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_percer

# 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 drop the rows with missing values entirely as imputing values here would skew the data and not be representative of the sampled population. We will drop the rows with missing values using the `dropna()` method.

```
df_clean = m_df.dropna()
df_clean.head()
```

	sex_dashboard	graft_dashboard2	reinjury	age	height_m	mass_kg	bmi	ikdc	acl_rsi	tss_das

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)

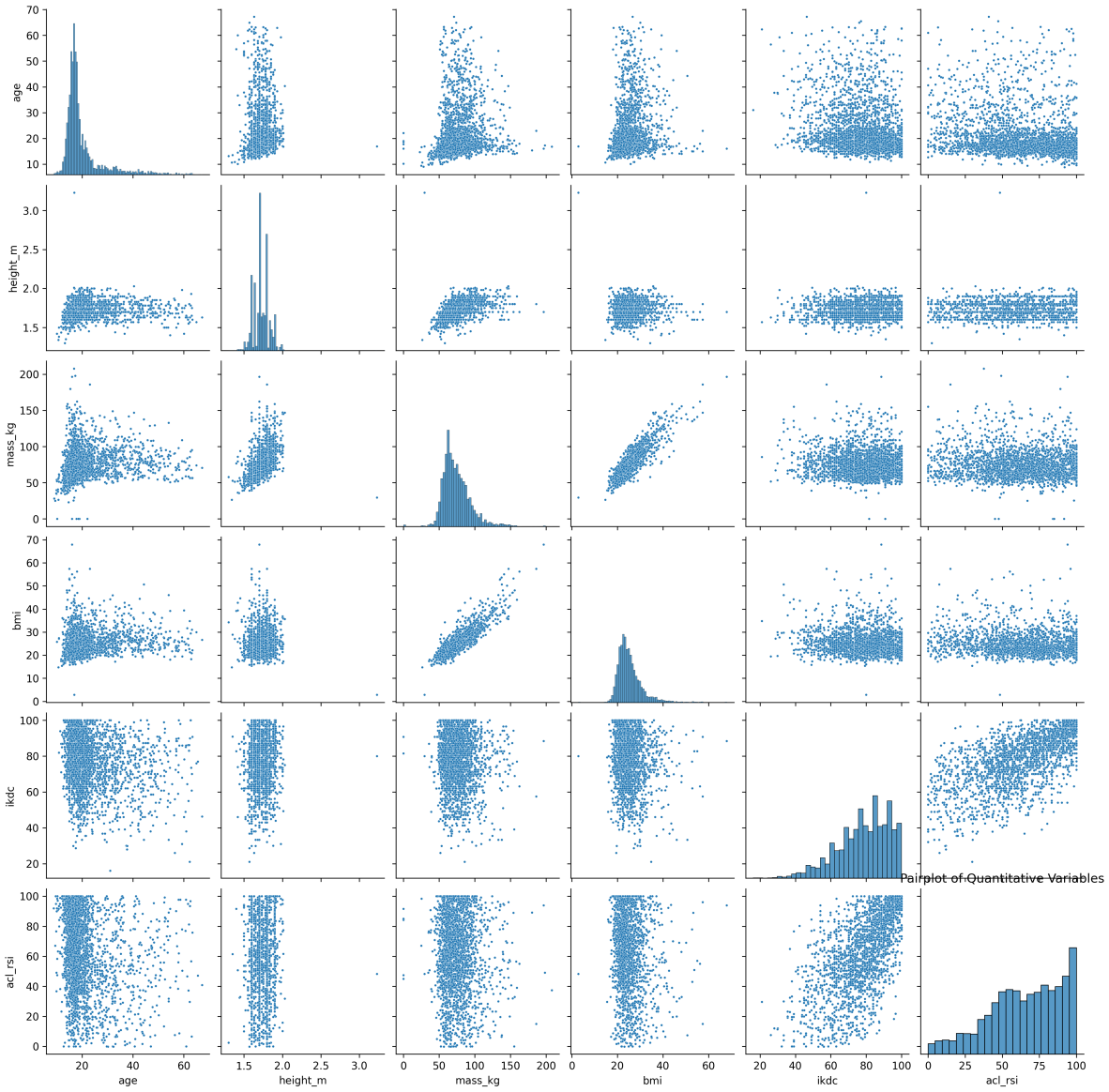
```
# setting up environment
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# importing data as dataframe:
df = pd.read_csv('aclr data(in).csv')
# Including relevant variables into our dataframe and dropping the others:
df = df[['sex_dashboard', 'graft_dashboard2', '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

```
# 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)
```

2.0.1 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()
```

