# IBEHS Stats Coding Reference Sheet

## Introduction

This coding reference sheet was adapted from the "Cheat sheet for Linear Algebra and Differential Equations"created by Drew Ulick under the Creative Commons CC BY 4.0 license. The hope is that this sheet acts as a resource for useful code that can be utilized throughout the duration of this course. The references used to develop this resource can be found at the end of the sheet if you want to explore more!

## Python Basics

**Data Types:**

Integer int(): x=11
Float float(): x=11.4
Boolean bool(): x=True
String str(): x="Hello There"
List list(): x=["I", 1, True]
Dictionary dict(): x={"name": "Bob","Grade":4}

**Libraries:**

The following shows how to import useful libraries and their conventional aliases:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

**NumPy:**

| | |
|---|---|
| Create an Array | *var=np.array([1,2,3])* |
| Populated Array [a,b) | *var=np.arange(a,b,step)* |
| Exponential $e^x$ | *var=np.exp(x)* |
| Squareroot $\sqrt{x}$ | *var=np.sqrt(x)* |
| Summation $\sum x$ | *var=np.sum(x)* |

## Data Manipulation

**Reading and Storing Data:**

Let df represent a Pandas dataframe:

| | |
|---|---|
| Read a .csv File | *df=pd.read_csv("File Name.csv")* |
| np.array to df | *df=pd.DataFrame(data)* |
| Create a Series | *var=pd.Series([1,2],name="A")* |
| Dataframe Info | *df.info()* |

## Data Manipulation (Cont.)

**Manipulation:**

Let A, B, C be labels for a data set and let a and b be numbers:

| | |
|---|---|
| Drop a Column | *df1=df.drop(columns=["A"])* |
| Return Rows [a,b) | *df1=df.iloc[a:b]* |
| Query Dataframe | *df1=df.query("a<A<b")* |
| Convert Type | *df.astype("int32")* |
| Convert to Numeric | *pd.to_numeric(df)* |
| Convert to Datetime | *pd.to_datetime(df)* |
| Categorical | *pd.Categorical(var)* |
| Remove Null Values | *df1=df.dropna()* |
| Reset Index | *df1=df1.reset_index()* |
| Find Unique Values | *df.A.unique()* |

## Statistics Basics

**Central Tendency Measures:**

| | |
|---|---|
| Mean | *np.mean(data)* |
| Median | *np.median(data)* |
| Mode | *np.mode(data)* |
| Standard Deviation | *np.std(data,ddof=0)* |
| Variance | *np.var(data,ddof=0)* |
| 5 Number Summary | *df.describe()* |
| Percentile | *np.percentile(data,percentile)* |

## Visualizations

**Box Plots:**

```
sns.boxplot(data=var) #Basic
plt.boxplot(data,labels=["A","B"]) #Alternative
```

The data can be grouped for presentation in many different ways:

```
sns.boxplot(data=var,hue="C")
sns.boxplot(data=var,x="A",y="B")
sns.boxplot(data=var,x="A",y="B",hue="C")
```

The whis parameter can be changed to adjust whisker length, which will be calculated as whis*IQR :

```
sns.boxplot(data=var,x="A",y="B",whis=1.5)
```

## Visualizations (Cont.)

**Histograms:**

There are 5 different types of histograms that can be created by changing the stat parameter:

```
sns.histplot(data=var,x="A", stat="count")
sns.histplot(data=var,x="A", stat="percent")
sns.histplot(data=var,x="A", stat="density")
sns.histplot(data=var,x="A", stat="frequency")
sns.histplot(data=var,x="A", stat="probability")
```

Change the number of bins using the bins parameter:

```
sns.histplot(data=var,x="A", stat="count", bins=10)
```

**Bar Plots:**

```
sns.barplot(data,x="A",y="B")
sns.barplot(data, errorbar="sd")#Error bars
sns.barplot(data, estimator="mean")#Use mean
```

**Other Useful Functions:**

| | |
|---|---|
| Swarm Plot | *sns.swarmplot(data=var)* |
| Scatter Plot | *sns.scatterplot(data=var)* |
| Distribution Plot | *sns.distplot(data=var)* |
| Strip Plot | *sns.stripplot(data=var)* |
| Violin Plot | *sns.violinplot(data=var)* |
| Line Plot | *sns.lineplot(data=var)* |
| X Label | *plt.xlabel("A")* |
| Y Label | *plt.ylabel("B")* |
| Title | *plt.title("C"))* |
| Legend | *plt.legend()* |
| Show Plot | *plt.show()* |

