ACLR Final Project

Rachel, Elaine, and Yuthi

Table of contents

Report

Audience

Our final report's audience/stakeholders include physicians and other researchers evaluating the health of patients who have undergone ACL reconstruction surgery.

Problem Statement

After athletes tear their anterior cruciate ligament (ACL), many undergo ACL reconstruction (ACLR) surgery. However, physicians and other researchers are still evaluating the recovery and health of patients who have undergone ACLR. It is important to note that a majority of patients do not get reinjured - in this study, 83% had no reinjuries after ACLR surgery. There are many features that can affect reinjury rates after the surgery, such as the gender of the patient, the graft type used, and even their mental readiness. Specifically, our stakeholders are interested in all of these factors and what combination will lead to the lowest reinjury rates in patients after ACLR surgery is performed.

Analysis

[insert graph analyses]

Conclusion

[insert conclusion]

1 Data Cleaning Outline:

Documentation for our data cleaning process, including decisions regarding how we handle missing values, outliers, and other data quality issues.

First, we import the necessary libraries and set the dataset which is a .csv file provided by the UVA School of Data Science and the UVA Department of Kinesiology as a pandas dataframe.

```
# Setting up our environment, importing all necessary libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Importing dataset as dataframe:
df = pd.read_csv('aclr data(in).csv')
```

```
# Previewing the dataframe:
df.head()
```

	record_id	redcap_event_name	redcap_repeat_instrument	sex_dashboard	graft_dashboa
0	1	baseline_arm_1	NaN	Male	Other
1	1	visit_1_arm_1	NaN	NaN	NaN
2	1	long_term_outcomes_arm_1	NaN	NaN	NaN
3	2	baseline_arm_1	NaN	Female	HS autograft
4	2	visit_1_arm_1	NaN	NaN	NaN

```
# Checking the dimensions of the dataframe:
print(df.shape)
```

```
(11150, 63)
```

The original dataframe has 11150 observations and 63 columns. We will be focusing on the variables we feel are most relevant to our hypothesis. We will be using the columns: `sex_dashboard`, `graft_dashboard2`, `reinjury`, `age`, `height_m`, `mass_kg`, `bmi`, `ikdc`, `acl_rsi` and dropping the rest from the dataframe.

```
df = df[['sex_dashboard', 'graft_dashboard2', 'reinjury', 'age', 'height_m', 'mass_kg', 'bmi
df.head()
```

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc	acl_rsi
0	Male	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	No	21.7	1.9	87.4	24.210526	95.4	87.5
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Female	HS autograft	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	No	14.5	1.6	72.2	28.203125	79.3	8.3

Now that we have our columns of interest, we will first check for missing values across the dataset. We will use the `isnull()` method to check for missing values and the `sum()` method to get the total number of missing values in each column, as well as the percentage of missing values in each column.

```
# Checking for missing values:
missing_values = df.isnull().sum()

# Checking the percentage of missing values:
missing_percentage = (missing_values / len(df)) * 100
# Displaying missing values and their percentage:
missing_values = pd.DataFrame({'Missing Values': missing_values, 'Percentage': missing_percen

# Displaying the missing values:
print(missing_values)
```

```
                  Missing Values  Percentage
sex_dashboard               6413   57.515695
graft_dashboard2            6413   57.515695
reinjury                    5975   53.587444
age                         6024   54.026906
height_m                    8632   77.417040
mass_kg                     7899   70.843049
bmi                         8633   77.426009
ikdc                        8199   73.533632
acl_rsi                     7750   69.506726
tss_dashboard               5913   53.031390
```

Now we will proceed by separating the variables into categorical and continuous variables. We will use the `select_dtypes()` method to select the categorical variables and the continuous variables. For our numerical variables, we will impute missing values with the respective mean for each column.

```
# Filtering for numeric columns:
numeric_columns = df.select_dtypes(include=['int', 'float']).columns

# Imputing missing values with the mean for each respective column/varibale:
mean_values = df[numeric_columns].mean()
m_df = df.fillna(mean_values)

# Displaying the first 5 rows of the modified dataframe:
(m_df.head(5))
```

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc
0	Male	Other	NaN	20.184761	1.725412	74.343033	25.201579	78.457377
1	NaN	NaN	No	21.700000	1.900000	87.400000	24.210526	95.400000
2	NaN	NaN	NaN	20.184761	1.725412	74.343033	25.201579	78.457377
3	Female	HS autograft	NaN	20.184761	1.725412	74.343033	25.201579	78.457377
4	NaN	NaN	No	14.500000	1.600000	72.200000	28.203125	79.300000

For our categorical variables, we have decided to fill the missing values with just an `Unknown` category, since this allows us to keep the rows with missing values without losing too much information so that we can continue with plotting later on.

```
# Filtering for Categorical columns:
categorical_columns = df.select_dtypes(include=['object']).columns
# Imputing missing values with the value 'Unknown' for each respective column/variable:
for column in categorical_columns:
    m_df[column] = m_df[column].fillna('Unknown')

# Displaying the first 5 rows of the modified dataframe:
(m_df.head(5))
```

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc
0	Male	Other	Unknown	20.184761	1.725412	74.343033	25.201579	78.45737

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc
1	Unknown	Unknown	No	21.700000	1.900000	87.400000	24.210526	95.40000
2	Unknown	Unknown	Unknown	20.184761	1.725412	74.343033	25.201579	78.45737
3	Female	HS autograft	Unknown	20.184761	1.725412	74.343033	25.201579	78.45737
4	Unknown	Unknown	No	14.500000	1.600000	72.200000	28.203125	79.30000

Now we have finished our early data cleaning process and are ready to explore relations in our EDA process.

