1 Simple Plots

2 Parallel Computing

3 Accessing Remote Servers
Plot 1: Treatment Effects on Responsiveness

- Threat of Collective Action
- Threat of Tattling to Superiors
- Claims of Loyalty

Response estimates, with 95% confidence intervals, for different treatments.
par(mar=c(8, 4, 1, 2) + 0.1)

plot(1:3,res.t[,1],pch=16,cex=2,ylim=c(-0.1,0.2),xlim=c(0.5, 3.5),
     main = "",xlab="",ylab="",axes=FALSE)
for(i in 1:3) segments(i,res.t[i,2],i,res.t[i,3],lwd=2)
abline(h=0,lty=2,col=2)

box()
axis(1,1:3,labels=c("...","...","..."),cex.axis=1.5,mgp=c(3,4,0))
axis(2,cex.axis=1.5)

mtext("Treatment",side=1,line=6.5,cex=1.5)
mtext("Treatment Effects on Responsiveness",side=2,line=3,cex=1.5)
```r
d <- d[order(d$est),]

par(mar=c(4,6,2,2),las=1)  # label turn 90 degree
plot(1,axes=F,xlab='',ylab='',xlim=c(-0.5,0.5),ylim=c(0.5,30.5))

abline(v=0,lty=2,col="gray")
points(d$est[d$low_mkt==0],c(1:30)[d$low_mkt==0],col=1,pch=16)
points(d$est[d$low_mkt==1],c(1:30)[d$low_mkt==1],col=2,pch=16)
for (i in which(d$low_mkt==0)) lines(c(d$ci1[i],d$ci2[i]),c(i,i))
for (i in which(d$low_mkt==1)) lines(c(d$ci1[i],d$ci2[i]),c(i,i),col=2)

axis(side=1,cex.axis=1.5)
axis(side=2,at=1:30,labels=d$lab,cex.axis=1)
mtext("Coefficient",1,line=3,cex=1.5)
box()
```
This example is borrowed from Chris Adolph’s *short course* on data visualization.
library(RColorBrewer) # gives nice colors

col<-brewer.pal(3,"Dark2")

par(las=1)
plot(1,xlim=range(age),ylim=c(0,1),axes=F,cex.lab=1.5,
     xlab="Age of Respondents",ylab="Probability of Voting")

polygon(c(rev(age), age), c(rev(coef.nohs[,3]), coef.nohs[,2]),
       col = paste(col[1],"30",sep=""), border = NA)
lines(age,coef.nohs[,1],col=col[1],lwd=1.5)
...

axis(1,cex.axis=1.2)
axis(2,cex.axis=1.2,lwd=0,lwd.ticks = 1)
text(60,0.30,"Less than HS",cex=1.2,col=col[1])
...
1. Simple Plots

2. Parallel Computing

3. Accessing Remote Servers
Why Going Parallel

- It is necessary
  - Before too long, you’ll need to perform some tasks repeatedly
  - In statistics, some (re)sampling techniques become more and more popular, e.g., bootstrap, Monte Carlo, MCMC, etc.

- It is feasible
  - Now almost everyone have computers with multiple cores
  - Cloud computing clusters have many nodes
There are multiple ways of parallel computing using R
Many of them are really simple
Packages like `foreach`, `snowfall` are readily available
We will be using the `foreach` package in this short introduction
If you can do loop (for), you can do parallel computing (foreach)

Check out Using The foreach Package
In our example, `foreach` is the frontend, `doParallel` is the backend
Most parallel computing procedures involve the following three steps:

1. Split the problem into pieces
2. Execute the pieces in parallel
3. Combine the results back together

`foreach` does the above all together

In our exercise, we need other two steps

0. At the beginning, register clusters
4. In the end, summarize the results
library(doParallel)
library(foreach)

# register clusters
cl<-makeCluster(4)
registerDoParallel(cl)

# loop (split, excite, and combine)
result<-foreach (i=1:sims,.combine=c,.inorder=FALSE) \%dopar\% {
  ...
  return(out)
}
stopCluster(cl)

# summarize
mean(result)
sd(result)
# loop (split, excite, and combine)
result<-foreach (i=1:sims,.combine=c,.inorder=FALSE) %dopar% {
  ...
  return(out)
}

In the above example

1. `i=1:sims` splits the tasks
2. `%dopar%` executes each task
3. `.combine=c` combines the results
The `.combine` Option

foreach can:

1. combine numbers to a vector: `.combine=c`
2. combine vectors to a matrix: `.combine=rbind`; `.combine=cbind`
3. combine matrix to an array (of three dimensions):
   need to define a new function, e.g. `.combine=f()`

```r
library(abind)
f <- function(){
  function(...) abind(...,along=3)
}
```

- We will have examples for each of the scenarios
- Time the algorithm using `Sys.time()` and see the difference
- That’s it!
1. Simple Plots
2. Parallel Computing
3. Accessing Remote Servers
Reasons for Using Remote Servers

- **It is useful**
  - Your computer not powerful enough (it just cannot do it)
  - Your computer not fast enough
  - You need your computer for some other purposes (e.g., R, Word, Skype, Netflix...)

- **It is feasible**
  - More and more available cheap/free cloud computing services
    e.g., HMDC, Amazon (EC2), Microsoft
  - MIT Athena is free, but it is not designed for cloud computation
  - We use it as an example; it is at least as powerful as your laptop
Connecting to Athena

First, we need to remotely display a unix windows

- If you are a windows user, you’ll need to download SecureCRT
- If you are a Unix/Mac OS user, you can work directly in the console
UNIX/Linux users and users of the Mac OS X “Terminal” application can also use the command line programs “ssh”, “scp”

- **Dial-up**
  
  `ssh username@athena.dialup.mit.edu`

- **Upload:**
  
  `scp file1 file2 username@ftp.dialup.mit.edu:/mit/username`

- **Download**
  
  `scp username@ftp.dialup.mit.edu:/mit/username/filename local_file`
Getting SecureCRT/FX to Work

Download SecureCRT from MIT IS&T: http://ist.mit.edu/securecrt-fx and install it

Configurations:

- New connection
- Hostname: athena.dialup.mit.edu
- User name: your Kerberos username
- Type of connection: SSH2
- File Transfer: SFTP
- Session name: whatever

You'll arrive at a unix console
Getting SecureCRT/FX to Work

New Session Wizard

What is the name or IP address of the remote host?
The user name can be left blank.

Hostname: athena.dialup.mit.edu
Port: 22
Firewall: None
Username: yqy

<上一步  下一步  取消>
Make a new directory: $ mkdir mathcamp
Open SecureFX for Data Transfer

Open Secure FX and Connect to Athena. You’ll able to see the dir you created.
Run R Codes at Athena

1. Create an R source file and thoroughly test it
2. Transfer the source file (and data files) to Athena using SecureFX
3. Type the following codes in the console
   
   ```
   $ cd mathcamp
   $ R
   $ source("athena_test.R")
   ```

   - Remember to install packages on the server if necessary
   - Be careful about using multiple nodes; the server manager may call you

4. Copy back the stored file or analyse it in the console
5. That’s it!
Finally, what if you want to shut down your computer (to save energy for humanity) and keep Athena working for you?

- Use `screen` command in Linux!
- Start a new screen: `screen -S xxxx`
- Kill an active screen: `Ctrl -d`
- Detach a screen without stopping it: `Ctrl -a d`
- List screens: `screen -ls`
- Reattach to an existing screen: `screen -x xxxx`
- Kill a detached screen: `screen -X -S xxxx quit`

Once you detach the working screen, you can go watching The Simpsons. Let’s try!