PROC SQL

Shannon Pileggi

STAT 330

OUTLINE

Overview

SELECT

Summarizing data

More

PROC SQL

Task	PROC SQL	DATA step	Other PROCs
Print results	✓	✗	✓
Sort data	✓	✗	✓
Summarize data	✓	∼	✓
Combine data	✓	✓	✗
Create new variables	✓	✓	✗
Subset data	✓	✓	∼
Create new data set	✓	✓	∼

Syntax

SAS Code

```
PROC SQL;
   CREATE TABLE table-name AS
   SELECT column(s)
   FROM table(s) | view(s)
   WHERE expression
   GROUP BY column(s)
   ORDER BY column(s)
   ;
QUIT;
```

SAS Code

- clauses **must** be specified in this order
- only the SELECT and FROM clauses are required, all other clauses are optional
- only one semi-colon for all clauses
- terminology:

SQL	SAS
table	data set
row	observation
column	variable

Explanations

`PROC SQL` calls sql procedure

`CREATE TABLE` create new data set (in lieu of print)

`SELECT` specifies the column(s) (variables) to be selected

`FROM` specifies the table(s) (data sets) to be queried

`WHERE` subsets the data based on a condition

`GROUP BY` classifies the data into groups based on the specified column(s)

`ORDER BY` sorts the resulting rows (observations) by the specified column(s)

`QUIT` ends the sql procedure

SAS vs SQL

Function	SQL	SAS
Drop columns	SELECT	DROP
Rename column	AS	RENAME
Add rows	INSERT INTO	OUTPUT
Delete rows	DELETE FROM / WHERE	WHERE / IF-THEN / DELETE
Delete duplicate rows	DISTINCT	NODUPLICATE
Create a table	CREATE TABLE	DATA
Sorting	ORDER BY	PROC SORT
Summarize data	GROUP BY	PROCs / FIRST. LAST.
Conditional statement	CASE-WHEN	IF-THEN
Displaying output	SELECT	PROC PRINT
Concatenating	OUTER JOIN	SET

Overview

SELECT

Summarizing data

More

Getting started with SELECT

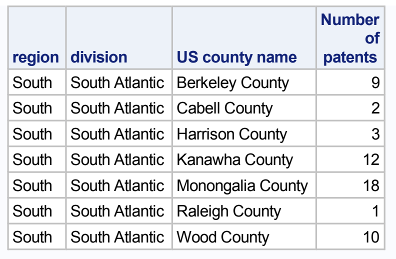
SAS Code

```
PROC SQL;
   SELECT *
   FROM patents
   ;
QUIT;


PROC SQL;
   SELECT region,
          division,
          county,
          patents
   FROM patents
   ;
QUIT;
```

SAS Code

- * specifies *all* variables
- indicate specific variables in a list separated by commas



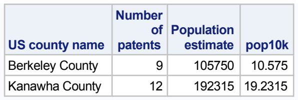
region	division	US county name	Number of patents
South	South Atlantic	Berkeley County	9
South	South Atlantic	Cabell County	2
South	South Atlantic	Harrison County	3
South	South Atlantic	Kanawha County	12
South	South Atlantic	Monongalia County	18
South	South Atlantic	Raleigh County	1
South	South Atlantic	Wood County	10

Calculating a new variable

SAS Code

```
PROC SQL ;
   SELECT  county, patents, population,
           (population/10000) AS pop10k
   FROM patents
   WHERE state = "WEST VIRGINIA"
         AND calculated pop10k > 10
        ;
QUIT ;
```

SAS Code



US county name	Number of patents	Population estimate	pop10k
Berkeley County	9	105750	10.575
Kanawha County	12	192315	19.2315

- Syntax: `(expression) as newvar`
- When using *newvar* in subsequent code, must refer to it as `calculated newvar`

Apply labels and formats

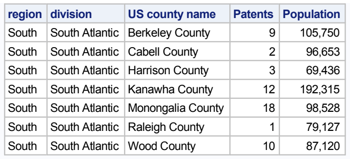
SAS Code

```
PROC SQL ;
   SELECT region, division, county,
      patents LABEL = "Patents",
      population FORMAT = COMMA15. LABEL = "Population"
   FROM patents
   WHERE state = "WEST VIRGINIA"
   ;
QUIT;
```

SAS Code

after variable name *before* comma:

- apply format with `format=myfmt.`
- apply label with `label="mylabel"`



region	division	US county name	Patents	Population
South	South Atlantic	Berkeley County	9	105,750
South	South Atlantic	Cabell County	2	96,653
South	South Atlantic	Harrison County	3	69,436
South	South Atlantic	Kanawha County	12	192,315
South	South Atlantic	Monongalia County	18	98,528
South	South Atlantic	Raleigh County	1	79,127
South	South Atlantic	Wood County	10	87,120