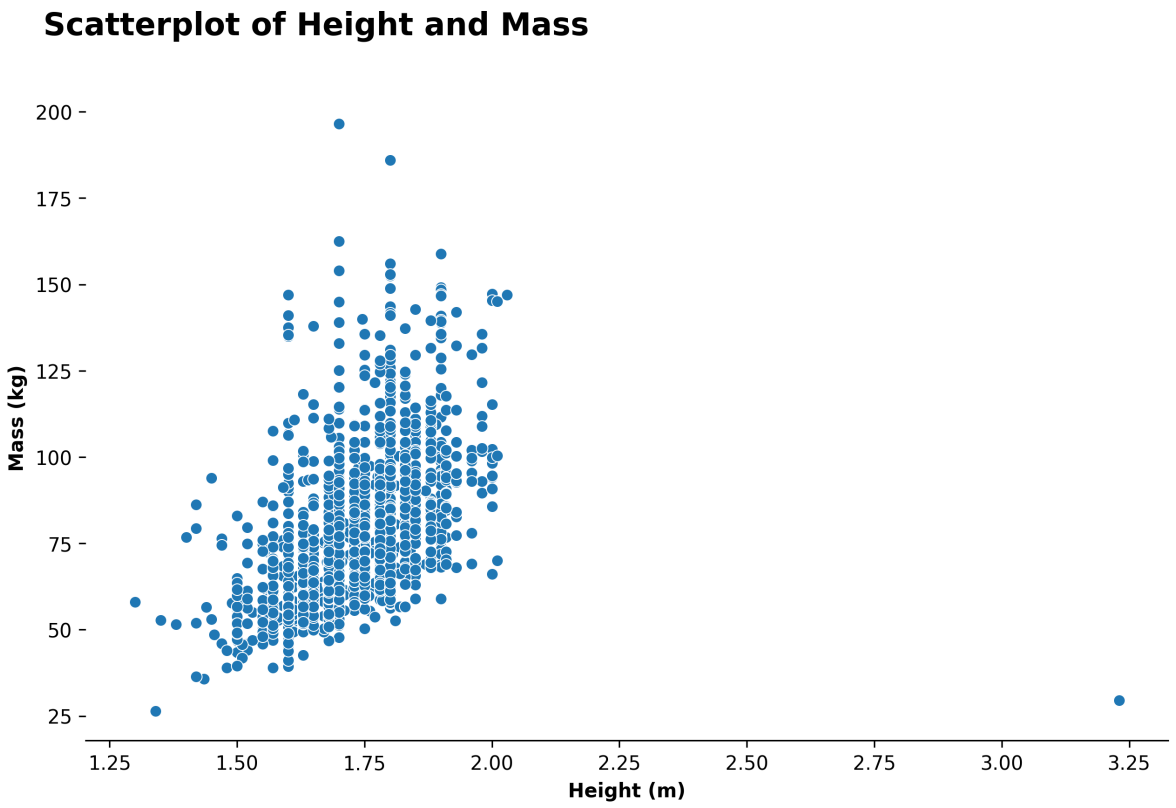


This pairplot illustrates the relationship between each pair of variables in our dataset. This is a quick and straightforward tool to see if there are any obvious correlations/clusters between different elements. We can see that BMI and mass have the most postively correlated relationship, which is to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

