An Introduction to R

Some examples drawn from Dalgaard, P. 2002. Introductory Statistics with R. Springer, NY.

The R Project for Statistical Computing: **http://www.r-project.org**

The Comprehensive R Archive Network (CRAN) at MTU: **http://cran.mtu.edu**

An introduction to R

R is just an overgrown calculator, at least depending on how you use it!

Most people invoke R as a command session, and either have a customized environment that allows them to edit and submit scripts directly, or cut and paste sections of scripts into R. In Windows and Mac OS X, there is a primitive GUI that acts really just as a frame for an R command session. On my Mac computer:

```
mulroney:~ rfroese$ R

R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The “greater than” (“>”) is your R prompt. You're off!

Basics

So try some things, assuming R is an overgrown calculator.

```
> 2+2
[1] 4

> exp(-2)
[1] 0.1353353

> rnorm(15)
```

```
 [1]  0.29043778 -0.01350052 -0.08649342 -0.15137987  0.62365504 -
0.26824288
 [7] -0.10575289 -0.56022855 -1.33810966 -0.58627471 -0.09385359
0.57250891
[13]  0.26510724  0.62065821  0.54348209
```

Assignments

```
> x <- 2
> x
[1] 2
> x+x
[1] 4
```

Vectorized arithmetic

```
> weight <- c(60, 72, 57, 90, 95, 72)
> weight
[1] 60 72 57 90 95 72

> height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
> bmi <- weight/height^2
> bmi
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630

> sum(weight)
[1] 446

> sum(weight)/length(weight)
[1] 74.33333

> xbar <- sum(weight)/length(weight)

> weight-xbar
[1] -14.333333  -2.333333 -17.333333  15.666667  20.666667  -2.333333

> (weight-xbar)^2
[1] 205.444444   5.444444 300.444444 245.444444 427.111111   5.444444

> sum((weight-xbar)^2)
[1] 1189.333

> sqrt(sum((weight-xbar)^2)/(length(weight)-1))
[1] 15.42293

> mean(weight)
[1] 74.33333

> sd(weight)
[1] 15.42293
```

Vectors

```
> c("Huey","Dewey","Louie")
[1] "Huey"  "Dewey" "Louie"

> c('Huey','Dewey','Louie')
[1] "Huey"  "Dewey" "Louie"

> c(T,T,F,T)
[1]  TRUE  TRUE FALSE  TRUE

 > bmi > 25
[1] FALSE FALSE FALSE FALSE  TRUE FALSE
```

Missing Values

```
> NA
[1] NA
> x <- NA
> x
[1] NA
> x <- c(NA,NA,NA,NA,NA)
> x
[1] NA NA NA NA NA
```

Functions that create vectors

```
> c(42,57,12,39,1,3,4)
[1] 42 57 12 39  1  3  4

> seq(4,9)
[1] 4 5 6 7 8 9

> seq(4,10,2)
[1]  4  6  8 10

> 4:9
[1] 4 5 6 7 8 9

> oops <- c(7,9,13)

> rep(oops,3)
[1]  7  9 13  7  9 13  7  9 13
> rep(oops,1:3)
[1]  7  9  9 13 13 13

> rep(1:2,c(10,15))
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Matrices and arrays

```
> x <- 1:12
> dim(x) <- c(3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

> matrix(1:12,nrow=3,byrow=T)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12

> x <- matrix(1:12,nrow=3,byrow=T)
> rownames(x) <- LETTERS[1:3]
> x
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12

> t(x)
     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12


> cbind(A=1:4,B=5:8,C=9:12)
     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12

> rbind(A=1:4,B=5:8,C=9:12)
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
```

Data types: factors

```
> pain <- c(0,3,2,2,1)
> fpain <- factor(pain,levels=0:3)
> levels(fpain) <- c("none","mild","medium","severe")
> fpain
[1] none   severe medium medium mild
Levels: none mild medium severe
```

```
> as.numeric(fpain)
[1] 1 4 3 3 2
```

