Introduction to R

**Part 1: Getting Started**

*download, install, run, help*

<http://www.r-project.org>

Select a mirror site near you

Windows

Download and Install R

base

(click exe of latest version)

OSX

MacOS X

(click .pkg of latest version)

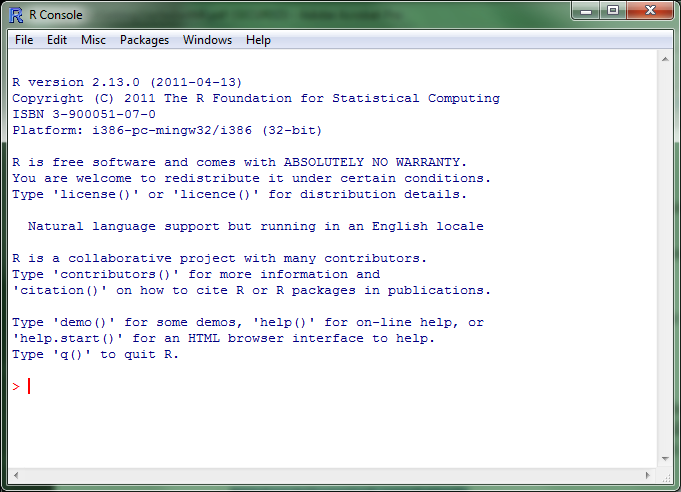
Optional: Download, install, and run Rstudio

<http://rstudio.org/download/desktop>

**Install R**

**Start R**

This is the R console



Type or paste code here.

**Working with R**

Typically when working with R, you will have some code you are working on in a text file in one window and the R will be open in a separate window. You will usually write your code in the text editor window and then copy / paste / send it into the R console to run. I will be showing you how to do this using Rstudio, but other options include Tinn-R or Wordpad (or your text editor of choice).

**Set up for this tutorial**

In this document, anything you see in red should be what you enter into the R console, and in blue will be the R output.

For the purposes of this tutorial, the following commands will create a folder on your H drive and copy some data to it that we'll be using in the tutorial.

dir.create("H:/introR")

file.copy(from="U:/mcm/presentations/CompTools/introR/data",to="H:/introR ",recursive=T)

Set your working directory

Your working directory will be where any files you generate end up. It can also be convenient (though not strictly required) to be the location of any data you might be reading in along with the location of any script you are currently working on.

setwd("H:/introR")

If you don’t explicitly set your working directory on windows, it will be set to

C:\Documents and Settings\mcm\My Documents *(Windows XP)*

or

C:\Users\mcm\Documents *(Windows Vista, Windows 7)*

**Get help on a function**

?mean

Browse Documentation

*Try asking for help on setwd*

help.start()

**Web resources**

R search engine - <http://rseek.org>

Stack Overflow - <http://stackoverflow.com>

R graph gallery - <http://addictedtor.free.fr/graphiques/>

**Entering commands**

You can use R like a calculator

1+1

[1] 2

To explain the result, the [1] before the 2 is because R considers the result of a calculation a vector with one element. So the first element of the vector is labeled with [1].

2 + 2

100 - 20

4 \* 6

10^2

100 / 4

sqrt(2)

log2(2)

Remember, to get help on a function, type question mark followed by function name

*What is 2 to the 12th?*

*What is the log (base 10) of 10000?*

?log2

**Create a vector and store it in a variable**

A vector in R can be a list of numbers, of strings (letters/words), or of TRUE/FALSE values. You can use the c() function to **combine** elements into a vector.

x<-c(1,3,6,8,13)

y<-c(2,5,4,7,12)

words<-c("hi","how","are","you")

sentences<-c("Hi, how are you?", "I am fine.")

torf<-c(TRUE,TRUE,FALSE,TRUE,TRUE)

If you mix strings and numbers, the numbers will be treated as strings.

v1<-c(6, 5,"hi")

[1] "6" "5" "hi"

A vector of two vectors gets flattened into one vector.

z<-c(x,y)

z

[1] 1 3 6 8 13 2 5 4 7 12

There is a shortcut in R, to create a vector with integers from one number to another.

