September 3, 2021

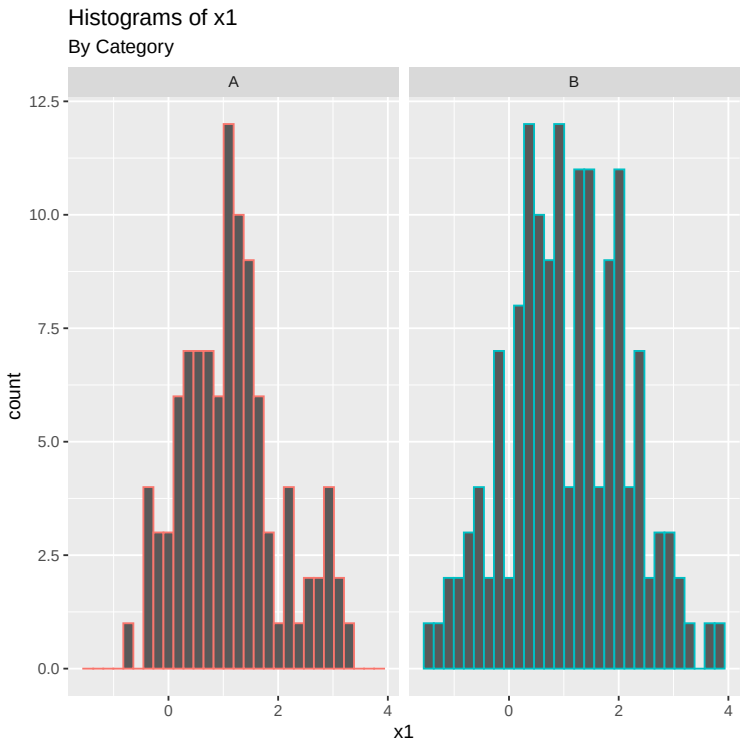
The results below are generated from an R script.

```
#### MFI R BOOTCAMP

# Case study: let's perform logistic regression on a simulated dataset

library(tidyverse)

## - Attaching packages --------------------------- tidyverse 1.3.1 -
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## - Conflicts ------------------------- tidyverse_conflicts() -
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(knitr)

set.seed(1729)

# Function definitions

logit <- function(p) {
  return( log(p/(1-p)) )
}

expit <- function(x) {
  return( 1/(1 + exp(-x)) )
}

norm <- function(x) {sqrt(sum(x^2))}


# Initializations and variable assignments
# Better use "<-" instead of "=" (<- was used in the original S-PLUS, and Google's R style guide insists

N <- 500 # number of observations
p <- 2 # number of covariates (excluding intercept)
beta <- c(-2, 2, 1) # true coefficients (beta0, beta1, beta2)

# Data structures
# There are several choices of data structures, depending on your application and preference. For data,

X <- matrix(0L, nrow=N, ncol=p)
colnames(X) <- c("x1", "x2")
```

```r
Y <- vector("numeric", length=N)

# Let's simulate some data

for (i in 1:N) {
  X[i,] <- c(rnorm(n=1, mean=1),
             rbinom(n=1, size=1, prob=0.6))

  eta_i <- beta %*% c(1, X[i,]) # matrix multiplication -- sizes must conform!

  Y[i] <- rbinom(n=1, size=1, prob=expit(eta_i))
}

# Always vectorize, if you can

X2 <- cbind(1, rnorm(n=N, mean=1), rbinom(n=N, size=1, prob=0.6))
eta <- rowSums(sweep(X2, 2, beta, "*"))
Y2 <- sapply(X=eta, FUN= function(eta) {rbinom(n=1, size=1, prob=expit(eta))}) # technically the apply f


myDat.frame <- data.frame(y=Y, x1=X[,1], x2=as.factor(X[,2]))
myDat.tib <- tibble(y=Y, x1=X[,1], x2=as.factor(X[,2]))

levels(myDat.frame$x2) <- list(A = "0", B = "1")
myDat.tib$x2 <- recode_factor(myDat.tib$x2, "0" = "A", "1" = "B")

summary(myDat.tib)

##        y               x1          x2
##  Min.   :0.000   Min.   :-1.9112   A:193
##  1st Qu.:0.000   1st Qu.: 0.3952   B:307
##  Median :1.000   Median : 1.0064
##  Mean   :0.574   Mean   : 1.0371
##  3rd Qu.:1.000   3rd Qu.: 1.6429
##  Max.   :1.000   Max.   : 3.8206

# We'll stick with the tibble going forward

myDat.train <- myDat.tib[1:(N/2),]
myDat.test <- myDat.tib[(N/2 + 1):N,]

myModel <- glm(y ~ x1 + x2, data=myDat.train, family=binomial(link="logit"))
summary(myModel)

##
## Call:
## glm(formula = y ~ x1 + x2, family = binomial(link = "logit"),
##     data = myDat.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7334  -0.6243   0.1606   0.6248   2.1046
##
## Coefficients:
```

