Sweave and Survival Analysis

Lecture 11

Nicholas Christian

BIOST 2094 Spring 2011

Outline

1. `xtable`
2. Sweave
3. Survival Analysis

xtable

- The `xtable` package contains a function `xtable()` that will convert an `R` object to an `xtable` object which can be printed as a LaTeX table.
- That is, create the code for a LaTeX table with all of the `'&'` and `\\`
- Basic syntax,

```
xtable(x, caption=NULL, label=NULL, align=NULL, digits=NULL)
```

`x`	An `R` object of some class where an `xtable` methods exists
`caption`	Table's caption
`label`	Table's label
`align`	Character vector giving the column alignment, can also include "|" for vertical lines
`digits`	Vector giving the number of digits to display in the corresponding columns

- `print.xtable()` has additional arguments for controlling the format of the table. Such as placement of horizontal lines; caption placement; table placement on page, etc.
- `xtable()` is a generic function and additional methods can be written, see the documentation

xtable - Example

```
# Install and load package xtable
install.packages("xtable")
library(xtable)

# Basic example
x <- matrix(rnorm(15), nrow=3, ncol=5,
      dimnames=list(paste("row", 1:3), paste("col", 1:5)))

# Create latex table
tex.x1 <- xtable(x, caption="Basic Example 1")

# Change alignment and number of digits
tex.x2 <- xtable(x, align=rep("c", 6),digits=1,caption="Basic Example 2")

# Can use xtable for any object where an xtable method exisits
methods(xtable)

# Use xtable for a linear models lm object
y <- rnorm(100)
x <- rexp(100)
fit <- lm(y~x)

xtable(fit, caption="Linear Regression Results")
```

What is Sweave?

- `Sweave` enables you to automatically generate reports by mixing `R` code with LaTeX files.
- The basic idea is to include `R` code in the LaTeX document where the final document only contains the output of the statistical analysis.
- Allows reports to be automatically updated when the data or analysis changes
- Reproducible research, all of the analysis (tables, graphs, etc) are performed when writing the report (technically when `Sweave()` is called to produced the LaTeX file)

Installing Sweave

- `Sweave` is part of the base package in `R`
- For LaTeX to be able to use `Sweave`, LaTeX needs access to the file `Sweave.sty`
- The quick and dirty way to install,
  1. Copy `Sweave.sty` from

     `C:\Program Files\R\R-2.12.0\share\texmf\tex\latex`
  2. Paste `Sweave.sty` into

     `C:\Program Files (x86)\MiKTeX 2.8\tex\latex\sweave`

     (will need to create the folder sweave)
  3. Refresh the file name database under, MiKTeX Options

Sweave

- To use `Sweave`,
  1. Write your LaTeX file and `R` code in a `Sweave` source file with extension `.rnw`
  2. Next call `Sweave()` to convert your `.rnw` file to a `.tex` file
  3. Compile the `.tex` file just like you would any other `.tex` file to get your `.pdf` file
- The function `Stangle()` is used to extract all of the `R` source code from the `.rnw` file

Sweave Source File

- A `Sweave` file contains *code chunks* embedded in a LaTeX document.
- `'<<...>>='` Marks the start of an `R` code chunk
- `'@'` Marks the end of an `R` code chunk
- To include `R` output within a line of text use `\Sexpr{}`. `Sweave` will replace the S/R expression with the corresponding output
- *All code chunks are evaluated by `R` in the order they appear in the document.*

Sweave Source File

- Within the angled brackets `'<<...>>='` we can specify options that control how the code and corresponding output appear in the final document

`echo=false`	Do not include `R` code
`results=verbatim`	Default, print output as is
`results=hide`	Do not include `R` output
`results=tex`	Results are regular TEX code and should be evaluated
`fig=true`	Indicates that the code chunk produces a figure
`width`	Optional argument to control figure width
`height`	Optional argument to control figure height

- To set a default option use `\SweaveOpts{}`, so for example, `\SweaveOpts{echo=false}` will suppress all `R` code

Sweave - Example

- Produce a simple report that summarizes the results of analyzing the `mtcars` dataset.
- The `system()` function executes operating system commands

```
Sweave("sweaveExample.rnw")                 # Create tex file
system("pdflatex sweaveExample.tex")        # Create PDF
system("open sweaveExample.pdf")            # View PDF
Stangle("sweaveExample.rnw")                # Extract R code
```

