Volpe R Course: Session 6, Expanded Statistical Toolbox

Instructors: Don Fisher, Dan Flynn, Jessie Yang
Course webpage: http://bit.ly/volpeR

5/25/2017

Overview

Last session: Data manipulation

- Merging data frames
- Basic manipulations
- Plotting spatial data

This session:

Expanded Statistical Toolbox

- Hierarchical modeling
- Repeated measures and time series

Project Presentations

Expanded Statistical Toolbox

To introduce hierarchical modeling, we will work with the fuel consumption and temperature data introduced in the last session. The temperature data have been updated to provide State-level average temperatures.

The data file to download is on the course Google Drive, http://bit.ly/volpeR:

- `ModelingDat.RData`

This is composed of the temperature and fuel consumption data

- `Avg T by State by Month.csv`
- `Net_generation_for_all_fuels_(utility-scale)_monthly.csv`

These have already been prepared following the steps outlined in the previous session, and results saved in the `.RData` file.

```
load('data/ModelingDat.RData')
ls()
```

```
## [1] "avg.temp"  "fuel.temp" "net.melt"
```

Hierarchical modeling

Hierarchical models are also called *mixed-effects models*, because they mix *fixed* and *random* effects.

- **Fixed effects** are typically predictors you are most interested in. If these are categorical, these are variables usually only have a few different levels, all of which are represented in you data (*all of few*). An example would be two levels of an experimental treatment.
- **Random effects** are predictors which may be very important, but which you would like to generalize across. Usually categorical, and you may only have a few of many possible levels in your data (*few of many*). An example would be several States.

In hierarchical modeling, you essentially analyze your fixed effects within each level of the random effects, then pool your results across all the random effects.

Lots more detail!

- http://lme4.r-forge.r-project.org/IMMwR/lrgprt.pdf
- https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf

Hierarchical modeling

The syntax in the `lmer` package for mixed-effect modeling is similar to what you have seen using `lm()`, but now you will have to add at least one random effect. The random effects take the two following common forms:

- `(1|RandomFactor)` Gives a different intercept for each level of the factor
- `(FixedContinuous|RandomFactor)` Gives a different slope and intercept for each level of the factor.

Additional forms you may use include:

- `(1|RandomFactor1/RandomFactor2)` Is a nested model, with a different intercept for each level of RandomFactor2. These are pooled to form a different intercept for each higher level, of RandomFactor1. For example, counties within states.
- `(FixedContinuousTime|RandomFactorSubject)` For repeated measures on a subject.

Hierarchical modeling

Here is an example, looking at our fuel consumption data by region.

```r
library(lme4) # install.packages(lme4)
# Region is random effect, avgt is fixed effect
m1 <- lmer(NetFuel ~ avgt + (1|Region), data = fuel.temp)
fixef(m1)
```

```
## (Intercept)        avgt
## 32633.61710    84.83312
```

```r
ranef(m1)
```

```
## $Region
##                     (Intercept)
## East.North.Central   15721.169
## East.South.Central   -6305.983
## Middle.Atlantic      -2022.976
## Mountain             -6684.872
## New.England         -26249.044
## Pacific.Contiguous   -6807.743
## South.Atlantic       27382.571
## West.North.Central  -10400.567
## West.South.Central   15367.446
```

Hierarchical modeling

This is the analogous to fitting a model for each region, then pooling across regions, for example doing this for each region separately:

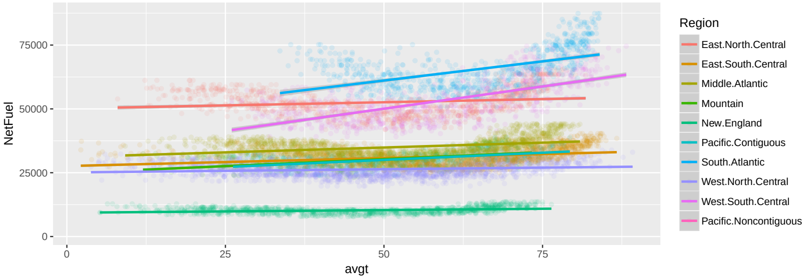
```
m1.1 <- lm(NetFuel ~ avgt, data = fuel.temp[fuel.temp$Region == "New.England",]
summary(m1.1)
```

```
##
## Call:
## lm(formula = NetFuel ~ avgt, data = fuel.temp[fuel.temp$Region ==
##     "New.England", ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2600.1  -958.4    62.9   771.5  3351.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9328.151    108.749  85.777   <2e-16 ***
## avgt          20.660      2.217   9.321   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1191 on 958 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.08314,    Adjusted R-squared:  0.08219
## F-statistic: 86.87 on 1 and 958 DF,  p-value: < 2.2e-16
```

Hierarchical modeling

Visually, you can think of the hierarchical model almost like this:

```r
library(ggplot2)
ggplot(fuel.temp) + aes(avgt, NetFuel, color = Region) +
  geom_point(alpha = 0.1) + geom_smooth(method = "lm")
```



Repeated measures modeling

We can use the hierarchical modeling framework to look at repeated measures or time series.

Example: subjects are deprived of sleep and are measured for their reaction times at increasing intervals (see `?sleepstudy`). How much does sleep deprivation matter? We want to track each subject separately, then pool the results to come up with general conclusions.

```r
rm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
summary(rm1)
fixef(rm1); ranef(rm1)
```

Each additional day increases reaction time by 10.5 +/- 1.5 ms.

You may be asking yourself, "but what about my p-values?". This is a normal reaction when encountering mixed-effects models! One solution:

```r
library(sjPlot) # install.packages(sjPlot) first
sjt.lmer(rm1)
```

```
## Computing p-values via Kenward-Roger approximation. Use `p.kr = FALSE` if co
```

Repeated measures modeling

Exercise:

Try to write models that estimate the trends in 1. net fuel consumption by month, and 2. average temperature by year, taking into account regional differences.

```r
# Modeling net fuel by Region and month
m2 <- lmer(NetFuel ~ avgt + mo + (1+mo|Region), data = fuel.temp)

# Modeling average temperature by Region and year (warning message)
m3 <- lmer(avgt ~ year + (1+year|Region), data = fuel.temp)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

Report Presentations!

Show the report you generated for this workshop (using `.Rmd`) and explain the story it tells.

Saving your work

After doing all this work, you will want to save the results. You can save R objects in your current workspace to file type called `.RData`. Then you can load these R objects back in at a later session, without having to carry out this preparation work.

For example, to save all objects in the current workspace:

```
save(list = ls(), file = "My_Prepped_Data.RData")
```

and load it back in using

```
load("My_Prepped_Data.RData")
```

To save specific R objects, you will name them in the list as strings, such as

```
save(list = c('fuel.temp', 'avg.temp'), file = "My_Prepped_Data.RData")
```

You can also consider making a script that does all this preparation work, and save it as a file called something like `Analysis Prep.R`. Then, in your data analysis script or `.Rmd` file, you can automatically run that script by inserting the command: `source('Analysis Prep.R')`.

Merging data for spatial analysis

Another application of the merged data is to plot it spatially.

```r
library(maps)
map('state')
# Plot avg temp
statenames  <- map('state', plot = FALSE)$names
statematches <- match(statenames, tolower(avg.temp$state))
tempmatches <- avg.temp$avgt[statematches]

colorpal <- heat.colors(100)

temp.colors <- colorpal[round(tempmatches)]

map('state',
    col = temp.colors,
    fill = T)
```