Making plots in R [things I wish someone told me when I started grad school]

Kirk Lohmueller
Department of Ecology and Evolutionary Biology
UCLA
September 17, 2015

In honor of “Talk Like a Pirate Day...”

What’s a pirates favorite computer language?

In honor of “Talk Like a Pirate Day...”

Rrrrrr!

In honor of “Talk Like a Pirate Day...”

But, why?

In honor of “Talk Like a Pirate Day...”

Because they get lost when they go to C

Learning objectives

Learning objectives

- Know how to make simple plots:

Learning objectives

- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot

Learning objectives

- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot
- Read “large” datasets into R

Learning objectives

- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot
- Read “large” datasets into R
- Perform manipulations of large datasets

Learning objectives

- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot
- Read “large” datasets into R
- Perform manipulations of large datasets
- Calculate some statistics in R

Learning objectives

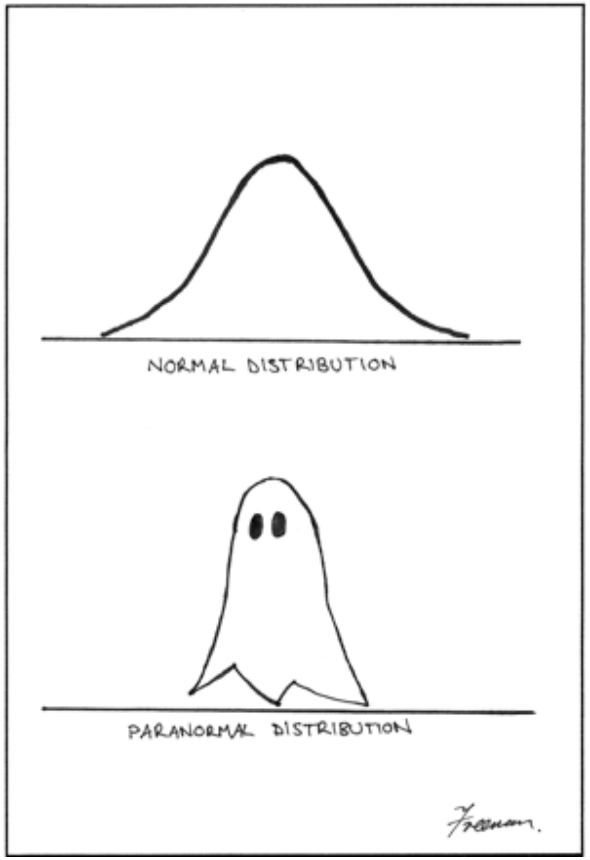
- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot
- Read “large” datasets into R
- Perform manipulations of large datasets
- Calculate some statistics in R
- Start to become familiar with simulation

Learning objectives

- Know how to make simple plots:
  - Scatterplot, histogram, density plot, bar plot
- Read “large” datasets into R
- Perform manipulations of large datasets
- Calculate some statistics in R
- Start to become familiar with simulation
- Begin to appreciate and enjoy Kirk’s corny humor

R can simulate data from a probability distribution

Let's look at the normal distribution:



