R Minicourse Workshop, Part 2

Presented to the
Washington State Deptment of Ecology
September 2–3, 2014

Dr. Robin Matthews, Institute for Watershed Studies
Dr. Geoffrey Matthews, Computer Science Department
Western Washington University

Graphical Presentation of Environmental Data

How *NOT* to Lie with Statistics

- One of the most important goals of research is to provide clear and unbiased summaries of the data
- Unfortunately, it is all too easy to obscure important elements from research, intentionally or unintentionally
- This portion of the `R-minicourse` will focus on creating effective graphical output while avoiding some of the more common problems

For more on data visualization:
How to Lie with Charts by Gerald Everett Jones (ISBN 978-1-419-65143-4)

Now You See It: Simple Visualization Techniques for Quantitative Analysis by Stephen Few (ISBN 978-0-970-60198-8)

Information Dashboard Design: Displaying Data for At-a-Glance Monitoring by Stephen Few (ISBN 978-1-938-37700-6)

Principles of Scientific Visualization

Memory Used for Processing Visual Information

We use three basic types of memory to process scientific information:

- *Iconic memory* (*pre-attentive processing*) for detecting visual information
- *Short-term memory* (*attentive or perceptual processing*) for temporary (limited) storage and is limited to ∼3–9 items
- *Long-term memory* for retaining information
    - Long-term memory can be created consciously or unconsciously
    - Information is stored more permanently, with cross-links that allow access back into short-term memory
    - Required for recognizing images, interpreting words and numbers, understanding context

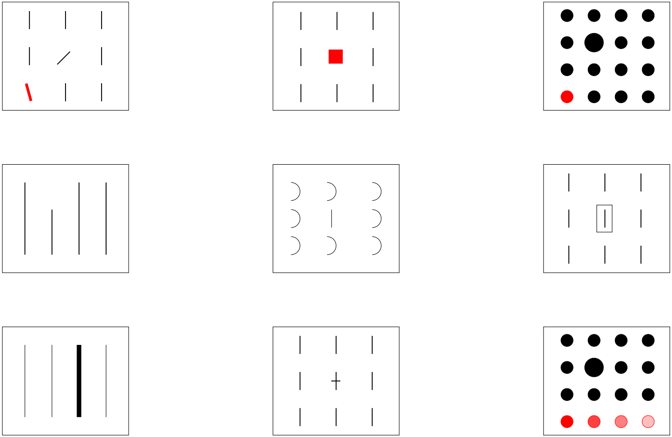
Principles of Scientific Visualization

Pre-Attentive Processing of Visual Information

- Iconic memory provides quick, subconscious processing of graphical information and is influenced by variations in:
    - form
    - color
    - spatial position
    - motion
- Graphics that make use of these features tend to make a strong impression on us, even when we don't know why

Principles of Scientific Visualization

Examples of Pre-Attentive Processing in Graphics



Figures modified from Show Me The Numbers by Stephen Few, Analytics Press, 2004

Principles of Scientific Visualization

Example of Pre-Attentive Processing in Tables

How many zeros are there?

6	4	4	2	1	5	7	2	2	2	2	8
9	8	9	3	6	5	5	5	7	8	7	6
1	3	5	9	5	6	0	6	7	6	6	6
7	4	2	5	7	7	1	5	5	5	4	2
5	2	1	1	4	2	6	6	4	9	6	3
5	7	2	0	6	1	6	8	0	6	0	2
9	8	7	4	4	5	4	4	9	1	5	1
2	1	3	7	8	6	2	0	2	9	4	9
3	4	9	6	2	1	7	9	4	8	2	8
2	5	5	2	2	4	5	5	8	7	1	5

How many zeros are there?