Data types: lists

```
> intake.pre <-
c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
> intake.post <-
c(3910,4220,3885,5160,5645,4680,5265,5975,6790,6900,7335)
> mylist <- list(before=intake.pre,after=intake.post)
> mylist
$before
 [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770

$after
 [1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335

> mylist$before
 [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
```

Data types: data frames

```
> d <- data.frame(intake.pre,intake.post)
> d
   intake.pre intake.post
1        5260        3910
2        5470        4220
3        5640        3885
4        6180        5160
5        6390        5645
6        6515        4680
7        6805        5265
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335

> names(d) <- c("before","after")

> d$before
 [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
```

Indexing

```
> intake.pre[5]
[1] 6390
> intake.pre[c(3,5,7)]
[1] 5640 6390 6805

> v <- c(3,5,7)
```

```
> intake.pre[v]
[1] 5640 6390 6805

> intake.pre[1:5]
[1] 5260 5470 5640 6180 6390
> intake.pre[-c(3,5,7)]
[1] 5260 5470 6180 6515 7515 7515 8230 8770
```

Conditional indexing

Comparison and logical operators:
== != >= <= > <
| (or) **&** (and) **!** (not)

```
> intake.post[intake.pre > 7000]
[1] 5975 6790 6900 7335
> intake.post[intake.pre > 7000 & intake.post <= 8000]
[1] 5975 6790 6900 7335
> intake.pre > 7000 & intake.post <= 8000
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

Indexing data frames

```
> d <- data.frame(intake.pre,intake.post)
> d[5,1]
[1] 6390
> d[5,]
  intake.pre intake.post
5       6390        5645

> d[d$intake.pre>7000,]
   intake.pre intake.post
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335

> i <- d$intake.pre>7000
> i
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
> d[i,]
   intake.pre intake.post
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335
>
```

Reading and writing data from external files

```
> redpine <- read.table(file="redpinedata.csv",sep=",",header=T)

> write.table(redpine,file="redpinedata.csv",sep=",")
```

```
> save(redpine,file="redpinedata.Rdata")

> load(file="redpinedata.Rdata")
```

Grouped data

```
> ex <- redpine[redpine$plot==1,]
> ex
   plot tree spp  dbh   ht
10    1    5  RP 15.9 52.6
3     1    8  RP 20.7 66.3
17    1   14  JP  9.6 39.2
4     1    6  RP 17.4 59.6
9     1    2  RP 15.9 53.9
5     1    1  RP 17.7 61.7
16    1   16  JP  9.9 32.9
15    1   17  JP 11.2 41.2
1     1   13  RP 22.8 71.3
13    1    4  RP 15.0 57.1
2     1   11  RP 19.0 54.6
6     1   15  RP 17.5 56.2
8     1   12  RP 17.7 65.8
14    1   10  JP 11.3 47.6
7     1    7  RP 17.9 62.5
12    1    3  RP 14.9 55.6
11    1    9  RP 15.9 55.3
> ex.JP <- ex[ex$spp=="JP",]
> ex.JP
   plot tree spp  dbh   ht
17    1   14  JP  9.6 39.2
16    1   16  JP  9.9 32.9
15    1   17  JP 11.2 41.2
14    1   10  JP 11.3 47.6
```

Sorting

```
> ex$spp
 [1] RP RP JP RP RP RP JP JP RP RP RP RP RP JP RP RP RP
Levels: JP RP
> sort(ex$spp)
 [1] JP JP JP JP RP RP RP RP RP RP RP RP RP RP RP RP RP
Levels: JP RP
> order(ex$spp)
 [1]  3  7  8 14  1  2  4  5  6  9 10 11 12 13 15 16 17
> ex[order(ex$spp),]
   plot tree spp  dbh   ht
17    1   14  JP  9.6 39.2
16    1   16  JP  9.9 32.9
15    1   17  JP 11.2 41.2
14    1   10  JP 11.3 47.6
10    1    5  RP 15.9 52.6
3     1    8  RP 20.7 66.3
```

