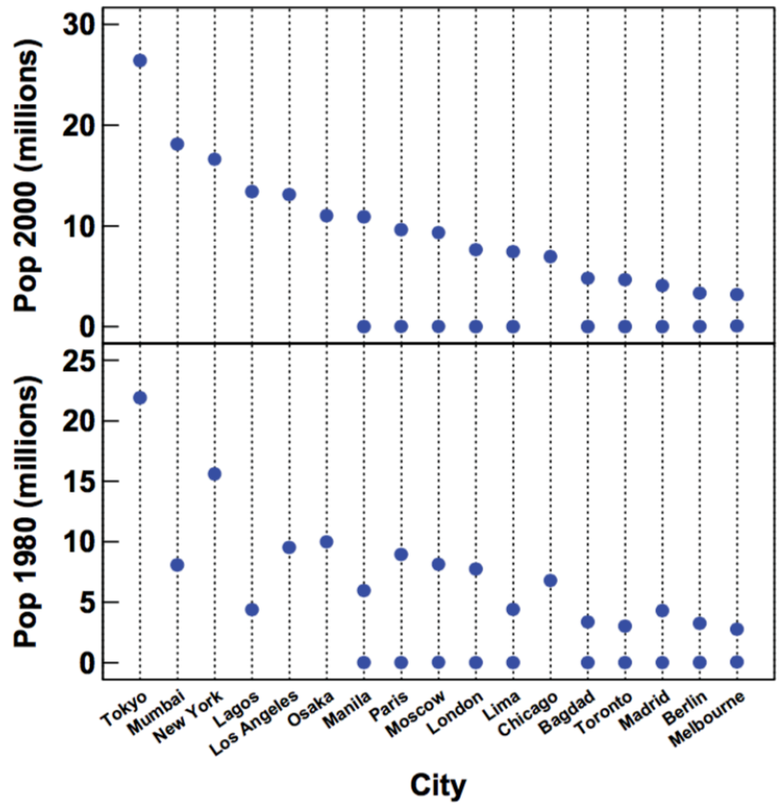


Figure 5.9

```
DATA: p1980 = "Pop 1980 (millions)"
DATA: p2000 = "Pop 2000 (millions)"
ELEMENT: point(position(city*(pop1980/p1980+pop2000/p2000)))
```



ggplot2로 ch5 예제 재구현

GOG ch5 Varget Algebra Example - Gist

ggplot2와 GoG

- ggplot2
  - Varset Algebra 개념을 지원하지 않음
  - 시각화하는 data.frame을 미리 조작해야함

scales = 'free' 옵션으로 nest와 비슷하게 구현

```
ggplot(pop, aes(x = city2, y = pop2000_d)) +
  geom_point(colour = I('darkblue')) +
  # scales = 'free'를 통해서 평면에서 공집합이 되는 (a, b) 카테고리를 제거
  facet_grid(. ~ group, scales = 'free') +
  scale_y_continuous(limits=c(0, 30))
```

Thank you!

@nacyo_t