Install.packages (“DEoptim”)

# r automatically kills the clusters if not used for a long time

# for i7 🡪 NEHALEM (use Rblas for speeding up)

A -< available.packages()

Head (rownames(a), 3) # show the names of the first few packages

Install.packages (c(“slidify”), “ggplot2”, “devtools”))

Source(“<http://bioconductor.org/biocLite.R>”)

bicoLite()

biocLite(c(GenomicFeatures,”AnnotationDbi”))

library(ggplot2)

search()

find.package(“devtools”)

install.packages(“devtools”)

library(devtools)

find\_rtools()

install\_github(‘slidify’, ‘ramnathv’)

install\_github(‘slidifyLibraries’, ‘ramnathv’)

dir()

str(Data)

subset(data,parameter, Name %in% c(‘index’, …))

aggregate(subset[,’Arithmetic.Mean’])

names(pallavg)[4] <- ‘new name’

newData = transform (data,Parameter.Name = factor (…))

Libraries to use:

library(bayesm) #for Rossi Bayes package

library(foreach) # for Parallel loop

library(doSNOW) # for parallelization

# define number of clusters for parallel execution

Library(“parallel”)

cl=makeCluster(6)

cl=makeCluster(detectCores()-1)

registerDoSNOW(cl)

# to stop the cluster

stopCluster(cl)

library(parallel)

vignette(parallel)

library(“doparalel”)

library(doMC)

registerDoMC(cores=6)

# to use parallel loop

matrix= foreach(i=1:10, .combine=rbind) % dopar%{ # be aware of function scope problem .export=c(“fccat”)

}

data = foreach(i=1:length(filenames),.packages=c(“ncdf”,”chron”,”stats”),.combine=rbind) %dopar%{

try({})

}

# just putting the variables in the list in the last line returns the list containing all

For (I in seq\_len(p)){}

#Memory function [Lesson: preallocate arry]

# right click on R icon> select properties > shortcut tab> Target field> --max-mem-size=6G

#"C:\Program Files\R\R-3.1.0\bin\i386\Rgui.exe" --max-mem-size=6G

memory.limit()

memory.size()

system.time(R expression) #time it takes to execute

proc.time()[3] # CPU usage in seconds, like tic toc print (end-begin)

getwd() #current working directory

setwd() #set current working directory

Rprof(file=”filename”) turns on profileing and writes to filename

Rprof(“) turns off profiling

summaryRprof(file=”filename”) summarizes output in profile file

#simple for

for (var in 1:10){}

#implicit loop

apply(df[,2:4],2,mean)

ts(apply(gibbsOut$omega\_theta[,,-burn],1:2,mean),start=start(y,),freq=4)

#Create diary file

con = file(“test.log”)

sink(con,append=TRUE)

sink(con,append=TRUE,type=”message)

par(ask=TRUE) # for pausing

#ESC for breaking the program

# read space seperated:

S1=read.csv(“d:/firefoxproject/stars1.csv”,header=F)

Library(xlsx)

Mydata=read.xlsx(“C:/Users/HE/Desktop?.../cleaned10232013.xlsx”,1)

df=read.table("data.txt",header=TRUE)

scan

tmp= ts(read.table(“…”,header=RUE),start=c(1978,1),frequency=12)\*100

Data.frame(matrix(…))

Write.csv(data,file=”thsdata.csv”)

#output

Write.table(m,file=”m.txt,sep=”,”,row.names=FALSE,col.names=FALSE)

#useful writing

cat("in my reg…")

print(var)

cat(“MCMC Iteration (est time to end – min)”,fill=TRUE)

cat(sprint(“<set name\%s\”value=\”%f\></set>\n”,df$timeStamp,def$Price))

print model.tables(aov,ex1,”means”),digits=3)

#debuging

debug(myreg)

Q: quit

undebug(myreg)

stop(“there are less than 10 predictors”)

# to get help

help.search("kewyord")

?doSNOW

?par # all graphic elements available

#inspect elements in workspace

ls()

fsh() # flash

rem(list=ls()) # clear workspace

rm(list=ls(pattern=”^tmp”))

# get structure of a variable , type or storage mode

str(Games)

mode(df$Y)

lmout=lm(Y~X1+X2,data=df)

summary(lmout)

columns(y) 🡨columnes(temp)[1:4]

names(expdFore)

library(Himsc)

describe(mydata)

library(pastecs)

stat.desc9mydata)

library(psych)

describe(mydata)

library™

inspect(firefoxcopy[0:10])

#matrix operations

Replicate (10,diag(2), simplify = F)

Install.packages(“corpcor”)

Make.positive.definite()

Is.positive.definite()

colSums

rowSums

colMeans

rowMeans

lowertri(…)

upper.tri(…) # index of upper and lower triangular matrix

%\*%

\* #element wise!

