DSCI353-353m-453: Class 05a-p Bootstrap for Mixed Effects Models

Profs: R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

TAs: W. Oltjen, K. Hernandez, M. Li, M. Li, D. Colvin

13 February, 2023

Contents

5.1.2.1 Class Readings, Assignments, Syllabus Topics

5.1.2.1.1 Reading, Lab Exercises, SemProjects

- Readings:
  - For today: DLwR1, DL06,07
  - For next class: DLwR2
- Laboratory Exercises:
  - LE2 is due next Tuesday
  - LE3 is given out next Tuesday
- Office Hours: (Class Canvas Calendar for Zoom Link)
  - Wednesdays @ 4:00 PM to 5:00 PM
  - Saturdays @ 3:00 PM to 4:00 PM
  - **Office Hours are on Zoom, and recorded**
- Semester Projects
  - Office Hours for SemProjs: Mondays at 4pm on Zoom
  - DSCI 453 Students Biweekly Updates Due
    - Update # is Due ** **
  - DSCI 453 Students
    - Next Report Out # is Due ** **

- All DSCI 353/353M/453, E1453/2453 Students:
  - Peer Grading of Report Out #1 is Due ** **
- Exams
  - MidTerm: **Thursday March 9th**, in class or remote, 11:30 - 12:45 PM
  - Final: **Thursday May 4th**, 2023, 12:00PM - 3:00PM, Nord 356 or remote

5.1.2.1.2 Textbooks

- Text Books for DSCI353/353M/453
  - R4DS: Wickham: R for Data Science
  - ISLR: Intro to Statistical Learning with R, 2nd Ed.
  - DLwR: Deep Learning with R, Chollet, Allaire,
  - DLGB: Deep Learning, Goodfellow, Bengio, Courville
- Magazine Articles about Deep Learning
  - DL1 to DL12 are "Deep Learning" articles in 3-readings/2-articles/
- Books from DSCI351/351M/451
  - Peng: R Programming for Data Science
  - Peng: Exploratory Data Analysis with R
  - Open Intro Stats, v4
  - R4DS: Wickham: R for Data Science

5.1.2.1.3 Tidyverse Cheatsheets, Functions and Reading Your Code

- Look at the Tidyverse Cheatsheet
  - **Tidyverse For Beginners Cheatsheet**
    - In the Git/20s-dsci353-353m-453-prof/3-readings/3-CheatSheets/ folder
  - **Data Wrangling with dplyr and tidyr Cheatsheet**

  Tidyverse Functions & Conventions

  - The pipe operator `%>%`
  - Use `dplyr::filter()` to subset data row-wise.
  - Use `dplyr::arrange()` to sort the observations in a data frame
  - Use `dplyr::mutate()` to update or create new columns of a data frame
  - Use `dplyr::summarize()` to turn many observations into a single data point
  - Use `dplyr::arrange()` to change the ordering of the rows of a data frame
  - Use `dplyr::select()` to choose variables from a tibble,
    - keeps only variables you mention
  - Use `dplyr::rename()` keeps all the variables and renames variables
    - rename(iris, petal_length = Petal.Length)
  - These can be combined using `dplyr::group_by()`
    - which lets you perform operations "by group".
  - The `%in%` matches conditions provided by a vector using the c() function
  - The **forcats** package has tidyverse functions
    - for factors (categorical variables)
  - The **readr** package has tidyverse functions
    - to read_…, melt_… col_…, parse_… data and objects

Reading Your Code: Whenever you see

- The assignment operator `<-`, think **"gets"**
- The pipe operator, `%>%`, think **"then"**