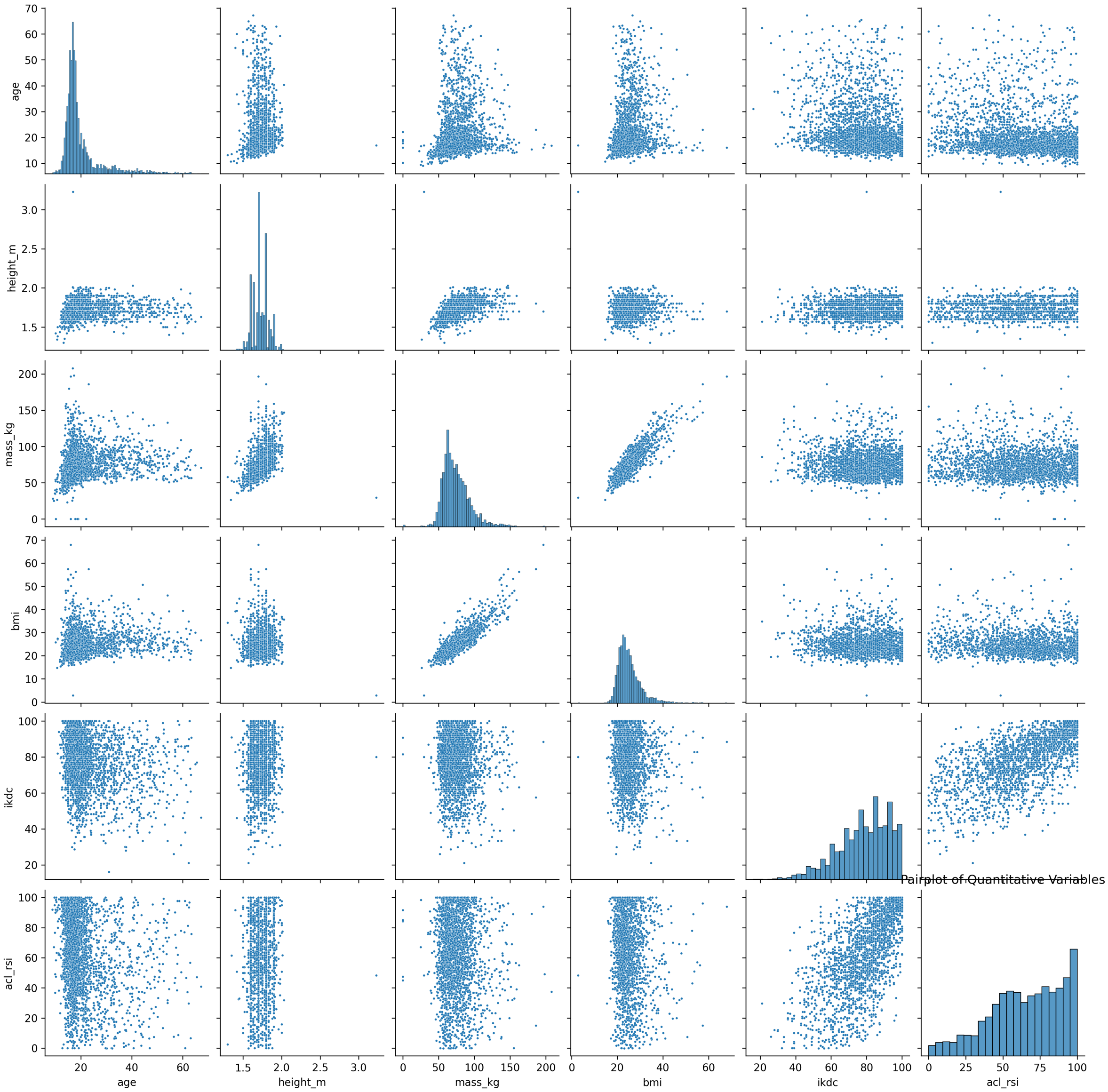
2 Exploratory Data Analysis

Preview of the cleaned dataset (first five rows)

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc	acl_rsi
0	Male	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	No	21.7	1.9	87.4	24.210526	95.4	87.5
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Female	HS autograft	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	No	14.5	1.6	72.2	28.203125	79.3	8.3