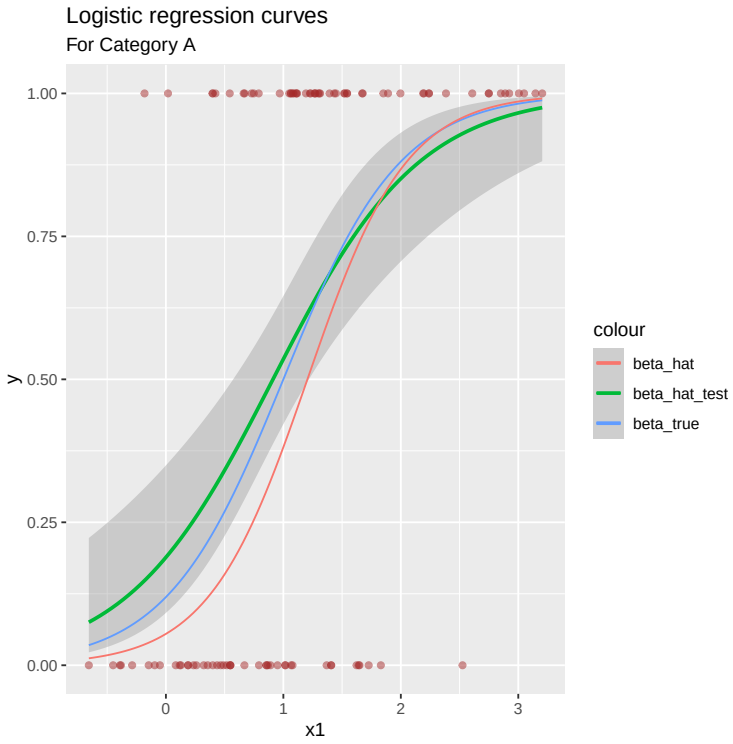
```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.8465     0.4333  -6.569 5.07e-11 ***
## x1            2.3628     0.3028   7.803 6.02e-15 ***
## x2B           1.5495     0.3781   4.098 4.17e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 343.86  on 249  degrees of freedom
## Residual deviance: 207.63  on 247  degrees of freedom
## AIC: 213.63
##
## Number of Fisher Scoring iterations: 5
```

```r
beta_hat <- myModel$coefficients
norm_diff <- norm(beta - beta_hat)

preds <- predict.glm(myModel, newdata=myDat.test, type=c("response"))
myDat.test <- cbind(myDat.test, y_pred = ifelse(preds > 0.5, 1, 0))
MSE <- mean(abs(myDat.test$y - myDat.test$y_pred))

# Plot some stuff

expit_beta_true <- function(x) { expit(beta[1] + x*beta[2])}
expit_beta_hat <- function(x) { expit(beta_hat[1] + x*beta_hat[2])}

ggplot(data=myDat.test, aes(x1, col=x2)) +
  geom_histogram(bins = 30) +
  facet_wrap(~ x2) +
  labs(title="Histograms of x1", subtitle="By Category") +
  theme(legend.position = "none")
```




```
ggplot(data=filter(myDat.test, x2 == "A"), aes(x=x1, y=y)) +
  geom_point(alpha = 0.5, col="brown") +
  geom_smooth(method="glm", method.args = list(family = "binomial"), aes(col="beta_hat_test")) +
  stat_function(fun=expit_beta_true, aes(col="beta_true")) +
  stat_function(fun=expit_beta_hat, aes(col="beta_hat")) +
  labs(title="Logistic regression curves", subtitle="For Category A") +
  theme(legend.position = "right")
```