5.1.2.1.4 Syllabus

Day:Date	Foundation	Practicum	Readings(optional)	Due(optional)
w01a:Tu:1/17/23	Markov Cluster	R, Rstudio IDE, Git		(LE0)
w01b:Th:1/19/23	Stat. Learning, Approach	Bash, Git, Class Repo	ISLR1,2 (R4DS-1-3)	
w02a:Tu:1/24/23	Lin. Regr. Bias-Var.	SemProjs; Regr. Ovrvw	ISLR3,(R4DS-4-6)	**(LE0:Due)** LE1
w02b:Th:1/26/23	Train/Test, Bias vs. Vari.	Tidyverse Review	DL01 DL02 (R4DS-7,8)	
w02Pr:Fr:1/27/23	**ADD DROP**	**DEADLINE**		**453 Update 1**
w03a:Tu:1/31/23	Logistic Regr. Classif	Pred. Analytics, Regr.	DL03,ISLR4	
w03b:Th:2/2/23	LDA/QDA	ggPlot2, Code Expect.	DL04, DL05	**LE1:Due**, LE2
w03:Sa:2/4/23				**LE1:Due**
w04a:Tu:2/7/23	Resample Cross-Valid.	ggplot	ISLR5	
w04b:Th:2/9/23	DL, ML Overview	Multilevel Mod.	ISLR6 (R4DS9-16)	
w04Pr:Fr:2/10/23				**453 Update 2**
w05a:Tu:2/14/23	Bootstrap	Bootstrap Mixed Effects	DL2R1, DL06,07	**LE2:Due**, LE3
w05b:Th:2/16/23	Subset Selec., Shrink.	Mixed Effects	DLwR2	
w05Pr:Fr:2/17/23				**453 Rep. Out 1**
w06a:Tu:2/21/23	Mod. Selec.	ML with NNs	DLwR3	
w06b:Th:2/23/23	Beyond Linear Modls	Feature Select., Caret	ISLR7 (R4DS22-25)	**LE3:Due**, LE4
w06Pr:Fr:2/24/23				**453 Update 3**
w07a:Tu:2/28/23	Dec. Trees, Rand. Forest.	Tidy Modeling	ISLR8, DL08,09	
w07b:Th:3/2/23	MidTerm Review, SVM	SVM, SVR, ROC	ISLR9 (R4DS26-30)	**Peer Review 1**
w08a:Tu:3/7/23	R-Keras/TensorFlow2		ISLR10	
w08b:Th:3/9/23	**MIDTERM EXAM**		DL10,11	**LE4:Due** LE5
w08Pr:Fr:3/10/23				**453 Update 4**
Tu:3/14/23	**SPRING**	**BREAK**	ISLR10	
Th:3/16/23	**SPRING**	**BREAK**	DL12,13	
w09a:Tu:3/21/23	Deep Learning	TF2 Keras Intro	Pocket Perceptron	ISLR10, DLR3
w09b:Th:3/23/23	Computer Vision, CNN	CNN w/TF2, Overfit	DLR4	
w09Pr:Fr:3/24/23				**453 Rep. Out 2**
w10a:Tu:3/28/23	Deep Learn Intro	NN Types	DLR5	
w10b:Th:3/30/23	DL CNN,RNN ImageNet	NN Types, CNN wTF2	Hinton ImageNet	
w10Pr:Fr:3/31/23				**453 Upd.5 & PrRev 2**
Sa:4/1/23				**LE5:Due** LE6
w11a:Tu:4/4/23	Fitting NNs	AUC,Prec,Recall Fruit		
w11b:Th:4/6/23	NLP, Graphs & ML		LeCun DL Rev. 2015	
w12a:Tu:4/11/23	Graphs & ML	NLP with sequences	DLR6	
w12b:Th:4/13/23	NLP w attention	Graph Repr Proc Wrkflw		**LE6:Due** LE7
w13a:Tu:4/18/23	DL Frameworks	Explaining DL w Lime		
w13b:Th:4/20/23	Linux Distros XGBoost	Explain Preds	Deep Dream	
w13Pr:Fr:4/21/23				**453 Rep. Out 3 Due**
w14a:Tu:4/25/23	Tranformers			
w14b:Th:4/27/23	Final Exam Review	Torch NN & DeepLearn		**LE7:Due**
w14Pr:Fr:4/28/23				**Peer Rev 3 Due**
	FINAL EXAM	**Th. 5/4/23, 12-3pm**	Nord 356 & Zoom	
	453 Final PDF Report	**Fr. 4/29, 11:59pm**		

Figure 1: DSCI351-351M-451 Syllabus

5.1.2.2 Bootstrap applied to Mixed Effects Models

- Bootstrap is one of the most famous resampling techniques
  - and is very useful way to get confidence intervals
  - in situations where classical approach (t- or z- tests) would fail.

