

Data Summarization

Introduction to R for Public Health Researchers

Data Summarization

- ▶ Basic statistical summarization
 - ▶ `mean(x)`: takes the mean of `x`
 - ▶ `sd(x)`: takes the standard deviation of `x`
 - ▶ `median(x)`: takes the median of `x`
 - ▶ `quantile(x)`: displays sample quantities of `x`. Default is min, IQR, max
 - ▶ `range(x)`: displays the range. Same as `c(min(x), max(x))`

Some examples

We can use the `mtcars` and Charm City Circulator datasets to explore different ways of summarizing data. The `head` command displays the first 6 (default) rows of an object:

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0

Statistical summarization

Note - the \$ references/selects columns from a data.frame/tibble:

```
mean(mtcars$hp)
```

```
[1] 146.6875
```

```
quantile(mtcars$hp)
```

0%	25%	50%	75%	100%
52.0	96.5	123.0	180.0	335.0

Statistical summarization

```
median(mtcars$wt)
```

```
[1] 3.325
```

```
quantile(mtcars$wt, probs = 0.6)
```

```
60%
```

```
3.44
```

Statistical summarization

`t.test` will be covered more in detail later, gives a mean and 95% CI:

```
t.test(mtcars$wt)
```

One Sample t-test

```
data:  mtcars$wt
```

```
t = 18.6, df = 31, p-value < 2.2e-16
```

```
alternative hypothesis: true mean is not equal to 0
```

```
95 percent confidence interval:
```

```
 2.864478 3.570022
```

```
sample estimates:
```

```
mean of x
```

```
 3.21725
```

Statistical summarization

Note that many of these functions have additional inputs regarding missing data, typically requiring the `na.rm` argument (“remove NAs”).

```
x = c(1,5,7,NA,4,2, 8,10,45,42)
mean(x)
```

```
[1] NA
```

```
mean(x, na.rm = TRUE)
```

```
[1] 13.77778
```

```
quantile(x, na.rm = TRUE)
```

0%	25%	50%	75%	100%
1	4	7	10	45

Data Summarization on matrices/data frames

- ▶ Basic statistical summarization

- ▶ `rowMeans(x)`: takes the means of each row of `x`
- ▶ `colMeans(x)`: takes the means of each column of `x`
- ▶ `rowSums(x)`: takes the sum of each row of `x`
- ▶ `colSums(x)`: takes the sum of each column of `x`
- ▶ `summary(x)`: for data frames, displays the quantile information

TB Incidence

Please download the TB incidence data:

http:

[//www.aejaffe.com/winterR_2017/data/tb_incidence.xlsx](http://www.aejaffe.com/winterR_2017/data/tb_incidence.xlsx)

Here we will read in a data.frame of values from TB incidence:

```
library(readxl)
tb <- read_excel("../..//data/tb_incidence.xlsx")
head(tb)
```

```
# A tibble: 6 × 19
```

```
  `TB incidence, all forms (per 100 000 population per year
```

```
<ch
```

```
1                                Afghanistan
```

```
2                                Albanian
```

```
3                                Algerian
```

```
4                                American Samoa
```

```
5                                Andorra
```

```
6                                Angola
```

Indicator of Mortality

We can rename the first column to be the country measured using the `rename` function in `dplyr` (we have to use the `'` things because there are spaces in the name):

```
library(dplyr)
tb = rename(tb,
             country = `TB incidence, all forms (per 100 000`)
```

Column and Row means

`colMeans` and `rowMeans` must work on all numeric data. We will subset years before 2000:

```
avgs = select(tb, starts_with("1"))  
colMeans(avgs, na.rm = TRUE)
```

	1990	1991	1992	1993	1994	1995	1996	1997
105.5797	107.6715	108.3140	110.3188	111.9662	114.1981	115.3140	116.5169	117.6435
	1998	1999						
121.5169	125.0435							

```
tb$before_2000_avg = rowMeans(avgs, na.rm = TRUE)  
head(tb[, c("country", "before_2000_avg")])
```