Length(array)

Mat[,1:2]

t()

list(rep(0,10))

chol(X)

chol2inv(chol(X)) #inverse using cholesky root

crossprod(X,Y) # very efficient t(X)%\*%Y => use more

diag # to create a diagonal matrix

bdiag # to creat block diagonal matrix

diag(5)\*c(1,2,3,4,5) # matrix with this diagonal

A%o%B outer product AB’

%x% # Kronker

backsolve() # inverse of triangular array

sqrt

Sort

Log

%% #100%%10=0

round

floor

ginv # Moore-Penrose Generalized Inverse of A (in MASS library)

eigen(A) # tgives $val of eigen values and $vec for eigne vectors

svd(A) # single value dcomposition d: singular value u: left singular v: right singular

chol(A) # cholesky R’R=A

qr(A) #QR decomposition upper triangular qr: upper,rank, qaux: addition. info on Q, pivot: pivot strategy

pmax(x,y) elementwise comparison of two vectors

matrix(double(nvar\*nz),ncol=nvar)

market = temp[,”MARKET”]-tmp[,”BKFREE”] #address by name

a= NCOL(y)

m2=aMatrix(0,nrow=1000,ncol=1000,sparse=TRUE)

library(‘slam’)

m1=matrix(0,nrow=1000,ncol=1000)

object.size(m1)

rbind

cbind

nrow(x)

ncol(x)

dim(x)[1]

dim(x)[2]

ceiling()

scale() # convert data to standardized score

min,max,median,sd,

mad # mean absolute deviation

table () # frequency table

cumsum #cumulative sum

cumproduct,cummax,cumin

rev() # reverse the order

order()

corr(x,y,use=”pair”) correlation matrix for pairwise complete data

is.finite(x)

is.infinite(x)

is.nan(x)

#function

myreg=function(y,X){

…

}

c<<-1 # for global assignment

#sequence

If (u<alpha){ }

While

y=ifelse(crabs$sattelite>0,1,0)

#distributions

Set.seed(1)

rnorm #normal

runif #uniform

rchisq #chi-square

rgamma (1, shape =sh1, rate=rate)

mean

var

quantile

#optimizer

library(optim) #general purpose

# Newton, quasi Newton, Nelder-mead, simulated annealing, BFGS, L-BFGS-B, SANN (simulated annealing)

library(nlminb)

# uses L-BFGS-B, Robust

library(nlm)

# uses Newton algorithm, Fast

library(nlme)

#rgenoud #genetic algorithm, allows parallel processing using SNOW

Library(DEoptim)

# uses different genetic optimization routine

library(Optimplex)

# simplex based algorithm, simplex method of Spendley et al., the method of Nelder and Mead, Box’s algorithm for constraint optimization, and the multi-dimensional search by Torczon

solve(fHess(exp(filt$par),func(x),dlmLL(y+build(log(x))))$Hessian)

#fdHess in nlme package: calculate hessian numerically and we can send transformed parameters

Ans=optimx(fn=function(x) sum(x\*x),par=1:2)

constrOptim(theta,f,grad,ui,ci,mu=1e-04,control=list(),method=if(isnull(grad)) “Nelder-Mead” else “BFGS”,outher.iterations=100,outer.eps=1e-05,…,hessian=FALSE)

constrOptim(c(0.99,0.001),func,NULL,ui=rbind(c(-1,1), #the –x-y>0

c(1,0), #the x>0

c(0,1)), #the y>0

ci=c(-1,0.0001,0.0001)) # the threshold

#Constraints in the form of: bounds = matrix(c(0,5,0,Inf,0,Inf,0,1),nc=2, byrow=True)

#columnes(bounds)=c(“lower”,”upper)

#n<-nrow(bounds)

#ui = rbind (diag(n),…,diag(n))

# ci = c(bound[,1],-bound[,2])

# I <- as.vector(is.finite(bounds))

#ui <- ui[I,]

# ci <- ci[i]

#constroptim (param, f, grad=NULL, ui=ui,ci=ci)

library(BBML)

mle.results=mle2(norm.fit,start=list(mu=1,sigma=1),data=list(x))

out = nlm(mlog,mean(x),x=x)

# send serveral parameters

Mlogl=function(theta,x){

alpha=theta[1]

lambda=theta[2]

return(-sum(dgamma(x,shape=alpha,rate=lambda,log=TRUE)))

}

Slpha.start=mean(x)^2/var(x)

out= nlm(mlogl,theta.start,x=x,hessian=TRUE,fscale=length(x),print.levle=2)

# Nedler-Mead: downhill simplex method : numerical method, unknown derivative, heuristic search, converges non stationary points