6	4	4	2	1	5	7	2	2	2	2	8
9	8	9	3	6	5	5	5	7	8	7	6
1	3	5	9	5	6	**0**	6	7	6	6	6
7	4	2	5	7	7	1	5	5	5	4	2
5	2	1	1	4	2	6	6	4	9	6	3
5	7	2	**0**	6	1	6	8	**0**	6	**0**	2
9	8	7	4	4	5	4	4	9	1	5	1
2	1	3	7	8	6	2	**0**	2	9	4	9
3	4	9	6	2	1	7	9	4	8	2	8
2	5	5	2	2	4	5	5	8	7	1	5

Principles of Scientific Visualization

Perceptual Processing of Visual Information

- Short-term and long-term memory require conscious interpretation of visual information
- As a result, it is easy to fool our visual perception of data, especially if you use pre-attentive processing
- In creating scientific graphics, careful use of color, shape, and position can **emphasize** or de-emphasize information
- Two major objectives in designing good tables or figures:
  - Highlight the data by enhancing "data ink" (reduce non-data ink)
  - Organize the data by grouping, prioritizing, and sequencing

Principles of Scientific Visualization

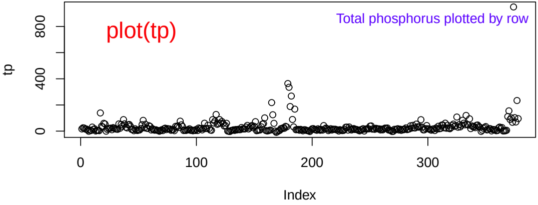
Perceptual Processing of Visual Information

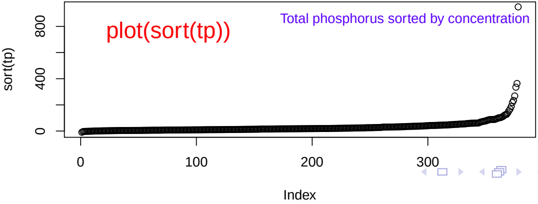
- Short-term and long-term memory involves conscious interpretation of visual information
- As a result, it is easy to fool our visual perception of data, especially if you use pre-attentive processing
- In creating scientific graphics, careful use of color, shape, and position can emphasize or de-emphasize information
- Two major objectives in designing good tables or figures:
  - Highlight the data by enhancing “data ink” (reduce non-data ink)
  - Organize the data by grouping, prioritizing, and sequencing

Building Simple Scatterplots Using plot()

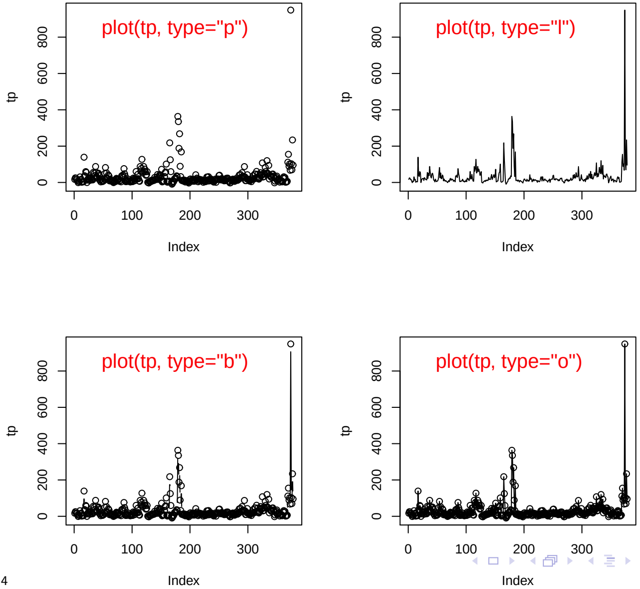
One of the most versatile plotting tools in R is the plot() function. In its simplest form, it can be used with very little modification to explore the data

If necessary, re-enter `read.table("lakes.csv", T, sep=","); attach(lakes)`

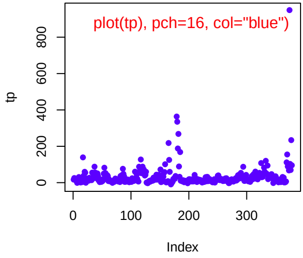




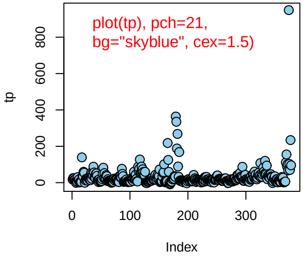
Plotting One Variable Using Points, Lines, or Both