Looking at Specific Distributions

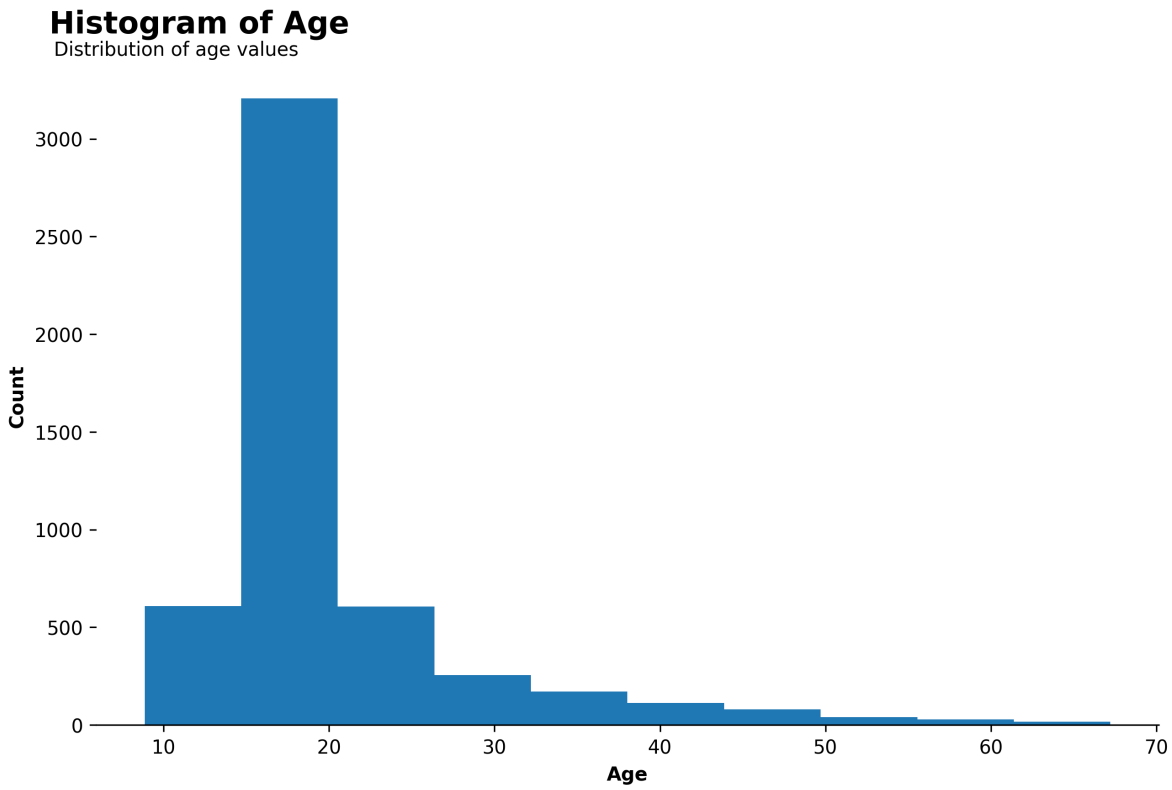
```
# Creating visualizations to illustrate distributions:
# SCATTERPLOT: Between height and mass_kg
```

```
plt.figure(figsize=(10,6))
sns.scatterplot(x='height_m', y='mass_kg', data=df)
# == SCAFFOLDING ==
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.

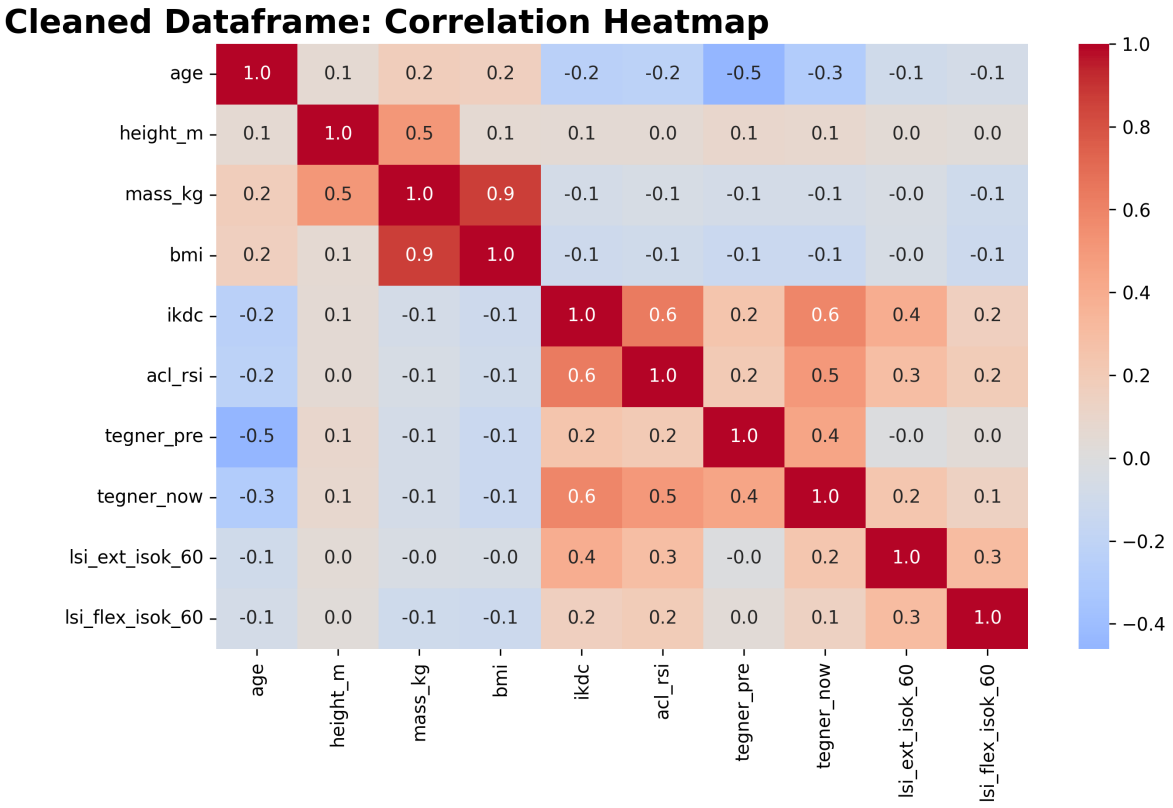
```
# Histogram for 'age':
plt.figure(figsize=(10,6))
df['age'].plot(kind='hist')
# == SCAFFOLDING ==
plt.suptitle('Histogram of Age', weight = 'bold', fontsize=16, x=0.20)
plt.title('Distribution of age values', fontsize=10, x=0.075)
plt.subplots_adjust(top = 0.91)
# axis labels:
plt.xlabel('Age', weight = '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()
```



6. Examine Correlations (if relevant)

Interpret findings: What variables appear related?

