Wrap-up

Marcel Neunhoeffer & Sebastian Sternberg

November 30, 2017 | December 4, 2017

Contents

Today:

1. Throwback Thursday
2. Bugfixing: sapply and for loops
3. Visualizing Interactions in Logit Models or Count Models

Goals for Today:

- See what you learned this semester
- Bugfixing
- How to visualize Interactions in Logit Models

```
rm(list = ls())
```

1.Throwback Thursday

Remember your first two lab exercises?

Exercise I:

- Create three objects:
  1. "my.lucky.number" it should contain your lucky number
  2. "my.firstname" it should contain your firstname
  3. "my.lastname" it should contain your lastname

After you created the objects, call them separately. Don't forget to add comments to your code.

```
my.lucky.number <- 5

my.first.name <- "Sebastian"

my.last.name <- "Sternberg"


my.lucky.number
```

```
## [1] 5
```

```
my.first.name
```

```
## [1] "Sebastian"
```

```
my.last.name
```

```
## [1] "Sternberg"
```

Exercise II: Generate a vector with two sequences of your choice.

a) give your vector a meaningful name
b) calculate the mean value of your vector and save the result as exercise.mean
c) calculate the variance of your vector and save the result as exercise.var
d) don't forget to add comments

```
vec.ex <- c(seq(1, 10, 2), seq(45, 98, 2.5))

mean(vec.ex)
```

```
## [1] 58.98148
```

```
exercise.mean <- mean(vec.ex)

var(vec.ex)
```

```
## [1] 902.1439
```

```
exercise.var <- var(vec.ex)
```

Remember your first homework?

It took quite some time back then...

How long do you think would it take you now?

So we think you are well prepared to master the Data Essay! It's really amazing what you learnt this semester.

So let's add two more things to the things you learnt...

2. Bugfixing: sapply and for loops

The Bug

Bugs are a part of coding. Every now and then something goes wrong. And then for a while no one notices. If someone does discover your bug you feel horrible. And wonder how that could happen. Then you start

Quantitative Methods in Political Science
University of Mannheim
Fall 2017
Gschwend, Neunhoeffer & Sternberg

Homework 1

Due: 14 September 2017

In our first lab-session we solved similar tasks in `R`, so you should be familiar with all the concepts.

Your write-up should include graphs, tables and full sentence answers to the questions. Please, also include all the `R`-code you wrote to answer a question in the write-up. Additionally, attach your whole script in the Appendix. Choose an appropriate form to present your answers, graphs, tables and code. Also remember to structure your code according to Google's R Style Guide and don't forget to add comments.

Please indicate how much each group member contributed to the final write up. (Indicate it in percent next to your names. E.g. Name 1 (20 %), Name 2 (30 %) & Name 3 (50 %).)

Your write up should not exceed 15 pages including all.

If you are doing the exercises on your own computer, you should first install `R`. Follow the instructions on `http://www.r-project.org/`. Additionally, you should also install Rstudio (`http://www.rstudio.org/download/desktop`) which includes a lot of helpful features that allow you to work more efficiently with `R`.

1. Download the US Presidential Elections data set `uspresidentialelections.dta` from the course ILIAS site. Load the data set in `R`.
2. Describe the dataset. What variables does it contain? How many observations are there? What time span does it cover?
3. Compute measures of central tendency and variability of the variables `vote` and `growth` using `R`. Use the numerical measures of central tendency and variability discussed in class. Describe them in your own words and make a nice table. Plot the distribution of both variables using a boxplot and histogram. Make sure to make your plots as nice-looking as possible. Especially, include a title and label the axes.
4. Make a bar plot of the party affiliation of incumbent presidential candidates.
5. During the presidential campaign in 1992, Bill Clinton's campaign coined the phrase "It's the economy, stupid!" Let's investigate the relationship between the economy and electoral success. Generate a nice-looking scatterplot of economic growth and vote share. Label the data points with the year of the election. Describe the pattern that you see in your own words.

Figure 1: Homework 1.

debugging...