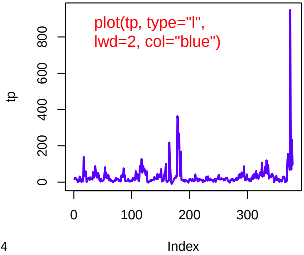
Changing Colors, Characters, Lines



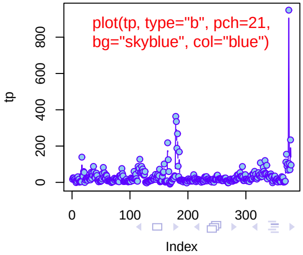
plot(tp), pch=16, col="blue")



plot(tp), pch=21,
bg="skyblue", cex=1.5)



plot(tp, type="l",
lwd=2, col="blue")



plot(tp, type="b", pch=21,
bg="skyblue", col="blue")

Saving and Copying R Figures

- R figures are directed to the graphics window
- Individual figures can be saved or copied from this window - select “emf” to minimize pixelation
- Each new figure overwrites the previous one unless you direct R to pause between figures:

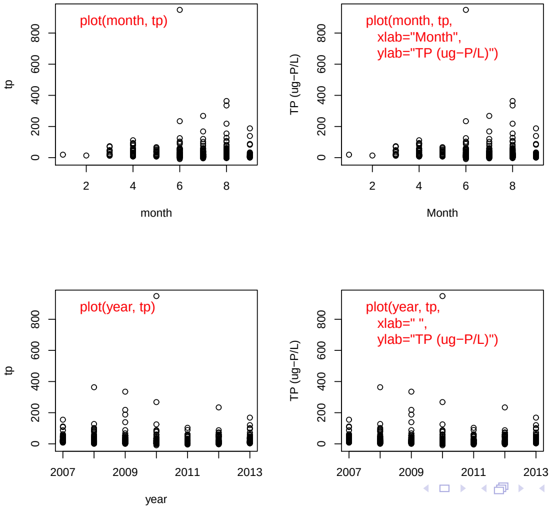
```
par(ask=T)  ### graphics window freezes between plots
            ### hit any key to see next figure
par(ask=F)  ### unfreezes graphics window
```

- A better approach is to save the output using a source file
  ⇒This would be a good time to use source files (`R-minicourse`, Part 1)
- The syntax for saving graphical output varies slightly for different operating systems; use `savePlot` for Windows:

```
plot(tp, chl)
savePlot(filename = "simpleplot", type="emf")
### type="pdf" also produces nice figures
```

Plotting One Variable vs. Time

We usually want an informative x-axis rather than the row number (Index). It is very simple to add month (column 2) and year (column 4)



```
plot(month, tp)
```

```
plot(month, tp,
  xlab="Month",
  ylab="TP (ug−P/L)")
```

```
plot(year, tp)
```

```
plot(year, tp,
  xlab=" ",
  ylab="TP (ug−P/L)")
```

Plotting Time Using the chron Library

- One of the most powerful features of R is that it is open-source and programmable, so individuals can contribute *libraries* or *packages* containing specialized programs

- The chron library is designed to recognize time in a variety of formats, and is easily integrated with other functions like plot()

- ⇒The chron may need to be installed on your computer. Click on "install package" at the top of the R window, select USA (WA1) as the mirror[1], then scroll down until you find chron. It should install automatically

- Before you can use the library you need to tell R to read the library:

```
library(chron) ### this will load chron during your work session
library()      ### this lists all active libraries
```

[1] Mirrors are sites that maintain exact copies of R libraries.