1:10

[1] 1 2 3 4 5 6 7 8 9 10

to store that in a variable, you would do

m<-1:10

*create a vector of words of your choosing*

*create a vector of numbers from 5 to 20*

Calculate some basic statistics on vectors

Note: when you see a # in front of something, it's just a comment and is ignored by R.

mean(x)

median(x)

#sample standard deviation

sd(x)

[1] 4.658326

summary(x)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1.0 3.0 6.0 6.2 8.0 13.0

#correlation

cor(x,y)

[1] 0.9442803

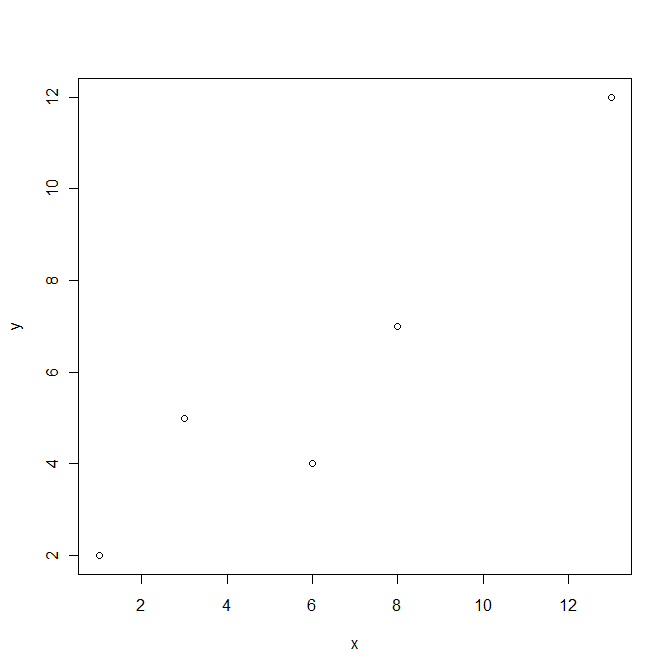
length(y)

[1] 5

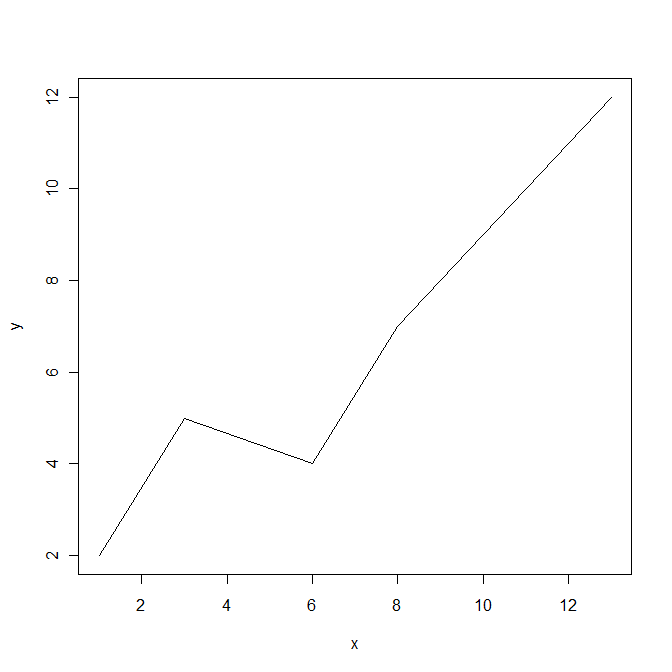
*what is the median of numbers 4 to 7?*

Plot x vs y

plot(x,y)



plot(x,y,type='l')



More on this later!

