

# R

Bioinformatics Applications (PLPTH813)

Sanzhen Liu

1/26/2017

# Outline

- R introduction
- Data structure
- Data input and output
- Basic graphics
- String operations

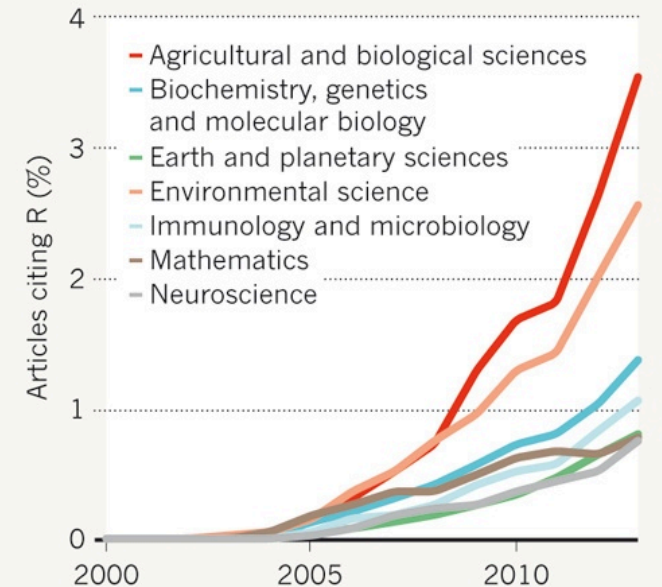
# R

- R is a programming language and a cutting-edge tool for data analysis, especially for statistical computing and graphics.
- R is free
- R is powerful. Applications are easily extensible by creating new functions. Functions are usually distributed through packages.
- It has great community support.

[www.r-project.org](http://www.r-project.org)

## A RISING TIDE OF R

An increasing proportion of research articles explicitly reference R or an R package.



<http://www.nature.com/news/programming-tools-adventures-with-r-1.16609#/rise>

# Examples

- Statistical test

```
d <- c(12, 36, 24, 70)
dm <- matrix(d, nrow=2, byrow=T)
chisq.test(dm)
```

data: dm

X-squared = 0, df = 1, p-value = 1

A	B	
	12	36
	24	70

- Graphic

```
source("http://plantgenomics.ksu.edu/liulab/wp-content/uploads/2016/01/xmas.card_.r")
```

# R commands, case sensitivity

- **Expression**

Print the value and not save the value in the environment

```
2 + 4
```

- **Assignment**

Assign values to a **variable**

```
y <- 2
```

```
y = 2
```

```
assign("y", 2)
```

```
Y <- 2 + 4
```

- **Comments (#)**

Notes/explanation to the scripts, starting with a hashmark ('#'), everything to the end of the line is a comment.

```
y <- 2 + 4 # an example of the assignment
```

# Data structure - vector

A vector is a single entity consisting of an ordered collection of numbers, characters, logical quantities, etc.

- **Numeric vector**

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

```
sum(x)
```

```
y <- 2
```

```
2*x + y
```

- **Logical vector**

```
lv <- c(TRUE, FALSE, TRUE, TRUE)
```

```
lv == FALSE
```

```
sum(lv)
```

```
# The logical operators are <, <=, >, >=, ==, and !=.
```

```
# == for exact equality and != for inequality.
```

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

```
lv2 <- x > 10
```

- **Character vectors**

```
cv <- c("a", "b", "c")
```

```
cv2 <- paste(cv, 1:3, sep="")
```

- **Missing values: NA, not available**

```
mvv <- c("a", "b", "c", NA)
```

```
is.na(mvv)
```

# Select a subset and modify a vector

- **Select a subset of a vector**

```
x <- c(4, 5, 7, 3, 9)
```

```
x[c(2, 3)]
```

```
x[x>10]
```

```
x[-c(1,5)]
```

- **Modify a vector**

```
x[3] <- 23.1
```

```
x <- c(x, 10.9)
```

```
names(x) <- c("a", "b", "c", "d", "e", "f")
```

# mode and length of a vector

- **Mode**

Vectors must have their values with the same mode, either numeric, character, logical, or other types.

```
z <- 0:9
```

```
is.numeric(z)
```

```
digits <- as.character(z) # convert to character
```

```
d <- as.integer(digits) # convert to integer
```

- **Length**

```
length(z)
```

```
length(z) <- 5 # retain just the first 5 values
```



# factor

Definition: A factor is a vector object used to specify a discrete classification (grouping) of the components of other vectors with the same length.