Using the chron Library, continued

- The IWS policy is to keep month, day, year (±time) in separate columns to minimize spreadsheet date conversion errors
- But chron expects the date to be in typical spreadsheet format (month/day/year and hour:min:sec), so we use a function to paste the columns together:

```
mdy.chron <- function(month, day, year) {
  chron(dates.=paste(month, day, year, sep="/"))
}
```

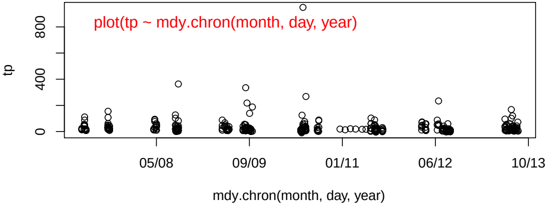
- Now we can use `mdy.chron(month, day, year)` as a variable for the x-axis (see top figure on page 17)

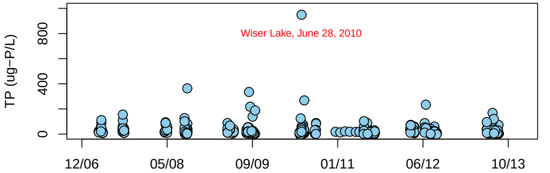
```
plot(tp ~ mdy.chron(month, day, year))
```

Using the chron Library, continued

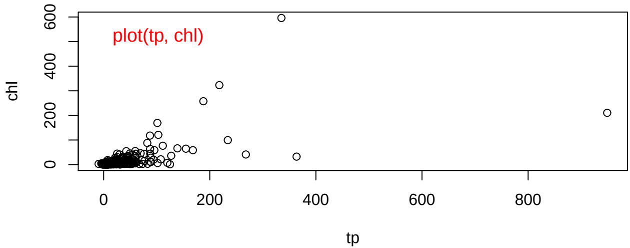
Here is a more advanced example (see bottom figure on page 17)

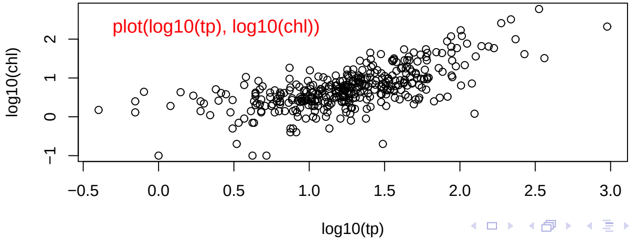
```
### create a "nice" date range for the x-axis
xdates <- c(mdy.chron(1,1,2007), mdy.chron(12, 31, 2013))

### plot the data, with x/y axis labels and annotations
plot(tp ~ mdy.chron(month, day, year),
     xlim=xdates,
     xlab=" ",
     ylim=c(0, 1000),
     ylab="TP (ug-P/L)",
     pch=21, bg="skyblue", cex=1.5)

### add a text line to identify the outlier
text(x=mdy.chron(6,28,2010), y=850, "Wiser Lake, June 28, 2010",
     cex=0.75, col="red")
```


plot(tp ~ mdy.chron(month, day, year))
tp
0
400
800
05/08
09/09
01/11
06/12
10/13
mdy.chron(month, day, year)


Wiser Lake, June 28, 2010
TP (ug−P/L)
0
400
800
12/06
05/08
09/09
01/11
06/12
10/13

Plotting Two Variables Using plot()

