Learning Objectives: Data Science & Big Data Analytics 2019

Working in R

- Be able to use R to perform mathematical calculations.
- Understand the different data types used by R.
- Know how to work with vectors (creating, indexing)
- Be able to use R to perform element-wise calculations to vectors/columns, (e.g. adding columns together, averaging columns, scaling columns).
- Be able to write a for loop in R e.g. to print a simple sequence 2,4,6,8 or perform a certain task 100 times and store the results in a data frame or vector.
- Understand how to use tests > < == != and combine them using & | to test for a given condition on a value or column of values
- Be able to use the ifelse function in R on a column to generate a new column based on the column values.
- Be able to write simple R functions that take some input arguments and return a calculated result.
- Be able to generate random number sets e.g. from a uniform or normal distribution, and set a random seed appropriately.

```
set.seed(100)

a <- runif(10)

b <- rnorm(20)
```

Working with datasets in R

- Be able to load data in R using a variety of formats (csv, xls, stata, Rdata, Rdat)
  library(readr)
  world_data_raw <- read_csv("QoG2012.csv")
- make selections of rows/columns
  ```
  student_grades <- select(student_data, read, write,  math, science)
  ```
- delete rows /columns
  ```
  student_data2 <- select(student_data, -read, -write)
  ```
- filter rows based on conditions (and sets of conditions)
  ```
  high_reading <- filter(student_data, read>50)
  ```

- use column data to create new columns e.g. based on a transformation of units or a formula involving other columns

```
student_data <- mutate(student_data, lang = (read+write)/2, overall = (read
+ write + math + science + socst)/5 )
```

- convert columns between types (e.g. between character/numerical/factor)

```
group_info_df$Gender <- as.factor(group_info_df$Gender)
group_info_df$Age <- as.integer(group_info_df$Age)
group_info_df$Height <- as.numeric(group_info_df$Height)
group_info_df$Been_before <- as.logical(group_info_df$Been_before)
```

- be able to use cut to convert a numerical column into a factor column

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

- be able to rename rows and columns

```
world_data <- rename(world_data_raw,
          judicary = h_j,
          gdp = wdi_gdpc,
          hdi = undp_hdi,
          corruption = wbgi_cce,
          stability = wbgi_pse,
          abs_lat = lp_lat_abst)
```

- be able to change the level labels for factor columns

```
student_data$racial_group <- factor(student_data$race, labels = c("Black", "Asian", "Hispanic", "White"))
```

- Be able to sort a dataframe by column values

```
# ascending order
student_data_sorted <- arrange(student_data,read)
# descending order
student_data_sorted <- arrange(student_data,desc(read))
```

- Be able to randomly sample a dataframe (e.g. to create training and test subsets, bootstrapped datasets, or to randomise row order)

```
set.seed(1)
n <- nrow(housing_data)
random_row_ids <- sample(1:n,size=n,replace=TRUE)
housing_data_rs <- housing_data[random_row_ids,]
```

- Be able to normalise columns (by shifting and scaling) e.g. so that it has mean = 0 and standard deviation = 1.

```
normalise <- function(x){
norm_x <- (x-min(x))/(max(x)-min(x))
return(norm_x)
}
```

- be able to use the dplyr group_by and summarise commands

  ```
  student_data$racial_group <- factor(student_data$race, labels = c("Black",
  "Asian", "Hispanic", "White"))
  student_data_by_racial_group <- group_by(student_data, racial_group)
  summarise(student_data_by_racial_group, n_students = n())
  summarise(student_data_by_racial_group, mean_read = mean(read), max_read =
  max(read), min_read = min(read), sd_read = sd(read))
  ```

- be able to use apply to apply a function to the rows or columns of a dataframe

  col.sums <- apply(x, 2, sum)

  row.sums <- apply(x, 1, sum)

- be able to build a dataframes (e.g by joining columns, adding columns to an existing dataframe, or by adding new rows to an existing dataframe)

  ```
  math_seq <- seq(from=0, to=100, by=1) student_data_for_fit <-
  data.frame(math=math_seq)
  fit_poly3 <- predict(polymodel3, newdata = student_data_for_fit )
  student_data_for_fit$fit_poly3 <- fit_poly3
  ```

Exploring datasets in R

- Be able to find and interpret mean, median of a column

  mean(student_data$science)

  median(student_data$science)

- Be able to find and interpret standard deviation and variance of a column

  var(student_data$science)

  sd(student_data$science)

- Be able to find and interpret range and quantiles (e.g. e.g. to find 95% interval) of a column

  range(birth_weights)

  quantile(birth_weights,c(0.025,0.975,NaN))

- Be able to inspect for outliers visually and find them in the dataset and take appropriate action.

