Stat 302
Statistical Software and Its Applications
Graphics

Yen-Chi Chen

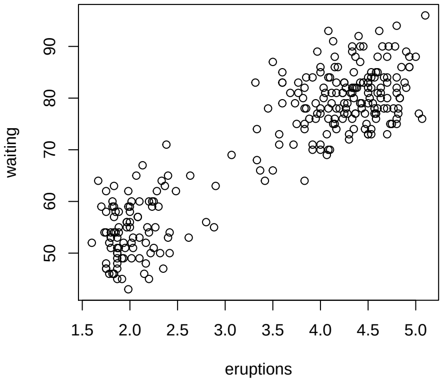
Department of Statistics, University of Washington

Autumn 2016

General Remarks on R Graphics

- A well constructed graph is worth a thousand words.
- Many people use R mainly for obtaining effective graphs.
- You can annotate graphs in many ways.
- You can even use mathematical expressions in annotations.
- There are many generic plot commands.
- Many further commands add graphics elements to plots.
- We will focus on 4 graphs: scatter plot, histogram, QQ plot, and box plot.
- We will not have time to cover all the details so I highly recommend you to do some practices on your own.
- See also: R Graphics by Paul Murrell, Chapman & Hall/CRC.

Scatter Plot: `plot(faithful)`



RStudio saves plots in various formats: ⇒ Plots ⇒ Export

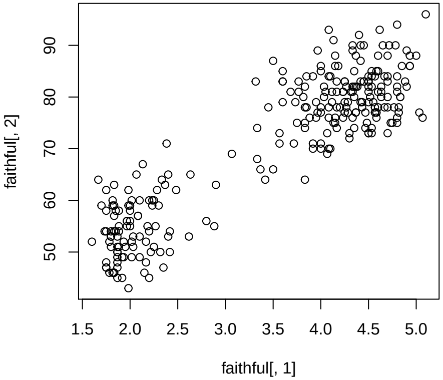
Comments on `plot(faithful)`

- `faithful` is a data frame with 2 columns: `eruptions` and `waiting`
- From the data frame nature of 2 columns the `plot` command knows to plot one column against the other.
- Normal usage is `plot(x,y)` with `x` and `y` numerical vectors of equal length.
- Note the resulting difference in the following commands

```
plot(faithful)

plot(faithful[,1],faithful[,2])
```

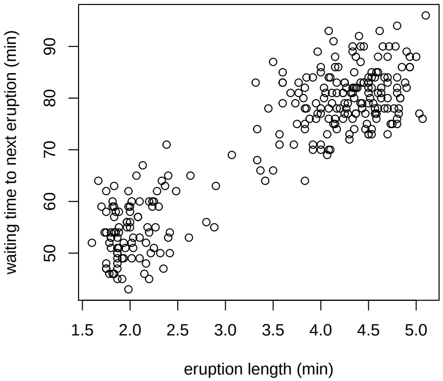
plot(faithful[,1],faithful[,2])

```
plot(faithful[,1],faithful[,2])
```



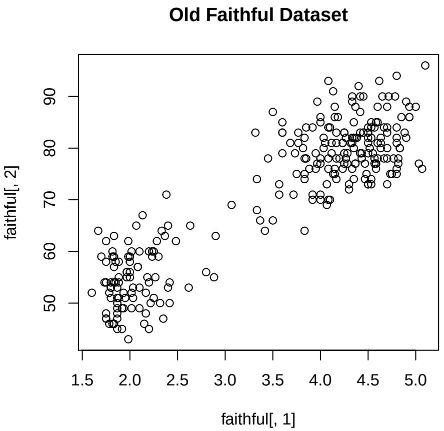
xlab/ylab: labels

```
plot(faithful[,1],faithful[,2],
 xlab="eruption length (min)",
 ylab="waiting time to next eruption (min)")
```