5.1.2.2.1 What is Bootstrap?

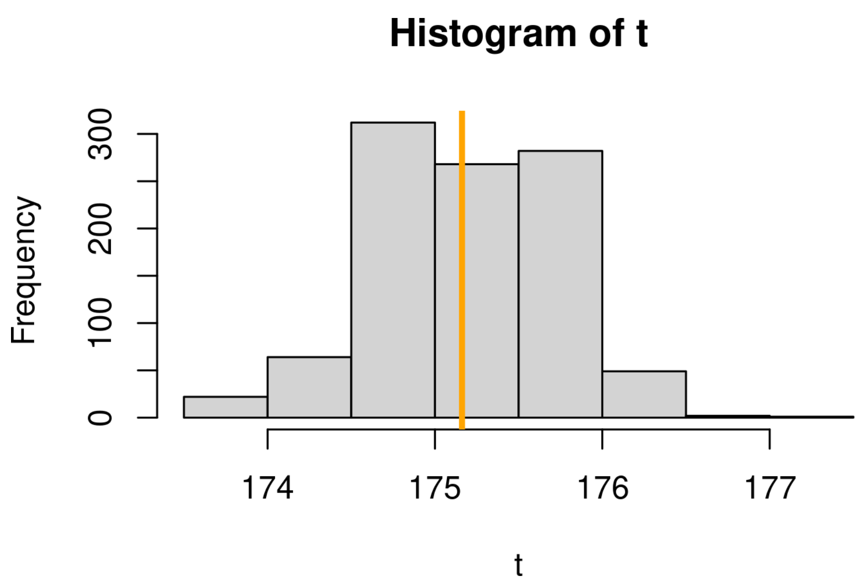
- Lets show a simple bootstrap example
  - using the height of 100 person in the population.

```
set.seed(20151101)
height <- rnorm(100, 175, 6)
```

Now we will resample

- with replacement 1000 times
- and compute the median:

```
t0 <- median(height)
t <-
  sapply(1:1000, function(x)
    median(sample(
      x = height, size = 100,
      replace = TRUE
    )))
hist(t)
abline(v = t0, col = "orange", lwd = 3)
```



And this is the histogram that we get.

And that's it, this is the essence of bootstrap:

- resampling the observed data with replacement
  - and computing the statistic of interest
  - (here the median)
  - many times on the resampled data
- to get a distribution of the statistic of interest.
- This distribution of the statistic of interest can then be used to compute,
  - for example, confidence intervals.

5.1.2.2.2 When to use Bootstrap?

- Bootstrap is used to enable inference on the statistic of interest
  - when the true distribution of this statistic is unknown.
- For example in a linear model the parameters of interest
  - have a known distribution
  - from which standard errors and formal tests can be performed.
- On the other hand for some statistics
  - (median, differences between two models …),
  - if the analyst do not want to spend time writing down equations,
- bootstrapping might be a great approach
  - to get standard errors and confidence intervals
  - from the bootstrapped distribution.

5.1.2.2.3 When will Bootstrap Fail?

- There are some situations where bootstrapped will fail:
  - (i) the statistic of interest is at the edge of the parameter space
    - (like minimum or maximum),
    - the bootstrapped distribution does not converge
      - (as the number of bootstrap sample increase to infinity)
      - to the true distribution of the statistic.

  (ii) sample size is small,
  - bootstrapping will not increase the power of statistical tests.
  - If you sample too few data to detect an effect of interest,
  - using bootstrap will not magically solve your problem
  - even worse the bootstrap approach will perform less well than others.

5.1.2.2.4 How many bootstrap samples

- As much as possible will be the answer.
- Note that in here we used low numbers to speed up the computations for class

5.1.2.3 Non-parametric and parametric bootstrap using the `boot` package

- The boot library in R is very convenient
  - to easily compute confidence intervals from bootstrap samples.

