- Demo .Rmd files
- Clicker Questions
  - Practice today & Friday
  - Live beginning next week
- Reader: Delay in hard copy
  - use copy on bcourses
  - Expect reader by next Monday
- Lab #1: released tomorrow at 8 a.m.

Data Frames

load(url('http://www.stat.berkeley.edu/users/nolan/data/afamily.rda'))

The Family

- We have all sorts of information about our family, height, weight, first name, gender, …
- The data frame gives us a way to collect all of these variables (vectors) into one object.

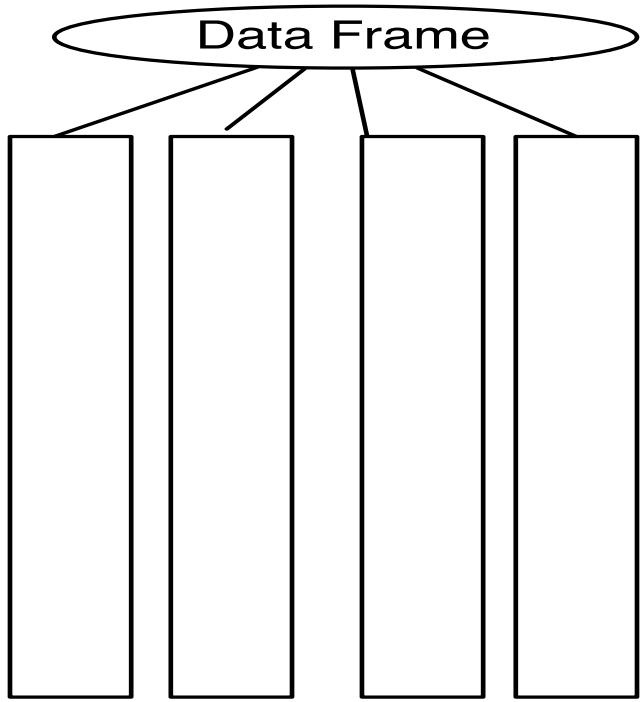
> data.frame(firstName = fnames,

sex = fsex, age = fage, height = fheight, weight = fweight, bmi = fbmi, overWt = foverWt)

```
> family
   firstName  sex age height weight      bmi overWt
1        Tom    m  77     70    175 25.16239   TRUE
2       Maya    f  33     64    124 21.50106  FALSE
3        Joe    m  79     73    185 24.45884  FALSE
4     Robert    m  47     67    156 24.48414  FALSE
5        Sue    f  27     61     98 18.51492  FALSE
6        Liz    f  33     68    190 28.94981   TRUE
7        Jon    m  67     68    185 28.18797   TRUE
8      Sally    f  52     65    124 20.67783  FALSE
9        Tim    m  59     68    175 26.66430   TRUE
10       Tom    m  27     71    215 30.04911   TRUE
11       Ann    f  55     67    166 26.05364   TRUE
12       Dan    m  24     66    140 22.64384  FALSE
13       Art    m  46     66    150 24.26126  FALSE
14       Zoe    f  48     62    125 22.91060  FALSE
```

Data Frame

1. *Ordered* container of vectors
2. Vectors must all be the *same length*
3. Vectors can be *different types*



```
> class(family)
[1] "data.frame"
> length(family)       - number of vectors in family
[1] 7
> dim(family)          - number of rows and columns
[1] 14  7
> names(family)        - names of the vectors in family
[1] "firstName" "gender"    "age"       "height"
[5] "weight"    "bmi"       "overWt"
```

Access a vector: dataframe$vector

```
> family$gender
 [1] m f m m f f m f m m f m m f
Levels: m f
> mean(family$height)
[1] 67.07143
> class(family$height)
[1] "numeric"
```

Subsetting Data frames

```
> family[ 10:13, -(3:14)]
   firstName sex
10       Tom   m
11       Ann   f
12       Dan   m
13       Art   m
```

We subset rows and columns of data frames

We subset by **position**, **exclusion**, **logical**, **name**, and **all**

```
family[  , c("sex", "firstName") ]
   sex firstName
1    m       Tom
2    f      Maya
3    m       Joe
4    m    Robert
5    f       Sue
6    f       Liz
7    m       Jon
8    f     Sally
9    m       Tim
10   m       Tom
11   f       Ann
12   m       Dan
13   m       Art
14   f       Zoe
```

Subset rows by **all** and columns by **name**

What's different about the return value?