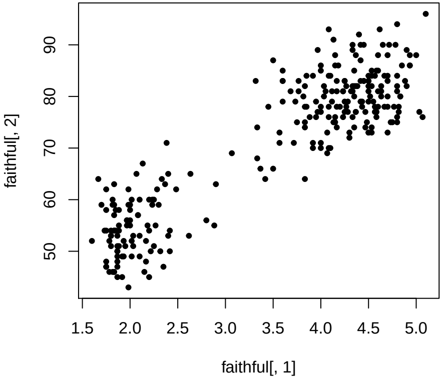
main: title

```
plot(faithful[,1],faithful[,2],
          main= "Old Faithful Dataset")
```



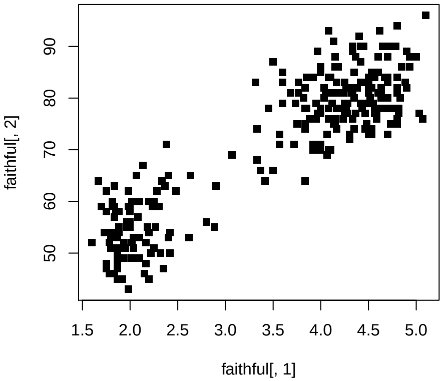
pch: type of points

```
plot(faithful[,1],faithful[,2],
        pch=20)
```



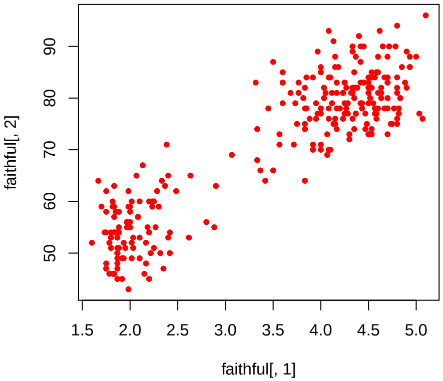
pch: type of points

```
plot(faithful[,1],faithful[,2],
        pch=15)
```



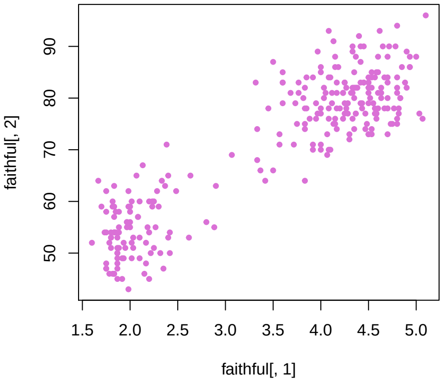
col: color

```
plot(faithful[,1],faithful[,2],
        pch=20, col="red")
```



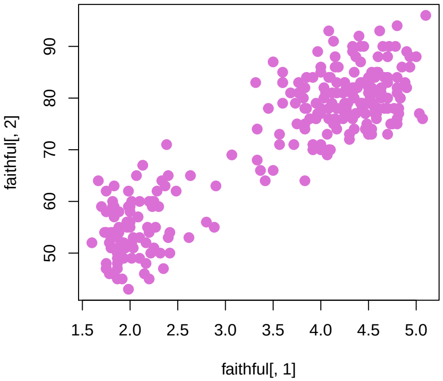
col: color

```
plot(faithful[,1],faithful[,2],
          pch=20, col="orchid")
```



cex: size of points

```
plot(faithful[,1],faithful[,2],
        pch=20, col="orchid", cex=2)
```

