

Data Science and Big Data Analysis Notes

```
library(readr)

read_csv() # load into dataframe
as.dataframe() # convert to dataframe
str() #structure function
factor() #force column to be Factors
rownames(group_info_df) <- group_info_df$name # set row names to use first names
order(group_info_df$grade) #gives order of rows in ascending grade order, print row num
group_info_df <- group_info_df[order(group_info_df$grade), ] # select the rows in order
group_info_df[3:7,] # selects rows 3,4,5,6,7 all columns
```

```
library(readxl)
read_excel("hsb2.xlsx")
```

```
library('dplyr')
select(student_data, read, write, math, science) #select
select(student_data, female, race, school_type=schtyp) #select and rename schtype to
school_type
```

```
filter(student_data, age>20) #select column that age >20
filter(student_data, read>50, write>50)
mutate(student_data, lang=...) # adding lang to data frame
arrange(data, column) #sort by column by ascending order
arrange(student_data, desc(read))
factor(student_data$sex, labels = c("Black", "Asian")) # assign labels to F and M
```

Bivariate linear regression models—Day3

```
gdp_mean<-mean(world_data$gdp, na.rm = TRUE)
n<-length(world_data$gdp[!is.na(world_data$gdp)])
se <- sd(world_data$gdp, na.rm = TRUE) / sqrt(n) # calculate standard error
```

```
# lower bound
lb <- gdp_mean - 1.96 * se
# upper bound
ub <- gdp_mean + 1.96 * se
```

```
library(texreg)
names(dat)[which(names(dat) == "..")] <- "." # change names
plot(
  UnemploymentRate ~ NoHighSchool, data = dat,
  xlab = "Adults without High School education (%)",
  ylab = "Unemployment (%)",
  frame.plot = FALSE,
  pch = 16,
  col = rgb(red = 110, green = 200, blue = 110, alpha = 80, maxColorValue = 255)
)
model1 <- lm(UnemploymentRate ~ NoHighSchool, data = dat)
summary(model1)
```

Call: **lm(formula = UnemploymentRate ~ NoHighSchool, data = communities)**

Residuals:

Min	1Q	Median	3Q	Max
-0.42347	-0.08499	-0.01189	0.07711	0.56470

Coefficients:

$$\text{UnemploymentRate} = 0.078952 + (0.742385 * \text{NoHighSchool})$$

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.078952	0.006483	12.18	<2e-16 ***
NoHighSchool	0.742385	0.014955	49.64	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1352 on 1992 degrees of freedom
Multiple R-squared: 0.553, Adjusted R-squared: 0.5527
F-statistic: 2464 on 1 and 1992 DF, p-value: < 2.2e-16

```
abline(model1, lwd = 3,
       col = rgb(red = 230, green = 150, blue = 0, alpha = 255, maxColorValue = 255))
```

```
screenreg(model1)
screenreg(list(model1, model2))
predict(model1, newdata = data.frame(NoHighSchool = 10)) # predict on single value =10
```

```
n <- nrow(world_data)
world_data$x <- rnorm(n, mean=0, sd=10)
```

```
rename(data, new name = old name)
mutate(gender = factor(female, labels=c("M", "F"))) # change 1 and 0 to M and F
```

Multiple linear regression models – Day4

```
library(haven) # used to load our data
library(texreg) # used to display fit info
library(dplyr) # used to manipulate data
library(tidyr) # used for the drop_na function
library(ggplot2) # in case we want to make ggplots
```

```
read_dta()
mutate(world_data, democracy = factor(democracy, levels=c(0,1),
labels=c("dictatorship", "democracy")))
world_data <- drop_na(world_data, latitude) # drop na value in latitude and return df
anova(latitude_model, inst_model)
```

Model 1 is nested in Model 2

Analysis of Variance Table

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	168	133.12				
2	166	74.28	2	58.841	65.749	< 2.2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Degrees of freedom
df = n - p - 1
n: number of data points
p: number of variables in model