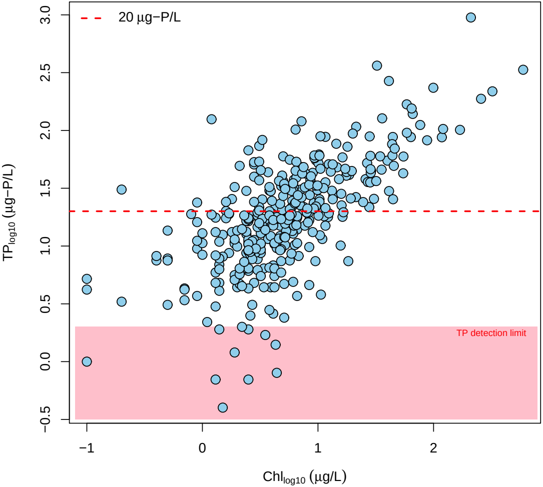




Advanced Scatterplot Features

Legends, Text, Expressions, Polygons

```
### Step #1: create an empty plot (type="n");
###          this will let us place the polygon layer beneath the data
plot(log10(chl), log10(tp), type="n",
     xlab=expression(paste("Chl"[log10] ~ (mu * "g/L"))),
     ylab=expression(paste("TP"[log10] ~ (mu * "g-P/L"))))

### Step #2: draw a shaded, borderless rectangle showing tp detection limit
rect(xleft=-1.1, ybottom=-0.5, xright=2.9, ytop=log10(2), col="pink", border=NA)

### Step #3: add the chlorophyll and total phosphorus data using points
points(log10(chl), log10(tp),
     pch=21, bg="skyblue", cex=1.5)

### Step #4: add a horizontal line at the tp "action" level (20 ug/L)
abline(h=log10(20), lty=2, lwd=2, col="red")

### Step #5: use text and legend to annotate the figure;
###          paste and expression are used to add math symbols and subscripts
text(x=2.5, y=0.25, "TP detection limit", cex=0.7, col="red")
legend(x="topleft", expression("20" * ~ mu * "g-P/L"),
       lty=2, lwd=2, col="red", bty="n")
```


20 µg−P/L
TP detection limit
TP_{log10} (µg−P/L)
Chl_{log10} (µg/L)
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−1
0
1
2

Summary of Scatterplot Syntax

Basic plotting syntax	
`plot(x,y)` or `plot(y~x)`	Plot x (horizontal) and y (vertical) type=points (p), lines (l), both (b), overplot (o)
`points(x,y)`, `lines(x, y)`	Can be used for scatterplots; adds points/lines to existing plot
`xlim`, `ylim`	set x- or y-limits (e.g., xlim=c(0, 100))
`lty`, `lwd`	Sets line type and line width
Syntax for annotating scatterplots	
`cex`	character expansion; subgroups can be specified (e.g., cex.axis)
`col`, `bg`	Sets color; subgroups can be specified (e.g., col.axis); `bg` used for pch 21-25 `col=NA` is transparent
`pch`	Plotting characters (1-16 regular; 21-25 dual color)
`xlab`, `ylab`, `main`	Adds main, x- or y-axis labels; can include spaces, etc
Misc	
`abline`	Add lines to existing plot; modify with `lty`, `lwd` h or v = numerical value (horizontal/vertical lines) a, b = intercept and slope (0,1 for 1:1 diagonal) `lm` object (regression line)
`rect`, `polygon`, `segments`	Add rectangles, polygons, line segments to existing plot
`legend`, `text`	Add legends or text to existing plot

`?par`, `?plot`, `legend`, etc. will bring up help screen

Plotting Examples Using Iris Data

Iris setosa

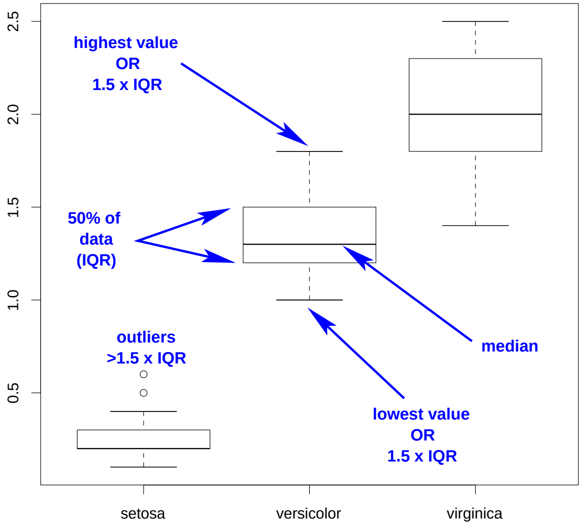
Iris versicolor

Iris virginica

- Boxplots are an excellent exploratory tool for summarizing *categorical* groups of data
- We will use Fisher's Iris data to illustrate scatterplot techniques
- The iris data consist of sepal and petal width and length measurements collected from 150 iris flowers representing three species of iris (n=50 for each species; species=*Iris setosa*, *Iris versicolor*, and *Iris virginica*)
- These data were first published by R.A. Fisher in "The use of multiple measurements in taxonomic problems" (Annals of Eugenics 7:179-188, 1936)
- The iris data are included with the R base library and can be loaded and attached using `data(iris); attach(iris)`