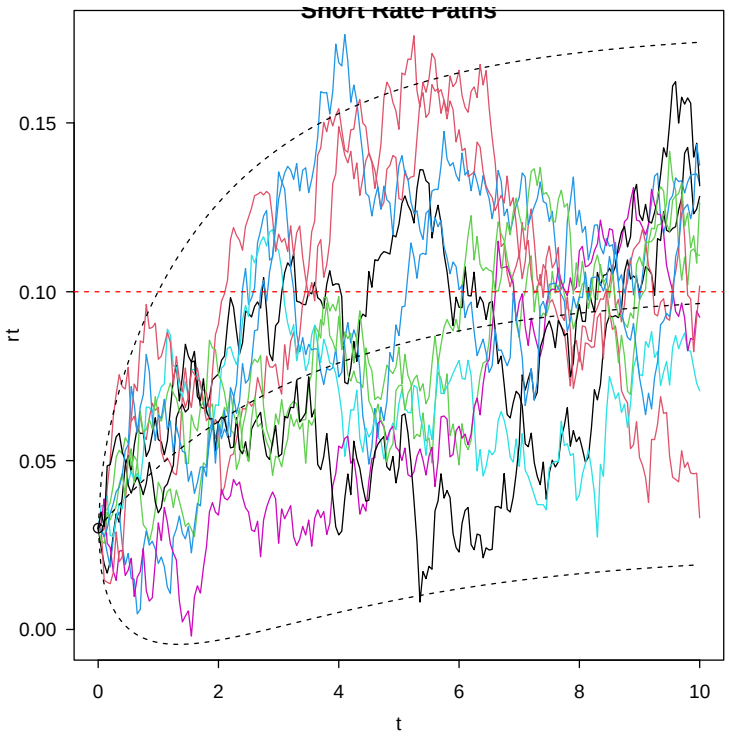
```
## `geom_smooth()` using formula 'y ~ x'
```




```
## Finding beta_hat directly using numerical optimization

mloglik <- function(beta) { # minus the log-likelihood
  ll <- 0
  for (i in 1:(N/2)) {
    eta_i <- beta %*% c(1, X[i,])
    ll <- ll + log( expit(eta_i) )*Y[i] + log(1 - expit(eta_i) )*(1 - Y[i])
  }
  return(-ll)
}

mloglik_grad <- function(beta) {
  gr <- rep(0, times=p+1)

  for (i in 1:(N/2)) {
    eta_i <- beta %*% c(1, X[i,])

    gr[1] <- gr[1] + (Y[i] - expit(eta_i))

    for (d in 1:p) {
      gr[d+1] <- gr[d+1] + (Y[i] - expit(eta_i))*X[i,d]
    }
  }
  return(-gr)
}

beta_hat_2 <- optim(par=rep(0,3), fn=mloglik, method="Nelder-Mead")
beta_hat_2 <- beta_hat_2$par # pretty close!
norm_diff_2 <- norm(beta_hat_2 - beta)
```

```r
beta_hat_3 <- optim(par=rep(0,3), fn=mloglik, gr=mloglik_grad, method="BFGS")
beta_hat_3 <- beta_hat_3$par # actually further away from the true value, but closer to what 'glm' (i.e
norm_diff_3 <- norm(beta_hat_3 - beta)



## working with packages

install.packages("glmnet")

##
## The downloaded binary packages are in
##  /var/folders/rr/1vbtqskj6sn7v9qgz25nn19m0000gn/T//RtmpQzTP5U/downloaded_packages

library(glmnet)

## Loading required package:  Matrix
##
## Attaching package:  'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loaded glmnet 4.1-2

rm(list = ls()) # clear environment
set.seed(1729)


N <- 500 # number of observations
p <- 10 # number of covariates (excluding intercept)
beta <- rnorm(n=p+1, mean=0, sd=4) # true coefficients (beta0, beta1, beta2)
ssq <- 2

X <- matrix(0L, nrow=N, ncol=p+1)

Y <- vector("numeric", length=N)

for (i in 1:N) {
  X[i,] <- c(1, rchisq(n=p, df=3))
  Y[i] <- beta %*% X[i,] + rnorm(n=1, mean=0, sd=sqrt(ssq))
}

beta_MLR <- solve(t(X) %*% X) %*% t(X) %*% Y
beta_MLR <- as.vector(beta_MLR)

dat <- data.frame(y=Y, X)

myModel <- lm(y ~. + 0, data=dat) # no intercept! Since we already have our own
myModel2 <- lm(y ~., data=subset(dat, select=-c(X1))) # with intercept, but without our column of ones
beta_OLS <- myModel$coefficients

## Stochastic processes (eg, a Vasicek model)

## stolen from https://www.r-bloggers.com/2010/04/fun-with-the-vasicek-interest-rate-model/
```

```r
## define model parameters
r0 <- 0.03
theta <- 0.10
k <- 0.3
beta <- 0.03

## simulate short rate paths
n <- 10    # MC simulation trials
T <- 10    # total time
m <- 200   # subintervals
dt <- T/m  # difference in time each subinterval

r <- matrix(0,m+1,n)  # matrix to hold short rate paths
r[1,] <- r0

for(j in 1:n){
  for(i in 2:(m+1)){
    dr <- k*(theta-r[i-1,j])*dt + beta*sqrt(dt)*rnorm(1,0,1)
    r[i,j] <- r[i-1,j] + dr
  }
}

## plot paths
t <- seq(0, T, dt)
rT.expected <- theta + (r0-theta)*exp(-k*t)
rT.stdev <- sqrt( beta^2/(2*k)*(1-exp(-2*k*t)))
matplot(t, r[,1:10], type="l", lty=1, main="Short Rate Paths", ylab="rt")
abline(h=theta, col="red", lty=2)
lines(t, rT.expected, lty=2)
lines(t, rT.expected + 2*rT.stdev, lty=2)
lines(t, rT.expected - 2*rT.stdev, lty=2)
points(0,r0)
```