- **factor = regular vector + Levels**

```
state <- c("tas", "sa", "qld", "nsw",  
          "nsw", "nt", "wa", "wa",  
          "qld", "vic", "nsw", "vic")  
statef <- factor(state)  
> statef  
[1] tas sa qld nsw nsw nt wa wa qld vic nsw vic  
Levels: nsw nt qld sa tas vic wa  
> levels(statef)  
[1] "nsw" "nt" "qld" "sa" "tas" "vic" "wa"
```

```
state2 <- as.character(statef)
```

# array and matrix

- array: 2 or more dimensions of data
- matrix: a special array with two dimensions

```
num <- 1:25  
numm <- matrix(num, nrow=5, byrow=T)  
nrow(numm)  
dim(numm)
```

matrices can be built up by using the functions `cbind()` and `rbind()`:

**`cbind()`** forms matrices by binding together matrices horizontally, or column-wise  
**`rbind()`** vertically, or row-wise.

Note: the result of `rbind()` or `cbind()` always has matrix status

# data.frame

name	age	>30?	gender
Josh	23	FALSE	male
Rose	35	TRUE	female
Jone	18	FALSE	male
Molly	21	FALSE	female
Lisa	36	TRUE	female

- **Data frame**

A data frame may be regarded as a matrix with columns possibly of differing modes and attributes. The data of a matrix are of the same type or mode.

- **Making data frames**

```
df <- data.frame(name=c("Josh", "rose"), age=c(23, 35))
```

- **Working with data frames**

```
> df$name
[1] Josh rose
Levels: Josh rose
> df[, 1]
[1] Josh rose
Levels: Josh rose
```

```
> df[[1]]
[1] Josh rose
Levels: Josh rose
> df[1]
  name
1 Josh
2 rose
```

```
head(df); tail(df); summary(df); str(df)
```

# list

A list is an object consisting of an ordered collection of objects.

- **Objects can be any types or modes**

```
lst <- list(name="Fred", wife="Mary", nkids=3, kid.ages=c(4,7,9))
```

```
> lst[1] # sublist
```

```
$name
```

```
[1] "Fred"
```

```
> lst[[1]] # first element in the list
```

```
[1] "Fred"
```

```
> lst$name # the element named "name"
```

```
[1] "Fred"
```

# Problem

```
df <- data.frame(name=c("Josh", "rose", "John"),  
age=c(23, 35, 18))
```

What are the values of

`df[2, 1]`

`df[3, 2]`

`df[2]`

`df[, 2]`

name	age
Josh	23
Rose	35
Jone	18

What is the difference between the last two?

# Data import

- **scan()**: to read data from a file to a vector or list

```
cat("lisa Jone", "28 21", file = "hrdb.txt", sep = "\n")  
hr <- scan("hrdb.txt", what=character())  
hr  
"lisa" "Jone" "28" "21"
```

- **read.table()**: to read a data frame (table) directly

read.delim, read.csv

```
d <- read.table(data)
```

Input file form with names and row labels:

	Price	Floor	Area	Rooms	Age	Cent.heat
01	52.00	111.0	830	5	6.2	no
02	54.75	128.0	710	5	7.5	no
03	57.50	101.0	1000	5	4.2	no
04	57.50	131.0	690	6	8.8	no
05	59.75	93.0	900	5	1.9	yes
...						

# Data export

- **write.table()** or **write.csv()**

```
## To write a tab-delimited file:
```

```
x <- data.frame(a = "pi", b = pi)
```

```
write.table(x, file="foo.txt", sep="\t", row.names=FALSE)
```

```
## and to read this file back into R one needs
```

```
read.table("foo.txt")
```

```
## Alternatively
```

```
write.csv(x, file = "foo.csv", row.names=FALSE)
```

```
read.csv("foo.csv")
```

# Outline

- R introduction
- Data structure
- Data input and output
- **Basic graphics**
- **String operations**