#SANN: variant of simulated annealing, stochastic global optimization (only func value), temperature decreases according to logarithmic cooling schedule, t current iteration step, ritically dependant upon setting control parameter, not general purpose method, very useful getting to a good rough surface, works for non differentiable function, metropolis function for acceptance probability (Gaussian Markov Kernel with scale proportional to the actual temperature), use for combinatorial problems (if func generates a new point)

#multidimensional array

ar=array(c(1,2,3,4,5,6),dim=c(3,2,2))

reject=array(0,dim(c(R/kee))

gibbsTheta=array(0,dim=c(TT+1,r,MC-1))

gibbsV[1,1,-burn] # remove elements

window(cbind(..,..,..),start=1880,end=1920) # to remove irrelevant part

#change

as.matrix(df)

as.numeric

as.vector

is.matrix

is.list

library(abind) #abind(array1,array2,along=2)

install.packages(“abind”)

matrix(c(0,1,2,0,1,2,0,1,2),byrow=T,ncol=3)

# vector

c(1,2,..,5)

c(0:13)

ts(retailSales,start=c(1995,1),end=c(1977,12),frequency12)

lrtsm2=diff(lrtsm,difference=1)

library(TTR)

library(forecast)

decompose

forecast(fit,24)

#list

l=list(num=1,char="a",vec=c(4,4),list=list(FALSE,2))

l$list

l$num

lgtdata[[i]]=list(y=y,X=X)

Mcmc=list(R=2000,sbeta=0.2,keep=20)

#access elements and subsetting

index=c(3:5) #3 4 5

vec = c(1,2,3,2,5)

Vec[index]

index=vec==2

vec[vec!=2]

Z[,1]=rep(1,nrow(Z))

# size of matrix

Dim

Assess Convergence and summary of posterior

BOA package in R: Robert and Casella (2004) #diagnostic tools

library(dlm)

mcmcMean(cbind(gibbsOut$dV[-(1:burn),],gibbsOut$dW[-(1:burn),]))

mcmcMeans(outGibbs$phi[-burn,],names=paste(“phi”,1:2))

apply(outGibbs$phi[-burn,],2,quantile,probs=c(0.5,.95))

apply(sqrt(outGibbs$vars[-burn,]),2,quantile,probs=c(0.05,.95))

#confidence interval using expected Fisher info

Z = qnorm ((1+0.95)/2)

Alpha.hat+c(-1,1)\*z/(sqrt(n\*trigamma(alpha.hat))

#using observed fisher info

Alpha.hat+c(-1,1)\*z/sqrt(out$hessian)

Hpd(qbeta,shape1=a,shape2=b)

Data Presentation and q-q plot

Round(mcmcMean(sqrt(gibbsV[-(1:burn),])),4)

Qqnorm(residual(damFilt,sd=False)

Tsdiag(damFilt)

Shapiro.test(res) # Shapiro normality test

#elementary Graphic

Par(mfrow=c(2,2)) # multiple images in one

hist(rnorm(1000),breaks=50,col="magenta")

plot(x,y)

line # to add to existing curve

which ((locInd == 1) %in% c(TRUE))

plot(x)

abline(c(0,1),lwd=2,lty=2) # for drawing extra lines

title ("Scatterplot") to put title on the figure

matplot(X) # sequence of plot of columns of X

acf(x) # autocorr func of time series

#Parameters:col,xlab,ylab,main

#other parameters: type="l" #connect scatter plot points with line

#lwd=x # width of line (1 is default, >1 ticker)

#lty=x #type of line (solid or dashed)

#xlim/ylim=c(z,w) #x/y axis runs for z to w

# sample: (col="red",col="magenta",xlab="2",pch=17,xlim=c(-4,4),ylim=(c(0,8),lty=2,lwd=2,

index=4\*c(0:13)+1

matplot(out$Deltadraw[,index],type=”l”,xlab=”Iterations/20”,ylab=” “,main=”Average Respondent Part-Worths”) # main is like title

#density diagram

par(mfrow=c(3,2),oma=(c,0,3,0)

plot(density(out$betadraw[250,1,500:1000]),main=”Medium Fixed Interest”,xlab=” “,xlim=c(-15,15),ylim=c(0,.35))

plot(dropFirst(CAPMsmooth$s[,m+1:ml],

lty=c(“13,”6413”,”431313”,”B4”),plot.type=”s”,xlab=” “, ylab=”Beta”)

legend(“bottomright”,legend=colnames(y),bty=”n”,lty=c(“13”,”6413”,”431313”,”B4”),inset=0.05)

lines(lower,lty=2,lwd=2) #confidence interval

lines(upper,lty=2,lwd=2) #confidence interval

plot(ergMean(sqrt(gibbsV[1,1,-burn])),type=”l”,main=””,cex.lab=1.5,ylab=expression(sigma[1]),xlab=”MCMC iteration”) # ergodic mean