```
# 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()
```



Positively correlated variables seem to include: - ikdc and acl_rsi - bmi and mass_kg - tegner_now and ikdc - tegner_now and acl_rsi

Negatively correlated variables seem to include: - tegner_pre and age - tegner_now and age

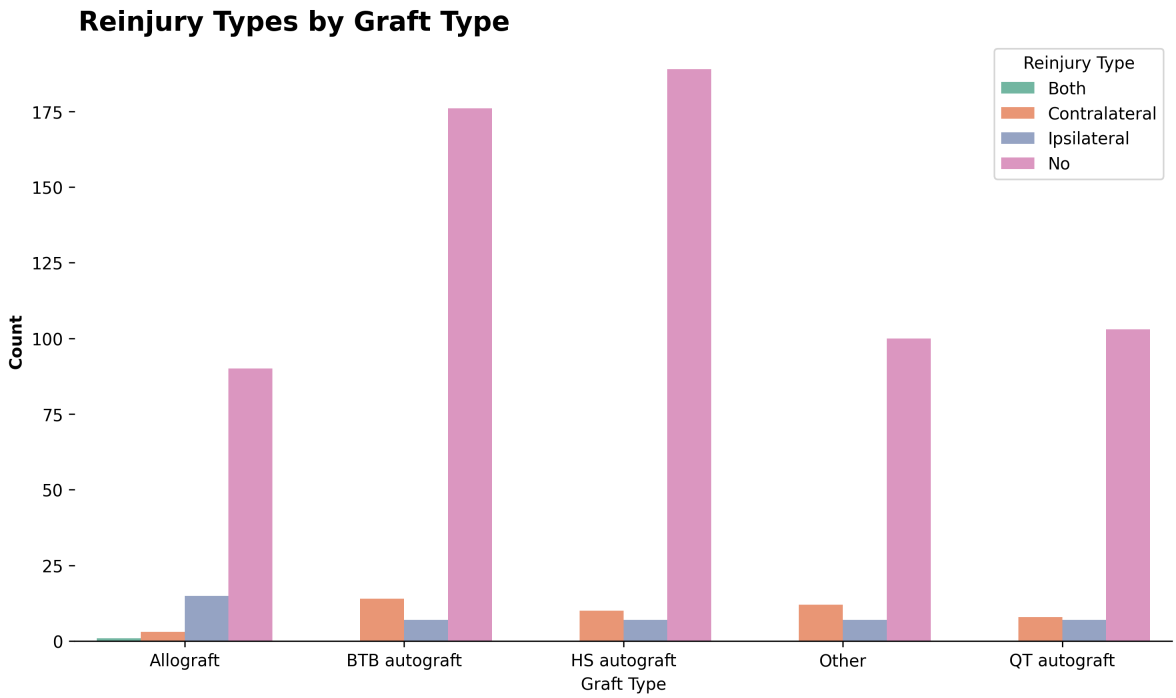
7. Explore Relationships (if relevant)

Dig into potential causal or descriptive relationships Use visualizations and statistical summaries.

```
# Records were mismatched so we shifted row values by 1
# (for every graft_type recorded, reinjury was blank so shifted by 1 to match)
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')
)

# #graft_order = ['HS allograft', 'BTB allograft', 'QT autograft', 'Allograft', 'Other']
# # Convert 'graft_dashboard2' to a categorical type w order
# #grouped_counts['graft_dashboard2'] = pd.Categorical(
#     grouped_counts['graft_dashboard2'],
#     categories=graft_order,
#     ordered=True
# )
# sort the DataFrame by this order for plotting
#grouped_counts = grouped_counts.sort_values('graft_dashboard2')