# Basic graphics

- **plot(); points(); lines(); abline(); text(); legend()**

**High-level plot: create a new plot**

```
plot(x, y, xlab, ylab, main, ...)
```

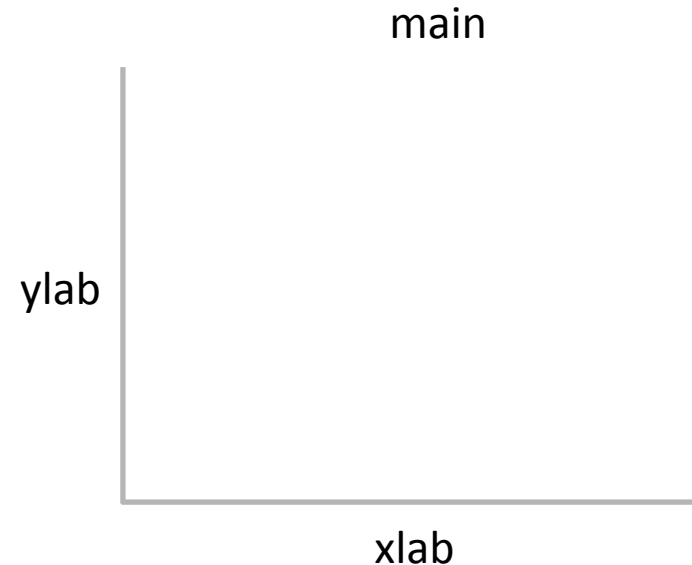
**Low-level plot: add to an existing plot**

```
# add points  
points(x, y)
```

```
# add lines  
lines(x, y)
```

```
# add horizontal or vertical lines  
abline(h, v)
```

```
# add text or legend  
text()  
legend()
```



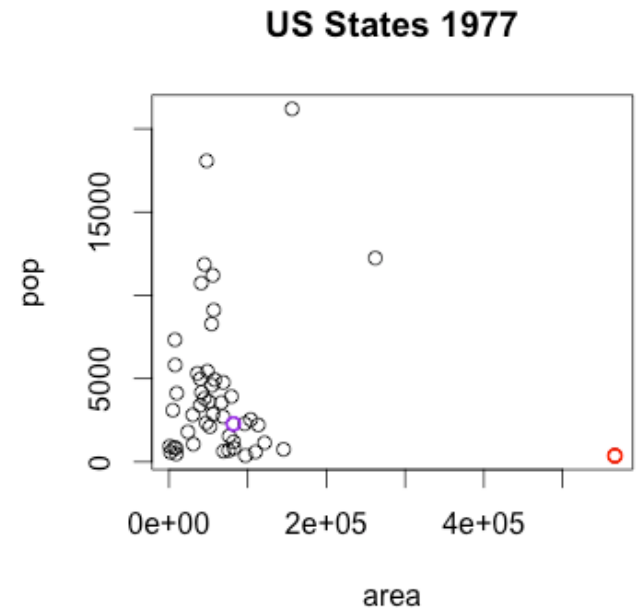
# Scatter plot

```
# data
area <- state.x77[, "Area"]
pop <- state.x77[, "Population"]

# scatter plot
plot(area, pop, main="US States 1977")

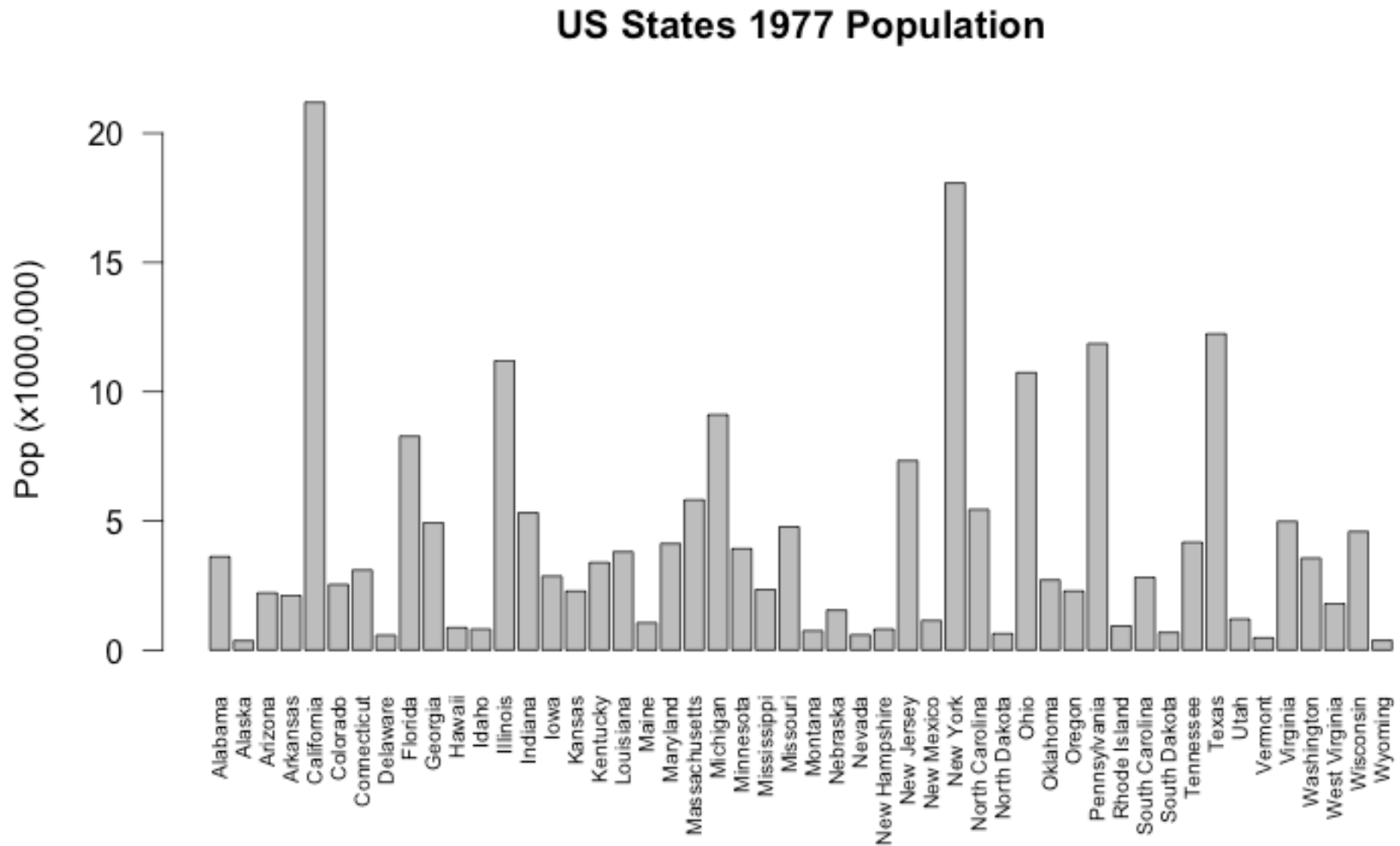
# label points
state.max.area <- which.max(area)
points(area[state.max.area],
       pop[state.max.area],
       col="red", lwd=2)

points(area["Kansas"], pop["Kansas"],
       col="purple", lwd=2)
```



