HMKANG@UMICH.EDU

MODULE 1 / UNIT 10

WRITING & SHARING R PACKAGES

Today

- How can we make an R package?
- How can we share my R package through GitHub?
- What else do I need to make an R package that contains Rcpp code?
- How can I document my R package conveniently?

Why R package?

- R packaging system has been one of the key factors in the overall success of the R project.

- Packages allow for easy, transparent, and cross-platform extensions of the R base system.

- R packages are a comfortable means to maintain collection of R functions and datasets

Pure R implementation of Viterbi algorithm

```
viterbiHMM <- function(obs, transMtx, emisMtx, pi) {
    T <- length(obs)
    ns <- nrow(transMtx)
    delta <- matrix(NA,ns,T)
    phi <- matrix(NA,ns,T)
    delta[,1] <- pi * emisMtx[,obs[1]]
    for(t in seq(2,T,1)) { ## works when T is 2 or greater
        for(i in 1:ns) {   ## no need to use another loop. Why?
            v <- delta[,t-1] * transMtx[,i]
            delta[i,t] <- max(v) * emisMtx[i,obs[t]]
            phi[i,t] <- which.max(v)
        }
    }
    path <- vector(length=T)
    ml <- max(delta[,T])
    path[T] <- which.max(delta[,T])
    for(t in seq(T-1,1,-1)) {
        path[t] = phi[path[t+1],t+1]
    }
    return(list(ml=ml,path=path,delta=delta,phi=phi))
}
```

R implementation of forward-backward algorithm

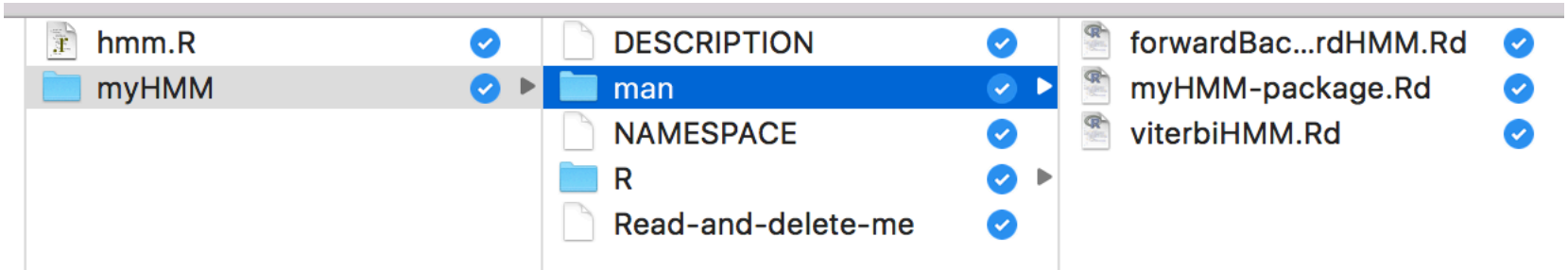
```
forwardBackwardHMM <- function(obs, transMtx, emisMtx, pi) {
    T <- length(obs)
    ns <- nrow(transMtx)
    alpha <- matrix(0, ns, T)
    beta <- matrix(0, ns, T)
    alpha[,1] <- pi * emisMtx[,obs[1]] ## same as viterbi
    for(t in seq(2,T,1)) {
        for(i in 1:ns) { ## alpha can be represented as a matrix operation
            alpha[i,t] = sum(alpha[,t-1] * transMtx[,i]) * emisMtx[i,obs[t]]
        }
    }
    beta[,T] <- 1
    for(t in seq(T-1,1,-1)) {
        for(i in 1:ns) { ## beta can be represented as a matrix operation
            beta[i,t] = sum(beta[,t+1] * transMtx[i,] * emisMtx[,obs[t+1]])
        }
    }
    condProb <- alpha * beta
    condProb <- condProb / matrix(colSums(condProb),ns,T,byrow=TRUE)
    return(condProb)
}
```

A simple & generic way to create an R package

1. Create an R code that contains everything
2. Run package.skeleton()
3. Edit help files
4. Edit the DESCRIPTION file
5. Check and build
6. Install

Running package.skeleton()

```
> package.skeleton("myHMM",code_files="hmm.R")
Creating directories ...
Creating DESCRIPTION ...
Creating NAMESPACE ...
Creating Read-and-delete-me ...
Copying code files ...
Making help files ...
Done.
Further steps are described in './myHMM/Read-and-delete-me'.
>
```



Edit help files under `man/` directory

- Running `package.skeleton()` will create many help files
- One file for the package : `[package-name]-package.Rd`
- One file for each function : `[function-name].Rd`
- Writing a comprehensive help file is very important when sharing / publishing your work with others.

