Cheat Sheet for comprehensive HarvardX Data Science Professional Certificate

Data Science Tools

R Programming

- **RStudio Shortcuts**

- `Ctrl + Enter`: Run current line or selection
- `Ctrl + Shift + M`: Pipe operator (`%>%`)
- `Alt + -`: Assignment operator (`<-`)
- `Ctrl + Shift + C`: Comment/uncomment lines

- **R Packages**

- `tidyverse`: Collection of packages for data manipulation and visualization
- `dplyr`: Data manipulation (filter, select, mutate, summarize)
- `ggplot2`: Data visualization
- `caret`: Machine learning
- `rmarkdown`: Creating reports

Python Programming

- **Jupyter Notebook Shortcuts**

- `Shift + Enter`: Run cell and move to next
- `Ctrl + Enter`: Run cell
- `Esc + M`: Markdown mode
- `Esc + Y`: Code mode

- **Python Libraries**

- `pandas`: Data manipulation and analysis
- `numpy`: Numerical computing
- `matplotlib`: Data visualization
- `seaborn`: Statistical data visualization
- `scikit-learn`: Machine learning

Data Manipulation

R (dplyr)

- **Basic Functions**

- `filter()`: Select rows based on conditions

- `select()`: Select columns
- `mutate()`: Create or transform variables
- `summarize()`: Summarize data
- `group_by()`: Group data by one or more variables

- **Examples**

```
library(dplyr)
data %>%
  filter(age > 30) %>%
  select(name, age) %>%
  mutate(age_group = ifelse(age > 40, "Old", "Young")) %>%
  group_by(age_group) %>%
  summarize(mean_age = mean(age))
```

Python (pandas)

- **Basic Functions**

- `df.loc[]`: Select rows and columns by label
- `df.iloc[]`: Select rows and columns by integer position
- `df.query()`: Query data using expressions
- `df.groupby()`: Group data by one or more columns
- `df.pivot_table()`: Create pivot tables

- **Examples**

```
import pandas as pd
df.loc[df['age'] > 30, ['name', 'age']]
df.iloc[0:5, 0:2]
df.query('age > 30')
df.groupby('age_group')['age'].mean()
df.pivot_table(index='age_group', values='age', aggfunc='mean')
```

Data Visualization

R (ggplot2)

- **Basic Structure**

```
ggplot(data, aes(x = var1, y = var2)) +
  geom_point() +
  labs(title = "Title", x = "X-axis", y = "Y-axis")
```

- **Common Geoms**

- `geom_point()`: Scatter plot
- `geom_line()`: Line plot
- `geom_bar()`: Bar plot
- `geom_histogram()`: Histogram
- `geom_boxplot()`: Box plot

Python (matplotlib & seaborn)

- **Basic Structure**

```
import matplotlib.pyplot as plt
plt.plot(x, y)
plt.title("Title")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

- **Common Plots**

- `plt.scatter()`: Scatter plot
- `plt.plot()`: Line plot
- `plt.bar()`: Bar plot
- `plt.hist()`: Histogram
- `sns.boxplot()`: Box plot

Machine Learning

R (caret)

- **Basic Workflow**

- `train()`: Train a model
- `predict()`: Make predictions
- `confusionMatrix()`: Evaluate model performance

- **Examples**

```
library(caret)
model <- train(y ~ ., data = train_data, method = "lm")
predictions <- predict(model, newdata = test_data)
confusionMatrix(predictions, test_data$y)
```

Python (scikit-learn)

- **Basic Workflow**

- `train_test_split()`: Split data into training and testing sets

- `model.fit()`: Train a model
- `model.predict()`: Make predictions
- `metrics.confusion_matrix()`: Evaluate model performance

- **Examples**

```
  from sklearn.model_selection import train_test_split
  from sklearn.linear_model import LinearRegression
  from sklearn.metrics import confusion_matrix

  X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
  model = LinearRegression()
  model.fit(X_train, y_train)
  predictions = model.predict(X_test)
  confusion_matrix(y_test, predictions)
```

Reporting and Communication

title: "Report Title"

output: html_document

Introduction

This is a report.

```
  summary(data)
```

```
#### Python (Jupyter Notebook)
- **Basic Structure**
```

Report Title

Introduction

This is a report.

```
  data.describe()
```

ver 1.0