The order of the columns is different than the order in the data frame. It matches the order of the names

```
> family[family$weight > 180, c("height", "bmi")]
   height      bmi
3      73 24.45884
6      68 28.94981
7      68 28.18797
10     71 30.04911
```

We subset the rows using a **logical** vector

We subset the columns by **name**

dataframe[ ]

```
> family["height"]
   height
1      70
2      64
3      73
4      67
5      61
6      68
7      68
8      65
9      68
10     71
11     67
12     66
13     66
14     62
```

```
> family[ , "height"]
 [1] 70 64 73 67 61 68 68 65 68 71 67 66 66 62
```

What's the difference between these two expressions?

```
> class(family["height"])
[1] "data.frame"
> class(family[, "height"])
[1] "numeric"
```

One returns a data frame and the other returns a vector

Reading Data Tables into R

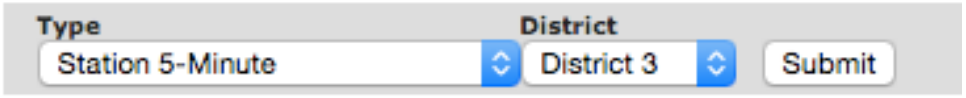
Traffic on I-80

PEMS Data Clearinghouse

Clearinghouse

The *Data Clearinghouse* provides a single access point for downloading PeMS data sets. You can use this page to quickly locate data by district, month and format.

After selecting the district, the type of data set, and clicking the submit button, you will be presented with a calendar for that data set. The chart shows you what months (and completeness) are available. We present a year of data at a time for ease of downloading.



D3 2016 Link 5-Minute

File Formats & Data Sets

PeMS exports data in a variety of file formats including HPMS and comma-delimited ASCII text. Each file format has an associated list of data sets that it supports. For example, the HPMS standard specifies four distinct record types: stations, volumes, vehicle classification and truck weights. The exact list of data sets depends on the data

Field Specification

utc_time_id

Plain Text Data

```
'Timestamp','Lane 1 Occ','Lane 1 Flow',\
'Lane 2 Occ’,'Lane 2 Flow','Lane 3 Occ',\
'Lane 3 Flow'
3/14/2003 00:00:00,.01,14,.0186,27,.0137,17
3/14/2003 00:05:00,.0133,18,.025,39,.0187,25
3/14/2003 00:10:00,.0088,12,.018,30,.0095,11
3/14/2003 00:15:00,.0115,16,.0203,33,.0217,19
3/14/2003 00:20:00,.0069,8,.0178,25,.0123,13
3/14/2003 00:25:00,.0077,11,.0151,24,.0092,13
```

What do you notice about the organization of the data?

Characteristics of the Traffic file

- First line has column names
- Lines are different lengths depending on the number of digits for a value
- Values are separated by commas (CSV)
- Time stamp has blanks and slashes and colons

Reading data into R

- Many data sets are stored in text files.
- An easy way to read these into R is to use the **`read_delim()`** function in the **`readr`** package.
- There are several arguments; 2 are required
  - file - name or URL
  - delim – specify the separator of elements in a row

```
require(readr)
traffic = read_delim(
   "flow-occ.txt", delim = ",")

head(traffic)
```

```
  'Timestamp'           'Lane 1 Occ'  'Lane 1
Flow'…
1 3/14/2003 00:00:00        0.0100              14
2 3/14/2003 00:05:00        0.0133              18
3 3/14/2003 00:10:00        0.0088              12
4 3/14/2003 00:15:00        0.0115              16
5 3/14/2003 00:20:00        0.0069               8
6 3/14/2003 00:25:00        0.0077              11
```

`readr()` determines variable classes

```
sapply(traffic, class)
```

```
'Timestamp'  'Lane 1 Occ' 'Lane 1 Flow'
"character"     "numeric"     "integer"

'Lane 2 Occ''Lane 2 Flow' …
   "numeric"     "integer"
```

What's interesting about these variables?