A tibble: 6 × 2

	country	before_2000_avg
	<chr>	<dbl>
1	Afghanistan	168.0
2	Albania	22.2

Summary

Using summary can give you rough snapshots of each column, but you would likely use mean, min, max, and quantile when necessary:

```
summary(tb)
```

country	1990	1991	1992
Length:208	Min. : 0.0	Min. : 4.0	Min. : 0.0
Class :character	1st Qu.: 27.5	1st Qu.: 27.0	1st Qu.: 27.0
Mode :character	Median : 60.0	Median : 58.0	Median : 60.0
	Mean :105.6	Mean :107.7	Mean :105.6
	3rd Qu.:165.0	3rd Qu.:171.0	3rd Qu.:165.0
	Max. :585.0	Max. :594.0	Max. :585.0
	NA's :1	NA's :1	NA's :1
1993	1994	1995	1996
Min. : 4.0	Min. : 0	Min. : 3.0	Min. : 0
1st Qu.: 27.5	1st Qu.: 26	1st Qu.: 26.5	1st Qu.: 25
Median : 56.0	Median : 57	Median : 58.0	Median : 60
Mean :110.3	Mean :112	Mean :114.2	Mean :115
3rd Qu.:171.0	3rd Qu.:171	3rd Qu.:177.5	3rd Qu.:175
Max. :585.0	Max. :585	Max. :594	Max. :585
NA's :1	NA's :1	NA's :1	NA's :1

Apply statements

You can apply more general functions to the rows or columns of a matrix or data frame, beyond the mean and sum.

```
apply(X, MARGIN, FUN, ...)
```

X : an array, including a matrix.

MARGIN : a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names.

FUN : the function to be applied: see 'Details'.

... : optional arguments to FUN.

Apply statements

```
apply(avgs,2,mean,na.rm=TRUE) # column means
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
105.5797	107.6715	108.3140	110.3188	111.9662	114.1981	115.3140	116.5169	117.6715	118.8289	120.9811
121.5169	125.0435	128.5701	132.0967	135.6233	139.1498	142.6764	146.2030	149.7296	153.2562	156.7828

```
apply(avgs,2,sd,na.rm=TRUE) # columns sds
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
110.6440	112.7687	114.4853	116.6744	120.0931	122.7119	126.1307	128.7494	131.3682	133.9869	136.6057
137.3754	146.0755	154.7756	163.4757	172.1758	180.8759	189.5760	198.2761	206.9762	215.6763	224.3764

```
apply(avgs,2,max,na.rm=TRUE) # column maxs
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
585	594	606	618	630	642	655	668	681	695	708
708	721	734	747	760	773	786	799	812	825	838

Other Apply Statements

- ▶ `tapply()`: 'grouping' apply
- ▶ `lapply()`: 'list' apply [tomorrow]
- ▶ `sapply()`: 'simple' apply [tomorrow]
- ▶ Other less used ones...

See more details here: <http://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/>

Youth Tobacco Survey

Please download the Youth Tobacco Survey data. You can also read it in directly from the web:

```
library(readr)
smoke = read_csv(
  "http://www.aejaffe.com/winterR_2017/data/Youth_Tobacco_S
```


Subsetting to specific columns

Let's just take smoking status measures for all genders using `filter`, and the columns that represent the year, state using `select` (covered more in detail later):

```
library(dplyr)
sub_smoke = filter(smoke,
                    MeasureDesc == "Smoking Status",
                    Gender == "Overall",
                    Response == "Current")
sub_smoke = select(sub_smoke, YEAR, LocationDesc, Data_Value)
head(sub_smoke, 4)
```

A tibble: 4 × 3

	YEAR	LocationDesc	Data_Value
	<int>	<chr>	<dbl>
1	2015	Arizona	3.2
2	2015	Connecticut	0.8
3	2015	Connecticut	5.6
4	2015	Connecticut	10.0

tapply()

From the help file: “Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.”