```
4     1    6  RP 17.4 59.6
9     1    2  RP 15.9 53.9
5     1    1  RP 17.7 61.7
1     1   13  RP 22.8 71.3
13    1    4  RP 15.0 57.1
2     1   11  RP 19.0 54.6
6     1   15  RP 17.5 56.2
8     1   12  RP 17.7 65.8
7     1    7  RP 17.9 62.5
12    1    3  RP 14.9 55.6
11    1    9  RP 15.9 55.3
> ex[order(ex$spp,ex$dbh),]
   plot tree spp  dbh   ht
17    1   14  JP  9.6 39.2
16    1   16  JP  9.9 32.9
15    1   17  JP 11.2 41.2
14    1   10  JP 11.3 47.6
12    1    3  RP 14.9 55.6
13    1    4  RP 15.0 57.1
10    1    5  RP 15.9 52.6
9     1    2  RP 15.9 53.9
11    1    9  RP 15.9 55.3
4     1    6  RP 17.4 59.6
6     1   15  RP 17.5 56.2
5     1    1  RP 17.7 61.7
8     1   12  RP 17.7 65.8
7     1    7  RP 17.9 62.5
2     1   11  RP 19.0 54.6
3     1    8  RP 20.7 66.3
1     1   13  RP 22.8 71.3
```

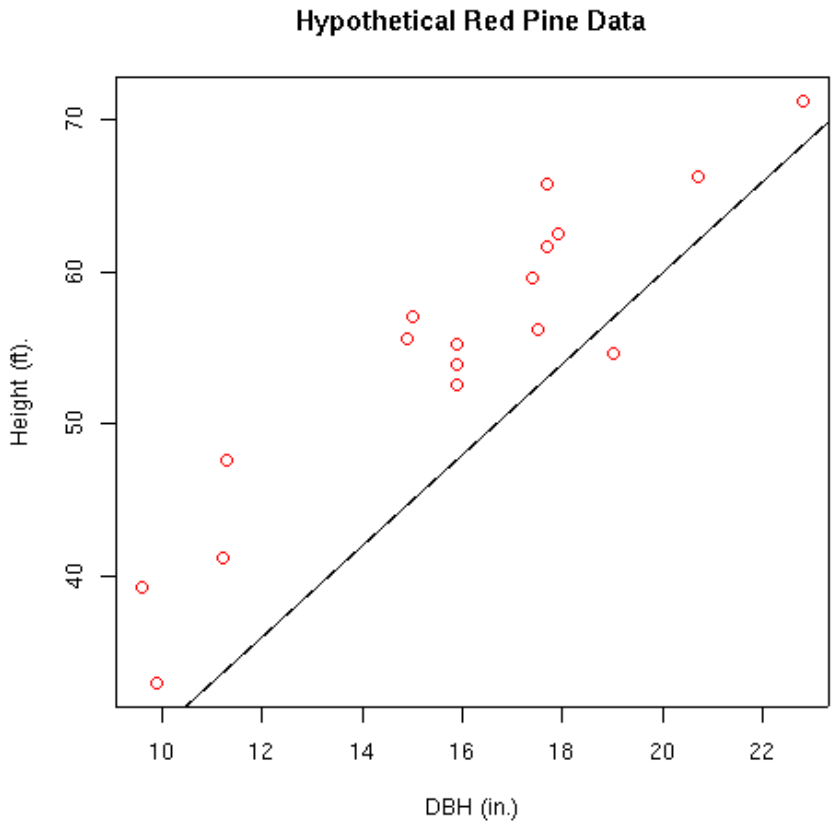
Implicit looping

```
> m <- matrix(rnorm(12),4)
> m
           [,1]       [,2]        [,3]
[1,]  0.2669397 -0.9701464  0.01002090
[2,] -1.4343255 -0.3486424 -1.19555032
[3,] -0.5960320 -1.3306065  0.72308871
[4,] -2.0791788  0.9822958 -0.27473615

> apply(m,1,min)
[1] -0.9701464 -1.4343255 -1.3306065 -2.0791788

> tapply(redpine$dbh,list(redpine$spp),mean)
       JP        RP
 9.211111 15.457143