2.1 Pairplot of our chosen variables

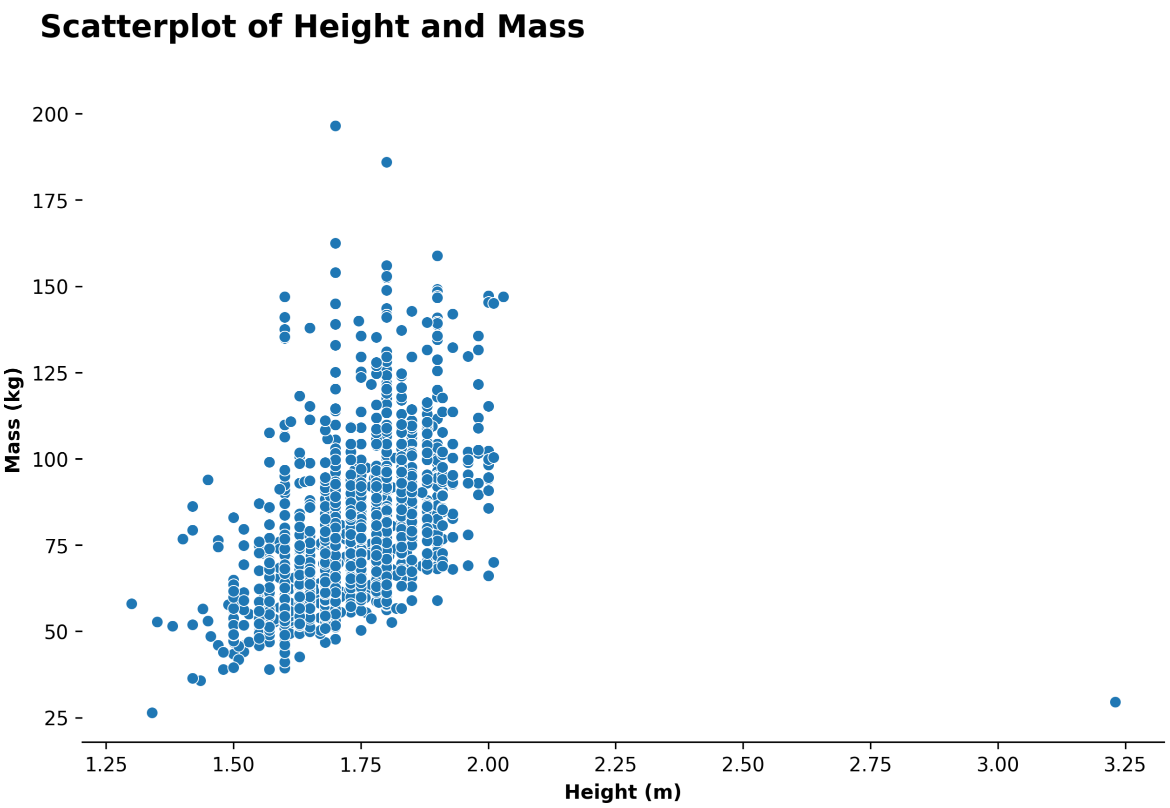
```
# Pairplot of filtered variables:
sns.pairplot(df[['age', 'height_m', 'mass_kg', 'bmi', 'ikdc', 'acl_rsi']], plot_kws={"s": 5}
plt.title('Pairplot of Quantitative Variables')
plt.show()
```



This pairplot illustrates the relationship between each pair of variables in our dataset. This is a quick and straightforward tool to see if there are any obvious correlations/clusters between different elements. We can see that BMI and mass have the most postively correlated relationship, which is to be expected (since mass is used to calculate BMI). Other than that, there are no glaringly obvious trends between variables.

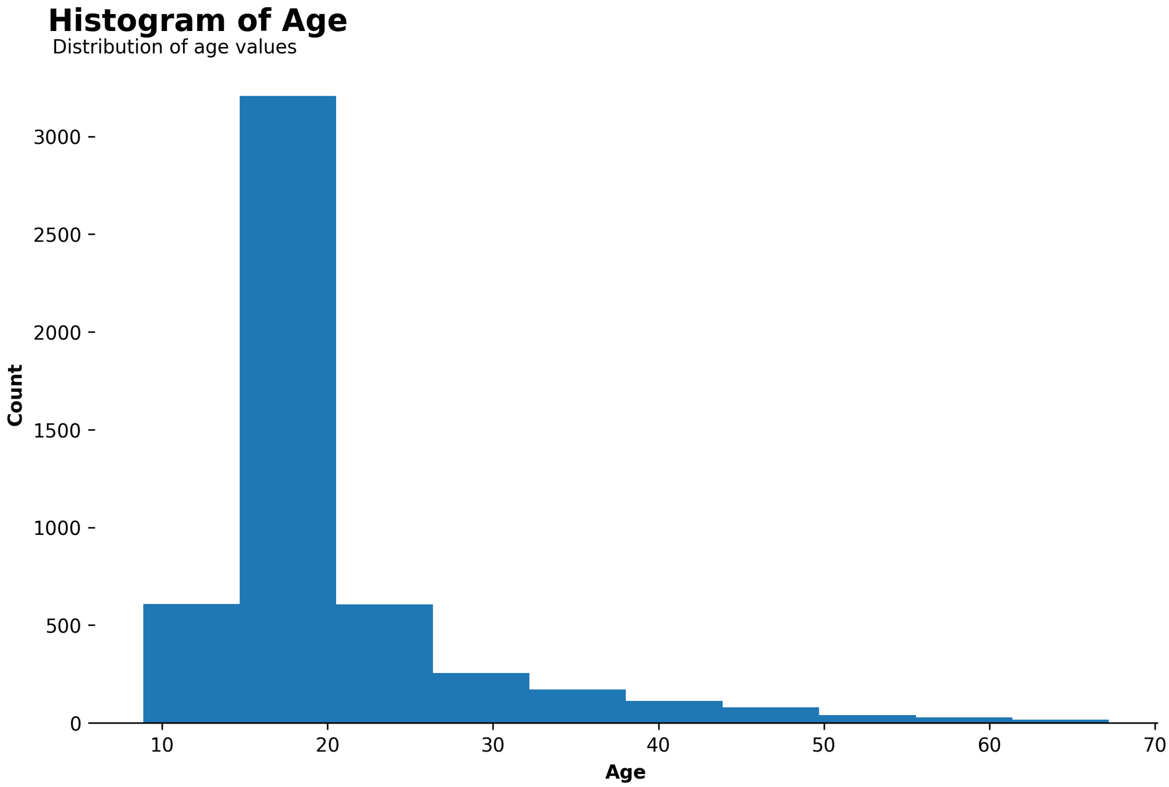
2.2 Looking at Specific Distributions

```
# Scatterplot of height v mass
plt.figure(figsize=(10,6))
sns.scatterplot(x='height_m', y='mass_kg', data=df)

plt.suptitle('Scatterplot of Height and Mass', weight = 'bold', fontsize = 16, x = 0.29)
# axis labels:
plt.xlabel('Height (m)', weight = 'bold')
plt.ylabel('Mass (kg)', weight = 'bold')
# removing spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.show()
```



This is a scatterplot that plots the distribution of height v mass between all the patients. There is a pretty positive correlation between the two variables, since as height increases, mass also tends to increase. There is one outlier where height is around 3.25 meters, or around 10 feet. This is most likely a typo and they intended to mark it as 1.25.