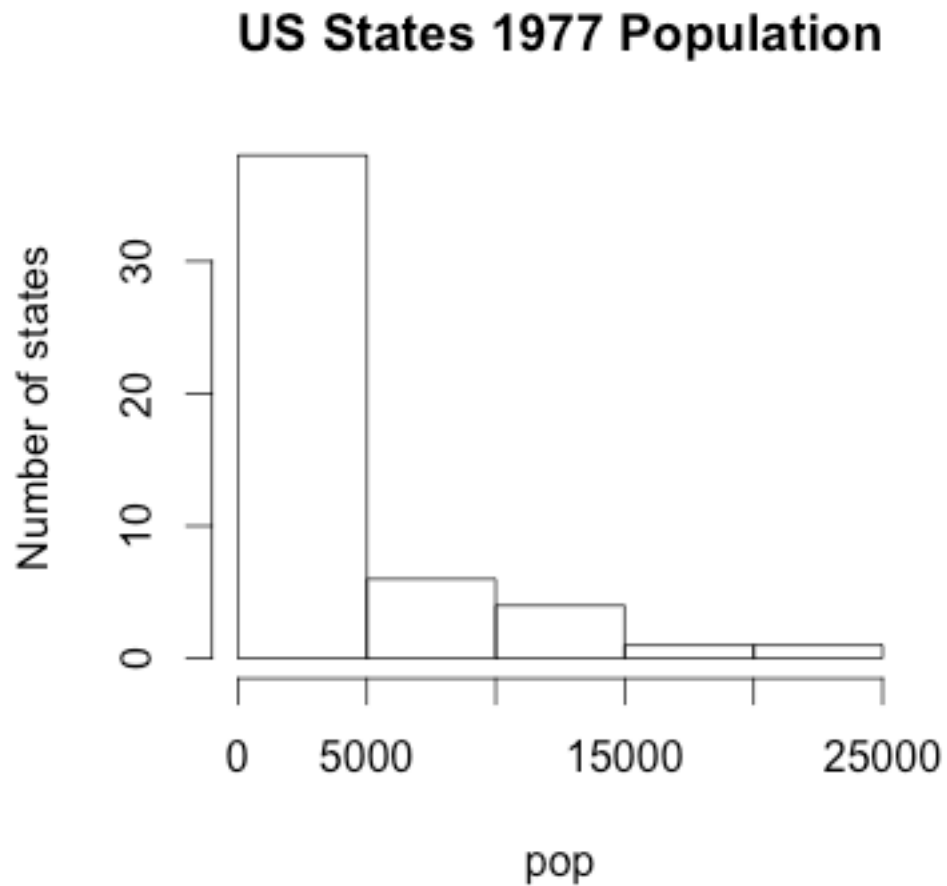
# Barplot

```
barplot(pop/1000, las=2, cex.names=0.65, ylab="Pop (x1000,000)",  
        main="US States 1977 Population")
```