Photographs by C. Hensler and D. Kramb; used with permission

boxplot (Petal.Width ~ Species)



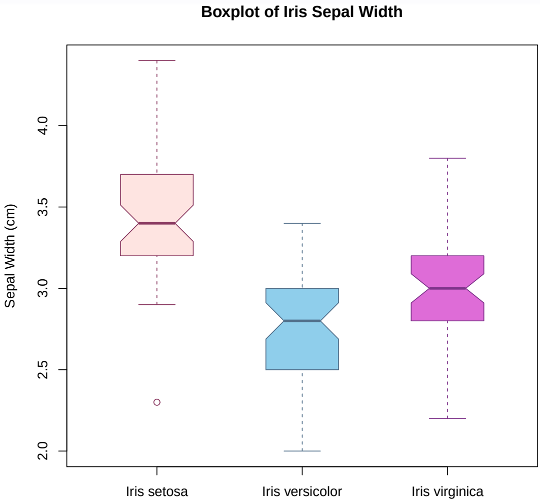
Annotated Boxplots of the Iris Data

- Many of the commands you learned for scatterplots will also work with boxplots
- This code produces notched, annotated boxplots (page 25) that shows intervals of significance:

$$\text{median} \pm 1.58 \times \frac{\text{IQR}}{\sqrt{n}}$$

If the notches overlap, the medians are not significantly different

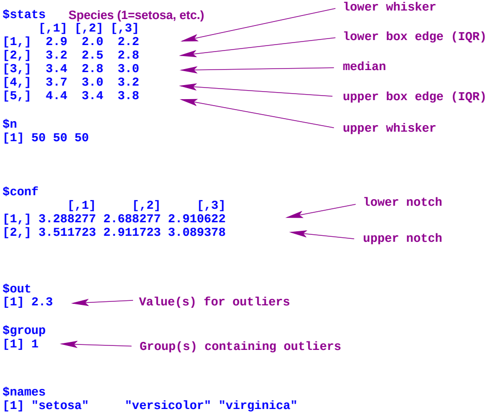
```
boxplot(Sepal.Width ~ Species,
   ylab="Sepal Width (cm)", main="Boxplot of Iris Sepal Width",
   notch = T, boxwex = 0.5,
   col = c("mistyrose", "skyblue", "orchid"),
   border=c("hotpink4", "skyblue4", "mediumorchid4"),
   names = c("Iris setosa", "Iris versicolor", "Iris virginica"))

### want a list of all colors? type colors()
```


Boxplot of Iris Sepal Width
Sepal Width (cm)
2.0
2.5
3.0
3.5
4.0
Iris setosa
Iris versicolor
Iris virginica

Requesting Boxplots Summary Statistics

You can request the plotting statistics, including notch intervals, by adding plot=F in the boxplot syntax

```
boxplot(Sepal.Width ~ Species, notch=T, plot=F)
```



```
$stats
     [,1] [,2] [,3]
[1,]  2.9  2.0  2.2
[2,]  3.2  2.5  2.8
[3,]  3.4  2.8  3.0
[4,]  3.7  3.0  3.2
[5,]  4.4  3.4  3.8

$n
[1] 50 50 50


$conf
          [,1]     [,2]     [,3]
[1,] 3.288277 2.688277 2.910622
[2,] 3.511723 2.911723 3.089378


$out
[1] 2.3

$group
[1] 1


$names
[1] "setosa"     "versicolor" "virginica"
```

Species (1=setosa, etc.)