- Be able to find the correlation between data columns

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

- Be able to make and interpret scatter plots

  plot

- Be able to make and interpret histograms (with defined cut/bin positions)

- Be able to make and interpret boxplots

  boxplot

- Be able to style plots: set x and y axis limits, set x and y axis labels, set title, set point marker types, set line styles, and set point colour based on a criteria or factor class.

  plot(medv ~ lstat,data = boston_housing,

  xlim(15, 20)

  col = chas,

  pch = 16,

  cex = 1,

  bty = "n",

  main = "Relation between median value and lower status by Charles River dummy variable",

  xlab = "Lower status",

  ylab = "Median value"

  lty = "dashed", lwd = 1.5)

- Be able to style plots: set x and y axis labels, set title

  xlab = "Lower status",

  ylab = "Median value"

- Be able to style plots: set point marker types and set line styles,

  pch= option to specify symbols

lty	line type. see the chart below.
lwd	line width relative to the default (default=1). 2 is twice as wide.

- Be able to style plots: set point colour based on a criteria or factor class.

  col = chas,

- Be able to use the table command to build a frequency table.

  table(student_data$socioeconomic_status,student_data$racial_group)

Learning Objectives: Data Science & Big Data Analytics 2019

- Be able to add annotations to plots: add text to a plot

- Be able to add annotations to plots: add vertical / horizontal lines at given values

- Be able to view a smoothed trend line onto scatter plot using scatter.smooth

  scatter.smooth(speed, dist, lpars =

  list(col = "red", lwd = 3, lty = 3))

- Be able to takes appropriate steps to identify where columns have missing (NA) values, and take appropriate measures (removal or exclusion from an analysis).

```
library(Amelia)
missmap(BostonHousing,col=c('yellow','black'),legend=TRUE)
drop_na(credit_data)
```

Hypothesis testing in R

- Understand the assumptions made in performing a t-test, how to carry it out in R and interpret and explain the results (t-value, p-value, confidence interval).

```
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)
# lower bound
lb <- gdp_mean - 1.96 * se
# upper bound
ub <- gdp_mean + 1.96 * se
t.value <- (gdp_mean - 10000) / se
# p-value calculation
2* ( 1 - pt(t.value, df = (n-1) ))
```

- Be able to carry out a t-test on two groups of values to find evidence for a difference in the means, or test a hypothesis that one mean is greater/less than the other.

```
t.test(gdp ~ judiciary, mu = 0, alt = "two.sided", conf = 0.95,
data=world_data)
```

- Be able to carry out a t-test on two columns to find evidence to test the hypothesis that the columns are correlated.

```
cor.test(~ hdi+ gdp, data=world_data, )
```

Bivariate Regression in R

Learning Objectives: Data Science & Big Data Analytics 2019

- Understand and be able to explain the principle of least-squares regression.

- Be able to carry out simple linear regression in R to make a linear model that predicts a response based on a single independent variable, and plot the line of best fit onto a scatter plot of the data.

  ```
  plot(medv ~ lstat,data = boston_housing,
     frame.plot = FALSE,
     pch = 16,
     col = rgb(red = 110, green = 200, blue = 110, alpha = 80, maxColorValue = 255))
  abline(modelm_l, lwd = 3,
       col = rgb(red = 230, green = 150, blue = 0, alpha = 255, maxColorValue = 255))
  ```

- Be able to display and understand the results of the fit, and the associated measures returned.

  ```
  summary
  ```

- Be able to recall and explain the assumptions of linear regression, use R to explore if these assumptions are met, and carry out suggested steps to identify and manage these issues (e.g. to perform a suggested transformation, or remove outliers).

  ```
  gvlma(lm.selected)
  ```

- Be able to explore how simple variable transformations improve/do not improve linear fitting.

- Be able to make predictions using the fit (both for the training data, or a new dataset)

- Be able to access the residuals, and calculate values of RSS, MSE, and RMSE.

  ```
  RSS
  sum(residuals(lm.1)^2)
  MSE
  mean(residuals(lm.1)^2)
  RMSE
  sqrt(mean(residuals(lm.1)^2))
  ```

Multivariate Regression in R

- Understand how to perform a multivariate fit in R.

  lm(~ +)
- Understand how to interpret the fit coefficients in a multivariate fit, and their associated p-values.

  summary
- Understand how to interpret the f-test value returned from a multivariate fit.