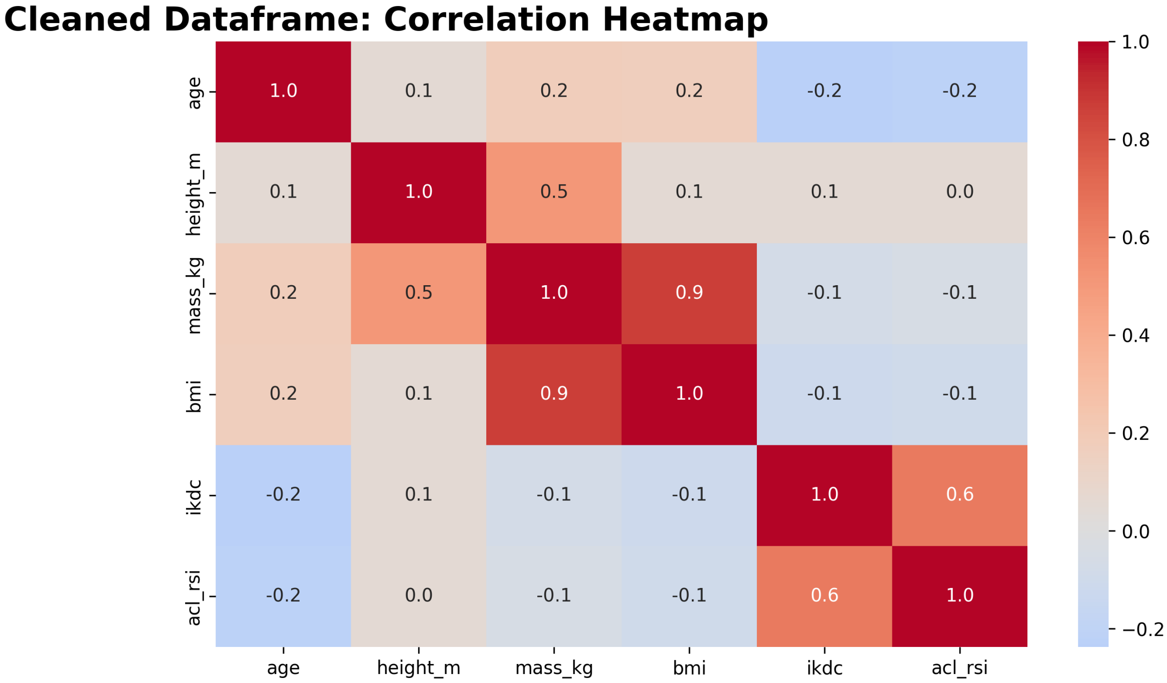
```
# Histogram for 'age'
plt.figure(figsize=(10,6))
df['age'].plot(kind='hist')

plt.suptitle('Histogram of Age', weight = 'bold', fontsize=16, x=0.20)
plt.title('Distribution of age values', fontsize=10, x=0.075)
plt.subplots_adjust(top = 0.91)
# axis labels:
plt.xlabel('Age', weight = 'bold')
plt.ylabel('Count', weight = 'bold')
# removing spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.show()
```



This is a histogram of the ages of all the patients in the study. As we can see there is a right tail skew; most participants are between the ages of 15-20. This makes sense since this study was likely done with many student athletes. There are a couple of older patients in their 50s and 60s, so it could be interesting to see how recovery is affected by age.