Matthew Freeman *J Epidemiol Community Health* 2006;60:6

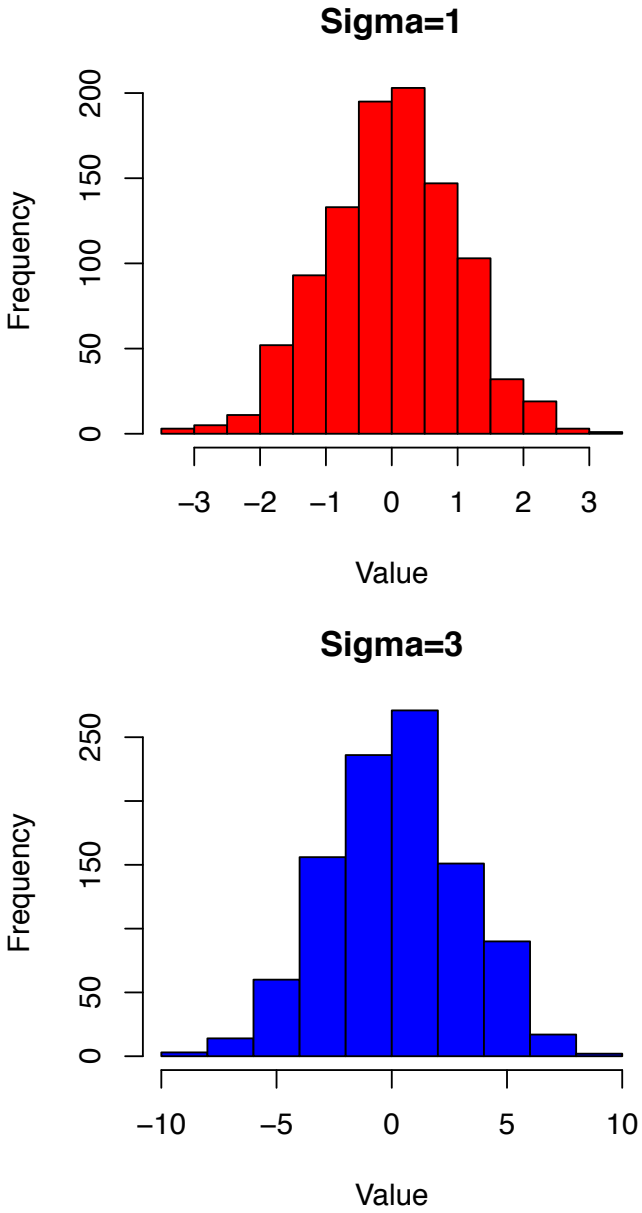
Simulating data from a normal distribution

```
> #first, draw 1000 random values from a standard normal distribution (SD=1):
> s1<-rnorm(1000,mean=0,sd=1)
>
> #now do 1000 drawn from a normal distribution with SD=3.
> s3<-rnorm(1000,mean=0,sd=3)
>
> head(s1)
[1]  0.26951848 -2.43530911  1.15968499  0.09647798 -0.74425935  0.40504897
> head(s3)
[1]  3.6718664  4.8193934 -0.6078601  2.1520862  2.9089759 -3.6002362
```

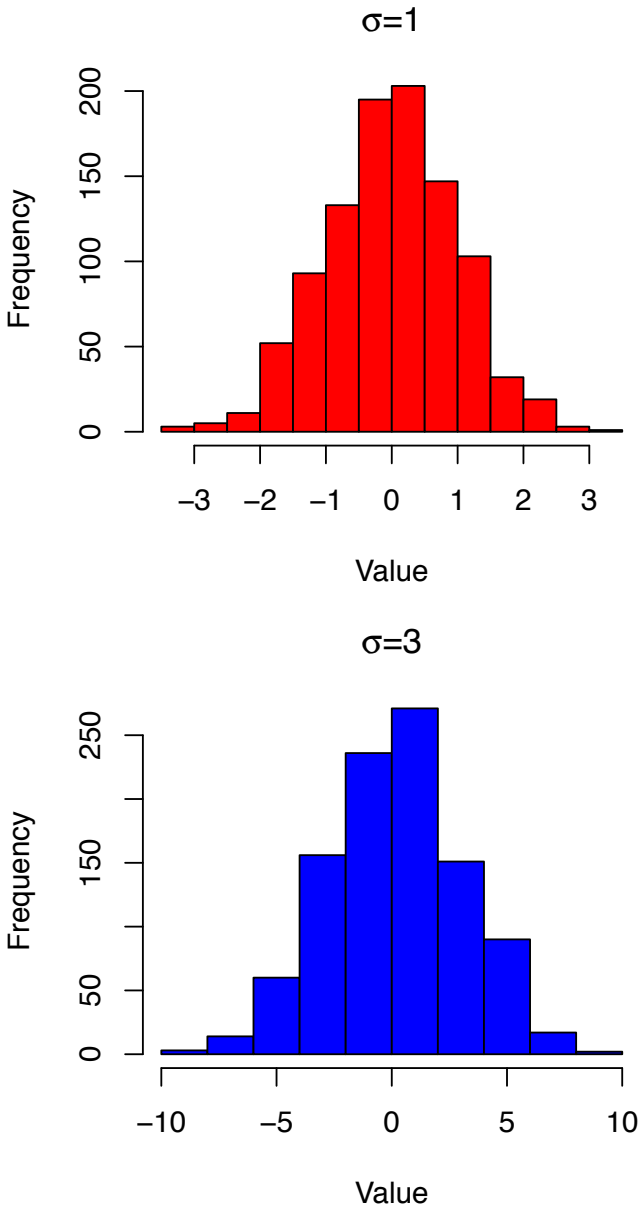
Basic histogram

```
> #plot histograms of both on same panel and save to a file:
> pdf(file="Normal_hist.pdf", width=4,height=7);
> #open the file
>
> par(mfrow=c(2,1), mar=c(4, 4, 3, 2)) #sets plotting area and margins
>
> hist(s1,col=2,xlab="Value",main="Sigma=1") #make first hist
>
> hist(s3,col=4,xlab="Value",main="Sigma=3") #make second hist
>
> dev.off() #shuts off current output device
quartz
     2
```

Basic histogram



Getting fancier...