  Anova()
- Understand how multivariate fits can include categorical variables, the use of dummy variables, and the interpretation of the resulting fit coefficients and their associated p-values.

```
model1 <- lm(science ~ math+gender,data=student_data)
summary
```
- Be able to include interaction terms in a multivariate fit, interpret the resulting coefficients, and test for their significance.

```
model1 <- lm(science ~ math+gender,data=student_data) model2 <-
lm(science ~ math*gender,data=student_data)
screenreg(list(model1,model2),digits=4)
```
- Be able to plot the result of multivariate fits appropriately (e.g. plots of the fit for simple models: 1 numerical predictor and 1 factor predictor (with/without interaction); plot of actual vs predicted values for more complex models; plot residuals vs fitted values)

```
plot(medv ~ lstat,data = boston_housing,
   col = chas,
   pch = 16,
   cex = 1,
   bty = "n",
   main = "Relation between median value and lower status by Charles River dummy
variable",
   xlab = "Lower status",
   ylab = "Median value")
```

```
abline( a=40.9 , b= - 0.9972 , col = 2)
abline( a=34.9 , b= - 0.9972 , col = 1)
legend(
 "topright",  # position fo legend
 legend = levels(boston_housing$chas), # what to seperate by
 col = c(1,2), # colors of legend labels
 pch = 16, # dot type
 lwd = 2, # line width in legend
 bty = "n" # no box around the legend
 )
```

Residual vs fitted

```
plot(lm.selected)
```

actual vs predicted

```
plot(house_data$logSalePrice~lm.selected$fitted.values)
```

- Understand the problem arising with highly correlated predictors, and be able to detect and suggest suitable steps to deal with these issues.

Model selection

- Be able to interpret fit results to evaluate the significance of the different predictors used, and suggest potential model improvements.
  ```
  screenreg(list(model1,model2),digits=4)
  ```
- Understand how to perform an ANOVA test on a multivariate model to compare the performance to a model containing only a subset of the predictors, and interpret the result.
  anova
- Be able to use the stepAIC function to perform stepwise model selection to optimise a model, understanding the arguments that can be used.
  ```
  lm.min <- lm(medv ~ 1, data=boston_housing)
  ```

```
lm.max <- lm(medv ~ ., data=boston_housing)
scp <- list(lower = lm.min, upper = lm.max)
lm.selected <- stepAIC(lm.min,
                       direction = 'forward',
                       scope = scp,
                       steps = 13)
```

- Be able to use the leaps function to test all possible predictor subsets and interpret the results and plots produced.

```
regsubsets.out <- regsubsets( medv ~ .,
                              data = boston_housing,
                              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')
```

- Be able to use and optimise multivariate fits. using Ridge Regression and Lasso methods to constrain fit coefficients.

```
# glmnet can do ridge regression alpha=0
#             or lasso regression alpha=1
# we can also use values between 0 and 1
# which uses a hybrid of the two penalty types
# (called elastic net)
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 coefficents
# 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)
```

Cross Validation

- Understand why the performance of a model on its training data may differ significantly in comparison to a test / validation / new data.

- Be able to describe the approaches of validation set (aka hold out) and be able to carry it out in R.

```
# valdiation set method
n = nrow(house_data_num)
n_training = n%/%2
# using %/% means integer division
# e.g. 3%/%2 = 1 not 1.5
# this is useful when we need whole numbers

# simple way to split (better to randomise...)
house_data_num.training <- house_data_num[1:n_training,]
house_data_num.test <- house_data_num[n_training:n,]
lm.selected.training <- lm(logSalePrice~OverallQual + OverallCond +
GrLivArea + YearBuilt, data=house_data_num.training)

pred <- predict(lm.selected.training, newdata=house_data_num.test)
residuals <- house_data_num.test$logSalePrice - pred
RMSE <- sqrt(sum((residuals)^2)/length(residuals))
RMSE
```

- Be able to describe the approaches of LOOCV and be able to carry it out in R using suitable functions

- `install.packages("cvTools")`
- `library(cvTools)`
- `n = nrow(house_data_num)`
- `cvFit(lm.selected, data = house_data_num, y = house_data_num$logSalePrice, K = n)`
- `RMSE = sqrt(sum((lm.selected$residuals)^2)/nrow(house_data_num))`
- `RMSE`

- Be able to describe the approaches of k-fold validation and be able to carry it out in R using suitable functions.