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

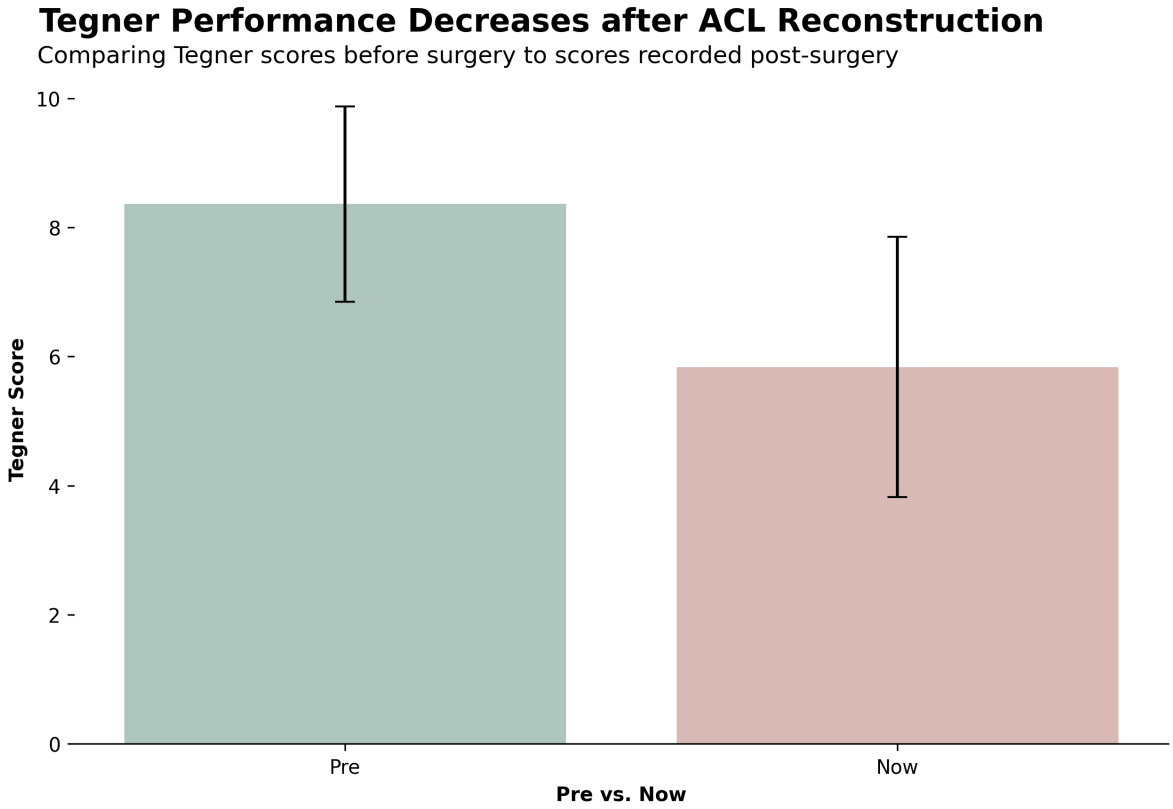
```
plt.ylabel('Count', weight = 'bold')
plt.title('Reinjury Types by Graft Type', weight = 'bold', fontsize = 16, x = 0.2)
plt.legend(title='Reinjury Type')
# removing spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)

plt.tight_layout()
plt.show()
```

```
   graft_dashboard2 reinjury
0             Other       No
3      HS autograft       No
8      HS autograft    BLANK
10     HS autograft       No
15     HS autograft       No
```