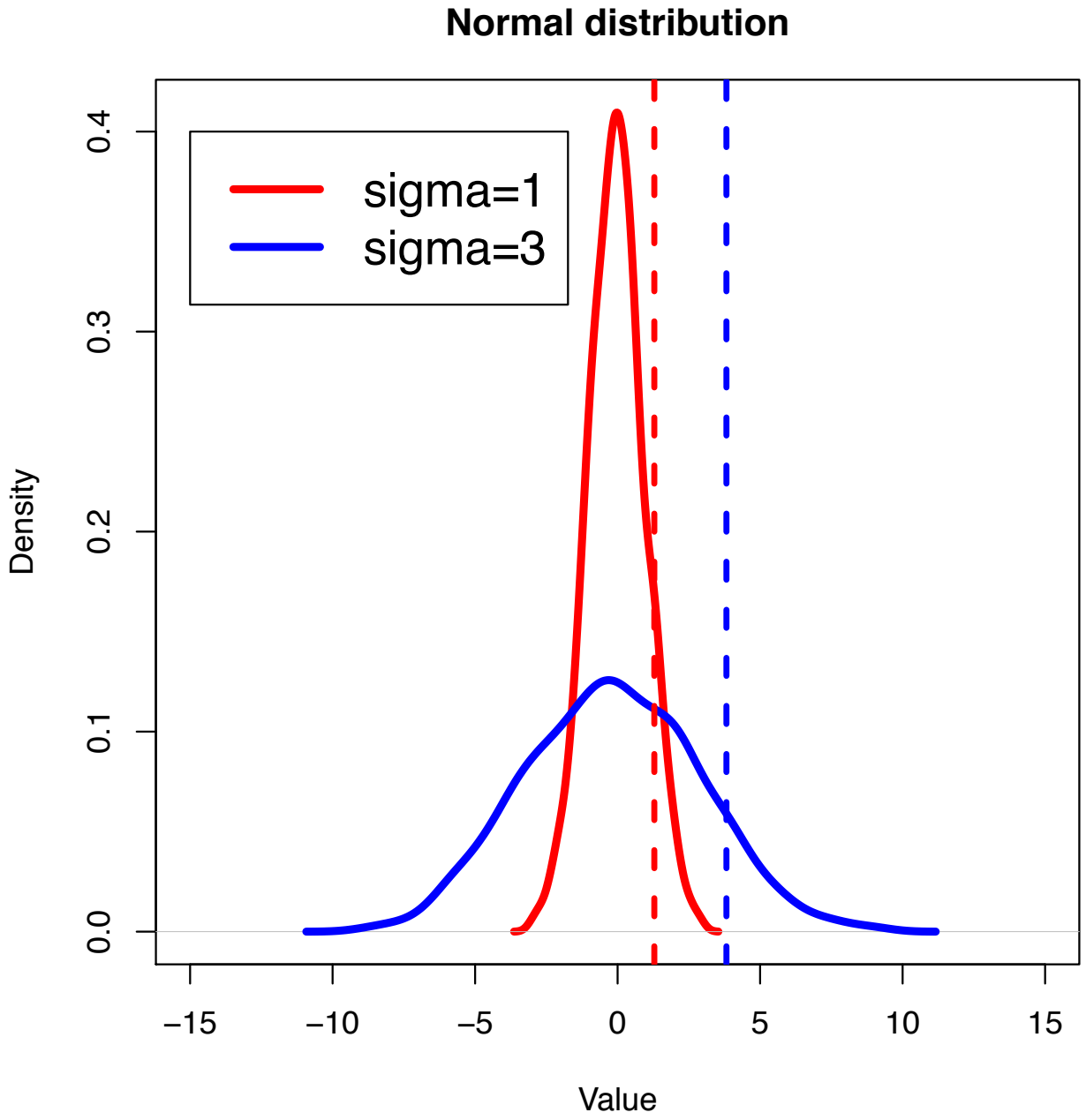
How did I do that?

```
>> #plot histograms of both on same panel and save to a file:
> pdf(file="Normal_hist.fancy.pdf", width=4,height=7);
> #open the file
>
> par(mfrow=c(2,1), mar=c(4, 4, 3, 2)) #sets plotting area and
margins
>
> hist(s1,col=2,xlab="Value",main=expression(paste(sigma,"=1")))
#make first hist
>
> hist(s3,col=4,xlab="Value",main=expression(paste(sigma,"=3")))
#make second hist
>
> dev.off() #shuts off current output device
pdf
  2
```