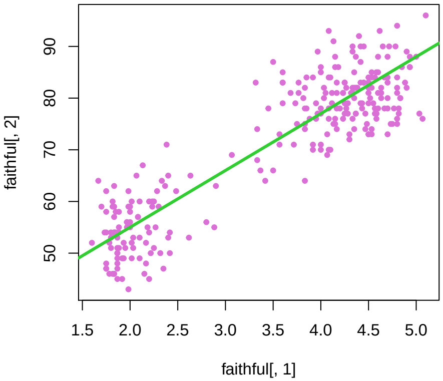


Controlling Plot Options

- Many graphics functions allow fine tuning control as follows.
- Plot dimensions are controlled by `xlim=c(x1,x2)` and `ylim=c(y1,y2)`, using your `x1,x2,y1,y2`.
- Axis labels are controlled by `xlab="your x-label"` and `ylab="your y-label"`.
- Set the main plot title by `main="Your Main Title"`.
- Set the plot sub title by `sub="Your Sub Title"`.
- See `par` for many graphics control options, like
  - `cex, cex.axis, cex.main, cex.sub`
    character expansion factors.
  - `col, col.axis, col.lab, col.main, col.sub`
    specifying colors.
  - `font, font.axis, font.lab, font.main, font.sub`
    font choices, 1 = plain text (the default), 2 = bold face, 3 = italic and 4 = bold italic.
- We do not have time to cover all of them but please try to practice changing each of them.

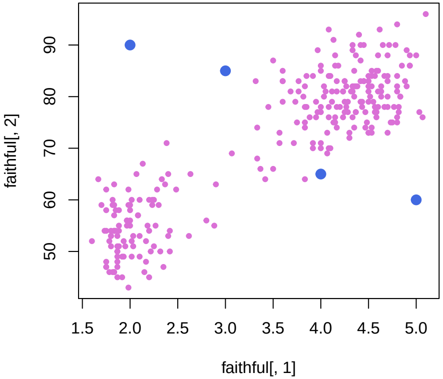
abline(a,b) : adding a line

```
plot(faithful[,1],faithful[,2],
        pch=20, col="orchid")
abline(a=33, b=11, lwd=3, col="limegreen")
```



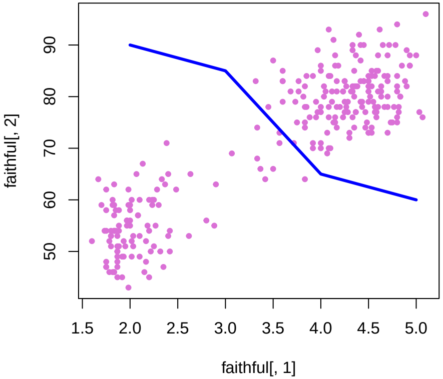
`points()`: adding points

```
plot(faithful[,1],faithful[,2],
        pch=20, col="orchid")
points(x=2:5, y=c(90,80,70,60),
        pch=20, cex=2, col="royalblue")
```



lines(): connecting points by lines

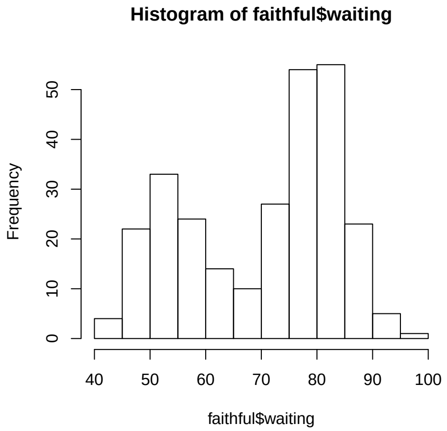
```
plot(faithful[,1],faithful[,2],
        pch=20, col="orchid")
lines(x=2:5, y=c(90,85,65,60),
        lwd=3, col="blue")
```