> tapply(redpine$dbh,list(redpine$spp,redpine$plot),mean)
          1     2     3
JP 10.50000  9.95  7.00
RP 17.56154 15.11 10.68
```

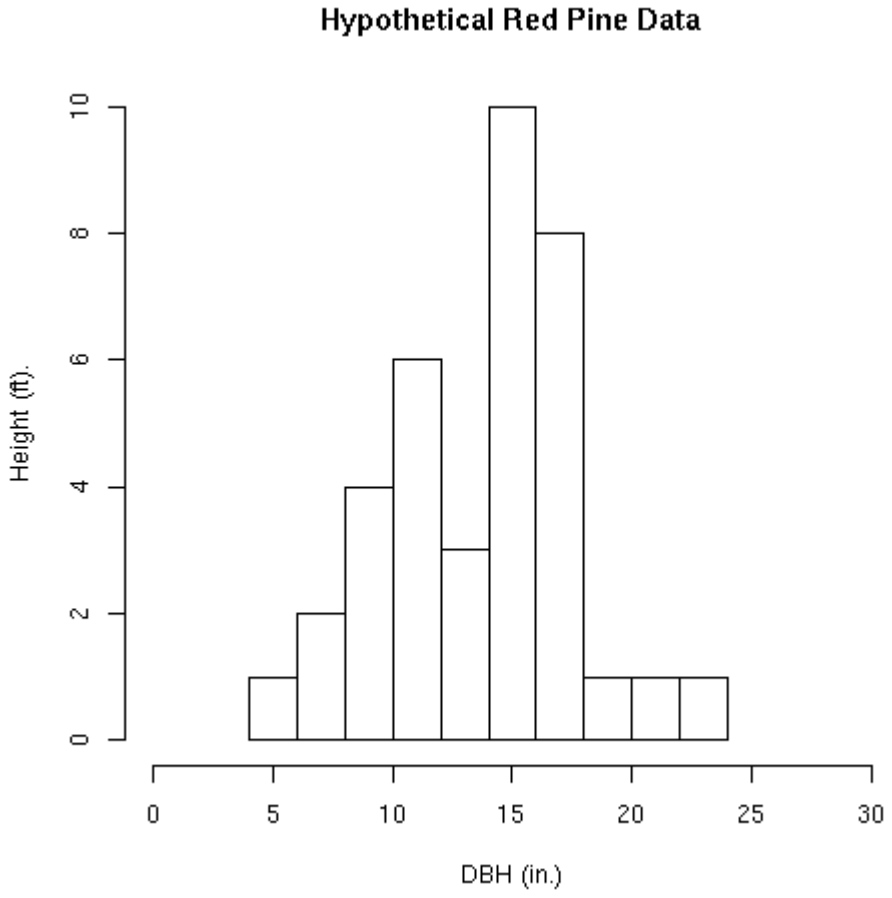
```
> table(redpine$spp,redpine$plot)

     1  2  3
  JP  4  2  3
  RP 13 10  5
>
```

Graphics

```
> plot(ex$dbh,ex$ht,xlab="",ylab="",col="red")
> title(main="Hypothetical Red Pine Data", xlab="DBH
(in.)",ylab="Height (ft).")
> abline(0,3)
```