Smooth density plot

```
> #make smooth density plot:
>
> pdf(file="Normal_density.pdf", width=6,height=6); #open the file
>
> par(mfrow=c(1,1), mar=c(4, 4, 3, 2)) #sets plotting area and margins
>
> plot(density(s1),col=2,lwd=4,xlab="Value",xlim=c(-15,15),main="Normal
distribution")
>
> lines(density(s3),col=4,lwd=4) #add the SD=3 values
>
> legend(-15,0.4,c("sigma=1","sigma=3"),lwd=4,col=c(2,4),cex=1.5) #put a legend on
>
> #we can highlight the upper 10% of each distribution with a vertical line:
> abline(v=quantile(s1,0.9),lty=2,lwd=3,col=2) #puts a vertical line onto the plot
for s1
> abline(v=quantile(s3,0.9),lty=2,lwd=3,col=4) #puts a vertical line onto the plot
for s3
> dev.off()
quartz
     2
```

Smooth density plot



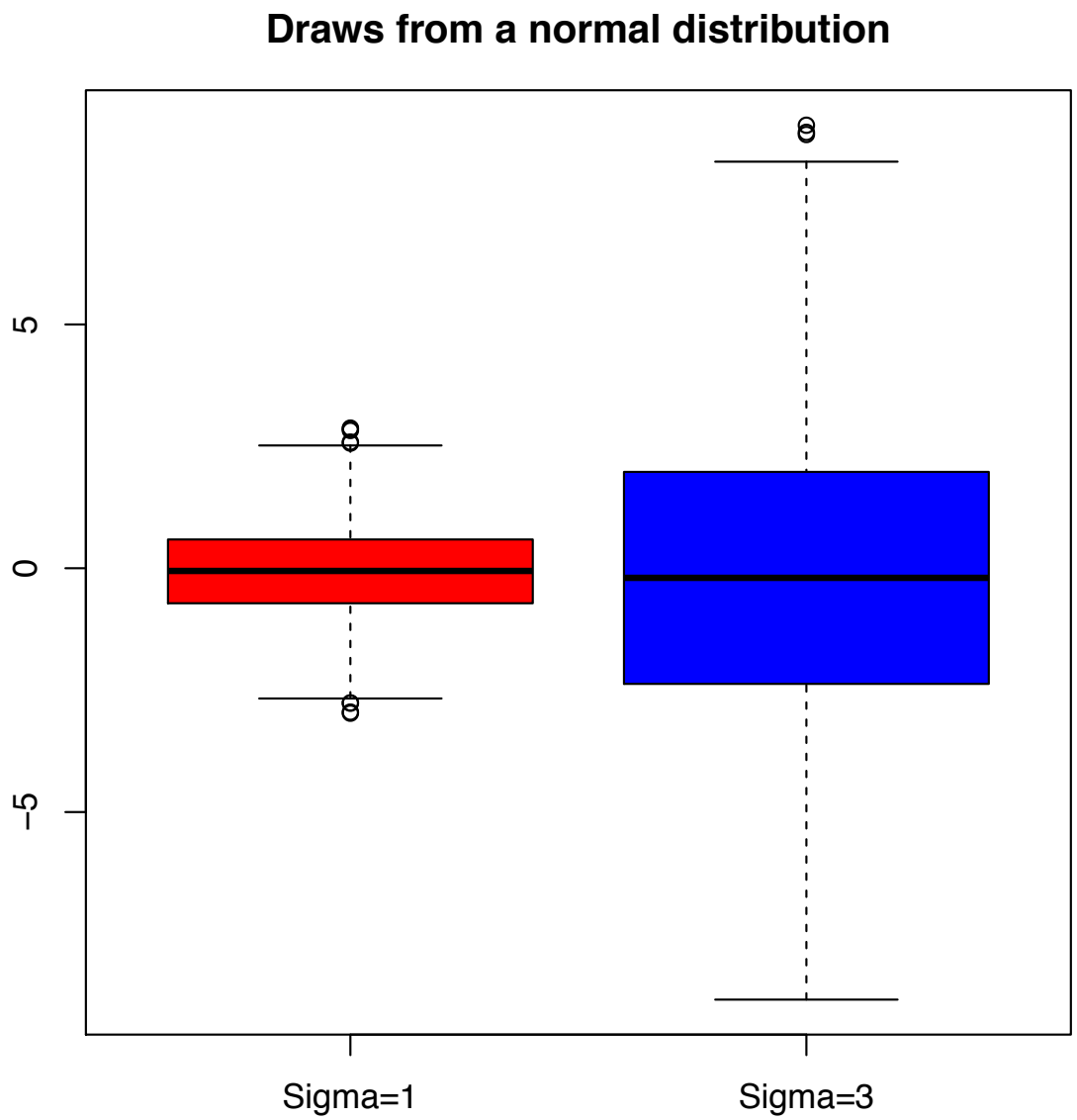
More on “quantile”

```
> #quantile take a vector of stuff, and returns the value q such that p%
of your distribution is less than q.
>
> #for example, find the 75th percentile of the standard normal
distribution:
> quantile(s1,0.75)
      75%
0.5899364
>
> #quantile with just a vector gives some interesting stuff:
> quantile(s1)
         0%         25%         50%         75%        100%
-2.97189479 -0.71435745 -0.05515638  0.58993639  2.87407876
```