2.3 Examine Correlations

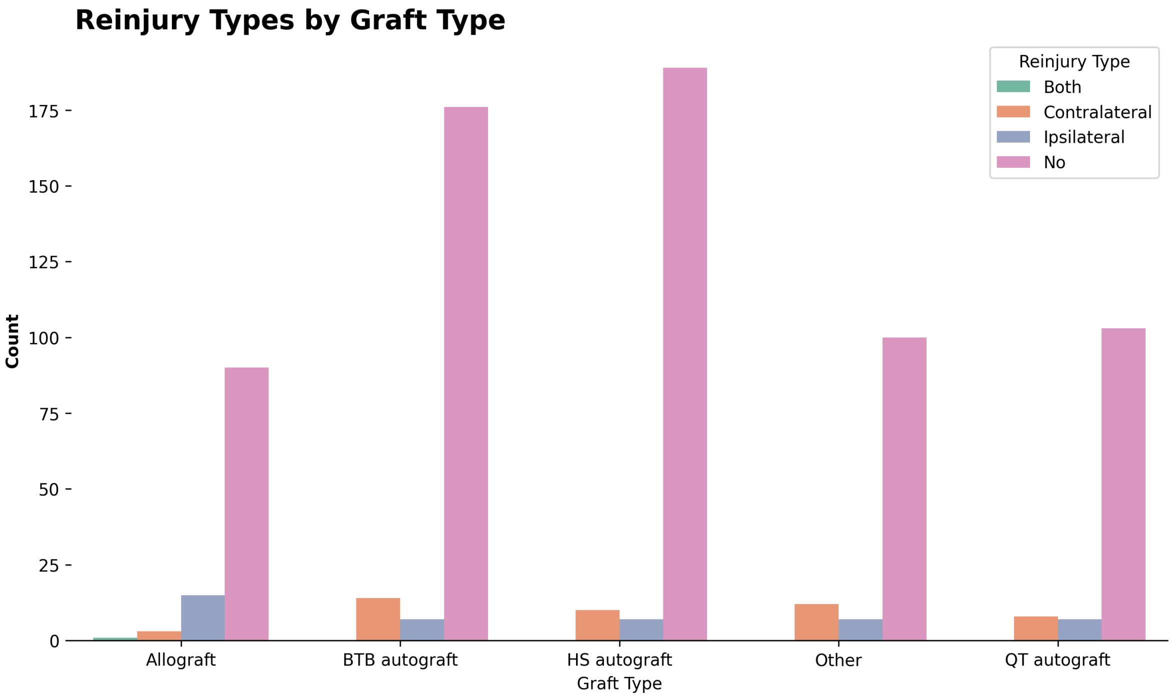
```
# setting filtered_df to be the relevant columns:
filtered_df = df[['sex_dashboard', 'graft_dashboard2', 'reinjury', 'age', 'height_m', 'mass_
# dropping categorical columns:
num_fil_df = filtered_df.drop(['sex_dashboard', 'graft_dashboard2', 'reinjury'], axis=1)
# creating correlation matrix of our filtered numerical columns
corr_matrix = num_fil_df.corr()

# plotting correlation heatmap:
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix, fmt=".1f", annot=True, cmap="coolwarm", center=0)
plt.title("Cleaned Dataframe: Correlation Heatmap",weight = 'bold', fontsize = 18, x=0.21)
plt.show()
```



What variables appear related? The most positively correlated variables are between bmi and mass. This supports our earlier scatterplot that showed a positive trend between height and mass as well, which is great. Something interesting is there is a relatively positive relationship between ikdc and acl_rsi. This is also to be expected because both are patient reported - ikdc is a measure of knee function, while acl_rsi is the return-to-sport-after-injury score.

2.4 Looking into Reinjury Type and Graft Type

```
# Records were mismatched so we shifted row values by 1
df['reinjury_shifted'] = df['reinjury'].shift(-1)
df_cleaned = df[df['graft_dashboard2'].notna()][['graft_dashboard2', 'reinjury_shifted']]
df_cleaned.columns = ['graft_dashboard2', 'reinjury']
# print(df_cleaned.head()) # previewing cleaned dataset

df_cleaned = df_cleaned[
    (df_cleaned['reinjury'].str.upper() != 'BLANK') &
    (df_cleaned['graft_dashboard2'].str.upper() != 'BLANK')]

grouped_counts = (
    df_cleaned.groupby(['graft_dashboard2', 'reinjury'])
    .size()
    .reset_index(name='count')
)

# plotting grouped bar chart:
plt.figure(figsize=(10, 6))
sns.barplot(
    data=grouped_counts,
    x='graft_dashboard2',
    y='count',
    hue='reinjury',
    palette='Set2'
)

plt.xlabel('Graft Type')
plt.ylabel('Count', weight = 'bold')
plt.title('Reinjury Types by Graft Type', weight = 'bold', fontsize = 16, x = 0.2)
plt.legend(title='Reinjury Type')
# removing spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.tight_layout()
plt.show()
```



We were curious if there was any relationship between reinjury types across different graft types, so we made this grouped barplot. It seems that HS autograft has the highest proportion of no reinjuries, while the BTB autograft seems to have the highest recorded count of contralateral reinjuries. This is a super interesting visualization, so we decided to include this relationship in our data visualizations, along with some other characteristics on the next page! HS Autograft has the highest proportion of no reinjuries, while the BTB autograft seems to have the highest recorded count of Contralateral reinjuries.