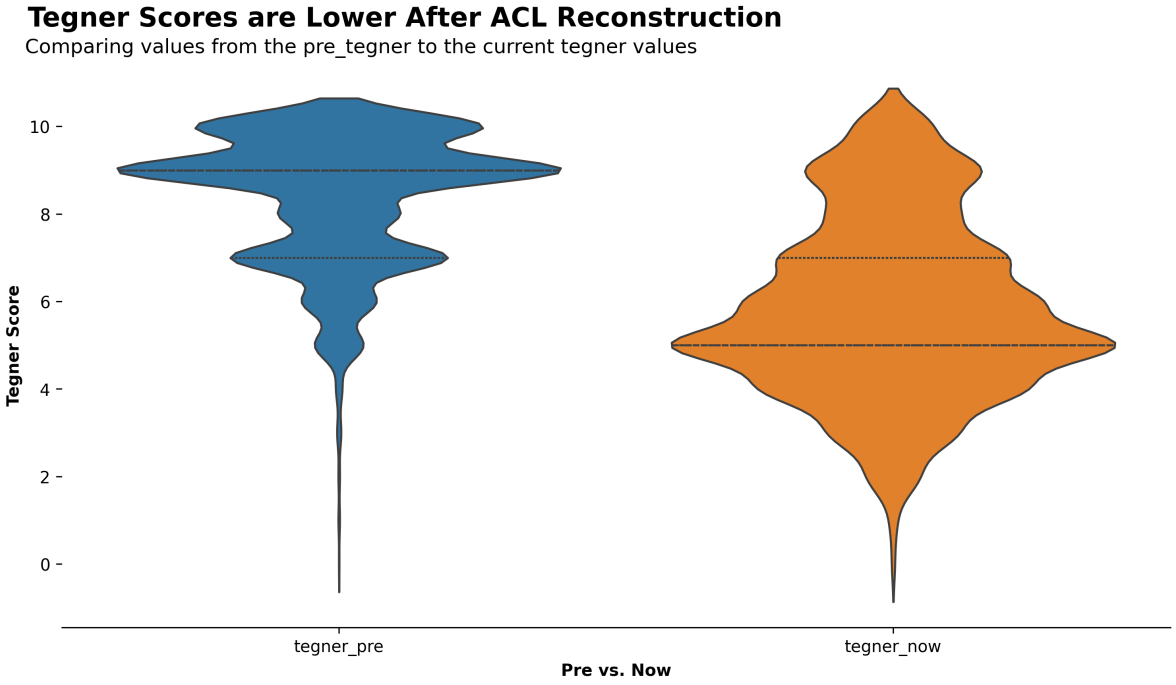


HS Autograft has the highest proportion of no reinjuries, while the BTB autograft seems to have the highest recorded count of Contralateral reinjuries.

```
# Bar Chart: Focusing on Tegner as a metric since it assesses overall functionality

# calculating mean and std for both Tegner values:
mean_val = [filtered_df['tegner_pre'].mean(), filtered_df['tegner_now'].mean()]
std_val = [filtered_df['tegner_pre'].std(), filtered_df['tegner_now'].std()]

# == PLOTTING ==
plt.figure(figsize=(10,6))
plt.bar(['Pre', 'Now'], mean_val, yerr=std_val, capsize=5, color=['#7aa095', '#bf8b85'], alp

# SCAFFOLDING:
plt.suptitle('Tegner Performance Decreases after ACL Reconstruction ', weight='bold', fontsi
plt.title('Comparing Tegner scores before surgery to scores recorded post-surgery', fontsize=

plt.subplots_adjust(top = 0.90) # adjusting space between the super title and the title
plt.ylabel('Tegner Score', weight='bold')
plt.xlabel('Pre vs. Now', 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()
```




```
# Creating a violin plot to compare the tegner_pre and tegner_now by age:

# PLOT
plt.figure(figsize=(12, 6))
sns.violinplot(data=filtered_df[['tegner_pre', 'tegner_now']], inner='quartile')

# SCAFFOLDING:
plt.suptitle('Tegner Scores are Lower After ACL Reconstruction', weight='bold', fontsize=16,
plt.title('Comparing values from the pre_tegner to the current tegner values', fontsize=12, 

plt.subplots_adjust(top = 0.90) # adjusting space between the super title and the title
plt.ylabel('Tegner Score', weight='bold')
plt.xlabel('Pre vs. Now', weight='bold')

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.show()
```


Tegner Scores are Lower After ACL Reconstruction
Comparing values from the pre_tegner to the current tegner values
Tegner Score
10
8
6
4
2
0
tegner_pre
tegner_now
Pre vs. Now

3 Data Visualization

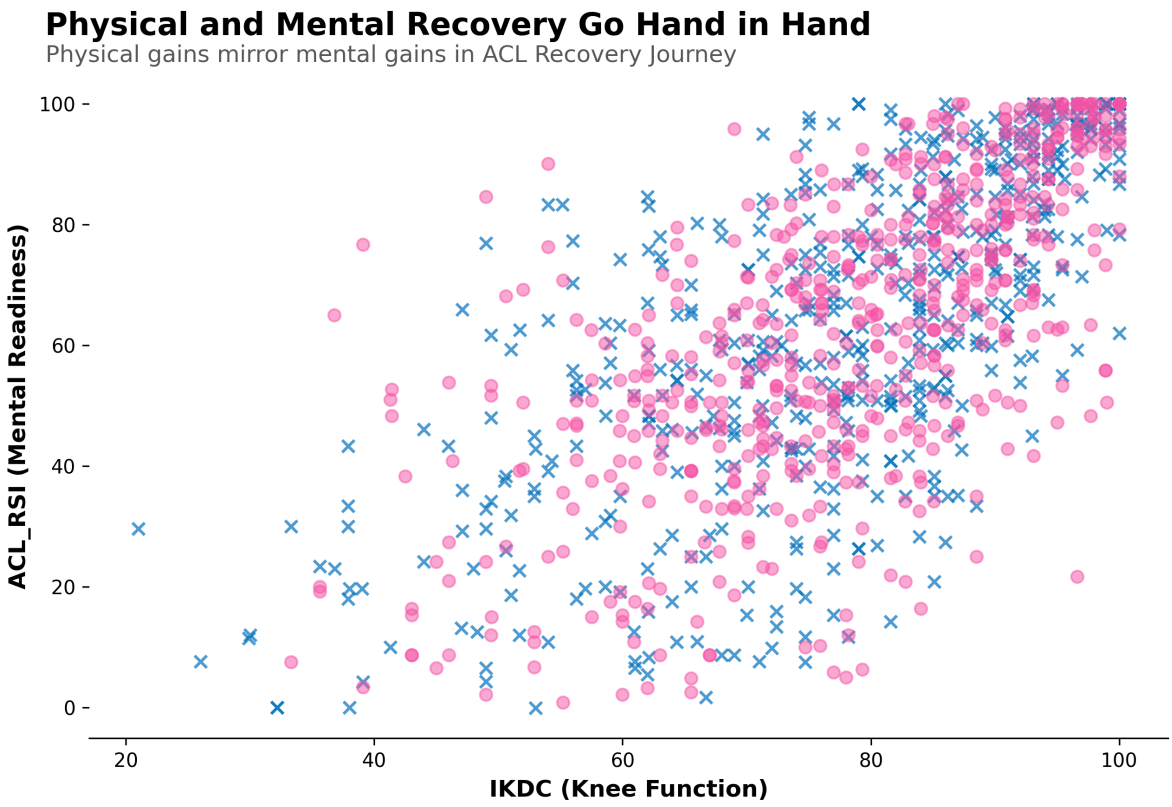
```
# Environment Setup:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
from matplotlib import font_manager
```

```
# 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', font
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', h
plt.show()
```



