Contents

---

elephants' graveyard of frequently used (and subsequently forgotten) code

tidyverse

Using the faraway package in R: Documentation

```
#install.packages("faraway")
library(faraway)
data(seatpos)
head(seatpos)
```

dplyr functions

filter, select, mutate, group by, summarise

```
seatpos %>%
  filter(Age < 60 & Weight < 200) %>%
  select(Age, Weight) %>%
  mutate(decades = case_when(Age <=30 ~ "twenties",
                             Age <= 40 & Age >30 ~ "thirties",
                             Age <= 50 & Age >40 ~ "forties",
                             Age <= 60 & Age > 50 ~ "fifties")) %>%
  group_by(decades) %>%
  summarise(mean_Weight = mean(Weight))
```

gather

gather columns into rows

```
data("mtcars")
head(mtcars)
mtcars <- rownames_to_column(mtcars, "Model") %>%
  select(mpg, cyl, Model) %>%
  filter(cyl == 4)

gather(mtcars, "attribute", "this", 1:2)
```

spread

spreads rows into columns

```
data(mtcars)
mtcars <- rownames_to_column(mtcars, "Model") %>%
  select(mpg, cyl, Model)

spread(mtcars, key = 'cyl', value = 'mpg')
```

linear modeling

linear models

anova

```
### one way anova
library(faraway)
data(coagulation)
head(coagulation)

# boxplot
plot(coag ~ diet,coagulation,xlab="Diet",ylab="Coagulation Time")

# anova
anova(lm(coag ~ diet,coagulation))

# pairwise
tk_coag <- TukeyHSD(aov(coag ~ diet,coagulation))
tk_coag
plot(tk_coag)
```

linear regression

```
# multiple regression
library(faraway)
data(savings)
head(savings)
```

```r
## overall test
m1 <- lm(sr ~ pop15 + pop75 + dpi + ddpi,savings)
summary(m1)

# f stat for the overall test
ssy <- sum((savings$sr - mean(savings$sr))^2) # sum square y
sse <- deviance(m1) # residual sum of squares
df.residual(m1) # degress of freedom for the sse
fstat <- ((ssy-sse)/4)/(sse/df.residual(m1)) # F statistic for the overall test
1 - pf(fstat,4,df.residual(m1)) # p value

## partial f test

m2 <- lm(sr ~ pop75 + dpi + ddpi,savings) # reduced model
sse2 <- deviance(m2) # SSE for reduced model
fstat2 <- (deviance(m2)-deviance(m1))/(deviance(m1)/df.residual(m1))
1-pf(fstat2,1,df.residual(m1))
```

interactions

```r
library(faraway)
data(teengamb)

# income and sex on outcome
m1 <- lm(gamble ~ income + sex,teengamb)
#income, sex, and interaction between both
m2 <- lm(gamble ~ income + sex + income*sex,teengamb)
```

correlation

```r
library(faraway)
data(stat500)

m1 <- lm(final ~ midterm,stat500)
summary(m1)
sum(coef(m1)* c(1,83))
# confidence interval for beta 1
confint(m1)

# correlation

cor(stat500$final,stat500$midterm)
```

functions and loops

sample for loop

```
data(GSSvocab)
EGE<-rep(0,500)
for (i in 1:500)
{
  temp0<-na.omit(GSSvocab)
  index<-sample(1:27360,27360,replace=T) # random index
  temp1<-temp0[index,] # random sample of all data
  NewTrain<-temp1[1:13680,] # training data
  NewTest<-temp1[13681:27360,] # test data
  out<-lm(vocab~gender+nativeBorn+ageGroup+
            educGroup+age+educ,data=NewTrain)
  EGE[i]<-var(NewTest$vocab-predict.lm(out, newdata=NewTest))
}
summary(EGE)
hist(EGE,breaks = 20)
qqnorm(EGE)
quantile(EGE,probs=c(.025,.975))
```

sample function

```
clean_live_frame_function <- function(livedata, albumname)  {
  artist_live_edits <- livedata %>%
    mutate(livemarker = ifelse(livedata$album_name == albumname, 1, 0))
  artist_live_edits2 <- artist_live_edits %>%
    mutate(omitvariable = case_when(
      (str_detect(album_name, "Sessions") |
         str_detect(album_name, "Live") |
         str_detect(album_name, "Festival") |
         str_detect(album_name, "Concerts"))& livemarker == 0 ~ "omit"))
 artist_live_edits2 <- artist_live_edits2 %>%
    mutate(omitvariable2 = case_when(
      (str_detect(track_name, "Live") & livemarker == 0 ~"omit")
    ))
 artist_live_edits2 <- artist_live_edits2 %>%
   filter(is.na(omitvariable) & is.na(omitvariable2))
 artist_live_edits2 <- artist_live_edits2 %>%
   distinct()
}
```

numpy

reticulate

```
#install.packages("reticulate")
library(reticulate)
use_python("/usr/bin/python")
```

```
#best way to install pandas may be to use this py_install function when in reticulate mode
py_install("pandas", pip = TRUE)
py_install("sklearn", pip = TRUE)
py_install("matplotlib", pip = TRUE)
```

arrays

```
#basic array
array_1d = np.array([10, 11, 12, 13])
print (array_1d)
#array from a list
list_1 = [11, 22, 33, 44, 55]
array_l1 = np.array(list_1)
print(array_l1)
#create 1D array
array_1d = np.array([6,4,4,4,4,5,5,5,5])
print (array_1d)
#remember that python starts at 0
print('element at index1:', array_1d[0])
```

pandas

basic

```
#summarize mean values
df.groupby(['sepal width (cm)']).mean()

#duplicates
df.shape
df.drop_duplicates

#rename columns
df2 = df.rename({'petal width (cm)': 'Width of Petals'})
print(df2)

#check null values
df2.isnull()
df2.isnull().sum()

#drop missing values
df2.dropna(axis =1)
```

more

```
#describe all values
df2.describe()

#count row totals
df2['Width of Petals'].value_counts().head(30)
```

```
#correlation
df2.corr()

#conditional selection
df2[df2['Width of Petals'] >= 2.3]

#complex conditional seection
df2[(df2['sepal length (cm)'] == 6.3) | (df2['sepal length (cm)'] == 6.4) ]
```

scikit-learn

train/test

```
#training and testing
X = df['sepal length (cm)'].values.reshape(-1,1)
y = df['sepal width (cm)'].values.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training the algorithm
#intercept
print(regressor.intercept_)
#slope
print(regressor.coef_)
```

evaluation

```
#testing
y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
df

#evaluation
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

### for multiple regression

X = df[['sepal width (cm)', 'sepal length (cm)']].values
```