The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 11.2.3
##
## Matrix products: default
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] glmnet_4.1-2    Matrix_1.3-3    forcats_0.5.1   stringr_1.4.0   dplyr_1.0.7
##  [6] purrr_0.3.4     readr_2.0.1     tidyr_1.1.3     tibble_3.1.3    ggplot2_3.3.5
## [11] tidyverse_1.3.1 knitr_1.33
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7       lubridate_1.7.10 lattice_0.20-44  assertthat_0.2.1 digest_0.6.27
##  [6] foreach_1.5.1    utf8_1.2.2       R6_2.5.0         cellranger_1.1.0 backports_1.2.1
## [11] reprex_2.0.1     evaluate_0.14    httr_1.4.2       highr_0.9        pillar_1.6.2
## [16] rlang_0.4.11     readxl_1.3.1     rstudioapi_0.13  labeling_0.4.2   splines_4.1.0
## [21] munsell_0.5.0    broom_0.7.9      compiler_4.1.0   modelr_0.1.8     xfun_0.25
## [26] pkgconfig_2.0.3  shape_1.4.6      mgcv_1.8-35      tidyselect_1.1.1 codetools_0.2-18
## [31] fansi_0.5.0      crayon_1.4.1     tzdb_0.1.2       dbplyr_2.1.1     withr_2.4.2
## [36] grid_4.1.0       nlme_3.1-152     jsonlite_1.7.2   gtable_0.3.0     lifecycle_1.0.0
```

```
## [41] DBI_1.1.1        magrittr_2.0.1   scales_1.1.1     cli_3.0.1        stringi_1.7.3
## [46] farver_2.1.0     fs_1.5.0         xml2_1.3.2       ellipsis_0.3.2   generics_0.1.0
## [51] vctrs_0.3.8      iterators_1.0.13 tools_4.1.0      glue_1.4.2       hms_1.1.0
## [56] survival_3.2-11  colorspace_2.0-2 rvest_1.0.1      haven_2.4.3

Sys.time()

## [1] "2021-09-03 11:56:54 EDT"
```