```
tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)
```

Simply put, you can apply function FUN to X within each categorical level of INDEX. It is very useful for assessing properties of continuous data by levels of categorical data.

tapply()

For example, we can estimate the average current smoking statuses over all states for each year:

```
tapply(sub_smoke$Data_Value, sub_smoke$YEAR, max, na.rm = TRUE)
```

[illegible]

Perform Operations By Groups: dplyr

`group_by` allows you group the data in a more intuitive way than `tidy::tapply`

We will use `group_by` to group the data by line, then use `summarize` (or `summarise`) to get the mean Average ridership:

```
summarize(group_by(sub_smoke, YEAR), year_avg = mean(Data_V
```

```
# A tibble: 17 × 2
  YEAR  year_avg
  <int>    <dbl>
1  1999  20.493333
2  2000  19.878431
3  2001  15.661111
4  2002  16.802326
5  2003  13.176190
6  2004  13.926923
7  2005  14.128571
8  2006  14.113636
```

Using the pipe (comes with dplyr):

Pipe sub_smoke into group_by, then pipe that into summarize:

```
smoke_avgs = sub_smoke %>%  
  group_by(YEAR) %>%  
  summarize(year_avg = mean(Data_Value, na.rm = TRUE))  
head(smoke_avgs)
```

```
# A tibble: 6 × 2
```

	YEAR	year_avg
	<int>	<dbl>
1	1999	20.49333
2	2000	19.87843
3	2001	15.66111
4	2002	16.80233
5	2003	13.17619
6	2004	13.92692

Data Summarization/Visualization

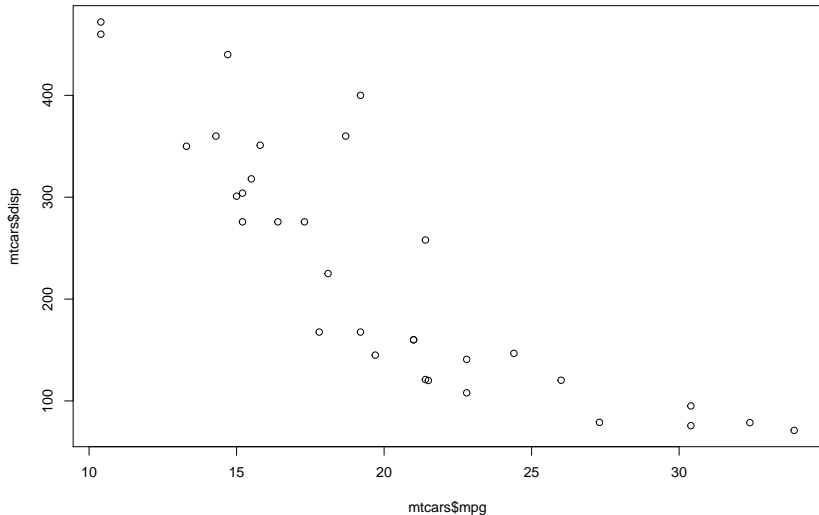
- ▶ Basic summarization plots
 - ▶ `plot(x,y)`: scatterplot of x and y
 - ▶ `boxplot(y~x)`: boxplot of y against levels of x
 - ▶ `hist(x)`: histogram of x
 - ▶ `density(x)`: kernel density plot of x

Basic Plots

Plotting is an important component of exploratory data analysis. We will review some of the more useful and informative plots here. We will go over formatting and making plots look nicer in additional lectures.

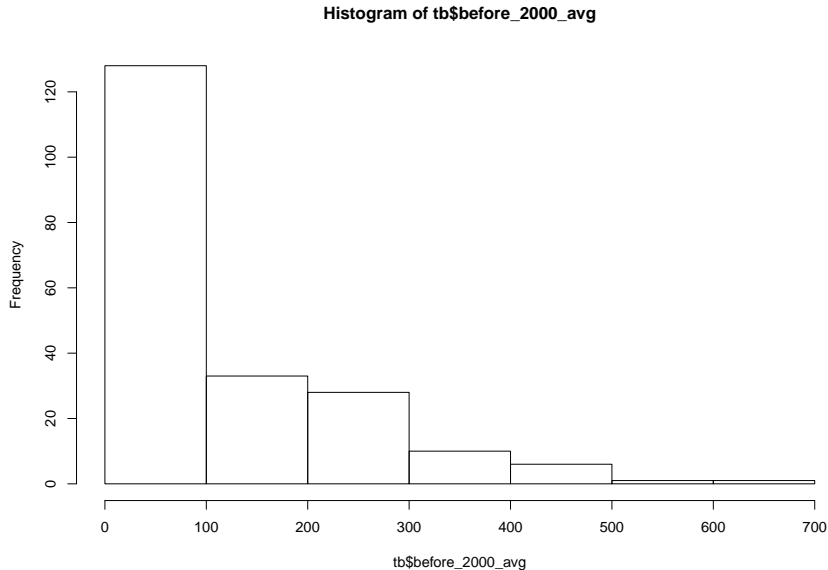
Scatterplot