lower whisker

lower box edge (IQR)

median

upper box edge (IQR)

upper whisker

lower notch

upper notch

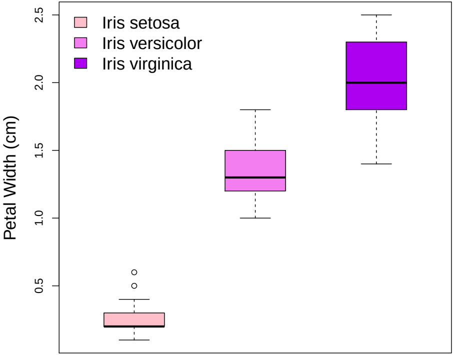
Value(s) for outliers

Group(s) containing outliers

Adding Legends to Simple Boxplots

Here is how to use `legend` to identify categorical groups:

```
legend(x="topleft", c("Iris setosa", "Iris versicolor", "Iris virginica"),
   fill=c("pink", "violet", "purple"), bty="n")
```



Paired Boxplots Using Guinea Pig Data

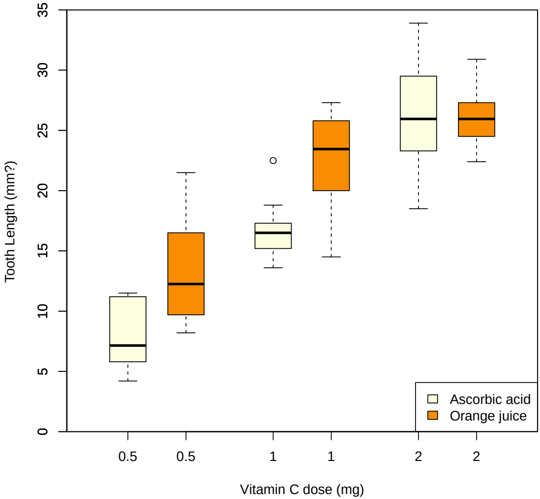
- You can used paired boxplots to plot more than one type of categorical data on the x-axis
- This example uses guinea pig tooth growth data receiving three doses of Vitamin C (0.5, 1, and 2 mg) from either orange juice or ascorbic acid

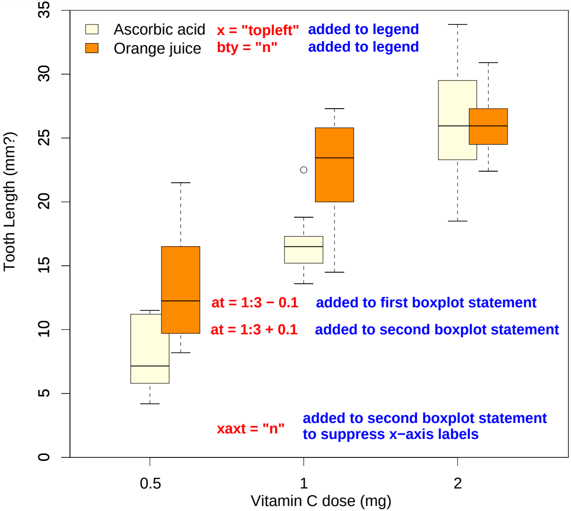
```
data(ToothGrowth) #data included with R base library
attach(ToothGrowth)
summary(ToothGrowth)
```

```
      len        supp         dose      
 Min.   : 4.20   OJ:30   Min.   :0.500  
 1st Qu.:13.07   VC:30   1st Qu.:0.500  
 Median :19.25           Median :1.000  
 Mean   :18.81           Mean   :1.167  
 3rd Qu.:25.27           3rd Qu.:2.000  
 Max.   :33.90           Max.   :2.000  
```

Paired Boxplots - Annotated R Syntax

```
boxplot(len ~ dose,
     boxwex = 0.25, at = 1:3 - 0.2,
     subset = supp == "VC", col = "lightyellow",
     xlab = "Vitamin C dose (mg)",
     ylab = "Tooth Length (mm?)", ylim = c(0, 35), yaxs = "i")
### the first plot sets up the template, including axis and main lagels
### yaxs = "i" helps create better axis intervals
### "at" lists the number of primary categories (1:3) for vitamin C doses
### and adds location (-0.2) to offset each box slightly left of center
### "subset" selects the ascorbic acid group (VC)

boxplot(len ~ dose, add = TRUE,
     boxwex = 0.25, at = 1:3 + 0.2,
     subset = supp == "OJ", col = "darkorange")
### "add=TRUE" will add the second plot to the same figure
### "at" matchs previous but offsets boxes in opposite direction

legend(x="bottomright", c("Ascorbic acid", "Orange juice"),
     fill = c("lightyellow", "darkorange"))
### note that default legend outline was NOT removed
```