In the last lab we introduced the following code (no excuse, but that bit of code was part of the lab for at least 4 years...).

```
exp.auto <- sapply(lambda1, function(x) mean(rnbinom(1000, size = theta, mu = lambda1[x])))
```

We said it would do the same thing as the following for loop.

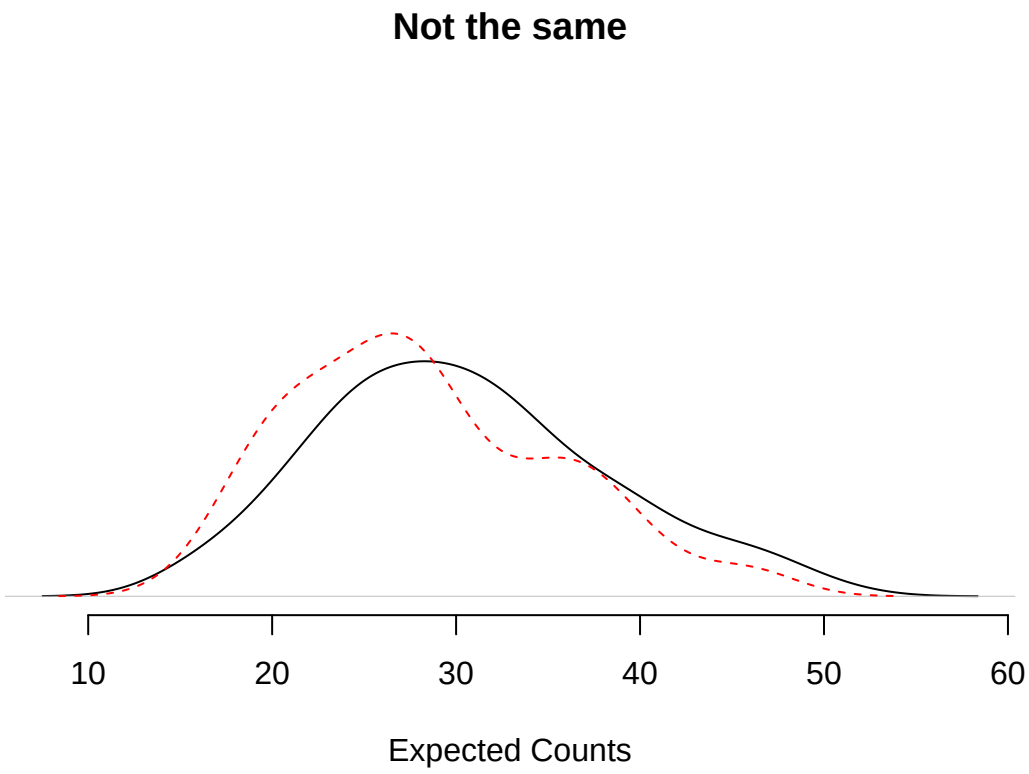
```
mean.auto <- rep(NA, length(lambda1))

for(i in 1:length(lambda1)){

mean.auto[i] <- mean(rnbinom(1000, size = theta, mu = lambda1[i]))

}
```

Spoiler... It doesn't.

Let's check that with an example.

See the wrong sapply in action.

```
theta <- 0.02

set.seed(123)
lambda1 <- runif(100, 22, 34)


set.seed(123)
exp.auto <- sapply(lambda1, function(x) mean(rnbinom(1000, size = theta, mu = lambda1[x])))

mean.auto <- rep(NA, length(lambda1))

set.seed(123)
for(i in 1:length(lambda1)){

mean.auto[i] <- mean(rnbinom(1000, size = theta, mu = lambda1[i]))

}


plot(density(exp.auto),
     ylim = c(0, 0.1),
     main = "Not the same",
     xlab = "Expected Counts",
     yaxt = "n", ylab = "",
     bty = "n")
lines(density(mean.auto), lty = "dashed", col = "red")
```



Unfortunately, we can all see now that the results are not the same.