Augmentation to Plots

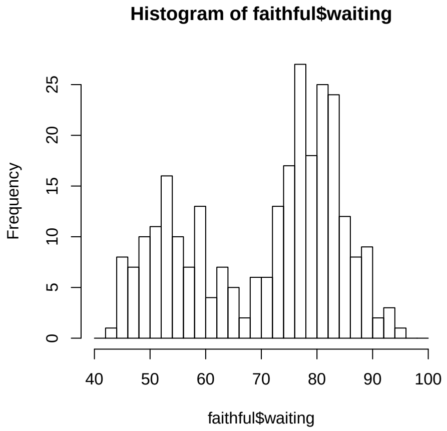
- Some commands only work after a plot has been initiated.
- `abline(a,b)` draws line with intercept `a` and slope `b`.
- `segments(...)` draws line segment(s) from P_1 to P_2.
- `arrows(...)` draws arrow(s) from P_1 to P_2.
- `lines(...)` draws curves through points by line segments.
- `points(...)` plots symbols (`pch`) at specified locations.
- `polygon(...)`, `rect(...)` draw polygons and rectangles.
- `text(...)` puts specified text at selected positions.
- `legend(...)` adds legends to plots.
- `mtext(...)` adds text to plot margins.
- and lots more ⇒ `help.start()` ⇒ An Introduction to R ⇒ 12 Graphical procedures ⇒ 12.2 Low-level plotting commands
- Please try to practice them on your own.

Histogram: `hist(faithful$waiting)`



`breaks`: break point for histogram

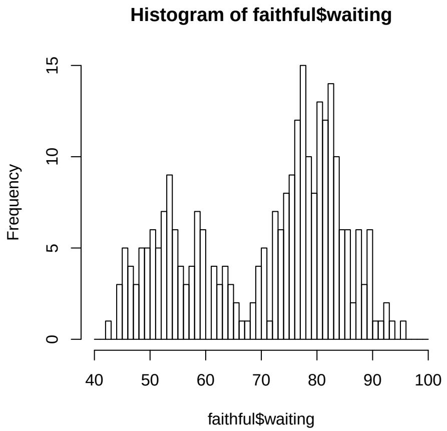
```
hist(faithful$waiting,
     breaks= seq(from=40,to=100, by=2))
```



$\longrightarrow$ the `by` in the `seq` now gives the bin width of the histogram.

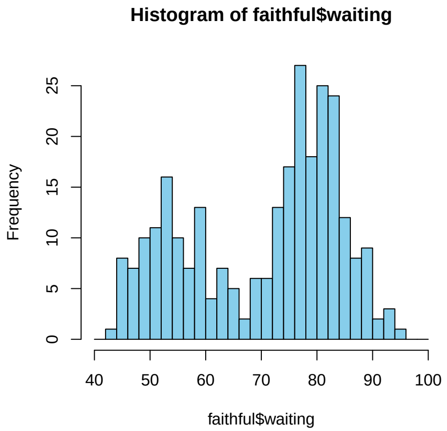
`breaks`: break point for histogram

```
hist(faithful$waiting,
     breaks= seq(from=40,to=100, by=1))
```



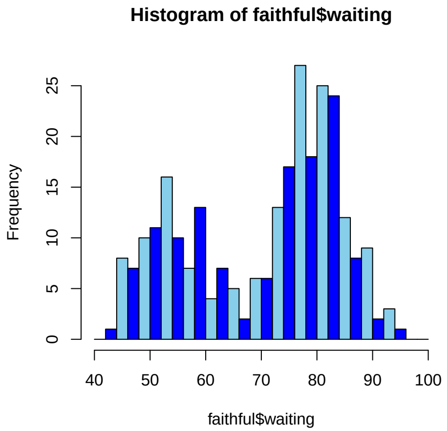
col: color of the histogram

```
hist(faithful$waiting, col="skyblue",
     breaks= seq(from=40,to=100, by=2))
```



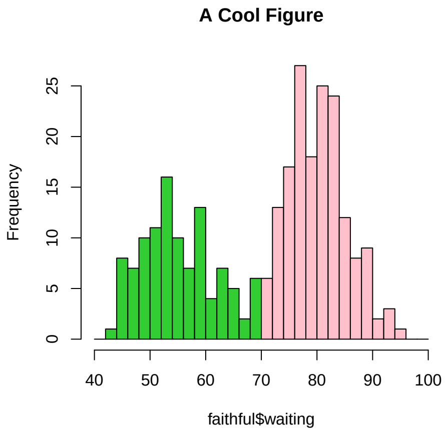
`col`: color of the histogram

```
hist(faithful$waiting, col=c("skyblue","blue"),
     breaks= seq(from=40,to=100, by=2))
```