Tooth Length (mm?)
0
5
10
15
20
25
30
35
0.5
0.5
1
1
2
2
Vitamin C dose (mg)
□ Ascorbic acid
■ Orange juice



x = "topleft" added to legend

bty = "n" added to legend

at = 1:3 − 0.1 added to first boxplot statement

at = 1:3 + 0.1 added to second boxplot statement

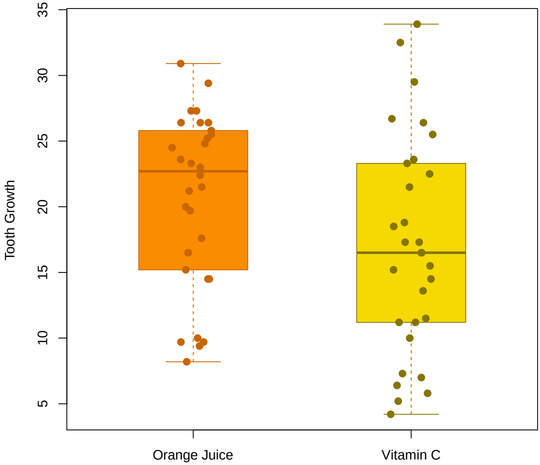
xaxt = "n" added to second boxplot statement to suppress x−axis labels

Advanced Boxplot Features

Adding Raw Data to Boxplots using points() and jitter()

```
with(ToothGrowth,
     {
      boxplot(len˜supp, border=c("darkorange3", "gold4"),
              col=c("darkorange", "gold"), boxwex=0.5,
              ylab="Tooth Growth",
              names=c("Orange Juice", "Vitamin C"))
### "with" defines the data set; the rest is basic boxplot syntax

      points(jitter(rep(1:2, each=30), 0.5),
             unlist(split(len, supp)),
             cex=1.25, pch=16,
             col=c("gold4", "darkorange3")[unclass(ToothGrowth$supp)])
     })
### "points" is similar to previous scatterplot examples
### "jitter" prevents points from plotting on top of each other
###  with 30 points in each group centered on boxes 1-2)
### "unlist(split) puts the tooth length data into "supp" groups
### "unclass" is used to assign correct colors based on "supp" groups
```

Boxplot With Jittered Data Points



Summary of Boxplot Syntax

Syntax	Description
Syntax for changing widths	
boxwex	Set scale for all boxes; values <1 make boxes narrower
staplewex	Set scale width of staple line; proportional to box width
outwex	Set scale width of outlier line; proportional to box width
Syntax for changing colors	
border	Use to add colors to the box borders (see iris examples)
col	Use to add colors to the boxes (see iris examples)
Syntax for adding features	
names	Use to add group labels (see iris examples)
notch	notch=T will add notches
range	Defines range for boxplot whiskers; default=1.5 ×box width; range=0 will extend whiskers to max/min values
Misc	
add	add=TRUE will add boxplot to current plot
at	Box locations; when add=TRUE, at = 1:n (n=number of boxes)
plot	plot=F suppresses plotting, but lists statistics

Also see scatterplot syntax (page 21)

Supplemental References

- Lander, Jared P. 2014. R for Everyone, Advanced Analytics and Graphics. Addison Wesley Data & Analytics Series, ISBN 978-0-321-88803-7.

- Murrell, Paul. 2011. R Graphics, CRC Press. ISBN 978-1-4398-3176-2.

- Teetor, Paul. 2011. The R Cookbook. O'Reilly Publishers. ISBN 978-0-596-880915-7.

- Wickham, Hadley. 2009. ggplot2: Elegant Graphics for Data Analysis (UseR!). Springer. ISBN 978-0-387-98140-6.