Non-parametric bootstrap with boot:

```
library(boot)
b1 <- boot(
  data = height,
  statistic = function(x, i)
    median(x[i]),
  R = 1000
)
boot.ci(b1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b1)
##
## Intervals :
## Level      Normal              Basic
```

```
## 95%   (174.0, 176.3 )   (174.1, 176.4 )
##
## Level     Percentile            BCa
## 95%   (173.9, 176.2 )   (173.9, 176.2 )
## Calculations and Intervals on Original Scale
```

Parametric bootstrap with boot:

```
x <- runif(100, -2, 2)
y <- rnorm(100, 1 + 2 * x, 1)
dat <- data.frame(x = x, y = y)
```

```
m <- lm(y ~ x)
```

5.1.2.3.1 Lets do a simple example with a linear model

We are interested in getting

- the confidence intervals
  - for the coefficient of the model:

```
foo <- function(out) {
  m <- lm(y ~ x, out)
  coef(m)
}
```

The function rgen

- generate new response vector from the model:

```
rgen <- function(dat, mle) {
  out <- dat
  out$y <- unlist(simulate(mle))
  return(out)
}
```

Now generate a 1000 bootstrap sample

```
b2 <- boot(
  dat,
  foo,
  R = 1000,
  sim = "parametric",
  ran.gen = rgen,
  mle = m
)
```

Compute the confidence intervals

- for the two coefficients:
  - index = 1 and index = 2

```
boot.ci(b2, type = "perc", index = 1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b2, type = "perc", index = 1)
##
## Intervals :
```

```
## Level     Percentile
## 95%   ( 1.039,  1.423 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(b2, type = "perc", index = 2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b2, type = "perc", index = 2)
##
## Intervals :
## Level     Percentile
## 95%   ( 1.832,  2.164 )
## Calculations and Intervals on Original Scale
```

In the non-parametric case

- `boot` expects two arguments in the function returning the statistic of interest:
  - the first one is the object from which to compute the statistic
  - and the second is a vector of
    * index (`i`),
    * frequencies (`f`)
    * or weight (`w`)
    * defining the bootstrap sample.
- In our example, `boot` will generate a series of indices (named `i`)
  - with replacement
  - and these will be used to subset the original height vector.

In the parametric case the function returning the statistic(s) of interest

- only needs one argument: the original dataset.
- We then need to supply another function (`ran.gen`)
  - describing how to generate the new data,
  - it needs to return an object of the same form as the original dataset.
- This random data generating function need two arguments:
  - the first one is the original dataset
  - and the second one contain maximum likelihood estimate
    * for the parameter of interest,
    * basically a model object.
  - The new dataset generated by the ran.gen function
    * will then be passed to the statistics function
    * to compute the bootstrapped value for the statistic of interest.
  - It is then straightforward to get
    * the confidence intervals for the statistic
    * using `boot.ci`.

5.1.2.4 Bootstrap applied to mixed-effect models

- Mixed-effect models are rather complex
  - and the distributions or numbers of degrees of freedom
  - of various outputs from them (like parameters …)
  - is not known analytically.
- Which is why the author of the `lme4` package recommends
  - the use of bootstrap to get
    * confidence intervals around the model parameters,

        - the predicted values
      - but also to get p-values from likelihood ratio tests.

In 3-readings/4-MatSci-And-SemProjReadings

- There is a good example of fixed- and mixed-effects modeling
  - In Gok et al. - 2017 - Predictive models of poly(ethylene-terephthalate) .pdf

```
library(lme4)
```

5.1.2.4.1 A simple random intercept model:

```
## Loading required package: Matrix
```

```
dat <- data.frame(x = runif(100, -2, 2), ind = gl(n = 10, k = 10))
dat$y <-
  1 + 2 * dat$x + rnorm(10, 0, 1.2)[dat$ind] + rnorm(100, 0, 0.5)
m <- lme4::lmer(y ~ x + (1 | ind), dat)
```

Get the bootstrapped confidence intervals

- for the model parameters:

```
b_par <- bootMer(x = m, FUN = fixef, nsim = 200)
boot.ci(b_par, type = "perc", index = 1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b_par, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (-0.3984,  1.1845 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

```
boot.ci(b_par, type = "perc", index = 2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b_par, type = "perc", index = 2)
##
## Intervals :
## Level     Percentile
## 95%   ( 1.834,  2.014 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