Boxplot

```
> #boxplot:
> pdf(file="Normal_boxplot.pdf", width=6,height=6); #open the file
>
> par(mfrow=c(1,1), mar=c(4, 4, 3, 2)) #sets plotting area and margins
>
> boxplot(cbind(s1,s3),names=c("Sigma=1","Sigma=3"),main="Draws from a
normal distribution",col=c(2,4))
>
> dev.off()
quartz
     2
```

Boxplot



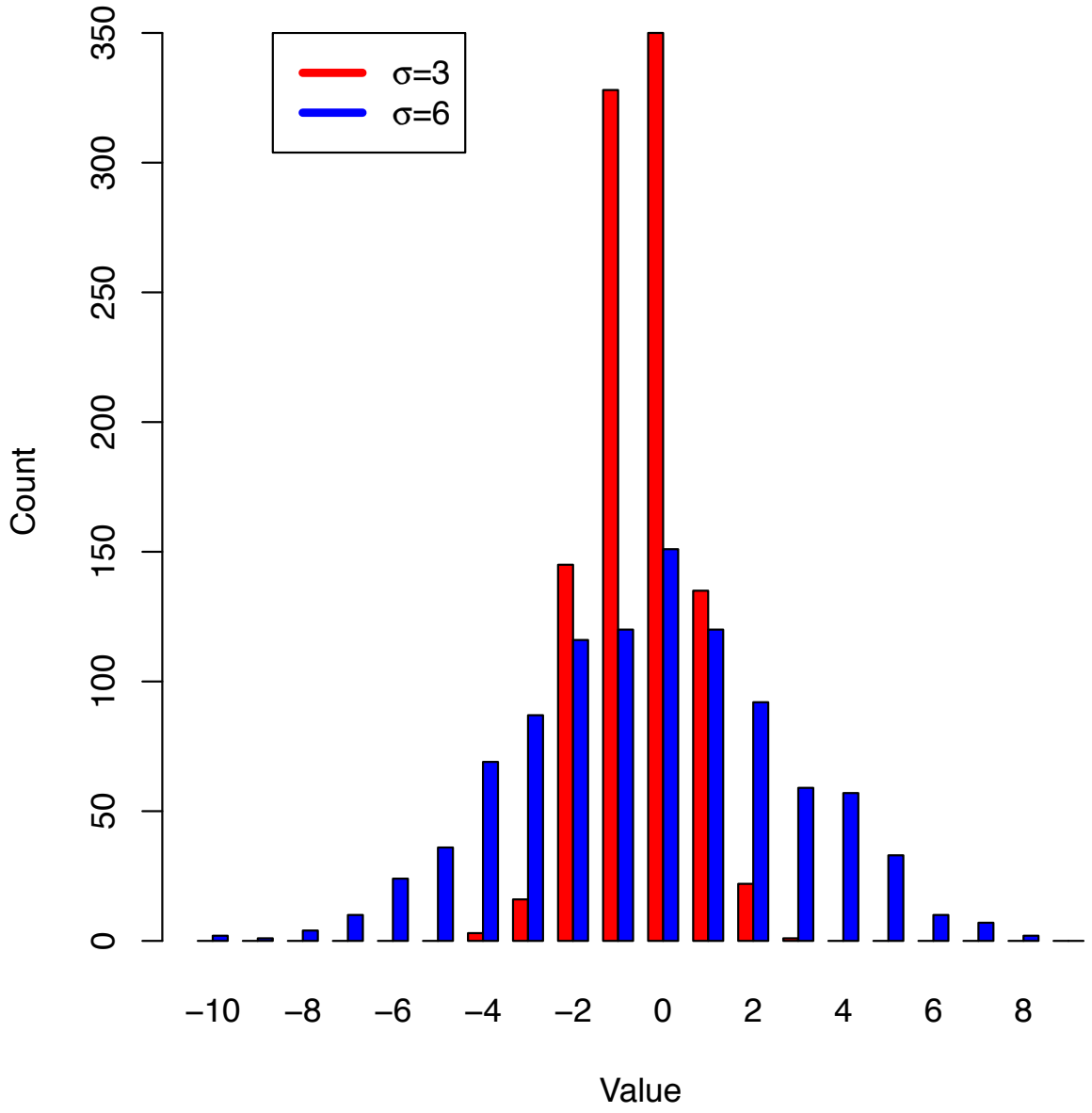
Histogram with both sets of data on same axes? Can we do it? YES WE CAN!

```
> #Let's make a histogram of these values, but putting both
on the same axes.
> #But, we need to have the same bin widths for both
datasets:
>
> bins<-seq(-10,10,by=1)
> hist(s1,breaks=bins)$breaks
 [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2
3   4   5   6   7   8   9  10
>
> hist(s3,breaks=bins)$breaks
 [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2
3   4   5   6   7   8   9  10
>
> #This looks good
>
> counts_s1<-hist(s1,breaks=bins)$counts