Trying to understand what it does

```
#?sapply
```

So what went wrong?

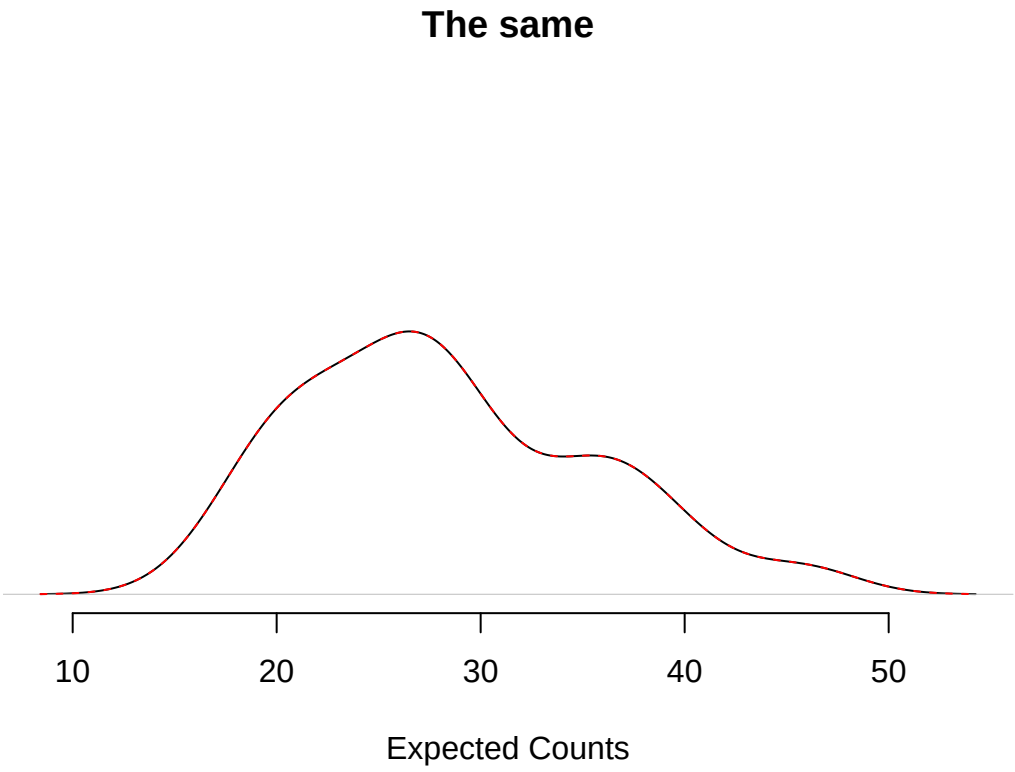
```
seq <- 4:10

res.for <- rep(NA, length(seq))

for(i in 1:length(seq)){
res.for[i] <-   seq[i] + 2
}

res.sapplyWrong <- sapply(seq, function(x) seq[x] + 2)

res.sapplyRight <- sapply(seq, function(x) x + 2)


cbind(res.for,
res.sapplyWrong,
res.sapplyRight)
```

```
##      res.for res.sapplyWrong res.sapplyRight
## [1,]       6               9               6
## [2,]       7              10               7
## [3,]       8              11               8
```

```
## [4,]       9              12               9
## [5,]      10              NA              10
## [6,]      11              NA              11
## [7,]      12              NA              12
```

How to do it right?

```r
theta <- 0.02

set.seed(123)
lambda1 <- runif(100, 22, 34)


set.seed(123)
exp.auto <- sapply(lambda1, function(x) mean(rnbinom(1000, size = theta, mu = x)))

mean.auto <- rep(NA, length(lambda1))

set.seed(123)
for(i in 1:length(lambda1)){

mean.auto[i] <- mean(rnbinom(1000, size = theta, mu = lambda1[i]))

}


plot(density(exp.auto),
     ylim = c(0, 0.1),
     main = "The same",
     xlab = "Expected Counts",
     yaxt = "n", ylab = "",
     bty = "n")
lines(density(mean.auto), lty = "dashed", col = "red")
```



Now we have correct code for the last lab. And sapply and the for loop finally do the same thing!