A cool figure: think about what happened

```
hist_break <-seq(from=40,to=100, by=2)
col_break <- rep("pink",length(hist_break))
col_break[which(hist_break<70)] <- "limegreen"
hist(faithful$waiting, col= col_break,
     breaks= hist_break, main="A Cool Figure")
```

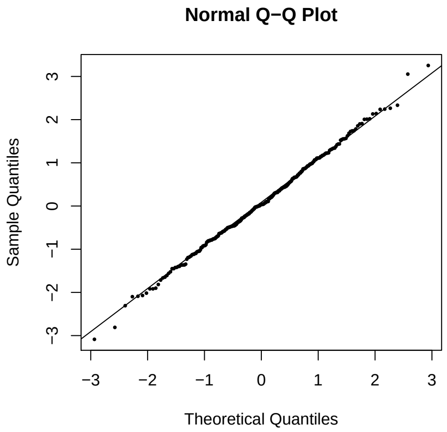


Normal QQ-Plot – 1

```
x <- rnorm(300)
# x is a standard normal random sample, n=300
qqnorm(x,pch=16,cex=.5)
# makes QQ-plot of sample
qqline(x)
# adds a fitted line to the previous plot.
# line is fitted through 1st and 3rd quartiles
```

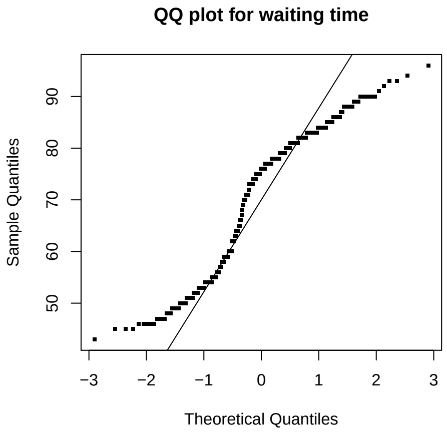
Normal QQ-Plot – 2

```
x <- rnorm(300)
qqnorm(x,pch=16,cex=.5)
qqline(x)
```



Normal QQ-Plot: `waiting` in the old faithful dataset

```
qqnorm(faithful$waiting, pch=15, cex=.5,
       main="QQ plot for waiting time")
qqline(faithful$waiting)
```

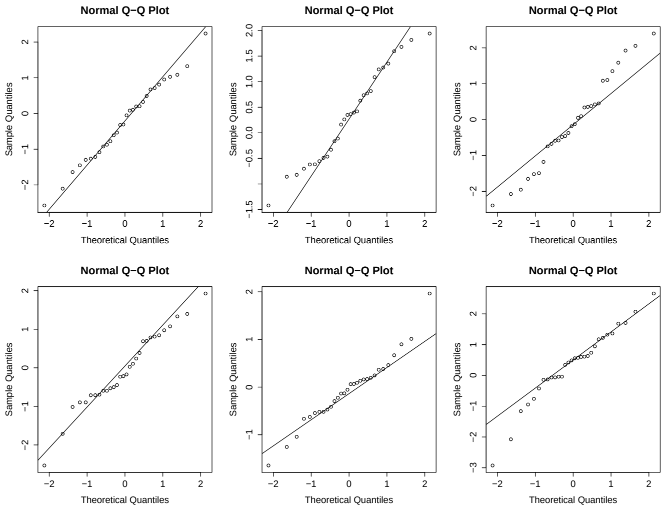


Normal QQ-Plot $n = 30$

```
par(mfrow=c(2,3))
x <- rnorm(30);qqnorm(x);qqline(x)
x <- rnorm(30);qqnorm(x);qqline(x)
x <- rnorm(30);qqnorm(x);qqline(x)
x <- rnorm(30);qqnorm(x);qqline(x)
x <- rnorm(30);qqnorm(x);qqline(x)
x <- rnorm(30);qqnorm(x);qqline(x)
```

- The `par` function controls many plotting parameters. $\Longrightarrow$ `?par`.
- Some plotting parameters work within the plotting function, others only within a prior `par(...)` call.
- The `;` separation allows several commands on one line.