unexplained errors

number of added variables

restricted model

unrestricted model

difference between RSS of models

F-value

p-value

```
new_data_democracy <- data.frame(institutions_quality = seq(from = 1.4, to = 9.3, by =
1), globalization = mean(world_data$globalization), latitude = mean(world_data$latitude),
democracy = "democracy")
```

```
pred <- predict(inst_model, newdata = new_data_democracy, se.fit = TRUE)
new_data_democracy$political_stability.se <- pred$se.fit
new_data_democracy <- mutate(new_data_democracy,
  political_stability.ub = political_stability.pred + 1.96*political_stability.se,
  political_stability.lb = political_stability.pred - 1.96*political_stability.se
)
```

```
# add lines for confidence intervals
# upper bound
lines(x = new_data_democracy$institutions_quality,
  y = new_data_democracy$political_stability.ub,
  lty = "dashed", lwd = 1.5)
# lower bound
lines(x = new_data_democracy$institutions_quality,
  y = new_data_democracy$political_stability.lb,
  lty = "dashed", lwd = 1.5)
```

```
abline(a=y0, b=slope, col="blue")
table(world_data$former_col)
```

```
better.model <- lm(human_development ~ poly(gdp_capita, 2), data = world_data)
screenreg( list(bad.model, better.model),
  custom.model.names = c("bad model", "better model"))
```

Variable Selection — Day5

```
library(MASS)
lm.1 <- lm(Boston$medv~Boston$lstat,data=Boston)
```

Rsquared, (summed squared of residuals) / (variance)

```
rsq = summary(lm.1)$r.squared
rsq
```

Adjusted Rsquared, larger better

```
rsq = summary(lm.1)$adj.r.squared
rsq
```

MSE, small better

```
mean(residuals(lm.1)^2)
MSE_model <- mean((actual data - predict)^2)
```

RMSE, small better

```
sqrt(mean(residuals(lm.1)^2))
```

```
AIC(lm.1) # lower AIC means better performance
BIC(lm.1) # lower BIC means better performance
```

```
lm.min <- lm(medv ~ 1, data=data.original)
lm.max <- lm(medv ~ ., data=data.original)
```

```
scp <- list(lower = lm.min, upper = lm.max)
```

```
lm.selected <- stepAIC(lm.min,
                      direction = 'forward',
                      scope = scp,
                      steps = 1)
```

```
# Using leaps
library(leaps)
```

```
regsubsets.out <- regsubsets( medv ~ .,
                             data = data.original,
                             nbest = 1,
                             nvmax = NULL,
                             force.in = NULL, force.out = NULL,
                             method = 'exhaustive')
```

```
summary(regsubsets.out)
```

```
as.data.frame(summary(regsubsets.out)$outmat)
plot(regsubsets.out, scale='adjr2', main='Adjusted Rsq')
```

```
library(corrplot)
corrplot(cor(boston_housing))
```

```
transport <- cut(boston_housing$rad, breaks = c(0,15,30),
labels=c("low", "high"))
# separate value 0~15 (low) and 15~30 (high)
```

```
credit_card$Balance_mean <- mean(credit_card$Balance)
credit_card$squared_dev <- (credit_card$Balance
                          - credit_card$Balance_mean)^2
credit_card$squared_resid <- (credit_card$Balance
                           - credit_card$Balance_pred)^2
TSS <- sum(credit_card$squared_dev)
RSS <- sum(credit_card$squared_resid)
Rsq <- 1 - RSS/TSS
```

```
lm.selected <- lm(logSalePrice~OverallQual + OverallCond +
GrLivArea + YearBuilt, data=house_data_num)
```

```
RSS = sum(lm.selected$residuals)^2
MSE = RSS/nrow(house_data)
RMSE = sqrt(MSE)
```

Validation — Day6

```
# create a column with NA
data.sample$cv_pred <- NA
```

LOOCV

```
library(cvTools)
cvFit(model, data = data.sample, y = data.sample$Y, K = # of data rows) # This returns sqrt(MSE)
or RMSE value
RMSE = sqrt(sum((model$residuals)^2)/nrow(house_data_num))
RMSE
```

K-fold CV

```
library(cvTools)
cvFit(model, data = data.sample, y = data.sample$Y, K = # of fold, R=.. )
#Setting k equal to n yields leave-one-out cross-validation.
# an integer giving the number of replications for repeated
K-fold cross-validation

cv_result = cvFit(model, data = house_data_num, y =
house_data_num$logSalePrice, K = 10, R=100)
cv_result$cv #RMSE
cv_result$se #standard error
cv_result$reps
```