An example help file for the package

```
\name{myHMM-package}
\alias{myHMM-package}
\alias{myHMM}
\docType{package}
\title{\packageTitle{myHMM}}
\description{\packageDescription{myHMM}}
\details{
The DESCRIPTION file:
\packageDESCRIPTION{myHMM}
\packageIndices{myHMM}
This package implements a generic HMM algorithm,
including Viterbi and forward-backward algorithms.
}
\author{
\packageAuthor{myHMM}
Maintainer: \packageMaintainer{myHMM}
}
\examples{
forwardBackwardHMM(c(1,3,2),matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE),
matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE),c(0.7,0.3))
}
```

An example help file for a function

```
\name{viterbiHMM}
\alias{viterbiHMM}
\title{
Viterbi algorithm for a generic HMM
}
\description{
  This function implements a Viterbi algorithm, given
  observed outcomes, transition and emission matrices,
  and initial probabilities.
}
\usage{
viterbiHMM(obs, transMtx, emisMtx, pi)
}
\arguments{
  \item{obs}{ size T observed outcomes as integer vectors from 1 to m }
  \item{transMtx}{ n x n matrix of transition from row to column}
  \item{emisMtx}{  n x m matrix of emission of observed data (column)
    given states (row)  }
  \item{pi}{ size n vector of initial state probabilities }
}
\details{ Dynamic programming is used for both algorithms. }
... (to be continued to the next page)
```

An example help file for a function

```
... (continued from the previous page)
\value{
\item{ml}{A numeric value describing the maximum likelihood}
\item{path}{A Viterbi path that provides the maximum likelihood}
}
\references{
Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models.
ieee assp magazine, 3(1), 4-16.
}
\author{Hyun Min Kang (hmkang@umich.edu)}
\note{ This function was developed as a part of BIOSTAT615 lecture
material }
\seealso{ \code{\link{forwardBackwardHMM}}}
\examples{
obs <- c(1,3,2)
A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
pi <- c(0.7,0.3)
viterbiHMM(obs, A, B, pi)
}
\keyword{ ~HMM }% use one of  RShowDoc("KEYWORDS")
\keyword{ ~Viterbi }% __ONLY ONE__ keyword per line
```

An example DESCRIPTION file

```
Package: myHMM
Type: Package
Title: Viterbi and forward-backward algorithms for
generic HMM
Version: 1.0
Date: 2017-10-28
Author: Hyun Min Kang
Maintainer: Hyun Min Kang <hmkang@umich.edu>
Description: This package contains a basic algorithm for
generic HMM
License: GPL-3
```

Running `R CMD check myHMM`

```
$ R CMD check myHMM
* using log directory '/blah/615_1_10/myHMM.Rcheck'
* using R version 3.4.2 (2017-09-28)
* using platform: x86_64-apple-darwin13.4.0 (64-bit)
* using session charset: UTF-8
* checking for file 'myHMM/DESCRIPTION' ... OK
* checking extension type ... Package
* this is package 'myHMM' version '1.0'
* checking package namespace information ... OK
... (omitted)
* checking examples ... OK
* checking PDF version of manual ... OK
* DONE
Status: 1 NOTE
See
  '/blah/615_1_10/myHMM.Rcheck/00check.log'
for details.
```

Running `R CMD build myHMM`

```
$ R CMD build myHMM
* checking for file ‘myHMM/DESCRIPTION’ ... OK
* preparing ‘myHMM’:
* checking DESCRIPTION meta-information ... OK
* installing the package to process help pages
* saving partial Rd database
* checking for LF line-endings in source and make files
and shell scripts
* checking for empty or unneeded directories
* building ‘myHMM_1.0.tar.gz’
```

Installing the package

```
> install.packages("myHMM_1.0.tar.gz",repos=NULL)
* installing *source* package ‘myHMM’ ...
** R
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (myHMM)
>
```

Example use of the package

```
> library(myHMM)
> help("myHMM")
> help(viterbiHMM)
> obs <- c(1,3,2)
> A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
> B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
> pi <- c(0.7,0.3)
> viterbiHMM(obs, A, B, pi)
$ml
[1] 0.0133056

$path
[1] 1 2 2

$delta
      [,1]     [,2]      [,3]
[1,] 0.616 0.009856 0.0014784
[2,] 0.030 0.036960 0.0133056

$phi
     [,1] [,2] [,3]
[1,]   NA    1    2
[2,]   NA    1    2
```

Example help pages

myHMM-package {myHMM} R Documentation

Viterbi and forward-backward algorithms for generic HMM

Description

This package contains a basic algorithm for generic HMM.