- `# K-fold cross validation (10 folds) repeated with 100 random fold sets`

- `cv_result = cvFit(lm.selected, data = house_data_num, y = house_data_num$logSalePrice, K = 10, R=100)`
- `cv_result$cv`
- `cv_result$se`
- `cv_result$reps`

- Understand how to interpret the results of validation testing to select optimum models.

- Be able to describe the relative advantages/disadvantages of validation set (aka hold out), LOOCV, k-fold validation methods.

- Understand and be able to carry out the method of resampling to generate bootstrapped datasets that can be used when evaluating model performance.

```
set.seed(101)

# Let's create a population with some
# variable of interest Y. We have an
# input variable X that has mean 800, stdev 10
# and 10000 members
N <- 10000
X <- rnorm(N,mean=800,sd=10)

# Let's make the true relationship that
# Y and X are positively correlated.
# with population regression relationship
# Y = 0.5*X + e
# where e is the variance in Y that is not explained
# by the variation in X
# assume this is fairly large
# e.g. mean = 0 but sd = 20
e <- rnorm(N,mean=0,sd=20)

# now we can calculate Y
Y = 0.5*X + e

# Store into data.frame
data.pop = data.frame(Y,X,e)

# Suppose a data scientist wants to explore the
# relationship between Y and X. However they don't have
# access to the full population sample, and need
# to infer it from a random sample of 200 points
n <- 200
sampled_rows <- sample(1:N,n )
data.sample <- data.pop[sampled_rows,]

# in bootstrapping we resample this many times
R=1000
# on each resampled set we calculate and store
# the fit parameter of interest
# here let's store the fitted slope value
# and mean squared sum of residuals MSE
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)
}
```

- Understand how the process of bootstrapping can be used to investigate the distribution of fit parameter results.

```
# examining the histograms of the variables of interest
# let's us estimate the possible distribution
# when testing new data
hist(coef_out)
hist(MSE_out)
mean(coef_out)
mean(MSE_out)
```

- Be able to carry out a bootstrap analysis to investigate e.g. fit performance or fit coefficients

```
# compare to fit to original sample
model <- lm(Y~X,data=data.sample)
coef_sample <- coef(model)[[2]]
MSE_sample <- mean(resid(model)^2)

coef_sample
MSE_sample

# note this doesn't improve the fit, it justs gives us
# infomation on the distributions of the parameter
# estimates
```

- Be able to use a for loop to carry out an optimisation analysis using a cross validation method, plot the results.

```
library(cvTools)
n_df_vals = 6
df_vals <- c(4,6,8,10,12,14)
result <- data.frame(df = rep(0,n_df_vals),
          rmse = rep(0,n_df_vals),
          se = rep(0,n_df_vals))
for (i in 1:6){
 df_i = df_vals[i]
 fit_spline = lm(logSalePrice~bs(GrLivArea,df=df_i),data=housing_data)
```

```
  cv_result = cvFit(fit_spline, data = housing_data, y = housing_data$logSalePrice, K =
10, R = 10)
  result$rmse[i] <- cv_result$cv
  result$df[i] <- df_i
  result$se[i] <- cv_result$se
}
head(result)
plot(result$rmse~result$df, pch=1,ylim=c(39.8,40.2)
points(result$df, result$rmse + 1.96*result$se,pch=2)
points(result$df, result$rmse - 1.96*result$se,pch=2)
```

Classification

- Be able to use R to calculate simple probabilities associated with sampling a dataset.

- Understand how the logistic function can be fitted to data to predict the probabilities associated with a binomial (0 or 1) variable in relation to the values of a set of predictors. Know why it is used in preference to a linear function.

```
logistic.model.prediction <-
  ifelse( predict(logistic.model,type='resp') > 0.5,
            'versicolor',
            'virginica')
table(predicted=logistic.model.prediction,actual=iris_simple$Species)
```

- Be able to carry out a logistic regression fit in R and interpret the results.

```
gender_height.glm=glm(gender~height,data=mf_training,family=binomial)
```

- Understand how to display the results of a classification model using table to display the confusion matrix comparing predictions to true values.

```
table(predicted=logistic.model.prediction,actual=iris_simple$Species)
```

Learning Objectives: Data Science & Big Data Analytics 2019

- Be able to use the results of a classification model to make predictions, and measure the performance according to the misclassification rate.

```
load("mf_test.Rda")
mf_test$mod1_value <- predict(gender_height.glm,newdata=mf_test)
mf_test$mod1_prediction <- rep(0,nrow(mf_test))
mf_test[which(mf_test$mod1_value > 0.5),]$mod1_prediction <- 1
total <- nrow(mf_test)
misclassified <- length(which(mf_test$mod1_prediction != mf_test$gender))
rate <- misclassified/total
rate
```

- Be able to describe the assumptions which Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) make to build a classification model.