**Read in some data**

df <-read.table("H:/introR/data/yeast/gene\_relationships.txt",header=T,sep='\t')

In this case, data.txt must be a text file with the following properties:

* Each line contains one record
* Within each record, fields are separated by a tab, comma, or other single-character
* Each record contains the same number of fields

At some point as you're learning R, you will likely run into problems reading in a data file. Try the following to troubleshoot:

* sep='\t'
* quote=''
* comment.char=''
* strip.white=T
* open in Excel or a text editor, look for weird characters or spacing problems, extra tabs, etc.
* try reading in the first few lines with nlines=5 or skipping the first line with skip=1

Data frames

A Data frame is a rectangular collection of columns. Each column can be a different type of data, numbers, strings, logical (TRUE/FALSE). If you use read.table, the result is already a data.frame, which you can check using

class(df)

[1] "data.frame"

Each column of the data frame is a vector, and each element of the data frame will be whatever type it is. You can access an individual row:

df[1,]

column:

df[,4]

or element:

df[2,7]

as well as little sections:

df[1:10,]

head(df)

df[1:3,1:4]

You can also access columns by name, two different equivalent ways:

df$gene

df[,"gene"]

If you wanted to access a specific gene's row by gene name, you could do the following:

df[df[,4]=="YIL162W",]

The inner part df[,4]=="YIL162W" is actually something we call an index vector, which has a logical value of TRUE/FALSE for every element in the column telling whether or not it is equal to "YIL162W". Then we are applying that index vector to every row in the data frame, only pulling out rows for which it is TRUE.

You could do something similar to find all overlapping genes:

df[df$left\_gene\_relationship == "overlapping",]

Or all genes without a close neighbor

*Save the 4th column from the data frame into a new variable called "genes".*

*Find the genes with a left\_gene\_dist less than 0.*

df[df$left\_gene\_dist > 5000 & df$right\_gene\_dist > 5000,]

Factors

Factors are a feature of R used for categorical data, but might trip you up at first. When reading in a file with read.table, string data will automatically be converted to a factor unless you specify as.is=T.

df <-read.table("H:/introR/data/yeast/gene\_relationships.txt",header=T,sep='\t',as.is=T)

More on scatter plots

Read in a data set listing temperature measurements across the US from 1851 until 2009.

weather<-read.csv("H:/introR/data/weather/us\_weather.csv",as.is=T,strip.white=T)

Make a version with all the temperatures in Fahrenheit

fweather<-weather

fweather[,7:18]<-weather[,7:18]\*9/5 + 32

read.csv is another function to read in a file specifically for comma-separated-values, you could also use read.table here with sep=','.

Select only the weather for Des Moines, Iowa for a moment.

iv<-fweather[,1]=="DES MOINES"

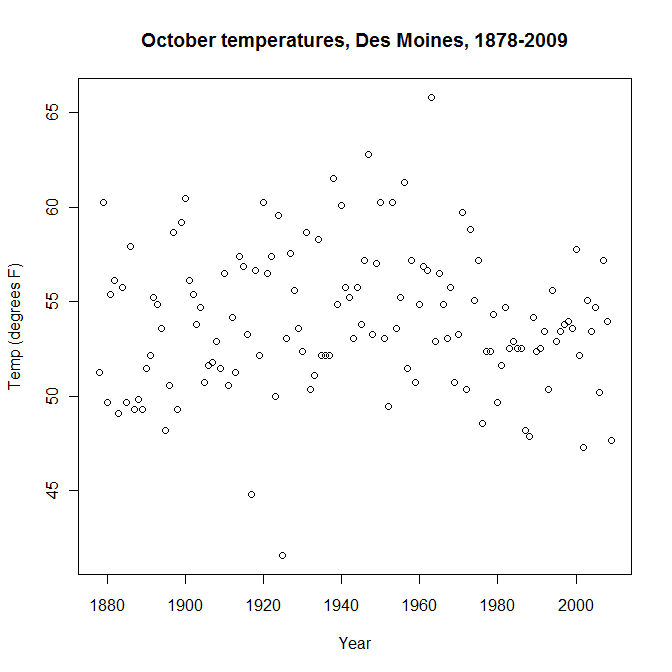
dm<-fweather[iv,]

dm[,"Oct"]

plot(dm[,"Period"],dm[,"Oct"])

Add a title, label axes.

plot(dm[,"Period"],dm[,"Oct"],main="October temperatures, Des Moines, 1878-2009",xlab="Year",ylab="Temp (degrees F)")



Histogram

Make a histogram of the October mean temperatures in Des Moines.