at=pretty(c(0,use),n=3)

at=at[at>=from]

axis(1,at=at-from,label=format(at))

legend(“topright”,legend=c(“mortgate rate”,”federal fund rate”),col=c(1,”darkgray”,Itly=c(1,2),bty=”n”)

lty=”longdash”, lty=”dotdash”, lty=solid # we can also define vector

plot.ts(outSr$s[-1,c(1,3)])

plot(ts(out$batch))

pacf(lrtsm1,lag.max=20)

acf(lrtsm1,lag.max=20)

qqnorm(cr,ylab=”Crime rate”, xlab=”Normal Scores”, main=”Normal probability plot”)

qqline(cr)

boxplot(DV~IV,data=data.ex1)

plot(table(x))

x=recordPlot(); replayPlot(x)

pie(rep(1,16),col=rainbow(16))

Install package through

install.packages("doSNOW")

update.packages() # to update

DLM functions

library (dlm)

investFilt =dlmFilter(invest,mod) #filtering

sdef=residuals(investFilt)$sd # standard deviation

lwr=investFilter$f+qnorm(0.25) \* sdef #confidence interval

upr=investFilter$f-qnorm(0.25) \* sdef #confidence interval

dlmModPoly(order=1,dV=parm[1], dW=parm[2])

dlmModReg(market)

dlmMLE(y,rep(0,2),build, lower=c(1e-6,0),hessian=TRUE) # define lower bound L-BFGS-B only accepts (max likelihood)

fit$convergence

unlist(build(fit$par)[c(“V”,”W”)])

avarLog=solve(fit$hessian)

dlmFilterDF(6,mod,DF=0.9) #0.9 is discount factor

tt=qt(0.95,df=2\*alpha)

lower=dropFirst(modFilt$m)-tt\*sqrt(Ctilde\*beta/alpha) #confidence interval

upper=dropFirst(modFilt$m)+tt\*sqrt(Ctilde\*beta/alpha) #confidence interval

dlmSmooth(modfilt) #smoothing

dlmBSample(filt) # sampling from dlm

MLE for variance of univDLM, type: “level” polynom, “trend” second order polynom, “BSM” second order polynom and seasonal component

dlmGibbsDIGt(y,mod=dlmPoly(2)+dlmModSeas(4),A\_y=1000,B\_y=10000,p=3,n.sample=MCMC,thin=2) #for t-distrib printed code, confidence interval not need to be constant width

dlmBSample(modfilt)

library(arms)

arms(mod$GG[3:4,3],ARfullCond,AR2support,1) #non standard distribution of AR given precession

dlmLL(y,mod)+sum(dnorm(u,sd=c(2,1)\*0.33,log=TRUE)

gdpGibbs(gdp,a.theta=1,b.theta=1000,n.sample=2050,thin=1,save.states=TRUE)

dlmModARMA: Auto Regressive moving average package of R

MSBVAR: package in R for Bayesian VAR models (Vector autoregressive models)

dlmForecast(expFilt,nAhead=12,sampleNew=10)

HWout = HoltWinter(lakeSup,gamma=0,beta=0) #holt winter filter

mod=dlmModSeas(frequency=4,dv=3.5,dw=c(4,2,0,0)) #DLM rep of seasonal component

mod1=dlmModTrig(s=12,dV=5.1118,dW=0)+dlmPoly(1,dV=0,dW=81307e-3) #specific periodic component

smoothTem1 = dlmSmooth(nottem,mod1)

Keywords

Sampler: Gibbs, marginal, hybrid to infer unobserved state

Conditional independance

Forward filtering backward sampling

Markov Kernel: invariant distribution

Thinning

Larger uncertainty

Posterior

Mean var

Precision independent

Bayes: random vector and not fixed like MLE => joint posterior

Sequential Monte Carlo: 91) One step ahead prediction density (2) Filtering density

Evolution of precision

Kalman Filter : New data comes in frequently, estimate error correction (recursion)

Independent inverse wishart prior: multivariate extension

Block diagonal : additive composition

Student t’s advantage: 2 degree of freedom: heavy tailed distribution, simple representation of scale mixture of normal distribution => still FFBS work conditionally

Structural break model over multiplicative model error structure

Proposal density

Factor model: extract common stochastic trend from multiple integrated time series

Basic idea: explain fluctuations of various markets or common latent factors that affect a set of economics or financial variables simultaneously

Common factor of two time variable: common stochastic trend, for identification set coefficient of first one to 1

Resampling methods: (1) multinomial (equal weight) (2) residual resampling

Filtering Recursion, particle Filter, up to date, online inference, problem is analytics