```
plot(mtcars$mpg, mtcars$displacement)
```



Histograms

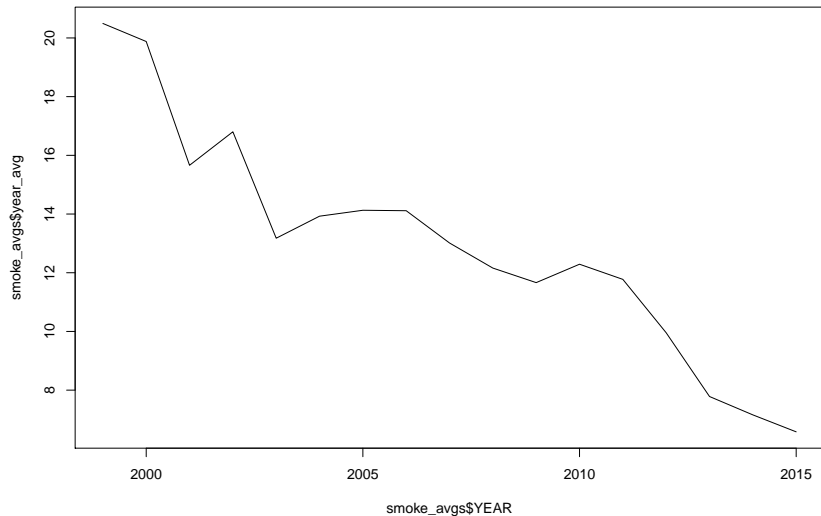
```
hist(tb$before_2000_avg)
```



Plot with a line

`type = "l"` means a line

```
plot(smoke_avgs$YEAR, smoke_avgs$year_avg, type = "l")
```

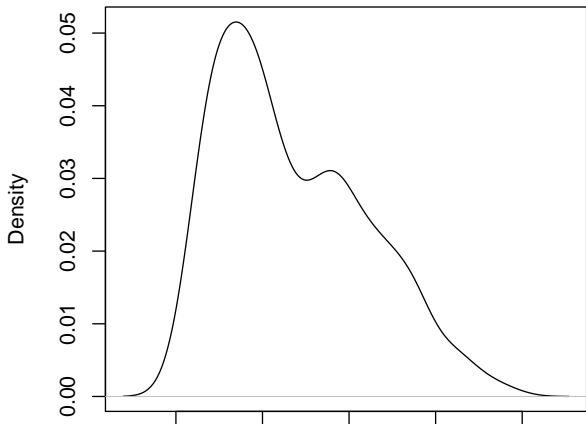


Density

Over all years and states, this is the density of smoking status incidence:

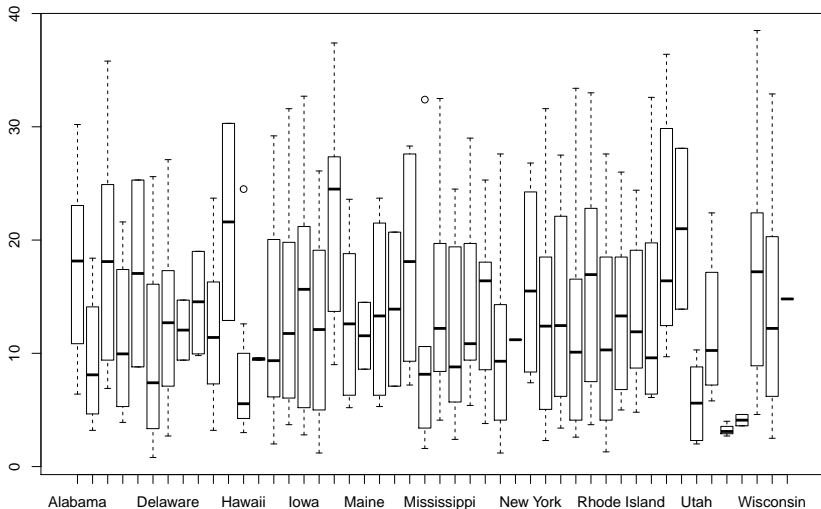
```
plot(density(sub_smoke$Data_Value))
```

density.default(x = sub_smoke\$Data_Value)



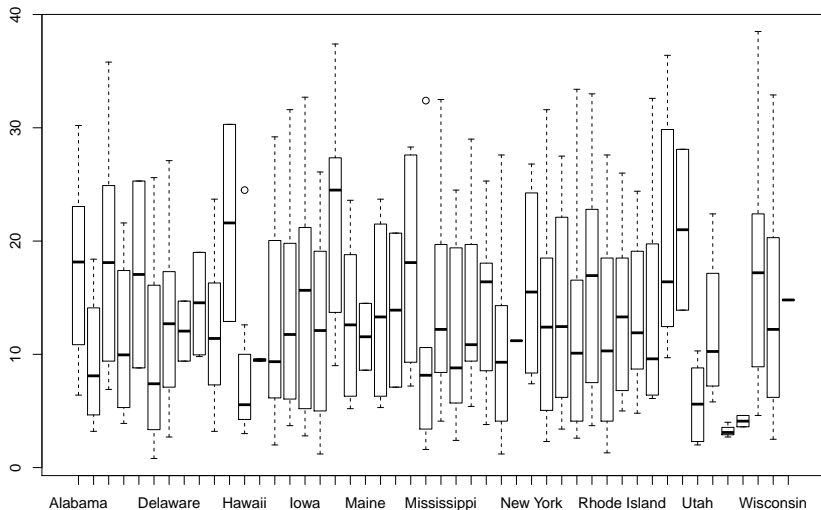
Boxplots

