**The Analytics Edge Mid-Term Cheat Sheet**

James Gan | 1003330

**Week 1 – Familiarization with R Syntax**

& - and

| - or

**1.1 List Operations**

x <- c(2,4,6)

* Count elements in list: ***length(x)***
* Sum elements in list: ***sum(x)***
* Generate a list of numbers:

***seq(start\_no, end\_no, by=step\_no, length = length\_no)***

We declare the start\_no and end\_no, which are inclusive, and optional parameters. When we set the length of list, the step\_no will be decided. And when we set the step\_no, the length\_no will be decided for us.

* Generate repeat numbers in list:

***rep(number, repeat\_no)***

*rep(2,4)*

*>>> 2 2 2 2*

We can also concatenate and repeat a few numbers

*rep(c(1,2), c(4,4))*

*>>> 1 1 1 1 2 2 2 2*

* Extract certain value from list

*x[c(2,5,7)]* : Extract values from list from index 2, 5,7

*x[2:4]* : Extract values from list from index 2 to 4

**1.2 Matrix Operations**

* Initializing a matrix

**x <- matrix(c(3,-1,2,-1), nrow=2, ncol=2)**

>>>

[,1] [,2]

[1,] 3 2

[2,] -1 -1

* Elementwise multiplication

***2 \* x***#Multiply 2 to every element of matrix

**x \* x** #Multiply each element with itself in the matrix

**x %\*% x** #Matrix Multiplication

* Transpose Matrix

**t(x)**

* Indexing Matrix

**x[row\_no, column\_no]**

* Index Multiple Column matrix

**x[, start\_column:end\_column]**

**1.3 DataFrame Operations**

* DataFrame Statistics

***summary(df)***#Statistics Output of all columns

***str(df)***#Output the character type and first few values

* Value\_count:

**table(df$column\_name)**

* Count null value in that column:

**sum(is.na(df$column\_name))**

* Count value with function: tapply()

The function allows us to apply various calculation such as mean, table(sum). Output is an array that we can index.

**tapply(df$column\_name, df$index\_column\_name, function)**

**tapply(df$column\_name, df$index\_column\_name, function, na.rm=TRUE)**

* Count number of rows that satisfies Condition

**nrow(subset(df,** **column\_name == 0 & column\_name\_2 > 10))**

**nrow(subset(df, column\_name == 0))**

* Take a portion of original dataframe: subset()

1)Taking a subset of dataframe with condition

**new\_df <- subset(df, column\_name == 0 & column\_name\_2 > 10))**

2) Taking a subset of dataframe w/o certain columns

**new\_df<-subset(df,select=-c(column\_name))**

* Number of rows in dataframe

**nrows(df)**

* Mean value of column

**mean(df$column\_name)** #Including count of na rows

**mean(df$column\_name, na.rm=TRUE)** #Removing count of na rows

* Summation of values of column with condition

**sum(df$column\_name == 10)**

* Largest number of rows that has same value for two columns

**max(table(****df$column\_name\_1, df$column\_name\_2))**

#Finds the max value of the value count of column\_name\_1 and column\_name\_2

**1.4 Plot Visualizations**

* Simple Histogram Plot

**hist(df$column\_name)**

or

**library("ggplot2")**

**ggplot(df)+geom\_histogram(aes(column\_name),na.rm=T,binwidth=5,color="black",fill="lightblue")**

* Preventing points from overlapping each other: jitter()

**plot(jitter(df$****column\_name\_1), jitter(df$column\_name\_2))**

or

**ggplot(df, aes(column\_name\_1, column\_name\_2)) +geom\_point(position="jitter",na.rm=T)**

* Star Plot:

**star(df, key.loc=c(18,2))**

#the key.loc is the coordinate position of the legend

If we want to create a linear plot with the points as star visualization.

We first select the data for star plot as matrix, then the x, y axis as the location then the labels.

**stars(as.matrix(swiss[,c(2,3,5,6)]), location = as.matrix(swiss[,c(4,1)]), axes = T, labels = NULL, len = 3, main = "Fertility against Education", xlab = "Education", ylab = "Fertility")**

**1.5 Histogram**

* Plot 2 diff histogram based on another column category

We do this by setting the column name of facet\_grid()

**ggplot(data = Parole, aes(x = Age)) + geom\_histogram(binwidth=5,closed=c("left"),center=17.5,color=c("blue"))+facet\_grid(Male~.)** #Top-Bottom Plot

**ggplot(data = Parole, aes(x = Age)) + geom\_histogram(binwidth=5,closed=c("left"),center=17.5,color=c("blue"))+facet\_grid(.~Male)** #Adjacent Plot

* Plot the category as values within histogram bar

**ggplot(data = Parole, aes(x = Age, fill = as.factor(Male))) + geom\_histogram(binwidth=5,closed=c("left"),center=17.5,color=c("blue"))**

**Week 2 – Linear Regression**

**2.1 Data Pre-Processing**

* Convert column to numeric from character data type

**df$column\_name <- as.numeric(as.character(df$column\_name)**

**2.2 Data Modelling**

* Model Fitting for linear model

**model1<- lm(dependent\_variable~column\_01, data=df)**

**summary(model1)**

* Confidence level of coefficients of regression

**confint(model1, level = 0.99)**

* Predict dependent variable with confidence level

**predict.lm(model1, newdata=data.frame(column\_name == “something”), interval =c(“confidence”), level=.99)**

**2.1 Data Analysis**

* Compute Correlation: cor()

The pearson correlation function here excludes the “NA” entries

**cor(****df$****column\_1, df$column\_2, use = "pairwise.complete.obs")^2**

* **Chart, scatter chart

  Description automatically generated**Plot 1 predictor variable against 1 dependent variable with least square regression line

**library(ggplot2)**

**ggplot(df,aes(column\_1,column\_2\_dependent))+geom\_point(na.rm=T)+geom\_smooth(method="lm",na.rm=T,se=F)**

* ggfortify: Residual, QQ plot and other stats plots

**Chart, scatter chart

Description automatically generated**

**library(ggfortify)**

**autoplot(model1,label.size = 3)**

* Scatterplot Matrix: pairs.panel()

This method outputs correlation plots and correlation coefficients

**A picture containing table

Description automatically generatedpairs.panels(df,ellipses = F, lm =T, breaks=10, hist.col="blue")**

* Multi-Collinearity

There’s the issue of multi-collinearity in the model where we have 2 independent variable that is highly correlated to one another, therefore 1 of them becomes statistically insignificant.

Therefore, we perform least squares regression to determine what variable to remove,

H0: βj == 0

We want to reject the null hypothesis to show that this particular variable is statistically significant. We can reject when p\_value < α.

* Hypothesis Testing

We can test the statistical significance of all the different independent variable by plotting each one against the predictor variable.

H0 : βj == 0

Ha : βj != 0

If we reject the null hypothesis, this suggest that the independent variable is statistically significant. Where the p value < level of significance.

* Polynomial Model

In addition to linear model, we can use the lm() function to plot polynomial functions:

**modelpoly1 <- lm(predictor\_var~poly(independent\_var,2,raw=TRUE), data = df)** #We set the polynomial to power 2 here

**summary(modelpoly1)**

* Prediction and SSE, SST, and R^2

**test<-subset(wine,vintage>=1982)**

**predtest<-predict(model1,newdata=test)**

**sse<-sum((log(test$price91)-predtest)^2)**

**sst<-sum((log(test$price91)-mean(log(train$price91)))^2)**

**testR2<- 1-sse/sst**

**testR2**

* Sffsdfsd
* Fdsfsdfs
* dfs