LDA assumes normal distributed data and a class-specific mean vector.

LDA assumes a common covariance matrix. So, a covariance matrix that is common to all classes in a data set.

Observation of each class are drawn from a normal distribution (same as LDA).

QDA assumes that each class has its own covariance matrix (different from LDA).

- Be able to use R to perform a LDA (Linear Discriminant Analysis) fit for classification.

```
lda.model = lda(Species ~ Sepal.Length+Sepal.Width, data = iris)
lda.model.prediction <- predict(lda.model)$class
table(predicted=lda.model.prediction,actual=iris$Species)
correct = sum(lda.model.prediction == iris$Species)
incorrect = sum(lda.model.prediction != iris$Species)
```

- Be able to use R to perform a QDA (Quadratic Discriminant Analysis) fit for classification.

```
qda.model = qda(Species ~ ., data = iris)
qda.model.prediction <- predict(qda.model)$class
table(predicted=qda.model.prediction,actual=iris$Species)
correct = sum(qda.model.prediction == iris$Species)
incorrect = sum(qda.model.prediction != iris$Species)
```

- Be able to apply a suitable methods of cross validation to measure performance of different classification methods and optimise models (e.g. to compare different fitting models).

```
lda.model = lda(Species ~ ., data = iris, CV=TRUE)
lda.model.prediction <-lda.model$class
table(predicted=lda.model.prediction,actual=iris$Species)
correct = sum(lda.model.prediction == iris$Species)
incorrect = sum(lda.model.prediction != iris$Species)
```

K- Nearest Neighbours algorithm

- Be able to describe the principle of the K-nearest neighbours algorithm for classification and regression problems.

```
library(dplyr)
iris.X <- mutate(iris, pl=normalise(Petal.Length), pw=normalise(Petal.Width))
set.seed(1)
library(dplyr)
train.X = iris.X[,c(6,7)]
train.Y = iris.X[,5]
test.X = train.X
knn.pred=knn(train.X,test.X,train.Y,k=10)
table(predicted=knn.pred,actual=train.Y)
x1_seq = seq(0,1,0.01)
x2_seq = seq(0,1,0.01)
gridpoints <- NULL
for (i in x1_seq){
  for (j in x2_seq){
    gridpoints<-rbind(gridpoints,c(i,j))
  }
}
gridpoints=data.frame(gridpoints)
colnames(gridpoints) <- c("pl", "pw")
knn_grid.pred=knn(train.X,gridpoints,train.Y,k=10)
```

gridpoints$Species<-as.character(knn_grid.pred)

plot(pw~pl,col= Species,pch=20, data=iris.X)

points(pw~pl,pch='.',col= unclass(knn_grid.pred), data=gridpoints)

- Understand KNN requires consideration of the scale of each variable used as a predictor, and how to apply normalisation when appropriate.

```
rescale_x <- function(x){
  x <- x-min(x)
  x <- x/max(x)
  return(x)
}
```

- Be able to explore how the performance of the KNN method varies with K for a particular dataset and select an optimal K value using a cross validation method.

```
# testing different k values
k_vals = 1:95
n = length(k_vals)
results <- data.frame(k_val=rep(0,n), mcr=rep(0,n))
for (i in 1:n){

  knn_test$pred <- knn(knn_train_x, knn_test_x, knn_train_y, k =
k_vals[i] )

  table(predicted =knn_test$pred, actual =  knn_test$Y)
  results$mcr[i] <- sum(knn_test$pred!=knn_test$Y)/nrow(knn_test)
  results$k_val[i] <- k_vals[i]
}
plot(mcr~k_val,data=results)
```

Comparisons of Fitting methods

- Understand that models can be used for both inference and prediction.
- Be able to carry out suitable cross validation testing to measure the performance of different fit methods and evaluate the results.
- Be able to describe the different fitting methods considered in terms of their assumptions, and relative usefulness for making predictions/inferences.

Working with R code files and notebooks

- Be able to create and run code in R scriptfiles (.R extension)
- Be able to write a report in R Markdown as a notebook, that consists of code sections and appropriately styled text (headers, inline code, font styles).

Learning Objectives: Data Science & Big Data Analytics 2019

- Know how to work with code that is contained in an R Notebook in RStudio.
- Be able to knit the notebook into HTML formatted report.