3 Data Visualization

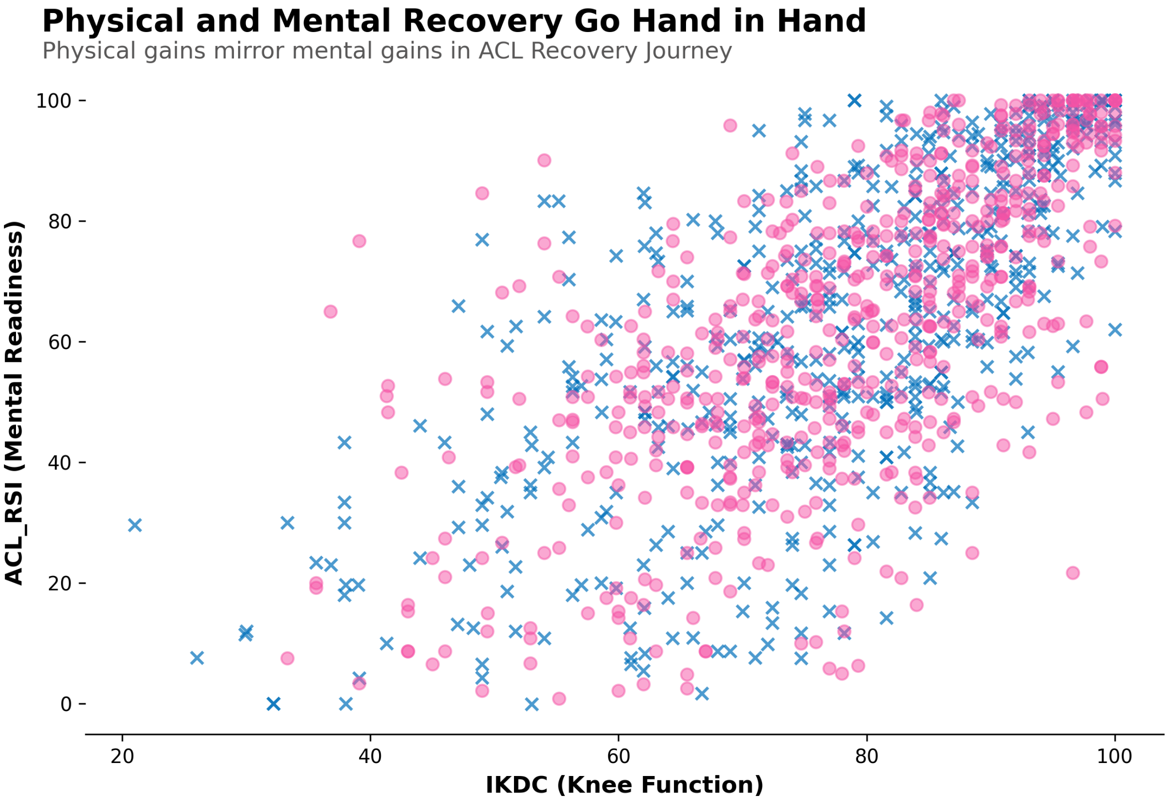
```
# Environment Setup:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
from matplotlib import font_manager
```

3.1 Graph 1

```
# importing data as dataframe:
df = pd.read_csv('aclr data(in).csv')

# filtering df to be consistent with relevant variables of interest:
df = df[['sex_dashboard', 'graft_dashboard2', 'reinjury', 'age', 'height_m', 'mass_kg', 'bmi
# previewing df with relevant columns:
#df.head()
# Filtering for numeric columns:
numeric_columns = df.select_dtypes(include=['int', 'float']).columns
# imputing missing values with the mean for each respective column
mean_values = df[numeric_columns].mean()
m_df = df.fillna(mean_values)
# previewing the imputed dataframe:
#print(m_df.head(5))
df['sex_dashboard'] = df['sex_dashboard'].shift(-1)
```

```
# == PLOT PREPARATION ==

# data for graphing (getting rid of NaN values):
df_clean = df[['ikdc', 'acl_rsi', 'sex_dashboard', 'tss_dashboard']].dropna()
```

```
# Filter by sex
df_male = df_clean[df_clean['sex_dashboard'] == 'Male']
df_female = df_clean[df_clean['sex_dashboard'] == 'Female']
# setting plot size:
plt.figure(figsize=(10, 6))
# male graph:
plt.scatter(df_male['ikdc'], df_male['acl_rsi'],
            marker='x', label='Male', color='#0070BB', alpha=0.7, s=40)
# Plot females with square markers
plt.scatter(df_female['ikdc'], df_female['acl_rsi'],
            marker='o', label='Female', color='#F653A6', alpha=0.5, s=40)

# == SCAFFOLDING ==

# setting titles:
plt.suptitle('Physical and Mental Recovery Go Hand in Hand', weight = 'bold', fontsize = 16,
plt.title('Physical gains mirror mental gains in ACL Recovery Journey', color='#585757', fon
plt.subplots_adjust(top = 0.905) # adjusting spacing between sub and main title


# setting x-axis and y-axis labels:
plt.xlabel('IKDC (Knee Function)', weight = 'bold', fontsize = 12)
plt.ylabel('ACL_RSI (Mental Readiness)', weight = 'bold', fontsize = 12)

# reducing clutter on the end of the x-axis:
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.locator_params(axis='x', nbins=8)  # reduces x-axis ticks

# extra formatting (removing spines for cleaner look):
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
# adding source as annotation:
plt.figtext(0.1, -0.05, 'Source: UVA Department of Kinesiology and School of Data Science', 
plt.show()
```



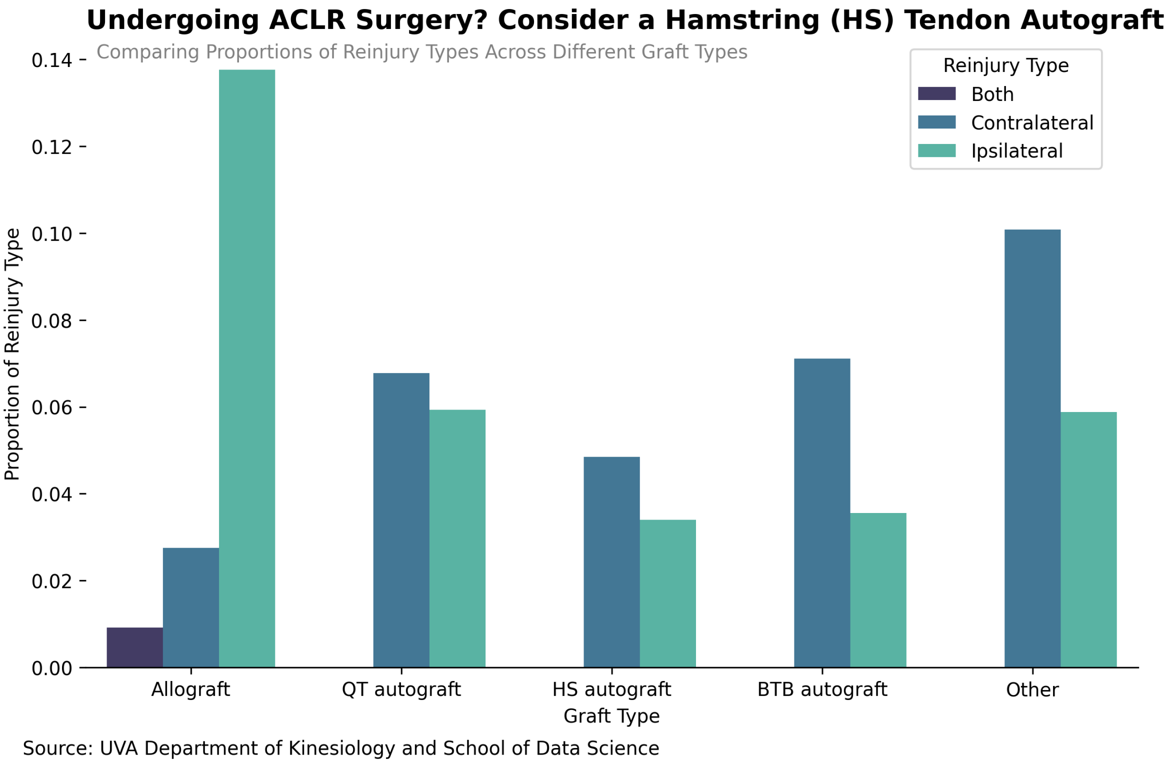
Source: UVA Department of Kinesiology and School of Data Science