MCMC does not lend itself easily to sequential usage

Updating process: 91) draw conditional component (2) update its weight (3) normalize

Importance density, transition density, markovian

Std is less precise unless we take more particles, while accurate approximation of filtering mean

Summary: we build distribution empirically

Performance of particle filter depends on specific of the important transition densities (Liu and West)

Advantage of auxiliary particle filter: allows use of one step prior distribution without loosing efficiency

Solution: MCMC on weekend, particle filter ourly basis

Interested in filtering mean and variance

Series stationary

Models

Fit=glm(satellites~weight,family=poisson(link=log),data=crabs)

Coeff(fit)

Summary(fit)

Glm.nb(stell~weight,link=log)

Lm(d~vc)

Glm(d~vc,family=poisson())

MCMC=list(R=iter,keep=slice)Sim1=runiregGibbs(dt1,Prior1,MCMC)

glm(y~x,family=binomial())

Library(mcmc)

Arima(lrtsm,order=c(0,1,11)) #order P,D,Q

Chisq.test(data)

Glm(snoring~scores,family=binomial(link=logit))

Glm(y~~weight,family=binomial(link=probit),datacrabs)

Vgam(y~s9weight),family=binomialff(link=logit),data=crabs) #generalized additive

Vglm((formula=cbind(y2,y3,y4,y5,y1) ~size+factor(lake),family=multinomial,data=alligators)

Library(tree)

Fit=rpart(y~color+width,method=”class”)

Distance=dist(x,method”manhattan”)

Hclust(distances,”average”)

Postscript(file=”dendogram-election.ps”)

Plot(democlust,labels=states)

Graphics.off

Betabin(cbind(y,n-y)~group,random=~1,data=rats)

Kmeans(m3,k) # k=3 means 8 clusters

Library(fpc)

Plotcluster(m3,kmeansResult$cluster)

Library(Rgraphviz) # use for cluster association matrix

Plot(myTdm, terms=findFreqTerms(myTdm,lowfreq=1)[1:20],corThreshold=0)

Library(ggplot2) # graphic package to draw plots

Qplot(names(termFrequency),termFrequency,geom=”bar”)+coord\_flip() # draw horizontal bar plot)

Barplot(termFrequency,las=2) # draw vertical bar plot

Library(wordcloud)

wordFreq=sort(rowSums(m),decreasing=Ture)

greyLevels=gray((wordFreq+10)/(max(wordFreq)+10))

wordcloud(words=names(wordFreq),freq=WordFreq,min.freq=2,random.order=F,colors=grayLevels)

library(fpc) # clustering with partitioning around mediods

library(igraph)

gg=raph.adjacency(termMatrix,weighted=T,mode=”undirected”)

g=simplify(g)

V(g)$label = V(g)$name

V(g)$degree=degree(g)

Egam=(log(E(g)$weight)+.4)/max(log(E(g)$weight)+.4)

E(g)$color = rgb(.5,.5,0,egam)

Aov(x~y,data=datafile) # analysis of variance, also power.anova.test(…), and power.t.test(…)

t.test(x,g)

pairwise.t.test(x,g)

Save to Pdf

Pdf(‘BGOLSM.pdf’,width=11,height=8.5,pointsize=12,paper=’special’)

Slidify

library(slidify)

#setwd("~/sample/projects/")

author("Meisam Hejazi Nia")

slidify('index.Rmd')

library(knitr)

browseURL('index.html')

publish\_github(user,repo)

#HTML5 Deck Frameworks: io2012, html5slides, deck.js, dzslide, landslides, Slidy

#YM widget [mathjax] $x^2$ # latex math formatting

# Another tool with the same capability: R presentation

Shiny

install.packages("shiny")

libray(shiny)

#ui.R

library(shiny)

shinyUI(pageWithSidebar(

headerPanel("Data science FTW!"),

sidebarPanel(

h3('Sidebar text')

),

mainPanel(

h3('Main Panel text')

)

))

# server.R

library(shiny)

shinyServer(

function(input, output) {

}

)

#ui.R

shinyUI(pageWithSidebar(

headerPanel("Illustrating markup"),

sidebarPanel(

h1('Sidebar panel'),

h1('H1 text'),

h2('H2 Text'),

h3('H3 Text'),

h4('H4 Text')

),

mainPanel(

h3('Main Panel text'),

code('some code'),

p('some ordinary text')

)

))

#another sample of Ui.R

shinyUI(pageWithSidebar(

headerPanel("Illustrating inputs"),

sidebarPanel(

numericInput('id1', 'Numeric input, labeled id1', 0, min = 0, max = 10, step = 1),

checkboxGroupInput("id2", "Checkbox",

c("Value 1" = "1",

"Value 2" = "2",

"Value 3" = "3")),

dateInput("date", "Date:")