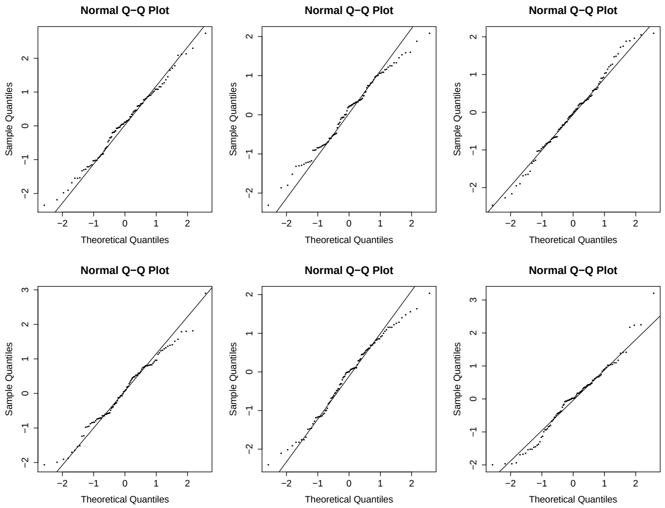
Judging Normality Takes Lots of Pratice



Normal QQ-Plot $n = 100$

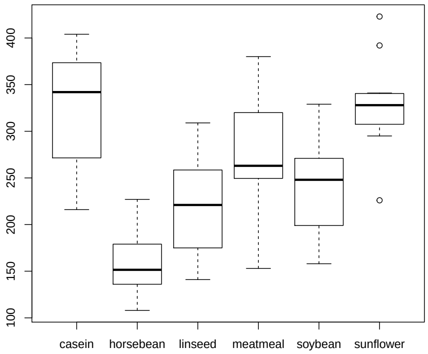
```
par(mfrow=c(2,3))
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
x <- rnorm(100);qqnorm(x,pch=16,cex=.5);qqline(x)
```

Increasing *n* to 100 Helps



Box Plots

```
boxplot(weight~feed,data=chickwts)
   # boxplot for variable weight, split
   # by the type of feed (factor)
```

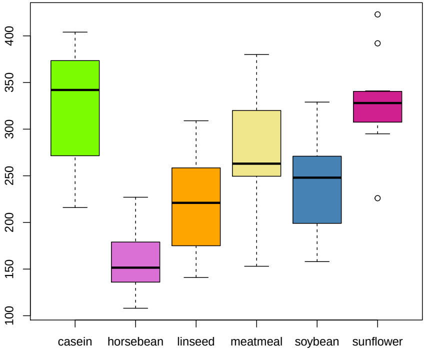


Comments on Box Plot

- The horizontal box lines $\equiv$ 3 quartiles $Q(.25), Q(.5), Q(.75)$.
- The dashed vertical lines extend to the adjacent values.
  - Compute the interquartile range $IQR = Q(.75) - Q(.25)$.
  - The upper adjacent value is the largest observation $\leq Q(.75) + 1.5 \times IQR$
  - The lower adjacent value is the smallest observation $\geq Q(.25) - 1.5 \times IQR$
- Points beyond adjacent values shown individually (outliers?)
- For $\mathcal{N}(\mu, \sigma^2) \approx .35\%$ are beyond each adjacent value.
- `data=chickwts` $\Rightarrow$ simpler reference to variables.
- `weight ~ feed` implies boxplots for the factor of `feed`.