## Probability

To use the relevant probability functions with more ease import the following:

```
from scipy.stats import norm
from scipy.stats import binom
from scipy.stats import poisson
from scipy.stats import uniform
from scipy.stats import expon
```

## Probability (Cont.)

The survival function (SF) outputs 1-CDF. The percentile poinf function (PPF) outputs the inverse of the CDF percentiles.
For a normal distribution, loc is the mean and scale is the standard deviation:

**Normal Distribution:**

| | |
|---|---|
| PDF | *norm.pdf(x,loc=a,scale=b)* |
| CDF | *norm.cdf(x,loc=a,scale=b)* |
| SF | *norm.sf(x,loc=a,scale=b)* |
| PPF | *norm.ppf(q,loc=a,scale=b)* |
| Z-Score | *stats.zscore(data)* |

For a binomial distribution where n is number of trials, k is successes, and p is probability of a success:

**Binomial Distribution:**

| | |
|---|---|
| PMF | *binom.pmf(k,n,p)* |
| CDF | *binom.cdf(k,n,p)* |
| SF | *binom.sf(k,n,p)* |
| PPF | *binom.ppf(q,n,p)* |

For a Poisson distribution, k is the number of events and $\mu$ is the rate:

**Poisson Distribution:**

| | |
|---|---|
| PMF | *poisson.pmf(k,$\mu$)* |
| CDF | *poisson.cdf(k,$\mu$)* |
| SF | *poisson.sf(k,$\mu$)* |
| PPF | *poisson.ppf(k,$\mu$)* |

## Hypothesis Testing

**Tests for Normality:**

A QQ plot can be used to show normality:

```
#Basic
sm.qqplot(data,line="45",loc=mean,scale=std)
#Alternative
import pingouin as pg
pg.qqplot(data,dist="norm")
```

Other tests of normality:

```
stats.shapiro(data)
stats.normaltest(data)#D'Agastino & Pearson's
```

**Tests for Equal Variance:**

```
stats.levene(dataA,dataB)
```

## Hypothesis Testing (Cont.)

Some tests require the following import:

```
import statsmodels.stats.weightstats as sms
```

**Parametric Tests:**

| | |
|---|---|
| Z-Test (1-Samp.) | *sms.ztest(x1=var1)* |
| Z-Test (2-Samp.) | *sms.ztest(x1=v1,x2=v2)* |
| T-Test (1-Samp.) | *stats.ttest_1samp(data,a)* |
| Student's T-Test | *stats.ttest_ind(x1=v1,x2=v2)* |
| Paired T-Test | *stats.ttest_rel(v1,v2)* |
| T-Dist. PDF | *stats.t.pdf(x,df=a)* |
| T-Dist. CDF | *stats.t.cdf(x,df=a)* |
| T-Dist. PPF | *stats.t.ppf(q,df=a)* |

**Change Variance Parameters:**

```
sms.ztest(x1=v1,x2=v2,usevar="pooled")
sms.ztest(x1=v1,x2=v2,usevar="unequal")
stats.ttest_ind(v1,v2,equal_var="True")
stats.ttest_ind(v1,v2,equal_var="False")#Welch's
```

**Change Alternative Hypothesis:**

```
sms.ztest(x1=v1,x2=v2,alternative="smaller")
sms.ztest(x1=v1,x2=v2,alternative="larger")
stats.ttest_ind(v1,v2,alternative="two-sided")
stats.ttest_ind(v1,v2,alternative="less")
stats.ttest_ind(v1,v2,alternative="greater")
```

**Generate Confidence Intervals:**

```
#Z CI
sms.zconfint(x1=v1,value=a)
#T CI
sms._tconfint_generic(mean,SEM,dof,alpha,'two-sided')
#Difference of Means
import statsmodels.stats as ss
v1=ss.DescrStatsW(data1)
v2=ss.DescrStatsW(data2)
cm=ss.CompareMeans(v1,v2)
cm.tconfint_diff(usevar="pooled")
```

**Non-Parametric Tests:**

```
#Mann-Whitney U
stats.mannwhitneyu(v1,v2,use_continuity=False)
#Wilcoxon Signed Rank Test
stats.wilcoxon(v1,v2)
```