),

mainPanel(

)

))

#another Ui.R

mainPanel(

h3('Illustrating outputs'),

h4('You entered'),

verbatimTextOutput("oid1"),

h4('You entered'),

verbatimTextOutput("oid2"),

h4('You entered'),

verbatimTextOutput("odate")

)

#sample of server.R

shinyServer(

function(input, output) {

output$oid1 <- renderPrint({input$id1})

output$oid2 <- renderPrint({input$id2})

output$odate <- renderPrint({input$date})

}

)

#another UI example

shinyUI(

pageWithSidebar(

# Application title

headerPanel("Diabetes prediction"),

sidebarPanel(

numericInput('glucose', 'Glucose mg/dl', 90, min = 50, max = 200, step = 5),

submitButton('Submit')

),

mainPanel(

h3('Results of prediction'),

h4('You entered'),

verbatimTextOutput("inputValue"),

h4('Which resulted in a prediction of '),

verbatimTextOutput("prediction")

)

)

)

#another sample of server

diabetesRisk <- function(glucose) glucose / 200

shinyServer(

function(input, output) {

output$inputValue <- renderPrint({input$glucose})

output$prediction <- renderPrint({diabetesRisk(input$glucose)})

}

)

#another sample of Ui.R

shinyUI(pageWithSidebar(

headerPanel("Example plot"),

sidebarPanel(

sliderInput('mu', 'Guess at the mean',value = 70, min = 62, max = 74, step = 0.05,)

),

mainPanel(

plotOutput('newHist')

)

))

#another sample of Ui.R

library(UsingR)

data(galton)

shinyServer(

function(input, output) {

output$newHist <- renderPlot({

hist(galton$child, xlab='child height', col='lightblue',main='Histogram')

mu <- input$mu

lines(c(mu, mu), c(0, 200),col="red",lwd=5)

mse <- mean((galton$child - mu)^2)

text(63, 150, paste("mu = ", mu))

text(63, 140, paste("MSE = ", round(mse, 2)))

})

}

)

#sample of Ui.R

shinyUI(pageWithSidebar(

headerPanel("Hello Shiny!"),

sidebarPanel(

textInput(inputId="text1", label = "Input Text1"),

textInput(inputId="text2", label = "Input Text2")

),

mainPanel(

p('Output text1'),

textOutput('text1'),

p('Output text2'),

textOutput('text2'),

p('Output text3'),

textOutput('text3'),

p('Outside text'),

textOutput('text4'),

p('Inside text, but non-reactive'),

textOutput('text5')

)

))

#sample of server.R

library(shiny)

x <<- x + 1

y <<- 0

shinyServer(

function(input, output) {

y <<- y + 1

output$text1 <- renderText({input$text1})

output$text2 <- renderText({input$text2})

output$text3 <- renderText({as.numeric(input$text1)+1})

output$text4 <- renderText(y)

output$text5 <- renderText(x)

}

)

#another sample of server.R

shinyServer(

function(input, output) {

x <- reactive({as.numeric(input$text1)+100})

output$text1 <- renderText({x() })

output$text2 <- renderText({x() + as.numeric(input$text2)})

}

)

#sample of Ui.R

shinyUI(pageWithSidebar(

headerPanel("Hello Shiny!"),

sidebarPanel(

textInput(inputId="text1", label = "Input Text1"),

textInput(inputId="text2", label = "Input Text2"),

actionButton("goButton", "Go!")

),

mainPanel(

p('Output text1'),

textOutput('text1'),

p('Output text2'),

textOutput('text2'),

p('Output text3'),

textOutput('text3')

)

))

#sample of server.R

shinyServer(

function(input, output) {

output$text1 <- renderText({input$text1})

output$text2 <- renderText({input$text2})

output$text3 <- renderText({

input$goButton

isolate(paste(input$text1, input$text2))

})

}

)

# if (input$goButton == 1){ Conditional statements }

output$text3 <- renderText({

if (input$goButton == 0) "You have not pressed the button"

else if (input$goButton == 1) "you pressed it once"

else "OK quit pressing it"

})

rCharts

require(rCharts)

haireye = as.data.frame(HairEyeColor)

n1 <- nplot(Freq ~Hair, group = ‘Eye’, type = ‘multiBarChart’, data = subset (haireye, Sex == ‘Male’))

n1$save(‘fig/n1.html’, cdn=TRUE)

cat(‘<iframe src=”fig/n1.html” width= 100%, height = 600)</frame>’)

#Slidify interactive yaml ex\_widgets: {rCharts: [“libraries/nvd3”]}

Yaml ext\_widgets: {rCharts: [“libraries/highcharts”,”libraries/nvd3”,”libraries/morris”]}