> counts_s3<-hist(s3,breaks=bins)$counts
```

Histogram with both sets of data on same axes? Can we do it? YES WE CAN!

```
> #now make the plot:
> pdf(file="normal_barplot.pdf", width=6,height=6); #open the
file
>
> par(mfrow=c(1,1), mar=c(4, 4, 3, 2)) #sets plotting area and
margins
>
>
barplot(rbind(counts_s1,counts_s3),col=c(2,4),beside=T,names.arg=
seq(-10,9.5,by=1),xlab="Value",ylab="Count")
>
>
legend(6,350,c(expression(paste(sigma,"=3")),expression(paste(sig
ma,"=6"))),col=c(2,4),lwd=4)
>
> dev.off()
pdf
  2
```

Histogram with both sets of data on same axes? Can we do it? YES WE CAN!



Finding extreme values

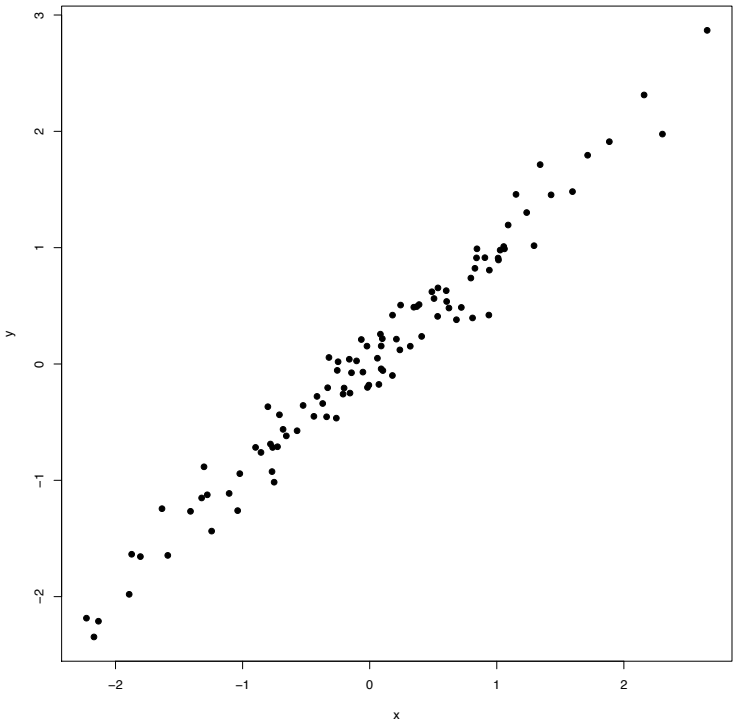
Say we want to find the % of values in a vector that are >X...

```
> > #We can find the % of values in s1 that are >3:
> mean(s1>3)
[1] 0.001
> #Only 1 of the 1000 values in s1 is >3
>
>
> mean(s3>3)
[1] 0.168
> #16.8% of values in s3 are >3
```