(We included it already in the solution file for the last lab. . .)

3. Visualizing Interactions in Logit Models

Unfortunately, the intuition about interaction terms form linear models does not extend to non-linear models.

However, we have one really powerful tool in our toolbox that can help us to look at and interpret interactions in any model. Simulation!

As some of you had problems with the interaction effect in a logit model in Homework 9, we will look at an interaction in a logit model here.

The same logic applies to any other non-linear model.

For the last time, we will start with some fake data.

```
# Population Size

n <- 100000

# True Parameters

beta0 <- -2
beta1 <- 0.3
beta2 <- 0.5
beta3 <- -0.2
```

```r
# Independent Variables

X1 <- rnorm(n, 20, 10)
X2 <- rnorm(n, 1, 0.5)

# Our Systematic component

mu <- beta0 + beta1*X1 + beta2*X2 + beta3*X1*X2

# Now we generate p via the logit response function

p <- (exp(mu)) / (1 + exp(mu))


# As we observe only 0 or 1 we need to put p in a binomial distribution

Y <- rbinom(n, 1, p)

# That's our full population.

pop <- data.frame(Y, X1, X2)

# Let's work with a sample from our population

data <- pop[sample(1:10000, 1000), ]


# Now we can run the model...

m1 <- glm(Y ~ X1 + X2 + X1*X2, data = data, family = binomial(link = logit))

summary(m1)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2 + X1 * X2, family = binomial(link = logit),
##     data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.77946  -0.77560   0.08492   0.73699   2.85333
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.47580    0.44922  -5.511 3.56e-08 ***
## X1           0.32581    0.02947  11.054  < 2e-16 ***
## X2           0.88222    0.37300   2.365    0.018 *
## X1:X2       -0.22482    0.02379  -9.450  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.99  on 999  degrees of freedom
```

```
## Residual deviance:  930.02  on 996  degrees of freedom
## AIC: 938.02
##
## Number of Fisher Scoring iterations: 6
```

So now we can't make sense of these coefficients...

To see what those coefficients mean, we use simulation.

A. Simulate Parameters - Remember the Steps?

Steps for Simulating Parameters:

- 1. Get the coefficients from the regression (gamma.hat)
- 2. Get the variance-covariance matrix (V.hat)
- 3. Set up a multivariate normal distribution N(gamma.hat,V.hat)
- 4. Draw from the distribution nsim times

```
gamma.hat <- coef(m1)
V.hat <- vcov(m1)

library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.3.2
```

```
S <- mvrnorm(1000, gamma.hat, V.hat)
```

B. Calculate Expected Values

Set up interesting scenarios. That's the important step here!

```
X1.sim <- seq(min(X1), max(X1), length.out = 100)

X2.lo <- quantile(X2, 0.25)
X2.hi <- quantile(X2, 0.75)

scenario.X2lo <- cbind(1, X1.sim, X2.lo, X1.sim*X2.lo)
scenario.X2hi <- cbind(1, X1.sim, X2.hi, X1.sim*X2.hi)

Xbeta.lo <- S %*% t(scenario.X2lo)
Xbeta.hi <- S %*% t(scenario.X2hi)

dim(Xbeta.lo)
```

```
## [1] 1000  100
```

```
dim(Xbeta.hi)
```

```
## [1] 1000  100
```

```
# To get expected values for p, we need to plug in the Xbeta values into
# the response function to get simulatd probabilities

p.sim.lo <- (exp(Xbeta.lo))/ (1 + exp(Xbeta.lo))
```