# Example 1 facetted scatterplot

Names(iris)= gsub([\\.,””,names(iris))](file:///\\.,)

R1<- rPlot(Sepa1Length~ SepWidth | Species, data = iris, color = ‘Species’, type =’point’)

R1$save(‘fig/r1.html’, cdn=TRUE)

Cat(‘iframe src=”fig/r1.html” width=100%, height= 600></iframe>’)















Google Visualization

suppressPackageStartupMessages(library(googleVis))

## Warning: package 'googleVis' was built under R version 3.0.3

M <- gvisMotionChart(Fruits, "Fruit", "Year", options = list(width = 600, height = 400))

print(M, "chart")

Motion charts: gvisMotionChart

Interactive maps: gvisGeoChart

Interactive tables: gvisTable

Line charts: gvisLineChart

Bar charts: gvisColumnChart

Tree maps: gvisTreeMap

G <- gvisGeoChart(Exports, locationvar = "Country", colorvar = "Profit", options = list(width =

height = 400))

print(G, "chart")

G2 <- gvisGeoChart(Exports, locationvar = "Country", colorvar = "Profit", options = list(width =

height = 400, region = "150"))

print(G2, "chart")

df <- data.frame(label=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))

Line <- gvisLineChart(df, xvar="label", yvar=c("val1","val2"),

options=list(title="Hello World", legend="bottom",

titleTextStyle="{color:'red', fontSize:18}",

vAxis="{gridlines:{color:'red', count:3}}",

hAxis="{title:'My Label', titleTextStyle:{color:'blue'}}",

series="[{color:'green', targetAxisIndex: 0},

{color: 'blue',targetAxisIndex:1}]",

vAxes="[{title:'Value 1 (%)', format:'##,######%'},

{title:'Value 2 (\U00A3)'}]",

curveType="function", width=500, height=300

))

print(Line, "chart")

G <- gvisGeoChart(Exports, "Country", "Profit",options=list(width=200, height=100))

T1 <- gvisTable(Exports,options=list(width=200, height=270))

M <- gvisMotionChart(Fruits, "Fruit", "Year", options=list(width=400, height=370))

GT <- gvisMerge(G,T1, horizontal=FALSE)

GTM <- gvisMerge(GT, M, horizontal=TRUE,tableOptions="bgcolor=\"#CCCCCC\" cellspacing=10")

print(GTM, "chart")

M <- gvisMotionChart(Fruits, "Fruit", "Year", options = list(width = 600, height = 400))

print(M)

Manipulate

library(manipulate)

myHist <- function(mu){

hist(galton$child,col="blue",breaks=100)

lines(c(mu, mu), c(0, 150),col="red",lwd=5)

mse <- mean((galton$child - mu)^2)

text(63, 150, paste("mu = ", mu))

text(63, 140, paste("MSE = ", round(mse, 2)))

}

manipulate(myHist(mu), mu = slider(62, 74, step = 0.5))

RPackage

# new R project in R studio and select package

# 1. Modify the Description file

# for header above each function:

#’ Bulding a Model with …

#’ this function develops a prediction ….

#’ @param x a n x p matrix of n observations and p predictors …

#’ @return a vector of coefficients for …..

#’ @author Meisam …

#’ This function runs a univariate …

#’ @seealso \code{lm}

#’ @export

#’ @importFrom stats lm

# then build the package (we can config the Build output as well)

# it documents in the Rd file our headers

# we can run Check to make sure that everything is alright

Key directives

Also important

export("<function>")

import("<package>")

importFrom("<package>", "<function>")

exportClasses("<class>")

exportMethods("<generic>")

export("mvtsplot")

importFrom(graphics, "Axis")

import(splines)

export("read.polyfile", "write.polyfile")

importFrom(graphics, plot)

exportClasses("gpc.poly", "gpc.poly.nohole")

exportMethods("show", "get.bbox", "plot", "intersect”, "union”, "setdiff", "[", "append.poly", "scale.poly", "area.poly", "get.pts", "coerce", "tristrip", "triangulate")

\name{line}

\alias{line}

\alias{residuals.tukeyline}

\title{Robust Line Fitting}

\description{ Fit a line robustly as recommended in \emph{Exploratory Data Analysis}.}

\usage{

line(x, y)

}

\arguments{

\item{x, y}{the arguments can be any way of specifying x-y pairs. See

\code{\link{xy.coords}}.}

}

\details{

Cases with missing values are omitted.

Long vectors are not supported.

}

\value{

An object of class \code{"tukeyline"}.

Methods are available for the generic functions \code{coef},

\code{residuals}, \code{fitted}, and \code{print}.

}

\references{

Tukey, J. W. (1977).