```
> hist(redpine$dbh,xlim=c(0,30),ylim=c(0,10),xlab="",ylab="",main="")
> title(main="Hypothetical Red Pine Data", xlab="DBH
(in.)",ylab="Height (ft).")
```

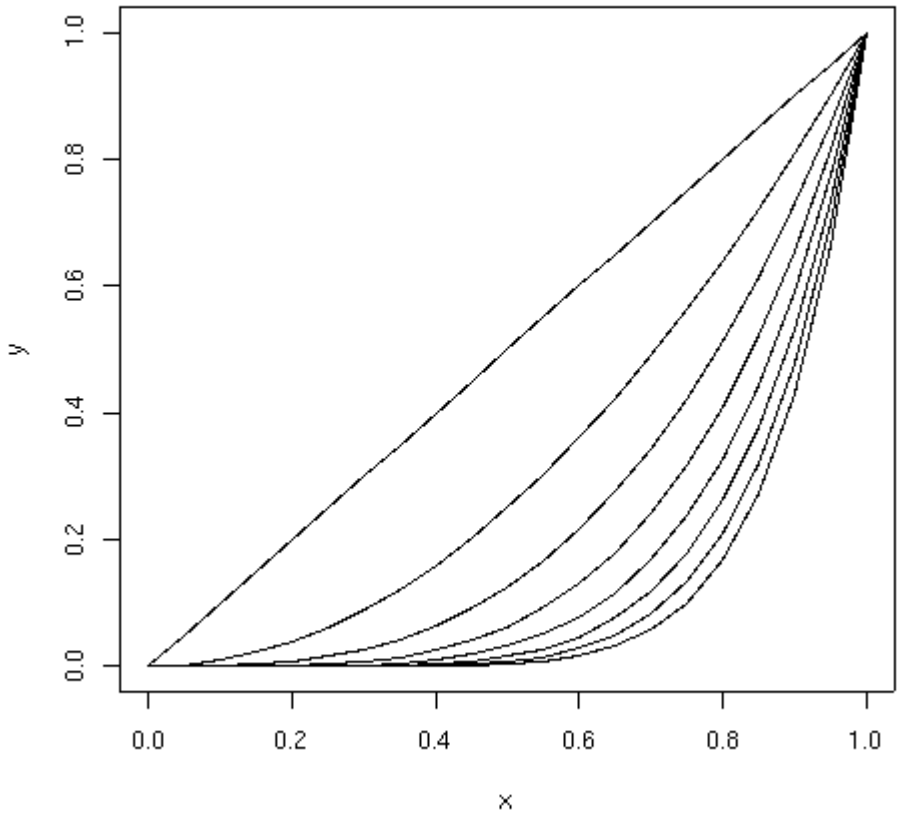


Writing graphics to files

```
> pdf(file="myplot.pdf")
> plot(ex$dbh,ex$ht,xlab="",ylab="",col="red")
> title(main="Hypothetical Red Pine Data", xlab="DBH
(in.)",ylab="Height (ft).")
> abline(0,3)
> dev.off()
null device
          1
>
```

Looping

```
> x <- seq(0,1,.05)
> plot(x,x,ylab="y",type="l")
> for (j in 2:8) {
+   lines(x,x^j)
+ }
```



Miscellaneous

```
> if(mean(ex$dbh)<10) stand <- "small" else stand <- "large"
> stand
[1] "large"

> ls()
 [1] "d"           "ex"          "ex.JP"       "fpain"       "i"
 [6] "intake.post" "intake.pre"  "m"           "mylist"      "pain"
[11] "redpine"     "stand"       "v"
> rm("stand")
> ls()
 [1] "d"           "ex"          "ex.JP"       "fpain"       "i"
 [6] "intake.post" "intake.pre"  "m"           "mylist"      "pain"
[11] "redpine"     "v"

>?plot
>help.search("regres*")
>fix(redpine)
```

Bringing it all together

```
par(mfrow=c(3,1))
for (p in 1:3) {
  i <- redpine$plot==p
  plot(redpine$dbh[i],redpine$ht[i],ylab="Height (ft)",xlab="DBH
(in)",main=paste("Sample Plot",p))
}

par(mfrow=c(1,1))
clist <- c("red","blue","green")
plot(0,0,type="n",xlim=c(0,max(redpine$dbh)),ylim=c(0,max(redpine$ht))
,ylab="Height (ft)",xlab="DBH (in)",main="Red Pine by Sample Plot")
for (p in 1:3) {
  i <- redpine$plot==p
  points(redpine$dbh[i],redpine$ht[i],col=clist[p])
}
legend(15,15,legend=c("Plot 1","Plot 2","Plot 3"),col=clist,pch=1)
```