Source: UVA Department of Kinesiology and School of Data Science

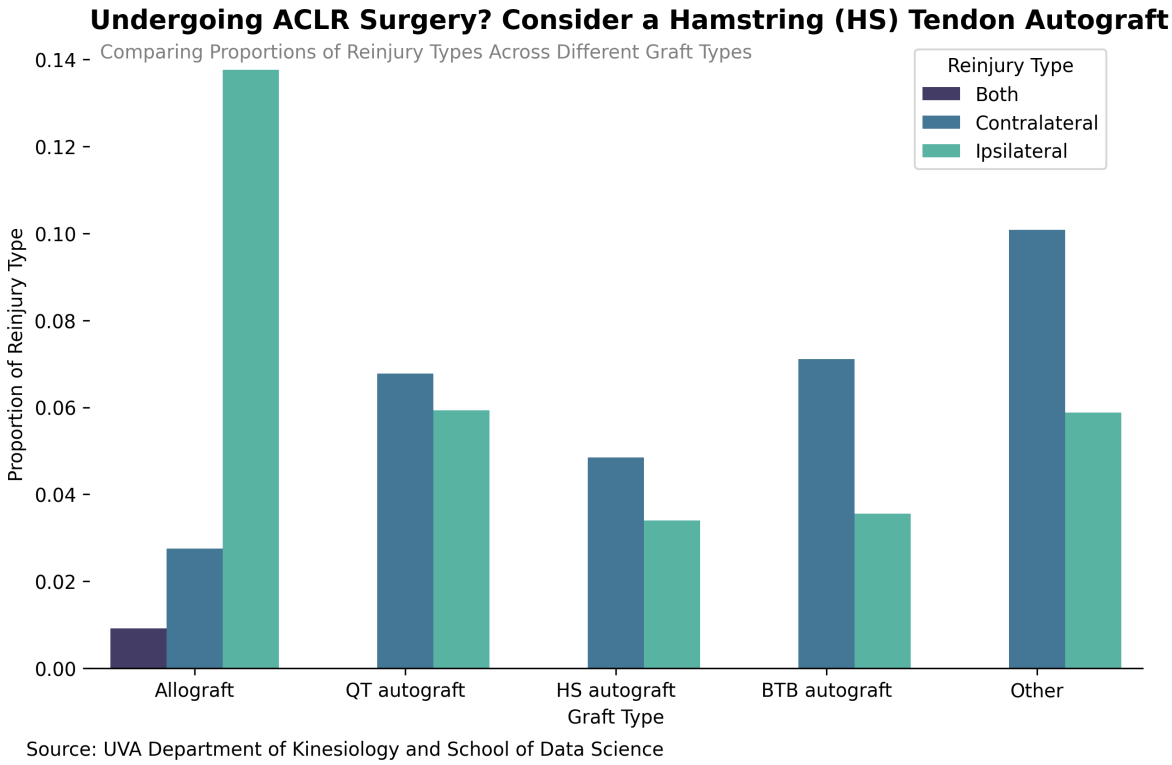
Graph 2:

Planning on creating a scatterplot showing how both physical and mental readiness affect each other and the individual's overall confidence in returning to play.

```
# 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
df = df[df['reinjury'].str.upper() != 'BLANK'] #get rid of all blank in reinjury

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

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

```
counts = ( #get counts of graft and reinjury
    df_cleaned.groupby(['graft_dashboard2', 'reinjury'])
    .size()
    .reset_index(name='count')
)

graft_order = ['Allograft', 'QT autograft', 'HS autograft', 'BTB autograft', 'Other'] #order
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
#show proportions
print(counts)
print('No reinjury proportions by Graft type:\n' +
'Allograft: 0.83\n' +
'BTB autograft: 0.89\n' +
'HS autograft: 0.92\n' +
'QT autograft: 0.87\n' +
'Other: 0.84\n')

counts_noreinjury = counts[counts['reinjury'] != 'No'] #get rid of no reinjury bar
plt.figure(figsize=(10, 6))
sns.barplot( #make grouped barplot
    data=counts_noreinjury,
    x='graft_dashboard2',
    y='proportion',
    hue='reinjury',
    palette='mako'
)
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_6503/934650589.py:27: 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%
   graft_dashboard2       reinjury  count  proportion
0         Allograft           Both      1    0.009174
1         Allograft  Contralateral      3    0.027523
2         Allograft    Ipsilateral     15    0.137615
3         Allograft             No     90    0.825688
4     BTB autograft  Contralateral     14    0.071066
5     BTB autograft    Ipsilateral      7    0.035533
6     BTB autograft             No    176    0.893401
7      HS autograft  Contralateral     10    0.048544
8      HS autograft    Ipsilateral      7    0.033981
9      HS autograft             No    189    0.917476
10            Other  Contralateral     12    0.100840
11            Other    Ipsilateral      7    0.058824
12            Other             No    100    0.840336
13     QT autograft  Contralateral      8    0.067797
14     QT autograft    Ipsilateral      7    0.059322
15     QT autograft             No    103    0.872881
No reinjury proportions by Graft type:
Allograft: 0.83
BTB autograft: 0.89
HS autograft: 0.92
QT autograft: 0.87
Other: 0.84
```

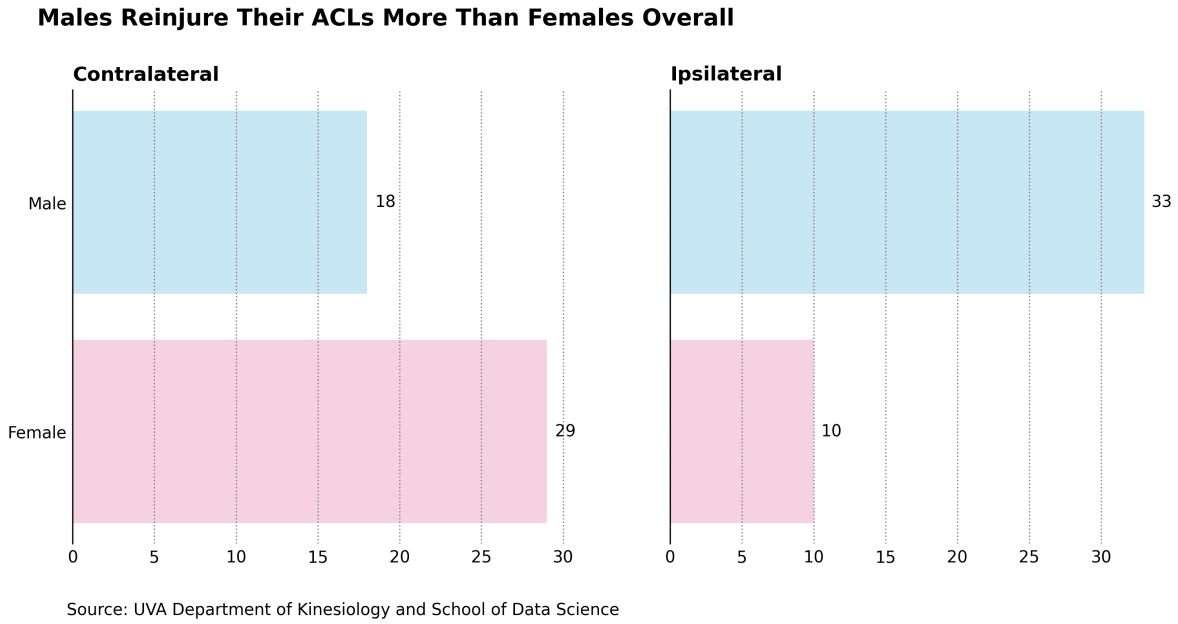


Graph 3: Looking at the relationship between different reinjuries between the sexes.

```
# 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()
```



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.