\emph{Exploratory Data Analysis},

Reading Massachusetts: Addison-Wesley.

}

system("R CMD build newpackage")

system("R CMD check newpackage")

Classes

library(methods)

setClass()

new() # to create new class

# class and methods

?Classes

?Methods

?setClass

?setMethod,

?setGeneric

class(1)

## [1] "numeric"

class(TRUE)

## [1] "logical"

class(rnorm(100))

## [1] "numeric"

class(NA)

## [1] "logical"

class("foo")

## [1] "character"

x <- rnorm(100)

y <- x + rnorm(100)

fit <- lm(y ~ x) ## linear regression model

class(fit)

## [1] "lm"

mean

## function (x, ...)

## UseMethod("mean")

## <bytecode: 0x7facdb660ad0>

## <environment: namespace:base>

print

## function (x, ...)

## UseMethod("print")

## <bytecode: 0x7facd9ccfd58>

## <environment: namespace:base>

methods("mean")

## [1] mean.Date mean.default mean.difftime mean.POSIXct mean.POSIXlt

show

## standardGeneric for "show" defined from package "methods"

##

## function (object)

## standardGeneric("show")

## <bytecode: 0x7facdb8034d8>

## <environment: 0x7facdb779868>

## Methods may be defined for arguments: object

## Use showMethods("show") for currently available ones.

## (This generic function excludes non-simple inheritance; see ?setIs)

showMethods("show")

## Function: show (package methods)

## object="ANY"

## object="classGeneratorFunction"

## object="classRepresentation"

## object="envRefClass"

## object="function"

## (inherited from: object="ANY")

## object="genericFunction"

## object="genericFunctionWithTrace"

## object="MethodDefinition"

## object="MethodDefinitionWithTrace"

## object="MethodSelectionReport"

## object="MethodWithNext"

## object="MethodWithNextWithTrace"

## object="namedList"

## object="ObjectsWithPackage"

## object="oldClass"

## object="refClassRepresentation"

## object="refMethodDef"

## object="refObjectGenerator"

set.seed(2)

x <- rnorm(100)

mean(x)

## [1] -0.0307

head(getS3method("mean", "default"), 10)

##

## 1 function (x, trim = 0, na.rm = FALSE, ...)

## 2 {

## 3 if (!is.numeric(x) && !is.complex(x) && !is.logical(x)) {

## 4 warning("argument is not numeric or logical: returning NA")

## 5 return(NA\_real\_)

## 6 }

## 7 if (na.rm)

## 8 x <- x[!is.na(x)]

## 9 if (!is.numeric(trim) || length(trim) != 1L)

## 10 stop("'trim' must be numeric of length one")

tail(getS3method("mean", "default"), 10)

##

## 15 if (any(is.na(x)))

## 16 return(NA\_real\_)

## 17 if (trim >= 0.5)

## 18 return(stats::median(x, na.rm = FALSE))

## 19 lo <- floor(n \* trim) + 1

## 20 hi <- n + 1 - lo

## 21 x <- sort.int(x, partial = unique(c(lo, hi)))[lo:hi]

## 22 }

## 23 .Internal(mean(x))

## 24 }

set.seed(3)

df <- data.frame(x = rnorm(100), y = 1:100)

sapply(df, mean)

## x y

## 0.01104 50.50000

set.seed(10)

x <- rnorm(100)

plot(x)

set.seed(10)

x <- rnorm(100)

x <- as.ts(x) ## Convert to a time series object

plot(x)

library(methods)

setClass("polygon", representation(x = "numeric",y = "numeric"))

setMethod("plot", "polygon",

function(x, y, ...) {

plot(x@x, x@y, type = "n", ...)

xp <- c(x@x, x@x[1])

yp <- c(x@y, x@y[1])

lines(xp, yp)

})

library(methods)

showMethods("plot")

## Function: plot (package graphics)

## x="ANY"

## x="polygon"

p <- new("polygon", x = c(1, 2, 3, 4), y = c(1, 2, 3, 1))

plot(p)

getS3method(<generic>, <class>)

getMethod(<generic>, <signature>)

yhat

#Create back end computing engine and then deploy on web so that others use it

Install.packages(“yhatr”)

Library(yhatr)

Model.require <- function(){ # loads R dependencies

}

Model.transform <- function(){ # transform data before feeding to model

}

Model.predict <- function(){ # is your prediction

}

# create an account on yhat

Yhat.config <- c(username = …, apikey = …., env=….)

Yhat.deploy(“ “)

Yhat.predict(“pollutant”,df)

Curl –X Post –H “content-type: application/json” –user meisam….@gmail.com:90d2…. –data ‘{“lon”:-76.61, ….}’ http://sandbox.yhathq.com/....