Creating a new variable with conditional logic

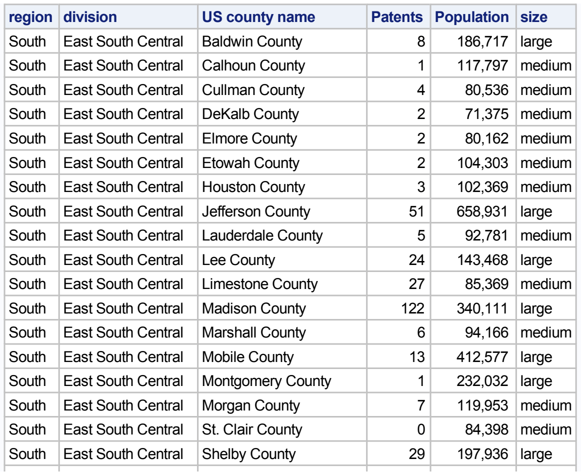
SAS Code

```
PROC SQL ;
   SELECT region, division, county, patents LABEL = "Patents",
   population FORMAT = COMMA15. LABEL = "Population",
   CASE
      WHEN population LE 70000 THEN "small"
      WHEN population BETWEEN 70001 AND 120000 THEN "medium"
      ELSE "large"
   END AS size
   FROM patents ;
QUIT ;
```

SAS Code

- CASE - WHEN/THEN/ELSE - END similar to IF-THEN-ELSE
- use `END AS newvar` to create a new variable
- when using *newvar* in subsequent code, must refer to it as `CALCULATED newvar`

Creating a new variable with conditional logic, output



region	division	US county name	Patents	Population	size
South	East South Central	Baldwin County	8	186,717	large
South	East South Central	Calhoun County	1	117,797	medium
South	East South Central	Cullman County	4	80,536	medium
South	East South Central	DeKalb County	2	71,375	medium
South	East South Central	Elmore County	2	80,162	medium
South	East South Central	Etowah County	2	104,303	medium
South	East South Central	Houston County	3	102,369	medium
South	East South Central	Jefferson County	51	658,931	large
South	East South Central	Lauderdale County	5	92,781	medium
South	East South Central	Lee County	24	143,468	large
South	East South Central	Limestone County	27	85,369	medium
South	East South Central	Madison County	122	340,111	large
South	East South Central	Marshall County	6	94,166	medium
South	East South Central	Mobile County	13	412,577	large
South	East South Central	Montgomery County	1	232,032	large
South	East South Central	Morgan County	7	119,953	medium
South	East South Central	St. Clair County	0	84,398	medium
South	East South Central	Shelby County	29	197,936	large

Overview

SELECT

Summarizing data

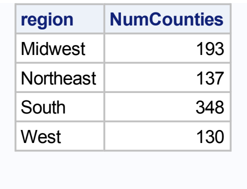
More

Getting started

SAS Code

```
PROC SQL;
   SELECT region,
          COUNT(county) AS NumCounties
   FROM patents
   GROUP BY region
   ;
QUIT ;
```

SAS Code



region	NumCounties
Midwest	193
Northeast	137
South	348
West	130

- Syntax: `FUNCTION(var) AS newvar`
- create variable with summary values on `SELECT` clause
- indicate how to summarize with `GROUP BY` clause

Summary functions

COUNT/FREQ/N	SUM	AVG/MEAN
MIN	MAX	RANGE
STD	VAR	USS
CSS	T	NMISS

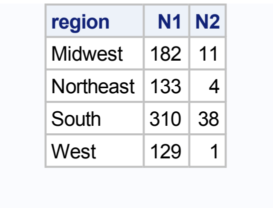
- these functions summarize data over observations
- think vertical summary, not horizontal
- so the `MEAN` function in `PROC SQL` works like `PROC MEANS` and **not** like the `mean` function in a `DATA step`

Counting missing values

SAS Code

```
PROC SQL ;
   SELECT region,
      COUNT(asian) AS N1,
      NMISS(asian) AS N2
   FROM patents
   GROUP BY region
   ;
QUIT;
```

SAS Code



region	N1	N2
Midwest	182	11
Northeast	133	4
South	310	38
West	129	1

- `COUNT` returns the number of non-missing observations
- `NMISS` returns the number of missing observations

Discussion

```
PROC SQL;
   SELECT
      region, summary
   FROM patents
   GROUP BY region
   ;
quit;
```

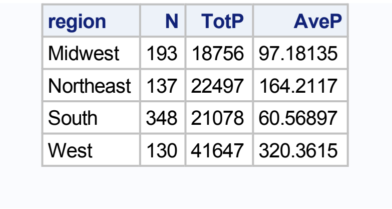
- COUNT(county) AS NumCounties
- COUNT(division) AS NumDivision
- COUNT(patents) AS NumPatents

Assume that there are no missing values in county, division, and patents. Identify the relationship.