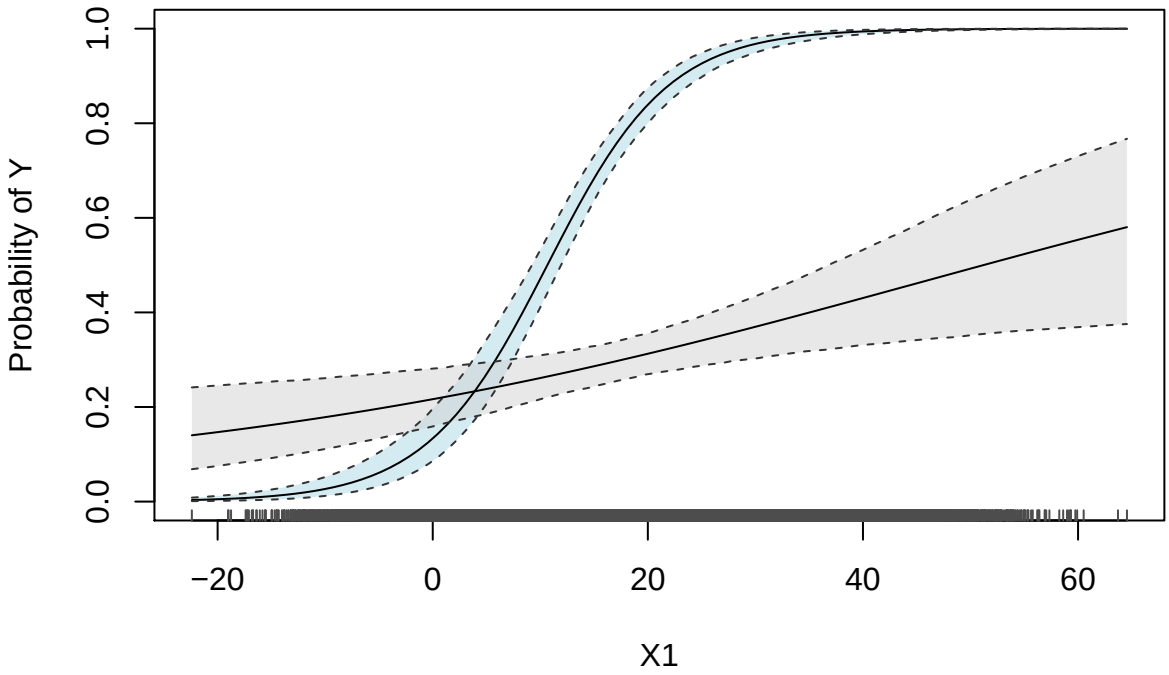
```r
p.sim.hi <- (exp(Xbeta.hi))/ (1 + exp(Xbeta.hi))

dim(p.sim.lo)
```

```
## [1] 1000  100
```

```r
dim(p.sim.hi)
```