Scatterplot pitfalls

```
> #Simple scatterplot:
> pdf(file="/Users/kirk/Dropbox/Kirk_stuff/KEL_bootcamp/scatter_small.pdf",
width=10,height=10); #open the file
>
> par(mfrow=c(1,1), mar=c(4, 4, 3, 2)) #sets plotting area and margins
>
> x<-rnorm(100)
> y<-x+rnorm(100,sd=0.2)
>
> plot(x,y,pch=19)
>
> dev.off()
quartz
     2
```

The most annoying thing in R...

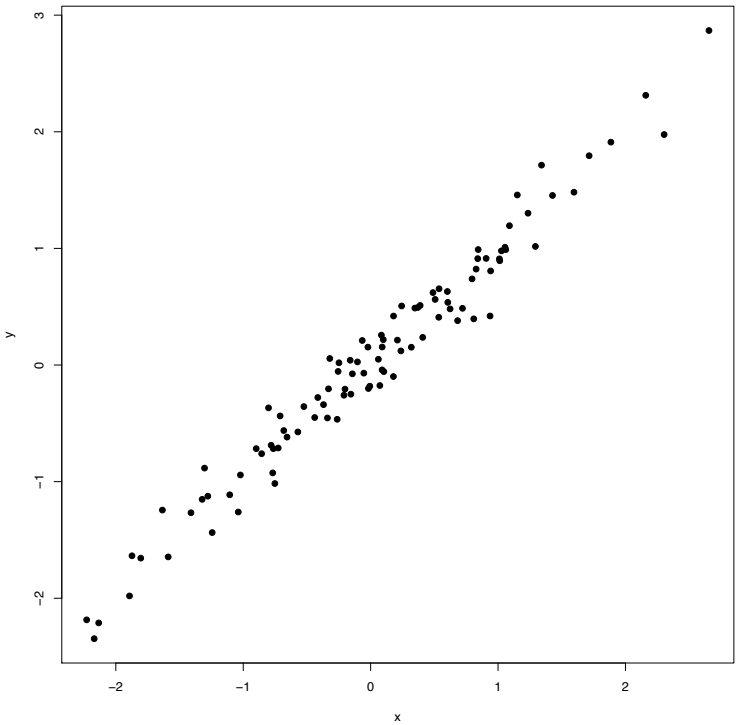


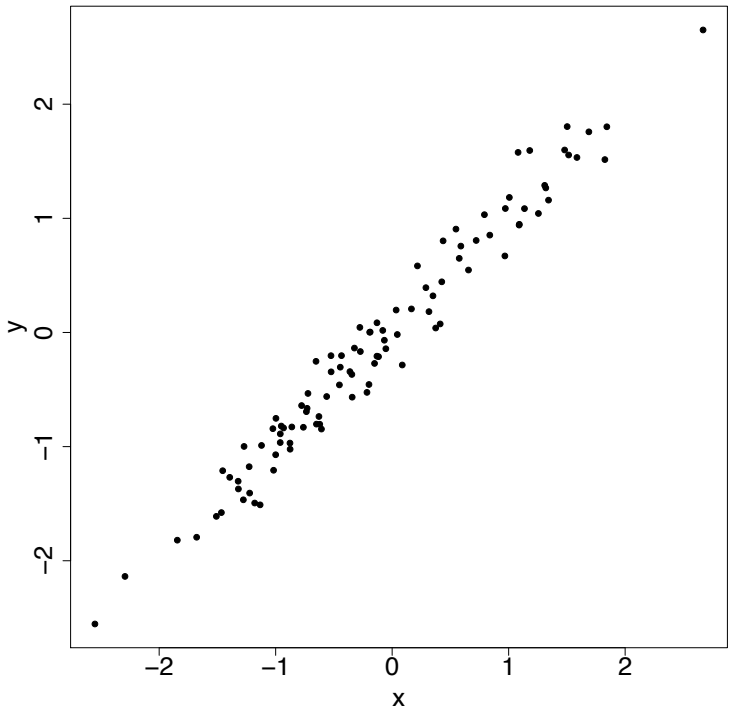
Huh?
What is plotted here?
My tired eyes can’t read this....