Details

The DESCRIPTION file:

Package:	myHMM
Type:	Package
Title:	Viterbi and forward-backward algorithms for generic HMM
Version:	1.0
Date:	2017-10-28
Author:	Hyun Min Kang
Maintainer:	Hyun Min Kang <hmkang@umich.edu>
Description:	This package contains a basic algorithm for generic HMM.
License:	GPL-3

Index of help topics:

```
forwardBackwardHMM      Forward-backward algorithm for a generic HMM
myHMM-package           Viterbi and forward-backward algorithms for
                        generic HMM
viterbiHMM              Viterbi algorithm for a generic HMM
```

This package implements a generic HMM algorithm, including Viterbi and forward-backward algorithms.

viterbiHMM {myHMM} R Documentation

Viterbi algorithm for a generic HMM

Description

This function implements a Viterbi algorithm, given observed outcomes, transition and emission matrices, and initial probabilities.

Usage

```
viterbiHMM(obs, transMtx, emisMtx, pi)
```

Arguments

`obs`	size T observed outcomes as integer vectors from 0 to m-1
`transMtx`	n x n matrix of transition from row to column
`emisMtx`	n x m matrix of emission of observed data (column) given states (row)
`pi`	size n vector of initial state probabilities

Details

Dynamic programming is used for both algorithms.

Value

`ml`	A numeric value describing the maximum likelihood
`path`	A Viterbi path that provides the maximum likelihood

Note

This function was developed as a part of BIOSTAT615 lecture material

Author(s)

Example manual pdf (in `.Rcheck` folder)

Package ‘myHMM’

October 28, 2017

Type Package

Title Viterbi and forward-backward algorithms for generic HMM

Version 1.0

Date 2017-10-28

Author Hyun Min Kang

Maintainer Hyun Min Kang <hmkang@umich.edu>

Description This package contains a basic algorithm for generic HMM.

License GPL-3

R topics documented:

---

myHMM-package *Viterbi and forward-backward algorithms for generic HMM*

Using `devtools/roxygen2` to create an R/Rcpp package

Install `devtools` and `roxygen2`

1. Run `devtools::create("pkgName",rstudio=FALSE)`
2. Add license `devtools::use_{gpl3|mit}_license("pkgName")`
3. Enable roxygen2 using `devtools::use_package_doc("pkgName")`
4. Enable Rcpp using `devtools::use_rcpp("pkgName")`
5. Add your R and C++ files in `pkgName/R` and `pkgName/src`
6. Run `devtools::document("pkgName")`
7. Edit `pkgName/R/pkgName-package.r`
8. Run `devtools::check("pkgName")` to test and check
9. Run `devtools::build("pkgName")` to build package

Using `roxygen2` for better documentation

```
#' Viterbi algorithm implemented in R
#'
#' This function implements a Viterbi algorithm, given observed outcomes,
transition and emission matrices, and initial probabilities.
#'
#' @param obs size T integer vector of observed outcomes, values from 1 to m
#' @param transMtx n x n transition matrix from row to column
#' @param emisMtx n x m emission matrix of observed data (column) given states
(row)
#' @param pi size n vector of initial state probabilities
#' @return a list containing maximum likelihood (ml) and the Viterbi path (path)
#' @examples
#' obs <- c(1,3,2)
#' A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
#' B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
#' pi <- c(0.7,0.3)
#' viterbiHMMr(obs, A, B, pi)
#' @export
viterbiHMMr <- function(obs, transMtx, emisMtx, pi) { ... }
```

roxygen2 documentation works for Rcpp, too

```
//' Forward-backward algorithm implemented in Rcpp
//'
//' This function implements a forward-backward algorithm of a generic HMM, given
observed outcomes, transition and emission matrices, and initial probabilities.
//'
//' @param obs size T integer vector of observed outcomes, values from 1 to m
//' @param transMtx n x n transition matrix from row to column
//' @param emisMtx n x m emission matrix of observed data (column) given states
(row)
//' @param pi size n vector of initial state probabilities
//' @return matrix of conditional probability of each state given the observed
outcomes
//' @examples
//' obs <- c(1,3,2)
//' A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
//' B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
//' pi <- c(0.7,0.3)
//' forwardBackwardHMMc(obs, A, B, pi)
//' @export
// [[Rcpp::export]]
NumericMatrix forwardBackwardHMMc(IntegerVector obs, NumericMatrix transMtx,
NumericMatrix emisMtx, NumericVector pi) {...
```