bootstrapping

```
R=1000
coef_out <- rep(NA,R)
MSE_out <- rep(NA,R)
for (i in 1:R){
  resampled_rows <- sample(1:n,n,replace=TRUE )
  data.resampled <- data.sample[resampled_rows,]
  model <- lm(Y~X,data=data.resampled)
  coef_out[i] <- coef(model)[[2]]
  MSE_out[i] <- mean(resid(model)^2)
}

hist(coef_out)
hist(MSE_out)
mean(coef_out)
mean(MSE_out)

plot(house_data$logSalePrice~lm.selected$fitted.values) # stepwise, actual~fittedvaule
plot(lm.selected$residuals~lm.selected$fitted.values)

#install.packages("gvlma")
library(gvlma)
gvlma(lm.selected) # assess linear model assumption
```

```
library(corrplot)
corrplot(cor(predictor_set))
res <- cor(my_data)
round(res, 2)
cor(y = world_data$hdi, x = world_data$corruption, use =
"complete.obs") #correlation coefficient
cor.test(~ hdi+ gdp, data=world_data, ) #estimated
correlation # find evidence to reject the null hypothesis
at the 5% significance level, and accept the hypothesis
that the variables are correlated.
```

```
residuals <- actual data - pred
RMSE <- sqrt(sum((residuals)^2)/length(residuals))
RMSE
```

bootstrapping, non-linear fit methods and shrinkage methods—Day7

Non-linear fit methods

```
library(ISLR)
library(splines)
```

```
# can specify knot points
fit_spline1=lm(wage~bs(age,knots=c(25,40,60)),data=Wage)
pred <- predict(fit_spline1,newdata=age_grid_df)
plot(wage~age, pch='.', data=Wage)
lines(age_grid,pred,lwd=2,col="red")
```

```
# can specify degrees of freedom (more=more flexible)
fit_spline2=lm(wage~bs(age,df=6),data=Wage)
pred <- predict(fit_spline2,newdata=age_grid_df)
plot(wage~age, pch='.', data=Wage)
lines(age_grid,pred,lwd=2,lty=2,col="red")
```

```
# loess to build a LOcal regrESSion
fit_loess1=loess(wage~age,span=.2,data=Wage)
fit_loess2=loess(wage~age,span=1,data=Wage)
plot(wage~age, pch='.', data=Wage)
lines(age_grid,predict(fit_loess1,newdata=age_grid_df),col="red",lwd=2)
lines(age_grid,predict(fit_loess2,newdata=age_grid_df),col="blue",lwd=2)
```

```
cut_points <- quantile(Wage$age,c(0,0.25,0.5,0.75,1.0))
Wage$age_groups <- cut(Wage$age, breaks = cut_points)
```

```
# assess spline with different degree of freedom
```

```

library(cvTools)
df_vals <- c(4,,8,12,16,20,24,28,32,36,40)
n_vals <- length(df_vals)
results <- 0
results <- data.frame(df = rep(0,n_vals),
                      rmse = rep(0,n_vals),
                      se = rep(0,n_vals))
)

for (i in 1:n_vals){
  df_i <- df_vals[i]
  fit_spline2=lm(wage~bs(age,df=df_i),data=Wage)
  cv_result <- cvFit(fit_spline2, data = Wage, y = Wage$wage, K =
10, R = 10)
  results$df[i] <- df_i
  results$rmse[i] <- cv_result$cv
  results$se[i] <- cv_result$se
}
head(results)
plot(results$rmse~results$df, pch=1, ylim=c(39.8,40.9))
points(results$df, results$rmse + 1.96*results$se, pch=2)
points(results$df, results$rmse - 1.96*results$se, pch=2)

```

glmnet Ridge Regression and Lasso method

```

fit_lasso <- glmnet(housing_data.x, housing_data.y,
                    alpha = 1, lambda = 1)
fit_ridge <- glmnet(housing_data.x, housing_data.y,
                    alpha = 0, lambda = 1)
# view fit coefficients
coef(fit_lasso)
# to predict new points based on model
pred <- predict(fit_lasso,newx=housing_data.x)

# if we do not specify lambda glmnet will test out
# a range of penalties and we can view how coefficients
# behave by plotting the result
fit_lasso <- glmnet(housing_data.x, housing_data.y,
                    alpha = 1)
plot(fit_lasso, xvar = "lambda") # lambda is on log-scale

# cross validation for best choice of lambda
results <- cv.glmnet(housing_data.x, housing_data.y,
                    alpha = 0, nfolds=10)
plot(results)

```

logistic regression, LDA and QDA classification —Day8

```

par(mfrow=c(2,2))