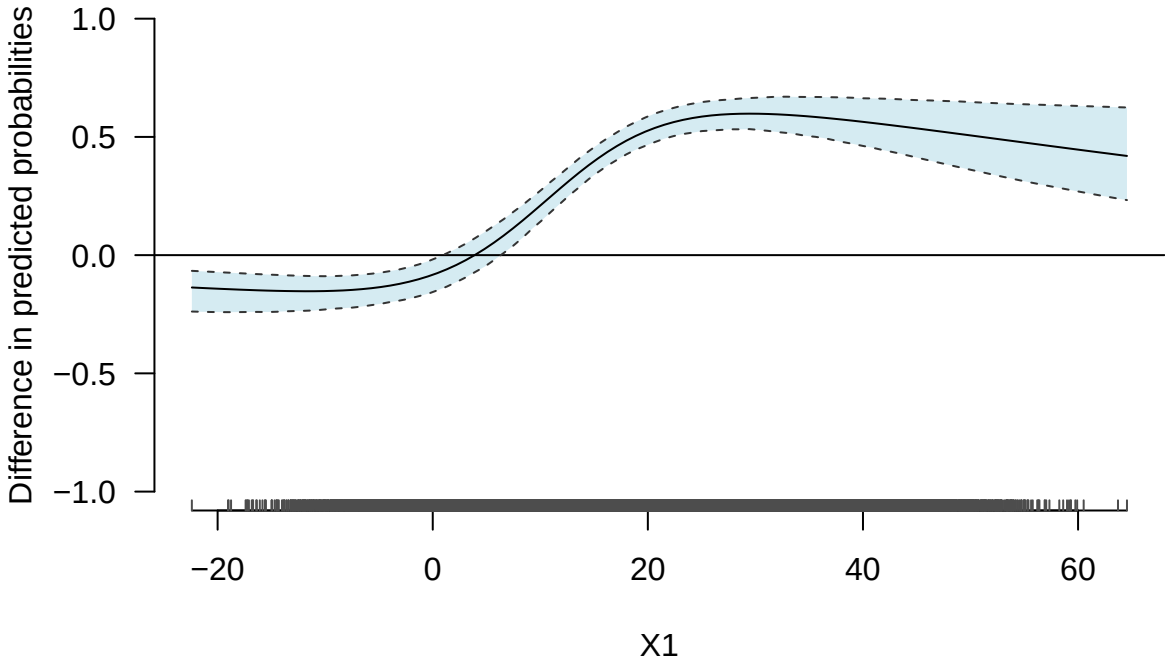
```
## [1] 1000  100
```

```r
# Means and Quantiles

p.mean.lo <- apply(p.sim.lo, 2, mean)
p.qu.lo <- t(apply(p.sim.lo, 2, quantile, prob = c(0.025, 0.975)))

p.mean.hi <- apply(p.sim.hi, 2, mean)
p.qu.hi <- t(apply(p.sim.hi, 2, quantile, prob = c(0.025, 0.975)))

## C. Plot

plot(X1.sim, p.mean.lo, type="n",
     ylim = c(0, 1),
     ylab = "Probability of Y",
     xlab = "X1")

polygon(c(rev(X1.sim), X1.sim), c(rev(p.qu.lo[,2]), p.qu.lo[,1]),
        col = adjustcolor("lightblue", alpha = 0.5),
        border = NA)

polygon(c(rev(X1.sim), X1.sim), c(rev(p.qu.hi[,2]), p.qu.hi[,1]),
        col = adjustcolor("lightgray", alpha = 0.5),
        border = NA)


lines(X1.sim, p.mean.lo, lwd = 1)
lines(X1.sim, p.qu.lo[, 1], lty = "dashed", col = "gray20")
lines(X1.sim, p.qu.lo[, 2], lty = "dashed", col = "gray20")


lines(X1.sim, p.mean.hi, lwd = 1)
lines(X1.sim, p.qu.hi[, 1], lty = "dashed", col = "gray20")
lines(X1.sim, p.qu.hi[, 2], lty = "dashed", col = "gray20")


# Add a "histogram" of actual X1-values.

axis(1, at = X1,
     col.ticks = "gray30",
     labels = FALSE, tck = 0.02)
```



How about looking at the first difference of the two scenarios directly?

```
fd <- p.sim.lo - p.sim.hi

fd.mean <- apply(fd, 2, mean)

fd.qu <- t(apply(fd, 2, quantile, prob = c(0.025, 0.975)))



plot(X1.sim, fd.mean, type="n",
     ylim = c(-1, 1),
     ylab = "Difference in predicted probabilities",
     xlab = "X1",
     bty = "n",
     las = 1)

polygon(c(rev(X1.sim), X1.sim), c(rev(fd.qu[,2]), fd.qu[,1]),
        col = adjustcolor("lightblue", alpha = 0.5),
        border = NA)

abline(h = 0)
lines(X1.sim, fd.mean, lwd = 1)
lines(X1.sim, fd.qu[, 1], lty = "dashed", col = "gray20")
lines(X1.sim, fd.qu[, 2], lty = "dashed", col = "gray20")


axis(1, at = X1,
     col.ticks = "gray30",
     labels = FALSE, tck = 0.02)
```



That looks funky (and not linear at all. . .).

Concluding Remarks

Great job everyone! You can be really proud of what you achieved.

Good luck with the Data Essay. We are confident that you have all you need to master it!