## Hypothesis Testing (Cont.)

**Power and Sample Size:**

The following function will solve for either effect size, number of observations, alpha, power, or ratio, depending on which parameter is not specified:

```
import statsmodels.stats.power as smp
#One Sample Independent Z-Test
smp.zt_ind_solve_power(effect_size=a,alpha=b,
                       nobs1=c,ratio=0)
#One Sample Independent T-Test
smp.tt_ind_solve_power(effect_size=a,alpha=b,
                       nobs1=c,ratio=0)
#Two Sample Independent Z-Test
smp.zt_ind_solve_power(effect_size=a,alpha=b,
                       nobs1=c,ratio=1)
#Two Sample Independent T-Test
smp.tt_ind_solve_power(effect_size=a,alpha=b,
                       nobs1=c,ratio=1)
```

## Regression

**Correlation and Covariance:**

| | |
|---|---|
| Covariance | *np.cov(data1,data2)* |
| Correlation | *np.corrcoef(data1,data2)* |
| Pearson Coeff | *stats.pearsonr(data1,data2)* |
| Spearman Coeff | *stats.spearmanr(data1,data2)* |
| Correlation Heatmap | *sns.heatmap(corr_matrix)* |
| Pair Plot | *sns.pairplot(data=var)* |

**Regression:**

| | |
|---|---|
| Create Model | *model=ols("Y~X",data=var)* |
| Fit Model | *results=model.fit()* |
| Model Summary | *results.summary2()* |
| Regression Plot | *sns.regplot(data=var)* |
| Residual Plot | *sns.residplot(data=var)* |
| Autocorr. Plot | *plt.acorr(data,maxlags=a)* |

**ANOVA:**

```
from statsmodels.stats.anova import anova_lm
anova_results=anoval_lm(results)
stats.f_oneway(data1,data2,data3)
```

**Confidence and Prediction Intervals:**

```
predictions=results.get_predictions(exog=dict(x="a"))
predictions.summary_frame(alpha=0.05)
```

## Regression (Cont.)

**Matrix Calculations:**

| | |
|---|---|
| Transpose | *np.transpose(M)* |
| Dot Product | *M1.dot(M2)* |
| Matrix Shape | *M.shape* |
| Inverse | *np.linalg.inv(M)* |
| Reshape | *np.reshape(X1,(a,b))* |
| Append | *np.append(X1,X2,axis=1)* |

**Regression Model Variations:**

```
#Multiple Predictors
model2=ols("Y~X1+X2",data=var)
#Multiple Predictors and Interaction Term
model3=ols("Y~X1*X2",data=var)
#Polynomial Term (X1^3)
model4=ols("Y~X1+I(X1*X1*X1)",data=var)
#Categorical Predictor
model5=ols("Y~X1+C(X3)",data=var)
```

**Transformations Examples:**

```
#Polynomial
model1=ols("Y~np.power(X1,5)",data=var)
#Log-Log
model2=ols("np.log(Y)~np.log(X1)",data=var)
#Reciprocal
model3=ols("Y~np.power(1/X1,1)",data=var)
```

## Design of Experiments

```
#Create Matrix and Model
X1=pd.Series([1,-1,1],name="X1")
X2=pd.Series([1,-1,-1],name="X2")
Y=pd.Series([12,10,12],name="Y")
df=pd.concat([X1,X2,Y],axis="columns")
model=ols("Y~X1+X2+X1*X2",data=df)
result=model.fit()
#Contour Plot
v1,v2,v3=model.exog_names[1:]
fig=plt.figure()
ax=fig.add_subplot(111)
x_surf=np.arange(-1,1,.1)
y_surf=np.arange(-1,1,.1)
xS,yS=np.meshgrid(x_surf,y_surf)
exog=pd.DataFrame({V1:xS.ravel(),V2:yS.ravel()})
zS=result.predict(exog=exog).values.reshape(xS.shape)
contour=ax.contourf(xS,yS,zS,cmap="coolwarm")
fig.colorbar(contour,shrink=0.5,aspect=5)
```

## References


[1] https://numpy.org/doc/stable/user/index.html
[2] https://pandas.pydata.org/docs/user_guide/index.html
[3] https://seaborn.pydata.org/api.html
[4] https://matplotlib.org/stable/users/index
[5] https://docs.scipy.org/doc/scipy/reference/index.html
[6] https://www.statsmodels.org/stable/user-guide.html
[7] https://www.w3schools.com/python/default.asp
[8] https://github.com/cbassim/Summer24_IBEHS4C03/tree/main