# Histogram

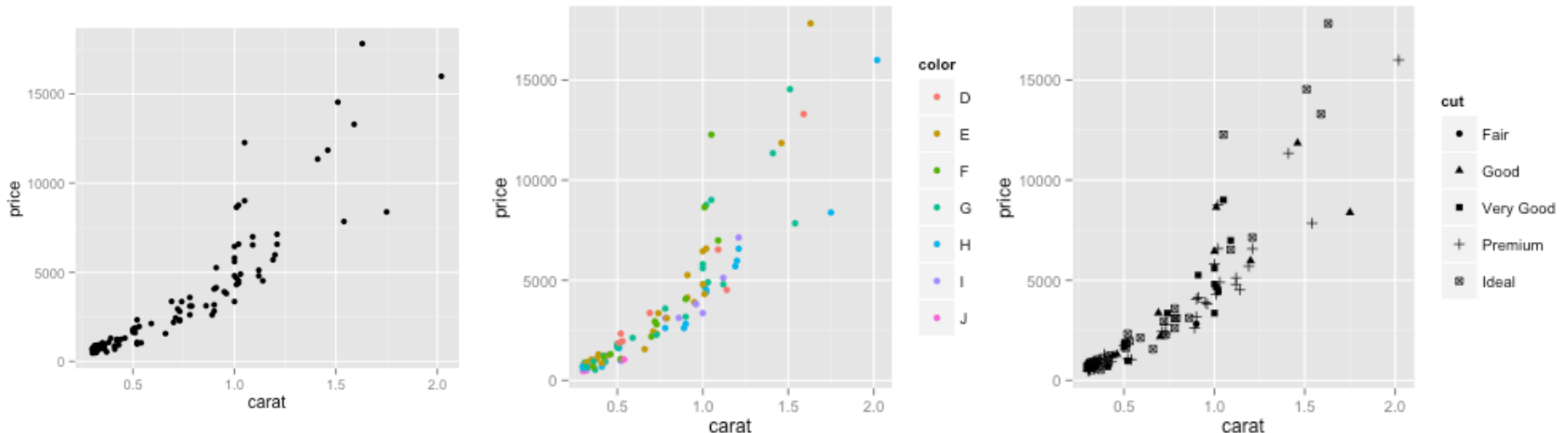
```
hist(pop, ylab="Number of states", main="US States 1977 Population")
```



# ggplot2 - an easy plotting package

**scatterplots** showing the relationship between the price and carats (weight) of a diamond\*.

```
qplot(carat, price, data = dsmall)
qplot(carat, price, data = dsmall, colour = color)
qplot(carat, price, data = dsmall, shape = cut)
```



\* from <http://ggplot2.org/book/qplot.pdf>

## ggplot2 - geom to control plot type

qplot is not limited to scatterplots, but can produce almost any kind of plot by varying the **geom**. geom has many options:

- "point" draws a scatterplot. This is the default.
- "smooth" fits a smoother to the data
- "boxplot" produces a box-and-whisker plot
- "line" draw lines between the data points.
- "histogram" draws a histogram
- "bar" makes a bar chart