1. NumDivision (<, >, =) NumCounties
2. NumPatents (<, >, =) NumCounties

Other summary stats

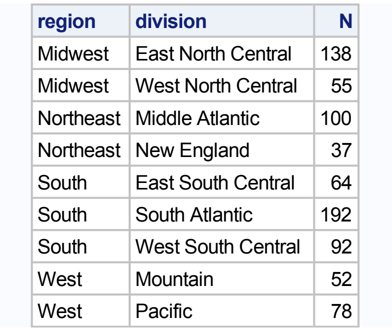
```
PROC SQL;
   SELECT
      REGION,
      COUNT(county) AS N,
      SUM(patents) AS TotP,
      MEAN(patents) AS AveP
   FROM patents
   GROUP BY region
   ;
QUIT;
```



region	N	TotP	AveP
Midwest	193	18756	97.18135
Northeast	137	22497	164.2117
South	348	21078	60.56897
West	130	41647	320.3615

Multiple groupings, ex1

```
PROC SQL ;
   SELECT
      region, division,
      COUNT(county) AS N
   FROM patents
   GROUP BY region, division
   ;
quit;
```

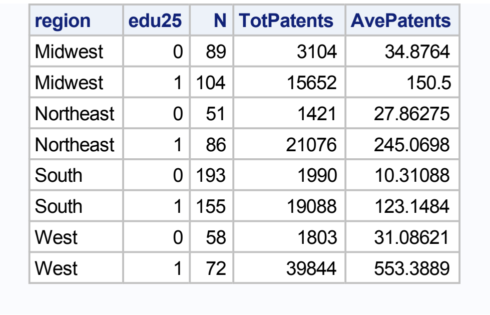


region	division	N
Midwest	East North Central	138
Midwest	West North Central	55
Northeast	Middle Atlantic	100
Northeast	New England	37
South	East South Central	64
South	South Atlantic	192
South	West South Central	92
West	Mountain	52
West	Pacific	78

- obtain sample size for region/division

Multiple groupings, ex2

```
PROC SQL ;
   SELECT
      region, edu25,
      COUNT(county) AS N,
      SUM(patents) AS TotP,
      MEAN(patents) AS AveP
   FROM patents
   GROUP BY region, edu25
   ;
QUIT;
```



region	edu25	N	TotPatents	AvePatents
Midwest	0	89	3104	34.8764
Midwest	1	104	15652	150.5
Northeast	0	51	1421	27.86275
Northeast	1	86	21076	245.0698
South	0	193	1990	10.31088
South	1	155	19088	123.1484
West	0	58	1803	31.08621
West	1	72	39844	553.3889

- obtain summary statistics for region/edu25

Overview

SELECT

Summarizing data

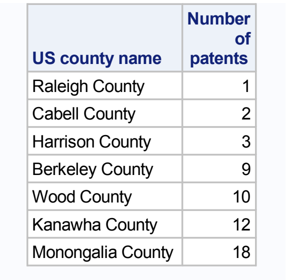
More

Sorting

SAS Code

```
PROC SQL;
   SELECT county, patents
   FROM patents
   WHERE state="WEST VIRGINIA"
   ORDER BY patents
    ;
QUIT;
```

SAS Code



US county name	Number of patents
Raleigh County	1
Cabell County	2
Harrison County	3
Berkeley County	9
Wood County	10
Kanawha County	12
Monongalia County	18

- can sort by character or numeric variables
- can sort by multiple variables, separated by comma
- sorting variable doesn't need to be in `select` clause

Creating a data set

SAS Code

```
PROC SQL ;
   CREATE TABLE wv AS
   SELECT region, division, county, patents
   FROM patents
   WHERE state="WEST VIRGINIA"
   ORDER BY patents
   ;
QUIT;

PROC PRINT DATA = wv ;
RUN ;
```

SAS Code

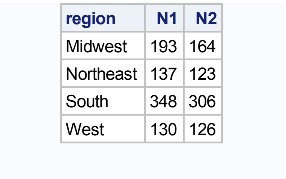
- Syntax: CREATE TABLE *datasetname* AS
- no output generated from PROC SQL

Using DISTINCT, example 1

SAS Code

```
PROC SQL;
  SELECT region,
    COUNT(county) AS N1,
    COUNT(DISTINCT county) AS N2
  FROM patents
  GROUP BY region
  ;
QUIT;
```

SAS Code

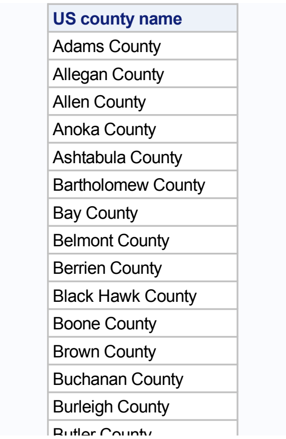


region	N1	N2
Midwest	193	164
Northeast	137	123
South	348	306
West	130	126

- N1 is the total number of counties in each region
- N2 is the number of unique county names in each region

Using DISTINCT, example 2

```
                    SAS Code
PROC SQL ;
   SELECT DISTINCT county
   FROM patents
   WHERE region="Midwest"
   ORDER BY county
   ;
QUIT;
                    SAS Code
```



US county name
Adams County
Allegan County
Allen County
Anoka County
Ashtabula County
Bartholomew County
Bay County
Belmont County
Berrien County
Black Hawk County
Boone County
Brown County
Buchanan County
Burleigh County
Butler County

- prints unique county names in Midwest

Joins (combining data)

Feature	PROC SQL	DATA step merging
Requires sorted data sets	✗	✓
Requires same name for identifying variable	✗	✓
Can handle many to many relationship	✓	✗

http://www.listendata.com/2014/06/proc-sql-merging.html