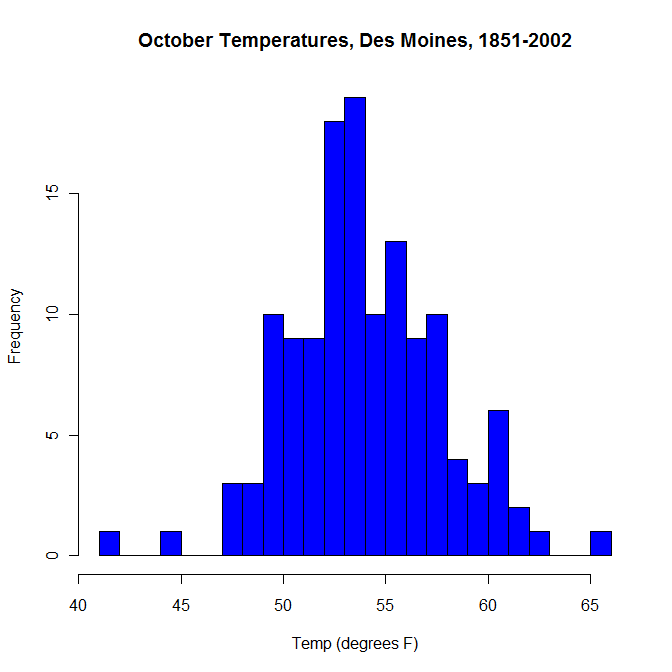
hist(dm[,"Oct"])

Make the histogram have more bins

hist(dm[,"Oct"],breaks=20)

Make the histogram a different color and give it a better title and labels

hist(dm[,"Oct"],breaks=20,col='blue',main="October Temperatures, Des Moines, 1878-2009",xlab="Temp (degrees F)",ylab="Frequency")



Linear Regression

Linear regression is a way to model the relationship between two (or more) variables using a linear function. Slope and intercept of the function are estimated from the data. (Note - you have to set up the formula like y~x)

lm(dm$Oct~dm$Period)

Call:

lm(formula = dm$Oct ~ dm$Period)

Coefficients:

(Intercept) dm$Period

57.571448 -0.001862

You could also store that in a variable if you want

model<- lm(dm$Oct~dm$Period)

What is that thing "model"? What does the function "lm" return? An object.

class(model)

[1] "lm"

What is in it? str is a nice way to examine a mysterious object in R, we'll see more of those later.

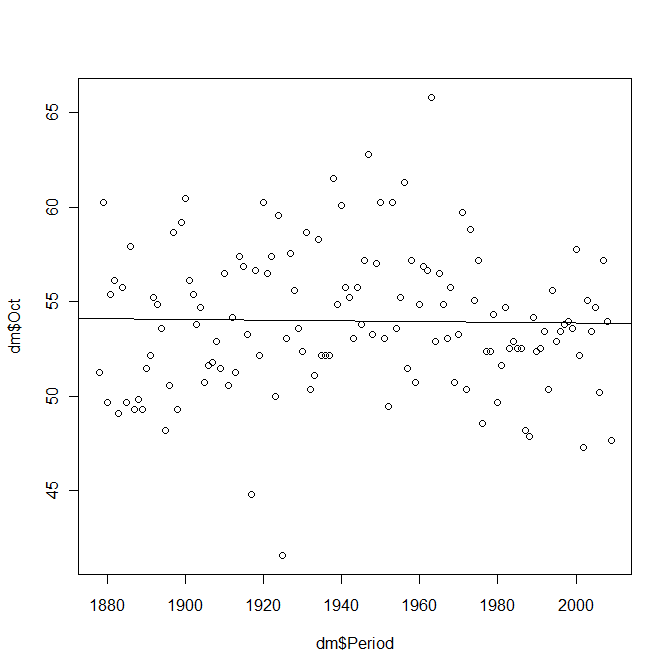
str(model)

Plot the regression line on the scatter plot.

plot(dm$Period,dm$Oct)

abline(lm(dm$Oct~dm$Period),col='blue')

Not much increase in temperatures for Des Moines in October over the period.



Let's add a new column to our data frame with the mean monthly temperatures for each year.