What the roxygen2 documentation looks like..

viterbiHMMc {hmm615} R Documentation

Viterbi algorithm implemented in Rcpp

Description

This function implements a Vitervi algorithm in generic HMM, given observed outcomes, transition and emission matrices, and initial probabilities.

Usage

```
viterbiHMMc(obs, transMtx, emisMtx, pi)
```

Arguments

`obs`	size T integer vector of observed outcomes, values from 1 to m
`transMtx`	n x n transition matrix from row to column
`emisMtx`	n x m emission matrix of observed data (column) given states (row)
`pi`	size n vector of initial state probabilities

Value

a matrix containing conditional probability of each possible states given the observed outcomes

Examples

```
obs <- c(1,3,2)
A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
pi <- c(0.7,0.3)
viterbiHMMc(obs, A, B, pi)
```

[Package *hmm615* version 0.0.0.9000 Index]

viterbiHMMr {hmm615} R Documentation

Viterbi algorithm implemented in R

Description

This function implements a Viterbi algorithm, given observed outcomes, transition and emission matrices, and initial probabilities.

Usage

```
viterbiHMMr(obs, transMtx, emisMtx, pi)
```

Arguments

`obs`	size T integer vector of observed outcomes, values from 1 to m
`transMtx`	n x n transition matrix from row to column
`emisMtx`	n x m emission matrix of observed data (column) given states (row)
`pi`	size n vector of initial state probabilities

Value

a list containing maximum likelihood (ml) and the Viterbi path (path)

Examples

```
obs <- c(1,3,2)
A <- matrix(c(0.8,0.2,0.4,0.6),2,2,byrow=TRUE)
B <- matrix(c(0.88,0.10,0.02,0.1,0.6,0.3),2,3,byrow=TRUE)
pi <- c(0.7,0.3)
viterbiHMMr(obs, A, B, pi)
```

Modify [pkgname]-package.r

```
#' hmm615.
#'
#' @name hmm615
#' @docType package
#' @author Your Name Here
#' @import Rcpp
#' @importFrom Rcpp evalCpp
#' @useDynLib hmm615
#' @name hmm615
NULL
```

Sharing your package with others via GitHub

- **You can share/publish your package via a repository**
- **Centralized repositories (CRAN, Bioconductor, ..)**
  - Becomes "official", and everyone has access to your package.
  - Submit your package, and the admin decides whether to accept/reject.
    - The admin double checks for the sanity of the package for you.
  - You do NOT have a full control of your own package once submitted.
- **De-centralized (user-owned) repositories (GitHub, Bitbucket)**
  - Better for "working" project, and works fine with "official" release too
    - Also works with "private" repository with password protection
  - You submit your package under your own GitHub or Bitbucket account
  - You have a full control of the package.
    - You also need to double check the sanity of the code yourself
    - ... and interact with users directly.

Steps to share your package on GitHub

(Adapted from http://kbroman.org/pkg_primer/pages/github.html)

Create your own GitHub account (if you don't have one)

1. **Change to your package directory**

```
$ cd hmm615/
```

2. **Initialize the repository**

```
$ git init
```

3. **Add files and commit everything**

```
$ git add .
$ git commit -m "Your note goes here"
```

4. **Create a new repository at https://github.com/new**
5. **Connect your repository to your GitHub repository**

```
$ git remote add origin https://github.com/username/reponame
```

6. Push your package to GitHub

```
$ git push -u origin master
```

Installing packages from GitHub

- **If you have devtools installed, you should be able to install a package in R console.**

```
> devtools::install_github("username/reponame")
```

- **For example, try to install my package at**

```
> devtools::install_github("hyunminkang/hmm615")
```

And remove it using `remove.packages("hmm615")`

Summary

- R package is an effective way to share your methods and tools with others.
- Writing R package is straightforward but requires additional work beside coding.
- Using **`devtools`** and **`roxygen2`** can make writing R/Rcpp packages in a more convenient and organized way.
- Putting your package on GitHub is one of the most effective ways to share your R/Rcpp code with others.