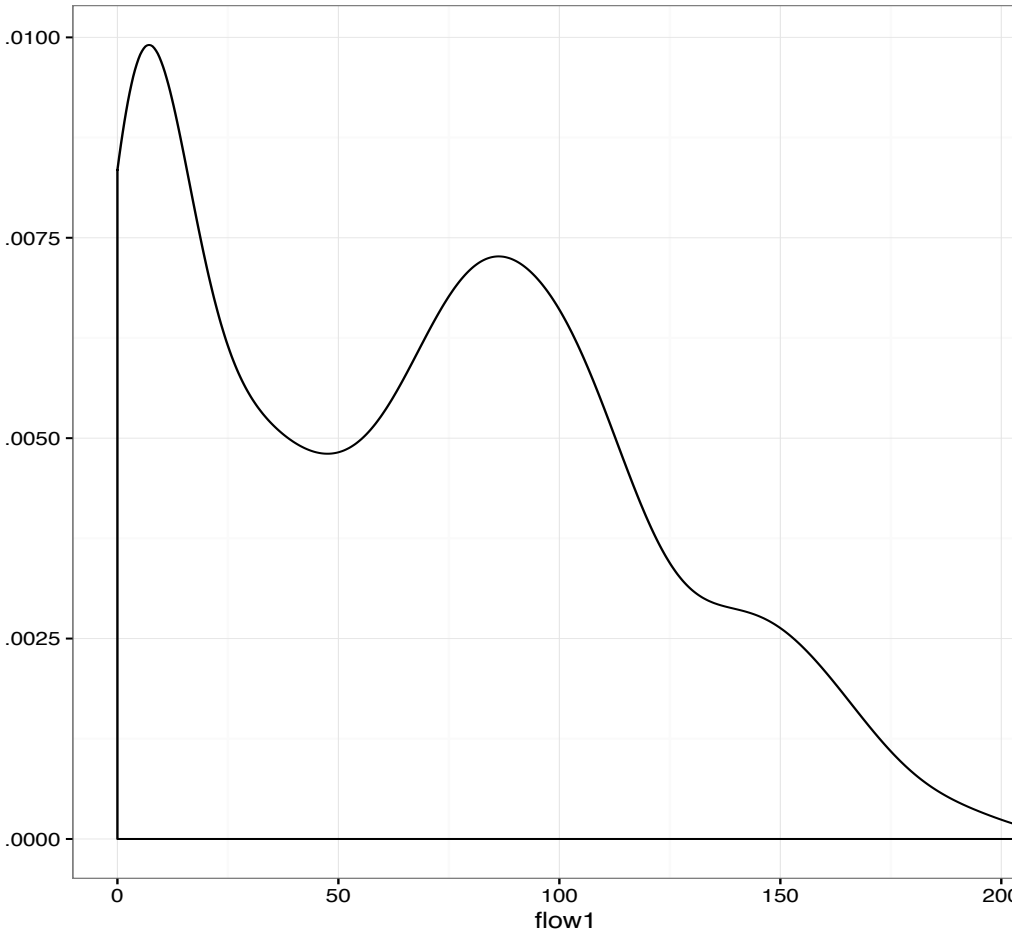
Exploration

1. What is the shape of the distribution of flow in the right lane?
2. Do you think this distribution is the same for all lanes?
3. How does flow vary with the time of day?
4. What does the relationship between flow and occupancy look like?