Survival Analysis

- The `survival` package comes with `R` but still needs to be loaded before you can use the functions.
- For an overview of other `R` packages available for survival analysis see, `http://cran.rproject.org/web/views/Survival.html`
- Almost all survival analysis functions use a survival object created by `Surv()` that consist of the event time and event indicator
- Basic syntax,

```
Surv(time, event)
```

`time`	For right censored data, follow-up time
`event`	Event indicator default is 1=event and 0=censor, for a different event value use `'=='`

Example - Surv()

```
library(survival)

head(aml) # Survival in patients with Acute Myelogenous Leukemia

# Event=1, Censor=0
x <- Surv(aml$time, aml$status)
unclass(x)

# Event=0, Censor=1
y <- Surv(aml$time, aml$status==0)
is.Surv(y)
```

Survival Analysis

- Functions useful for survival analysis

`survfit.formula()`	Kaplan-Meier estimate
`survfit.coxph()`	Predicted survival curve from a Cox model
`survdiff()`	Log-rank and Harrington and Fleming weighted log-rank test; $w(t) = \hat{S}(t)^{\rho}$, $\rho = 0$ for log-rank test
`survreg()`	Parametric Proportional Hazards Model
`coxph()`	Cox proportional hazards model
`cox.zph()`	Tests the proportional hazards assumption
`summary()`	Summarize results
`anova.coxph()`	Analysis of deviance table for one or more Cox models
`confint()`	Confidence intervals of parameter estimates
`drop1()`	Test each factor individually
`step()`	Stepwise algorithm using the AIC
`plot.survfit()`	Plot a survival curve

- Remember the function `methods()` is very useful for finding the methods that correspond to a generic function or the methods for a particular class

Example - Survival Analysis

```
head(aml) # Survival in patients with Acute Myelogenous Leukemia

# Kaplan-Meier estimate of the survival function
fit <- survfit(Surv(time, status) ~ x, data = aml)
summary(fit)
plot(fit, col=c("blue", "red"))
plot(fit, col=c("blue", "red"), mark.time=FALSE)

# Extract results
names(fit)
fit$n.risk

# Log-rank test
lr <- survdiff(Surv(time, status) ~ x, data = aml, rho=0)
lr

# Weighted log-rank test
wt.lr <- survdiff(Surv(time, status) ~ x, data = aml, rho=1)
wt.lr
```

Example - Survival Analysis

```
# Chemotherapy treatment for colon cancer
# Two records for each person one for recurrence and one for death
head(colon)

# Fit Cox model using just recurrence
# Use the breslow method to handle ties
fit <- coxph(Surv(time, status)~rx+sex, data=colon, subset=(etype==1),
                method="breslow")
summary(fit) # Return hazard ratios and
             # global test that all covariates are 0
anova(fit)   # Sequential tests
drop1(fit, ~., test="Chisq")
confint(fit) # Confidence interval for the parameters

# Estimated survival function with 95% point-wise confidence interval
plot(survfit(fit), xlab="Time", ylab="Survival")
```

Example - Survival Analysis

```
# Check proportional hazards assumption
# Tests of the proportional hazards assumption for each covariate,
# by correlating the corresponding set of scaled Schoenfeld residuals
# with a suitable transformation of time; as well as a global test
cox.zph(fit)

# Plot Schoenfeld residuals against transformed time for each covariate
# Smoother should be close to horizontal
par(mfrow=c(2,2))
plot(cox.zph(fit))

# Use residuals.coxph() to calculate different kinds of residuals
# which can be used for other types of diagnostics
residuals(fit, "martingale")

# Model building - need to use cases with complete data
fit1 <- coxph(Surv(time, status)~rx+sex, data=na.omit(colon),
          subset=(etype==1), method="breslow")
fit2 <- coxph(Surv(time, status)~rx+sex+nodes+perfor,
          data=na.omit(colon), subset=(etype==1), method="breslow")
anova(fit2, fit1) # Compare two models

step(fit2, direction="both") # Stepwise selection using AIC
```