3.2 Graph 2:

```
# setting font
georgia_font = font_manager.FontProperties(family='Georgia')
plt.rcParams['font.family'] = georgia_font.get_name()

df = df[['sex_dashboard', 'graft_dashboard2', 'reinjury', 'age', 'height_m', 'mass_kg', 'bmi
# cleaning reinjury variable
df = df[df['reinjury'].str.upper() != 'BLANK']

#proportion of patients with no reinjury
prop_noreinjury = df['reinjury'].value_counts(normalize=True).get('No', 0)
print(f"Proportion of patients with no reinjury: {prop_noreinjury:.2%}")
# cleaning dataframe
```

```
df['reinjury_shifted'] = df['reinjury'].shift(-1) #align reinjury with other values
df_cleaned = df[df['graft_dashboard2'].notna()][['graft_dashboard2', 'reinjury_shifted']] #ge
df_cleaned.columns = ['graft_dashboard2', 'reinjury']

#get counts of graft and reinjury
counts = (
    df_cleaned.groupby(['graft_dashboard2', 'reinjury'])
    .size()
    .reset_index(name='count')
)
#order of grafts
graft_order = ['Allograft', 'QT autograft', 'HS autograft', 'BTB autograft', 'Other']
counts['graft_dashboard2'] = pd.Categorical( #set order in counts
    counts['graft_dashboard2'],
    categories=graft_order,
    ordered=True
)
total_per_graft = counts.groupby('graft_dashboard2')['count'].transform('sum') #get sums
counts['proportion'] = counts['count'] / total_per_graft #calculate proportions
# print(counts)
# print('No reinjury proportions by Graft type:\nAllograft: 0.83\nBTB autograft: 0.89\nHS au

#get rid of no reinjury bar for readability
counts_noreinjury = counts[counts['reinjury'] != 'No']
#make grouped barplot
plt.figure(figsize=(10, 6))
sns.barplot(
    data=counts_noreinjury,
    x='graft_dashboard2',
    y='proportion',
    hue='reinjury',
    palette='mako'
)

# graph labels and scaffolding
plt.xlabel('Graft Type')
plt.ylabel('Proportion of Reinjury Type')
plt.title('Undergoing ACLR Surgery? Consider a Hamstring (HS) Tendon Autograft', fontsize=14
plt.text(0.01, 0.98, 'Comparing Proportions of Reinjury Types Across Different Graft Types',
         ha='left', va='center', transform=plt.gca().transAxes, fontsize=10, color='gray')  #
plt.legend(title='Reinjury Type', loc='upper left', bbox_to_anchor=(0.775, 1.0))
plt.text(-0.8, -0.02, 'Source: UVA Department of Kinesiology and School of Data Science', ha
sns.despine(top=True, right=True, left=True) #get rid of axes
plt.show()
```

```
findfont: Font family ['Georgia'] not found. Falling back to DejaVu Sans.
/tmp/ipykernel_23503/1847392337.py:30: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a future version o

Proportion of patients with no reinjury: 87.47%
```