1. Shape of Distribution of Flow

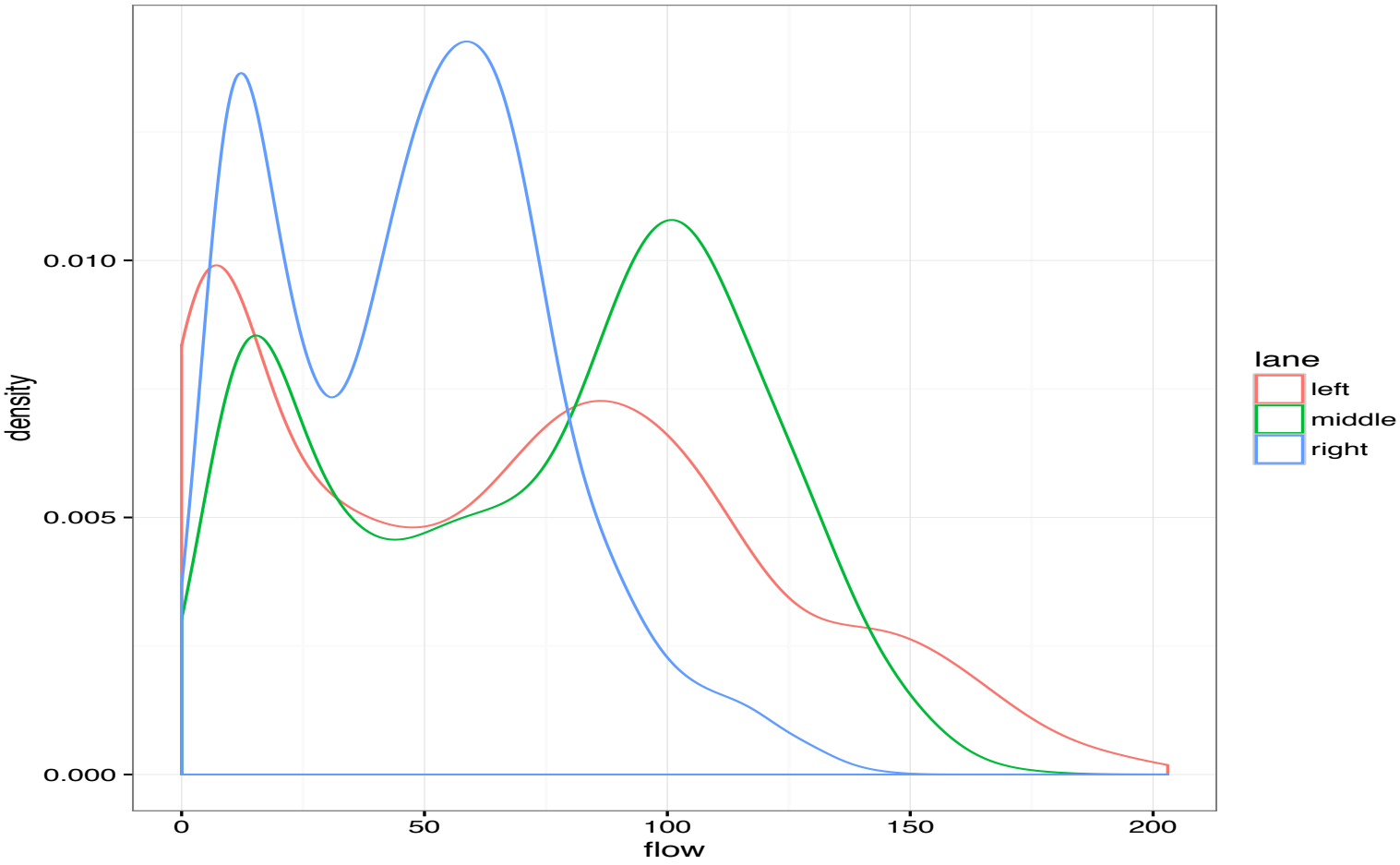
A. Symmetric
B. Skew right
C. Skew left

Bimodal,
skew right

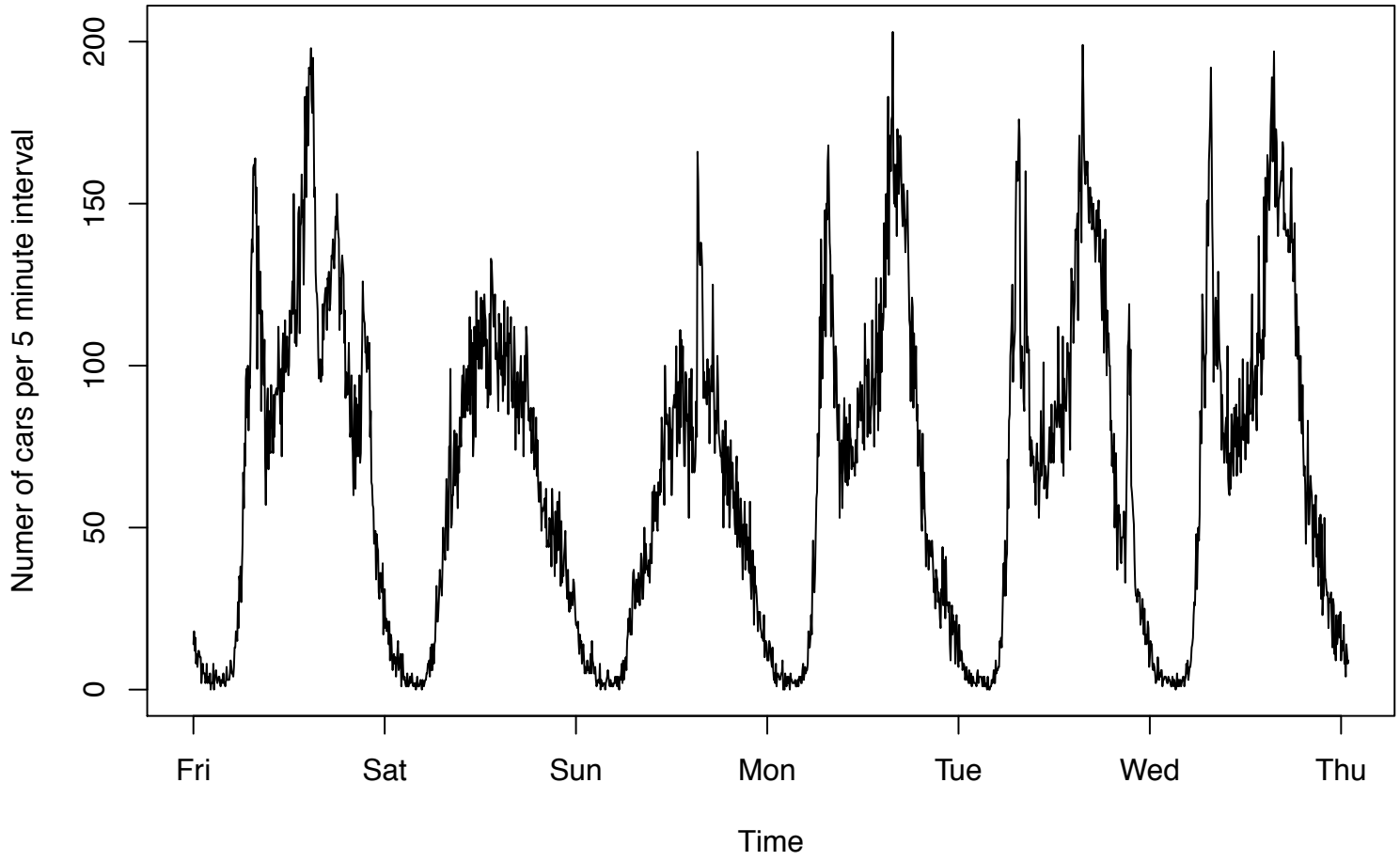


2. Distribution same for all lanes?

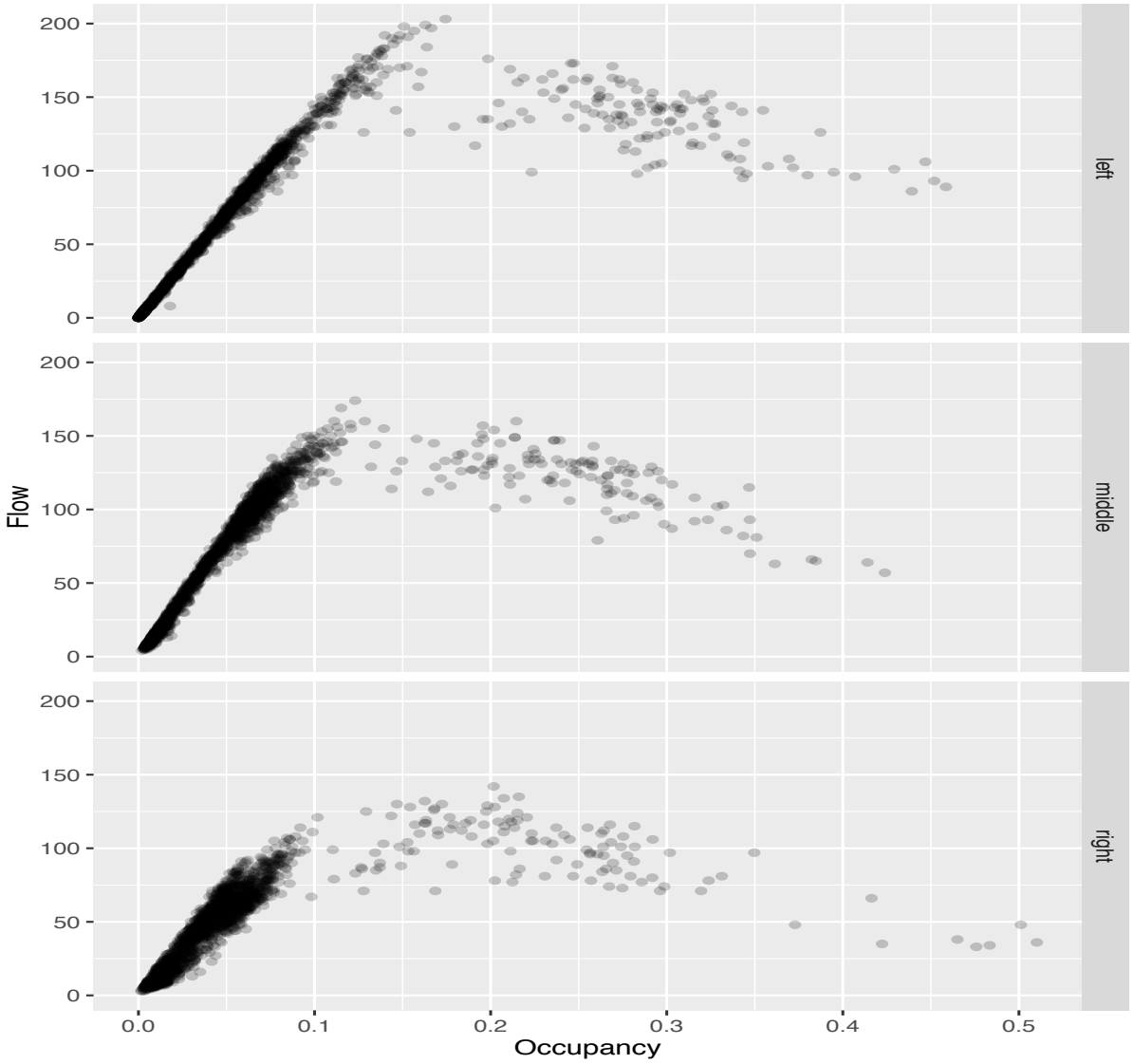
A. YES
B. NO



3. Flow in time



4. Relationship Flow and Occupancy?



Implications

- Lane matters – distributions but location of modes and spread are different
- Relationship between Flow and Occupancy
  - Linear relationship is not adequate
  - Traffic breaks down
  - Lane matters for slope and break down
- Distinct patterns over time of day and day of week

Get the Data Ready for Analysis

- Change variable names to something easier to work with
- Change time from strings to dates
- Stack the flow from all 3 lanes into one variable