One way to fix it

```
> #now, try again, make the labels bigger:
> #Simple scatterplot:
> pdf(file="/Users/kirk/Dropbox/Kirk_stuff/KEL_bootcamp/scatter_large.pdf",
width=10,height=10); #open the file
>
> par(mfrow=c(1,1), mar=c(5, 5, 3, 2)) #sets plotting area and margins
>
> x<-rnorm(100)
> y<-x+rnorm(100,sd=0.2)
>
> plot(x,y,pch=19,cex.lab=2,cex.axis=2)
>
> dev.off()
quartz
     2
```

One way to fix it





Using R

A basic data type in R is a vector. As an example:

Declaring a vector and manipulating it

```
>x<-c(1,5,10,45)    # declare a vector
>x           # display the vector
>mean(x)
>1/x
>sum(x)
>sd(x)
>length(x)
>range(x)
```

Other nice ways to declare vectors

```
>x=seq(0,20)
>x=seq(0,20,by=2)
>x=rep(5,10)
>x=rep(c(1,2,3),10)
>x=sample(20)
```

Using R

Accessing vector elements

```
x=seq(50,60)
x[2]
x[2:5]
x[c(2,4,8)]
x[-2]
x[-c(2,4,8)]
```

Declaring and accessing matrices

```
x=seq(1,10)
y=matrix(x,nrow=2,ncol=5)
y[3,5]
y[,3]
y[2,]
t(y)    # Matrix transpose
y%*%t(y)   #  %*% is the matrix multiplication operator
```

Using R

Data frames

Different vectors can be joined together into one object with the constraint that there is the same number of elements per vector.

```
z=factor(c("Ctrl","Ctrl","A","A","B","B"))
x=c(5,3,4,NA,10,4)
y=c(TRUE,TRUE,FALSE, TRUE,FALSE,TRUE)
d=data.frame(labels=z,heights=x,outcome=y)
```

Missing data

The NA construct can be used to specify missing data:

```
x=c(5,3,4,NA)
is.na(x)
mean(x)
mean(x, na.rm=TRUE)
```

Using R

The apply functions

There are three main functions: apply,sapply, tapply

```
# apply:  Apply function to all rows/columns of a matrix
x<-seq(1,10)
y<-matrix(x,nrow=2,ncol=5)
apply(y,1,mean)
apply(y,2,mean)

# sapply:  Apply function to each element of a list or dataframe
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
sapply(x,mean)

# tapply: Apply a function to each set of elements with the
#               same level of a factor
z<-factor(c("Ctrl","Ctrl","A","A","B","B"))
x<-c(5,3,4,NA,10,4)
tapply(x,z,mean,na.rm=TRUE)
```

Extended example

Part 1: Hardy-Weinberg in practice

- Read in genotype data from a file (4,014 SNPs in 60 individuals)
- Exploratory plot of heterozygosity vs. allele frequency.
  - Recall that the Hardy-Weinberg expected proportion of heterozygotes H as a function of allele frequency p is: $H = 2p(1-p)$
- Formal test of Hardy-Weinberg proportions using a χ^2-test for each SNP

Part 2: Finding a quantitative trait locus via association mapping

- Read in phenotype data from a file (fasting glucose in units of mmol/L)
- Test for each SNP whether genotype is correlated ("associated") with phenotypic trait value using a linear model framework (and the lm function)
- Find which SNP has the association and visualize its effect in a boxplot

Data source

- 2.3 million SNPs genotyped on Hapmap CEU founders (60 individuals). Data downloaded in plink format from plink website.
- SNPs from chromosome 2 and LD pruning undertaken (`--indep-pairwise 50 5 0.2`) to make the data set smaller