Source: UVA Department of Kinesiology and School of Data Science

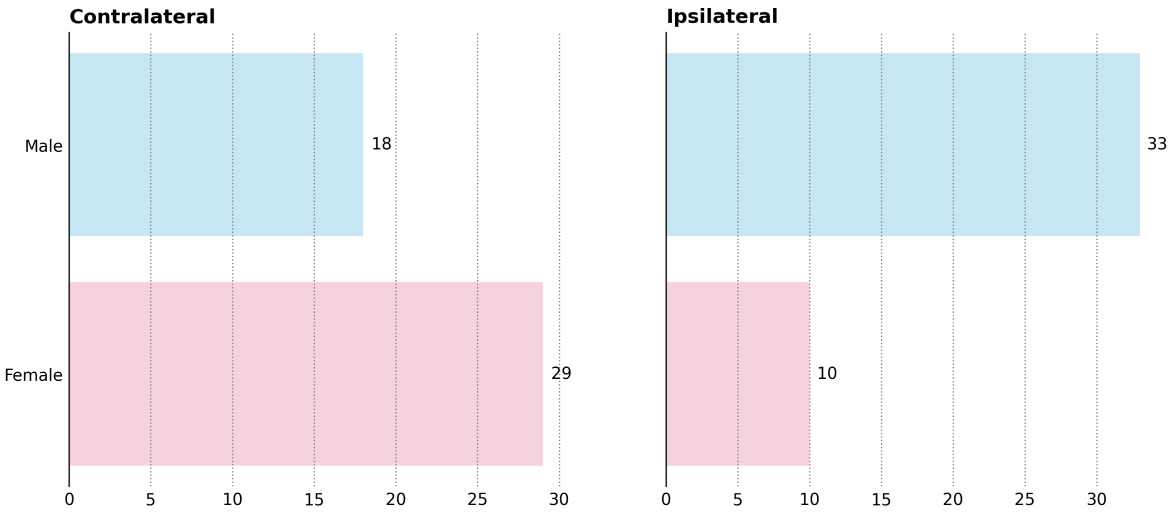
3.3 Graph 3:

Looking at the relationship between different reinjuries between the sexes.

```
# small multiples bar chart

# Set font family to Georgia
georgia_font = font_manager.FontProperties(family='Georgia')
plt.rcParams['font.family'] = georgia_font.get_name()

# Records were mismatched so we shifted row values by 1
# (for every graft_type recorded, reinjury was blank so shifted by 1 to match)
m_df['reinjury_shifted'] = m_df['reinjury'].shift(-1)
df2 = m_df[m_df['sex_dashboard'].notna()][['sex_dashboard', 'reinjury_shifted']]
df2.columns = ['sex_dashboard', 'reinjury']
# print(df2.head()) # previewing cleaned dataset

df2 = df2[
    (df2['reinjury'].str.upper() != 'BLANK') &
    (df2['sex_dashboard'].str.upper() != 'BLANK')]

df2 = df2[df2['reinjury'].str.upper() != 'NO'] # dropping 'no' reinjury records
df2= df2[df2['reinjury'].str.upper() != 'BOTH'] # dropping 'both' reinjury records


grouped_counts2 = (
    df2.groupby(['sex_dashboard', 'reinjury'])
    .size()
    .reset_index(name='count')
)


# Create sub-dataframes for Contralateral and Ipsilateral
df_contra = grouped_counts2[grouped_counts2['reinjury'] == 'Contralateral']
df_ipsi = grouped_counts2[grouped_counts2['reinjury'] == 'Ipsilateral']

# Set up 1x2 subplot grid
fig, axs = plt.subplots(1, 2, figsize=(12, 5), sharey=True)

# Title
fig.suptitle('Males Reinjure Their ACLs More Than Females Overall', fontsize=14, weight='bol

# Contralateral subplot
colors_contra = df_contra['sex_dashboard'].map({'Male': '#C8E7F5', 'Female': '#F6D2E0'})  #
bars_contra = axs[0].barh(df_contra['sex_dashboard'], df_contra['count'], color=colors_contra
axs[0].set_title('Contralateral', loc='left', weight='bold', color='black')
```

```
axs[0].grid(axis='x', linestyle=':', color='gray')
axs[0].spines['top'].set_visible(False)
axs[0].spines['right'].set_visible(False)
axs[0].spines['bottom'].set_visible(False)
axs[0].tick_params(axis='x', length=0)
axs[0].tick_params(axis='y', length=0)

for bar in bars_contra:
    xval = bar.get_width()
    axs[0].text(xval + 0.5, bar.get_y() + bar.get_height()/2,
                round(xval), va='center', ha='left', fontsize=10)

# Ipsilateral subplot
colors_ipsi = df_ipsi['sex_dashboard'].map({'Male': '#C8E7F5', 'Female': '#F6D2E0'})
bars_ipsi = axs[1].barh(df_ipsi['sex_dashboard'], df_ipsi['count'], color=colors_ipsi)
axs[1].set_title('Ipsilateral', loc='left', weight='bold', color='black')
axs[1].grid(axis='x', linestyle=':', color='gray')
axs[1].spines['top'].set_visible(False)
axs[1].spines['right'].set_visible(False)
axs[1].spines['bottom'].set_visible(False)
axs[1].tick_params(axis='x', length=0)
axs[1].tick_params(axis='y', length=0)

for bar in bars_ipsi:
    xval = bar.get_width()
    axs[1].text(xval + 0.5, bar.get_y() + bar.get_height()/2,
                round(xval), va='center', ha='left', fontsize=10)

# Final layout
plt.text(-42, -0.8, 'Source: UVA Department of Kinesiology and School of Data Science', ha='
# plt.tight_layout()
plt.show()
```

Males Reinjure Their ACLs More Than Females Overall


Contralateral

Ipsilateral

Source: UVA Department of Kinesiology and School of Data Science

4 Data Dictionary

Here are the relevant variables we used to complete our analysis with their meanings.

Variable	Description
acl_rsi	The return-to-sport-after-injury score is self-reported by the patient.
age	The age at which the patient received surgery.
bmi	Body mass index of the patient.
graft_dashboard2	The types of grafts used in surgery are allograft, QT autograft, HS autograft, BTB autograft, and others.
height_m	The height of the patient in meters.
ikdc	A patient-reported outcome measure used to assess knee function and symptoms.
mass_kg	The weight of the patient in kilograms.
reinjury	The different types of reinjuries: contralateral, ipsilateral, and both.
sex_dashboard	The gender of the patient: male or female.
tss_dashboard	Categorizes the months post-surgery into subsets.