- Ditto for occupancy
- Create a new vector indicating the lane

`names()` on the left

```
names(traffic)
[1] "'Timestamp'"   "'Lane 1 Occ'"
"'Lane 1 Flow'" "'Lane 2 Occ'" …
```

Fix – Reassign

```
names(traffic) =
  c("time", "occ1", "flow1","occ2",
    "flow2", "occ3", "flow3")
```

Special Data Type for Dates

- POSIX - a standard format developed by the IEEE
- Recognized by many R functions

```
traffic$time =
   as.POSIXct(traffic$time,
                format = "%m/%d/%Y %H:%M:%S")
```

Stack flow for the 3 lanes

```
 flow1 flow2 flow3
1   14     27     17
2   18     39     25
3   12     30     11
4   16     33     19
5    8     25     13
…
1738 11    20     13
1739  8    12      8
1740  9    11     12
```

```
head(flow)
[1] 14 18 12 16  8 11

tail(flow)
[1] 18  9 18 13  8 12
```

```
flow = stack(
  traffic[ , c("flow1", "flow2", "flow3")]
  )$values
```

Create a vector for lanes

```
lane =
  factor(
    rep(c("left", "middle", "right"),
        each = nrow(traffic)) )

time = rep(traffic$time, 3)

trafficLong =
   data.frame(lane, flow, occ,
                time = newtime)
```

	lane	flow	occ	time
1	left	14	0.0100	2003-03-14 00:00:00
2	left	18	0.0133	2003-03-14 00:05:00
3	left	12	0.0088	2003-03-14 00:10:00
4	left	16	0.0115	2003-03-14 00:15:00
5	left	8	0.0069	2003-03-14 00:20:00
6	left	11	0.0077	2003-03-14 00:25:00

...

	lane	flow	occ	time
5215	right	18	0.0199	2003-03-20 00:30:00
5216	right	9	0.0059	2003-03-20 00:35:00
5217	right	18	0.0234	2003-03-20 00:40:00
5218	right	13	0.0206	2003-03-20 00:45:00
5219	right	8	0.0063	2003-03-20 00:50:00
5220	right	12	0.0105	2003-03-20 00:55:00