```
boxplot(sub_smoke$Data_Value ~ sub_smoke$LocationDesc)
```



Boxplots

```
boxplot(Data_Value ~ LocationDesc, data = sub_smoke)
```

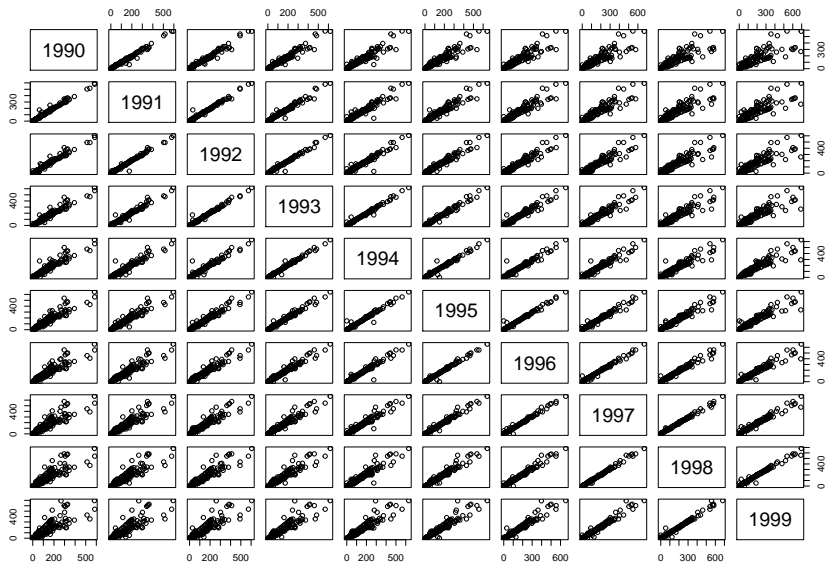


Data Summarization for data.frames

- ▶ Basic summarization plots
 - ▶ `matplot(x,y)`: scatterplot of two matrices, x and y
 - ▶ `pairs(x,y)`: plots pairwise scatter plots of matrices x and y, column by column

Matrix plot

`pairs`(avgs)



Conclusion

- ▶ Base R has apply statements that perform things repeatedly.
- ▶ dplyr has a lot of more intuitive syntax.
 - ▶ `group_by` is very powerful, especilly with `summarise/summarize`
- ▶ Base R has good things for quickly summarizing rows or columnns of all numeric data.
 - ▶ The `matrixStats` package extends this to `colMedians`, `colMaxs`, etc.