```

```
# the following command creates a 2x2 plot but
# adjusts margins (plot spacing) to be smaller
par(mfcol=c(2,2), mar=c(4,4,0.5,0.5), oma=c(1.5,2,1,1))
```

```
# with command lets you run a
# command using a given data set
# similar to using the data=... argument
with(mf_training, scatter.smooth(height ~ hair,
                                pch=20,
                                col = ifelse(gender==1,'red','blue')
                                )
)
plot(height ~ eye, data=mf_training)
plot(height ~ as.factor(gender), data=mf_training)
plot(height ~ as.factor(glasses), data=mf_training)
```

```
par(mfrow=c(1,2))
plot(regsubsets.out, scale="adjr2")
adj_rsquare_values <- summary(regsubsets.out)$adjr2
n_predictor_vals <- 1:5
plot(adj_rsquare_values~n_predictor_vals)
```

```
nfemales <- length(which(mf_training$gender==1)) # number of female in data
```

```
rows_glasses = which( mf_training$glasses==1)
result <- mean(mf_training[rows_glasses,]$gender) #P(female|glasses) , if female if they wear
glasses
```

```
result <- mean(mf_training[-rows_glasses,]$gender) #P(female|no-glasses)
```

```
mean(mf_training[which(mf_training$hair>10),]$gender) #P(female|hair>10)
```

```
lm(gender ~ height,mf_training)
predict(gender_height.lm, data.frame(height = c(150, 180))) # probability for female when height
is 150 and 180
```

```
b0 = coef(gender_height.lm)[["(Intercept)"]]
b1 = coef(gender_height.lm)[["height"]]
result <- (0.5 - b0)/b1 # take x out of y = b0 + b1*x
```

```
total <- nrow(mf_training)
misclassified <- length(which(mf_training$mod1_prediction != mf_training$gender))
rate <- misclassified/total
```

```
hist(subset(mf_training,gender==1)$height, col='blue',
     xlab='height',main='Male/Females height distributions')
hist(subset(mf_training,gender==0)$height, col=rgb(1,0,0,0.5), add=T)
```


Logistic

```
#predicts the probability given height, is female
gender_height.glm=glm(gender~height,data=mf_training,family=binomial)
plot(gender~height,data=mf_training)
curve(predict(gender_height.glm,data.frame(height=x),type="resp"),add=TRUE)
```

logistic fit formula is $P(\text{female}|\text{height}=H) / y = 1 / (1 + \exp(-B_0 - B_1 * X))$

```
mf_training$prediction <- factor(mf_training$model_prediction, levels=c(0,1),
labels=c("male","female"))
```

```
plot(iris$Petal.Length, iris$Petal.Width, pch=24,
     bg=c("red","green","blue")[unclass(iris$Species)],
     main="Iris Species\n R: Setosa, G: Versicolor, B: Virginica")
```

```
unclass() #converts a factor to its integer representation
levels(iris$Species) # print factor
subset(iris,Species=="virginica" | Species=="versicolor") #subset rows with only species
iris_simple[,c("Species","Sepal.Length","Sepal.Width")]
```

```
correct = sum(logistic.model.prediction == iris_simple$Species)
correct
```

LDA

library(MASS)

```
lda.model = lda(Species ~ Sepal.Length+Sepal.Width, data = iris_simple)
lda.model.prediction <- predict(lda.model)$class
lda.model = lda(Species ~ ., data = iris, CV=TRUE)
```

QDA

```
qda.model = qda(Species ~ ., data = iris)
qda.model.prediction <- predict(qda.model)$class
```

```
plot(iris$Sepal.Length, iris$Sepal.Width, pch=24,
     bg=c("red","green","blue")[unclass(iris$Species)],
     main="Iris Species\n R: Setosa, G: Versicolor, B: Virginica")
```

```
misclassified = sum(iris_test$pred_lda!=iris_test$Species)
misclassification_rate_lda = misclassified/nrow(iris_test)
misclassification_rate_lda
```

KNN —Day9

```
normalise <- function(x){
```

```

    norm_x <- (x-min(x))/(max(x)-min(x))
    return(norm_x)
}

```

```

Set.seed(xx)
sample(1:nrow(iris),size=floor(nrow(iris)/2))

```

```

knn.pred=knn(iris_X_train,iris_X_test,iris_Y_train,k=30)

```

```

knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)

```

Arguments

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
cl	factor of true classifications of training set
k	number of neighbours considered.

```

table(predicted=knn.pred,actual=iris_Y_test)
knn.cv() performs LOOCV
plot(X2~X1,col=Y,pch=2, data=data.cols)

```