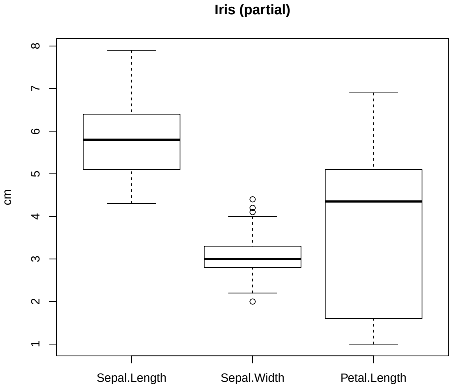
Box Plots: col

```
col_tmp <- c("lawngreen","orchid","orange",
             "khaki", "steelblue","violetred")
boxplot(weight~feed,data=chickwts,
        col = col_tmp)
```



Box Plots: many inputs

```
boxplot(iris$Sepal.Length,iris$Sepal.Width,
        iris$Petal.Length, names=c("Sepal.Length",
        "Sepal.Width", "Petal.Length"),
        main="Iris (partial)", ylab="cm")
```



⟶ try to change each argument a bit to understand their functions.

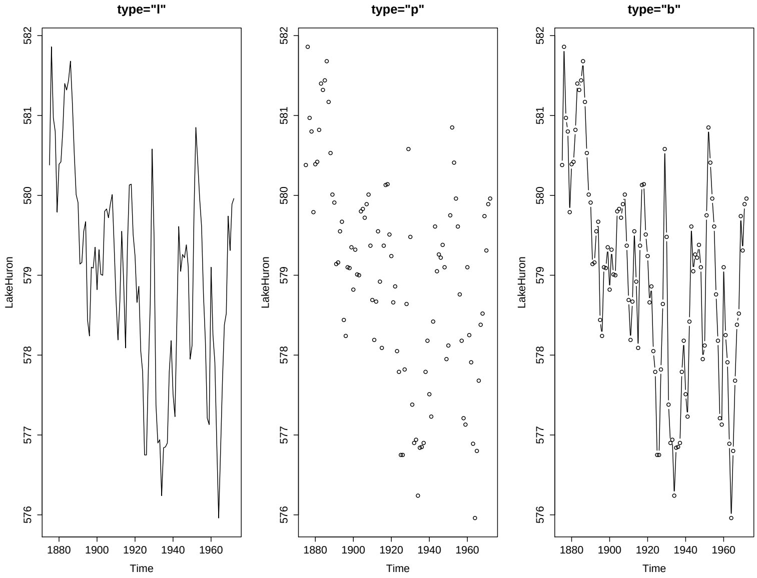
Back to `plot()`

- The `plot()` function has some very power features.
- Here I will show you two features.

Lake Huron Water Level: illustrating plot argument `type`

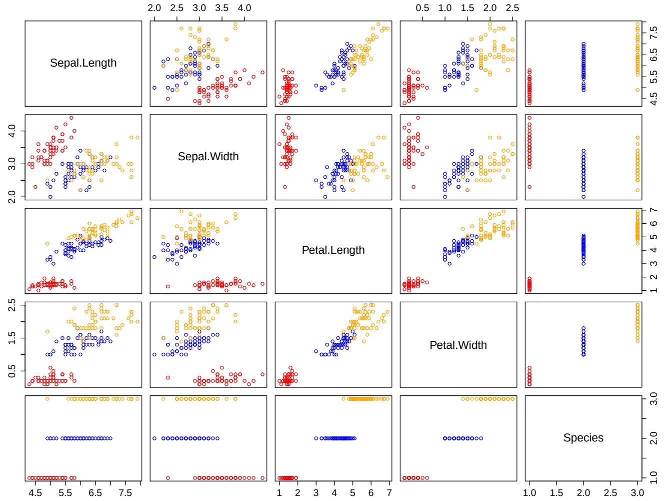
```
par(mfrow=c(1,3))

plot(LakeHuron,type="l",main='type="l"')
# points connected by lines

plot(LakeHuron,type="p",main='type="p"')
# only points are plotted

plot(LakeHuron,type="b",main='type="b"')
# both points and lines are plotted

# see ?plot for more on the type argument
```

Lake Huron Plots: a time series dataset



Visualizing a multivariate data

```
plot(iris,col=
 rep(c("red","blue","orange"),each=50))
```



Saving Plots

- We indicated the interactive way within the RStudio interface.
- There are also various other ways by direct commands.
- `pdf(file="myplot.pdf", width=8,height=6)` opens pdf-file "myplot.pdf". `width, height` are in inches.
- Any subsequent graphics commands produce output to that file, until `dev.off()` is issued, or the R session terminates.
- Similar commands exist for other graphics formats $\Rightarrow$ `?Devices` for `tiff`, `jpeg`, `bmp`, `png`, `postscript`, `quartz` (Mac).