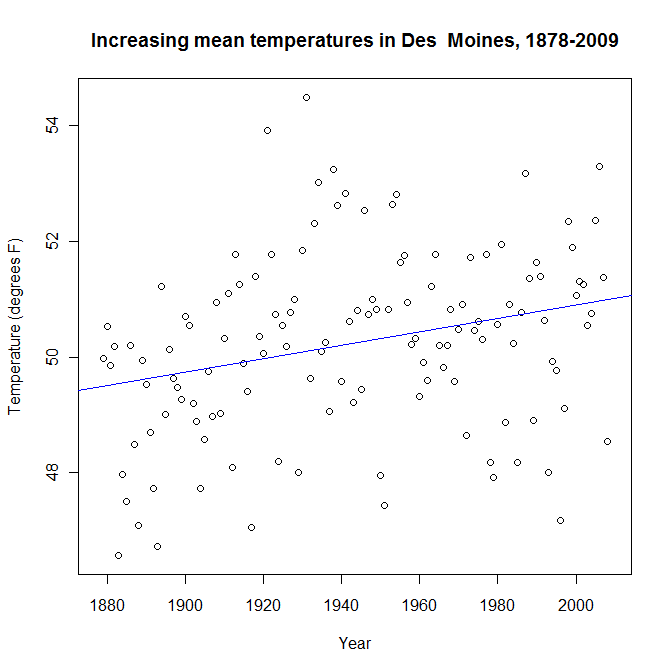
head(dm)

dm$mn<-rowMeans(dm[,7:18],na.rm=T)

If we look at the means of all the monthly temperatures, there is an increase.

plot(dm$Period, dm$mn, main="Increasing mean temperatures in Des Moines, 1878-2009",xlab="Year",ylab="Temperature (degrees F)")

abline(lm(dm$mn~dm$Period),col='blue')



Highlighting points on a scatter plot

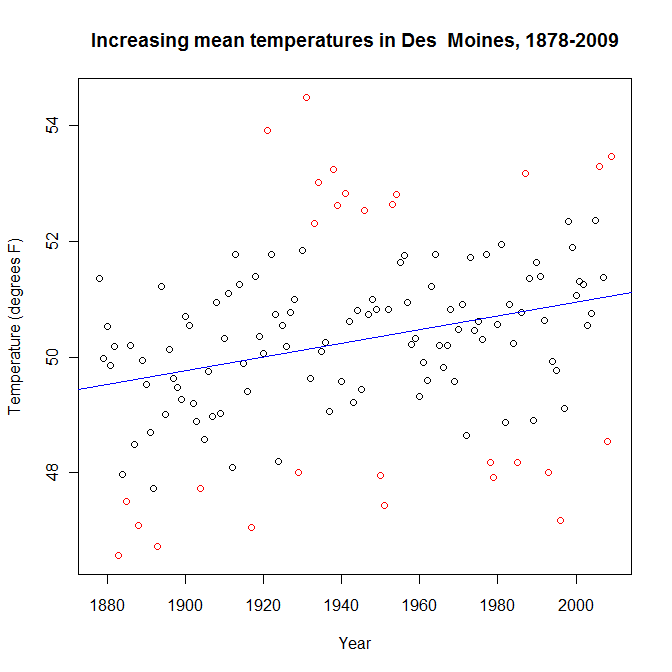
Highlight some outliers on the plot (based on residual, which is their distance from the regression line).

meanmodel<-lm(dm$mn~dm$Period)

iv<-abs(meanmodel$resid)> 2

iv is what we call an index vector, a way of selecting some part of your data in R. If you type *iv*, you see a lot of TRUE and FALSE values. iv is the same length as weather, with a TRUE or FALSE for each element depending on whether or not its residual > 3.

points(dm[iv,"Period"],dm$mn[iv], col='red')



Now that we have our index vector, we can use it to examine our data frame as well.

dm[iv,]

Create a barplot

*Read the following file into a variable called "microarray":*

*H:/introR/data/array/microarray1.txt*

*Is there a header on the file? Don't forget header=T*

*Once you have read in the file, try to make a scatter plot of Cy3 vs Cy5.*

*Create an index vector selecting points that have log2(Cy5/Cy3) > 1.*

*Highlight those points on the plot in blue.*

Read in some hox gene qPCR data (RA induction in mouse ES cells).