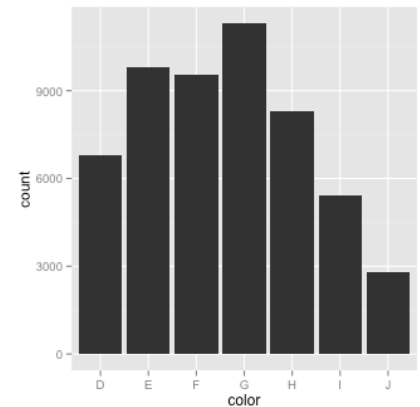
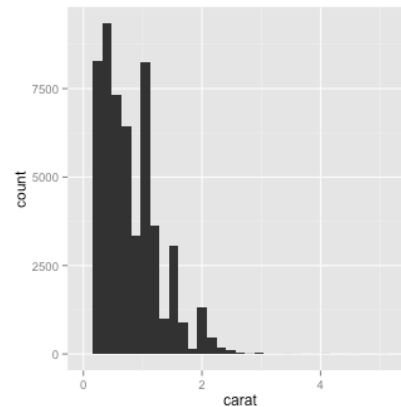
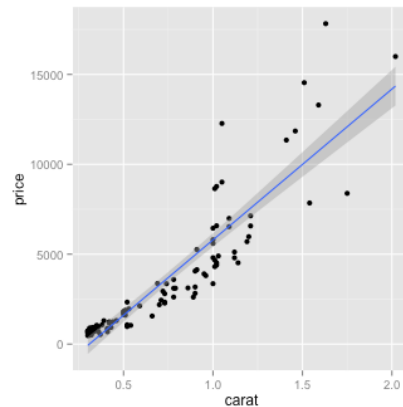
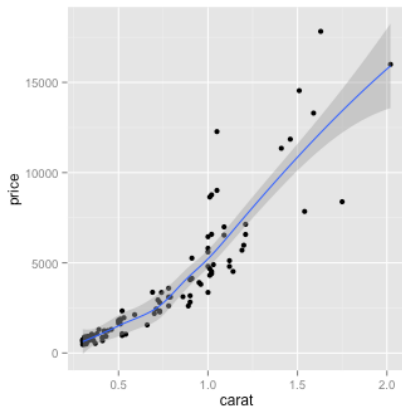
# ggplot2 – a flexible tool to plot various plots

## Adding a smooth line or a fitted line

```
qplot(carat, price, data = dsmall, geom = c("point", "smooth"))  
qplot(carat, price, data = dsmall, geom = c("point", "smooth"),  
      method = "lm")
```

## Histogram and barplot

```
qplot(carat, data = diamonds, geom = "histogram")  
qplot(color, data = diamonds, geom = "bar")
```



## String operations - nchar

- **nchar()**

nchar the sizes of the corresponding elements of a vector.

nchar(cvec)

```
> cvec
```

```
[1] "google" "hello"  "the"    "world"
```

```
> nchar(cvec)
```

```
[1] 6 5 3 5
```



## String operations - grep

- **grep()**

grep searches for matches to argument pattern within each element of a character vector

`grep("o", cvec)`

```
> cvec  
[1] "google" "hello"  "the"    "world"
```

```
> grep("o", cvec)  
[1] 1 2 4
```

## String operations – sub and gsub

- **sub()** and **gsub()**

sub and gsub perform replacement of the first and all matches respectively.

```
sub("o", "O", cvec)
```

```
gsub("o", "O", cvec)
```

```
> cvec
```

```
[1] "google" "hello"  "the"    "world"
```

```
> sub("o", "O", cvec)
```

```
[1] "gOogle" "hellO"  "the"    "wOrld"
```

```
> gsub("o", "O", cvec)
```

```
[1] "gOOgle" "hellO"  "the"    "wOrld"
```

## Get help

- `help(ls)`
- `?ls`
- `??colsum`: ambiguous search
- [R reference card](#)
- `stackoverflow`
- Google is the best helper!

R learning: <http://swirlstats.com/>

# Rstudio

Rstudio is an open source integrated development environment (IDE) for R

- On your own machine (Rstudio Desktop)

Download and install [R](#)

Download and install [Rstudio](#)

- Use Rstudio at Beocat (Rstudio server)

**rstudio.beocat.cis.ksu.edu**

Your KSU ID and password to login

# Adventures with R