More Powerful Graphics

- Add-on packages provide more graphics capabilities.
  We mention just two.
- These are too complex to delve into here. Good as projects.
- The `lattice` package.
- $\Rightarrow$ Book: *Lattice: Multivariate Data Visualization with R*, Springer 2008, by Deepayan Sarkar, creator of the package.
- The `ggplot2` package, not covered here, but see
  R Graphics Cookbook by Winston Chang, O'Reilly, 2013.
- *Interactive and Dynamic Graphics for Data Analysis with R and GGobi*, Springer 2007, by Dianne Cook and Deborah Swayne.

In-class Exercises

- Try the following:

```
col_tmp <- rep("limegreen",nrow(faithful))
col_tmp[which(faithful$eruptions<3)]<- "orchid"
plot(faithful, pch=16, col=col_tmp)
abline(v=3, lwd=3, col="brown")
```

- Also try the following:

```
hist(faithful$waiting,
     breaks= seq(from=40,to=100, by=2),
     col=1:8)
```

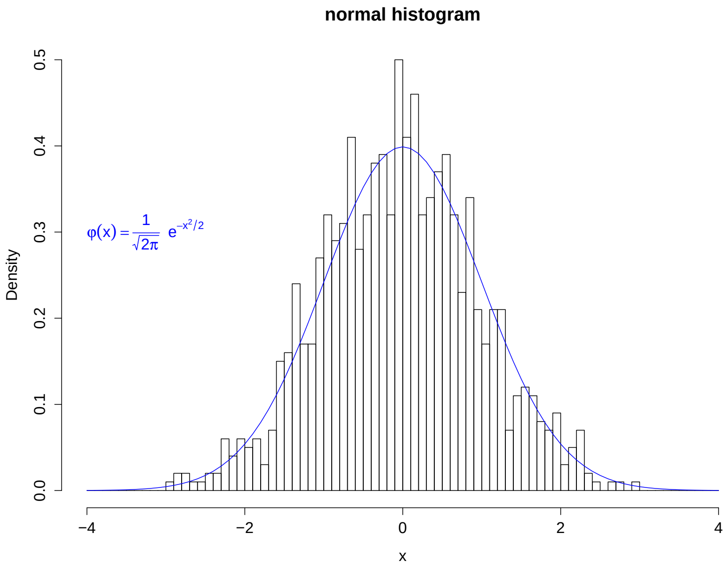
- Think about what happened? what do each line/argument do? you may change them a bit to understand these commands.
- You can learn more in the following link: https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Graphics

Appendix: Math Annotations

- $\Rightarrow$ `?plotmath` gives documentation on it.
- `> demo(plotmath)` gives examples by commands and results.
- Murrell, P. and Ihaka, R. (2000)
  "An approach to providing mathematical annotation in plots."
  Journal of Computational and Graphical Statistics, 9, 582-599.

Appendix: Normal Sample Histogram and Density

```
normalhist <- function(n=1000){
    x <- rnorm(n)
    xx <- seq(-4,4,.1)
    hist(x,breaks=xx,probability=T,
         main="normal histogram")
    yy <- dnorm(xx)
    lines(xx,yy,col="blue")
    text(-4,.3,expression(varphi(x)==
        over(1,sqrt(2*pi))*phantom(0)*
        e^{-x^2/2}),adj=0,col="blue")
}
```



$$\varphi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}$$