Or alternatively:

```
confint(
  m,
  parm = c(3, 4),
  method = "boot",
```

```
  nsim = 200,
  boot.type = "perc"
)
```

```
## Computing bootstrap confidence intervals ...

##                   2.5 %   97.5 %
## (Intercept) -0.211244 1.359807
## x            1.841195 2.041312
```

Get the bootstrapped confidence intervals

- around the regression curves:

```
new_dat <- data.frame(x = seq(-2, 2, length = 20))
mm <- model.matrix(~ x, data = new_dat)
predFun <- function(.)
  mm %*% fixef(.)
bb <- bootMer(m, FUN = predFun, nsim = 200) #do this 200 times
```

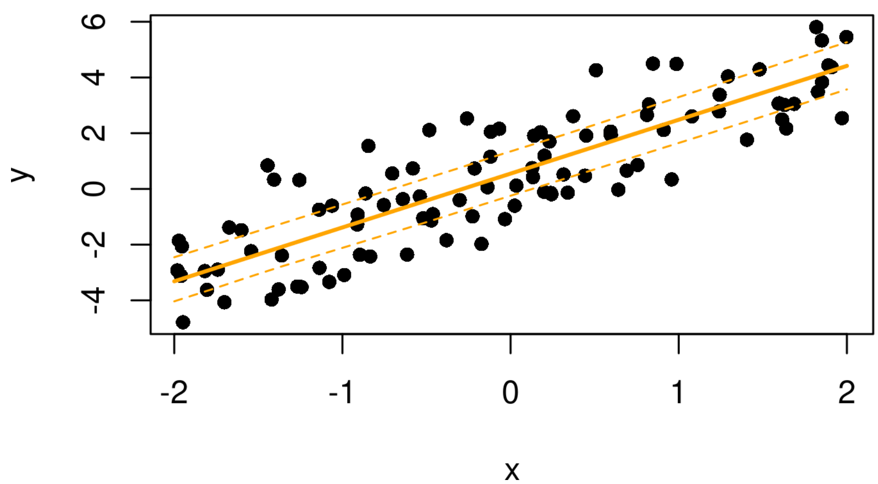
As we did this 200 times

- the 95% CI will be bordered
- by the 5th and 195th value:

```
bb_se <- apply(bb$t, 2, function(x)
  x[order(x)][c(5, 195)])
new_dat$LC <- bb_se[1,]
new_dat$UC <- bb_se[2,]
new_dat$pred <- predict(m, newdata = new_dat, re.form =  ~ 0)
```

Plot the results

```
plot(y ~ x, dat, pch = 16)
lines(pred ~ x, new_dat, lwd = 2, col = "orange")
lines(LC ~ x, new_dat, lty = 2, col = "orange")
lines(UC ~ x, new_dat, lty = 2, col = "orange")
```



Finally get bootstrapped p-values

- from the likelihood ratio test between two models.

Drawing confidence intervals around the regression curves

- is tricky due to the random effect estimated values
  - which comes with there own uncertainty
  - (have a look at `dotplot(ranef(m,condVar=TRUE))` to see it).

Bootstrapping is an efficient way to take these uncertainties into account

- since the random deviates are re-computed for each draw.

Finally getting p-values for the effect of a fixed-effect term

- can be done using a parametric bootstrap approach as described here
- and implemented in the function `PBmodcomp`
  - from the `pbkrtest` package.
- In the output of `PBmodcomp`
  - the bootstrapped p-values is in the PBtest line,
  - the LRT line reports the standard p-value
    - assuming a chi-square distribution
    - for the LRT value.

```
library(pbkrtest)
?PBmodcomp
```

5.1.2.5 Links

- Lionel Hertzog, "Introduction to bootstrap with applications to mixed-effect models", Nov. 25, 2015.
- T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, 2nd ed. New York: Springer-Verlag, 2009 [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn/.
- Bradley Efron and Robert J Tibshirani, An introduction to the bootstrap, vol. 57. Chapman & Hall/CRC Monographs on Statistics and Applied Probability, 1993 [Online]. Available: https://www.routledge.com/An-Introduction-to-the-Bootstrap/Efron-Tibshirani/p/book/9780412042317
- D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting Linear Mixed-Effects Models using lme4," Journal of Statistical Software, vol. 67, no. 1, pp. 1–48, 2015, doi: 10.18637/jss.v067.i01.