hox<-read.table("H:/introR/data/hox/hox\_qpcr.txt",sep='\t',header=T)

Change the column names (sorry, kind of messy)

colnames(hox)<- gsub("X","t",colnames(hox))

colnames(hox)<-tolower(colnames(hox))

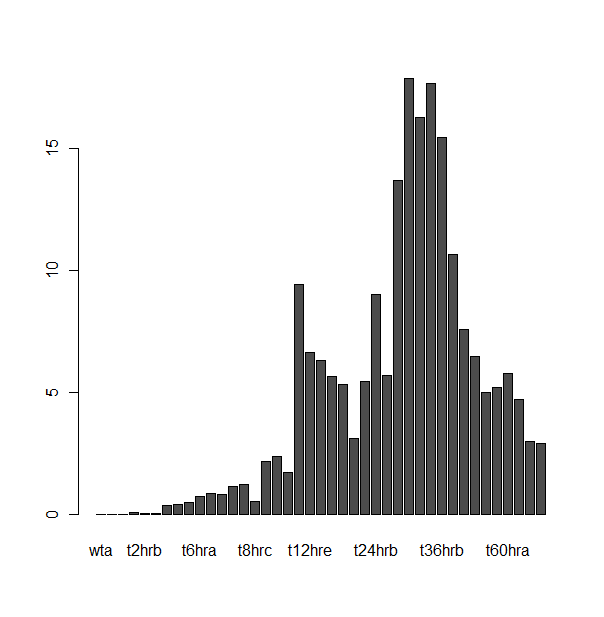
Try making a barplot of one gene. First, let's just pull out the data.

hoxb1<- hox[hox$gene=="Hoxb1",2:42]

For a barplot, R expects a matrix or a vector. A matrix is like a data.frame, but all the columns are the same type - numbers or strings.

*If height is a vector, the plot consists of a sequence of rectangular bars with heights given by the values in the vector. If height is a matrix and beside is FALSE then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked sub-bars making up the bar. If height is a matrix and beside is TRUE, then the values in each column are juxtaposed rather than stacked.*

barplot(as.matrix(hox[hox$gene=="Hoxb1",2:42]))



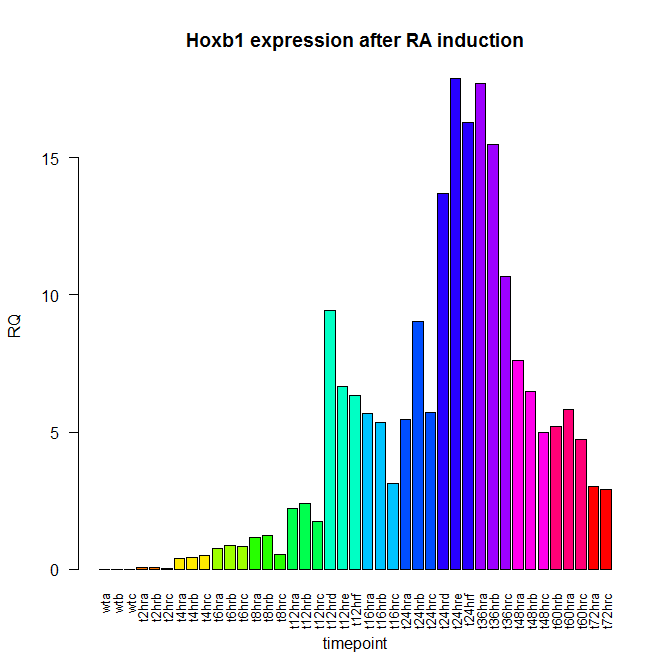
Sort of. But let's fix a few things. Let's make prettier colors, for one thing.

cols<-rainbow(13)

cols<-rep(rainbow(13),each=3)

Barplot with colors, make labels horizontal, add axis labels and a title.

barplot(as.vector(t(hox[hox$gene=="Hoxb1",2:42])),col=cols,las=2,names.arg=colnames(hox)[2:42],ylab="RQ",xlab="timepoint",main="Hoxb1 expression after RA induction",cex.names=.8)



I did a few tricky things in there. t() is a function to transpose a data frame.