Untitled

MT

2023-09-22

# Data Analysis for PhiCB5 Adsorption Data

## Introduction

The experiment is run in biological replicates (have their own unique id) and technical triplicate for the biological replicates. The data set is data\_master\_final from the PhiCB5 adsorption data. This document outlines the statistical tests carried out on the dataset.

#### Data Preparation

The dataset was loaded and the ‘unique\_id’ column was specified as a string to prevent it from being parsed as a date.

We’ll use Python for data manipulation, statistical testing, and plotting. The analysis includes:

* Data Import
* Data Normalization
* Summary Statistics
* ANOVA and Tukey’s Test
* Plotting

#### Library R

#### Initiate Python Env

#### Library Python

Three types of normalization were performed:

1. Direct Subtraction
2. Ratio Normalization
3. Z-Score Normalization

# Load the newly uploaded dataset with the 'unique\_id' column specified as string  
df\_new = pd.read\_excel("Data/data\_master\_final.xlsx", dtype={"unique\_id": str})  
  
# To check unique values in the 'unique\_id' column  
print(df\_new['unique\_id'].unique())

## ['2/13/2023\_AG' '2/13/2023\_MT' '2/13/2023\_TS' '2/8/2023\_AG' '9/9/2023'  
## '9/10/2023' '9/12/2023']

# Calculate the mean titer of the "No Cell Control" for each unique ID  
no\_cell\_control\_means\_new = df\_new[df\_new['genotype'] == 'No Cell Control'].groupby('unique\_id')['titer'].mean().reset\_index()  
no\_cell\_control\_means\_new.rename(columns={'titer': 'mean\_titer\_no\_cell\_control'}, inplace=True)  
  
# Merge this information with the original DataFrame  
df\_merged\_new = pd.merge(df\_new, no\_cell\_control\_means\_new, on='unique\_id', how='left')  
  
# Perform the normalization calculations  
  
# Direct Subtraction  
df\_merged\_new['normalized\_titer\_direct\_sub'] = df\_merged\_new['titer'] - df\_merged\_new['mean\_titer\_no\_cell\_control']  
  
# Ratio Normalization  
df\_merged\_new['normalized\_titer\_ratio'] = df\_merged\_new['titer'] / df\_merged\_new['mean\_titer\_no\_cell\_control']  
  
# Z-Score Normalization  
df\_merged\_new['normalized\_titer\_zscore'] = (df\_merged\_new['titer'] - df\_merged\_new['mean\_titer\_no\_cell\_control']) / df\_merged\_new['mean\_titer\_no\_cell\_control'].std()

## Plotting the different normalizations

# Initialize the figure with modified dimensions for a 2x2 layout  
fig, axes = plt.subplots(2, 2, figsize=(20, 20))  
  
# Original Data  
sns.violinplot(x='genotype', y='titer', data=df\_merged\_new, ax=axes[0, 0], inner=None, color='lightgray')  
sns.stripplot(x='genotype', y='titer', data=df\_merged\_new, hue='unique\_id', dodge=True, marker='o', alpha=0.7, jitter=True, ax=axes[0, 0])  
axes[0, 0].set\_title('Original Data')  
axes[0, 0].set\_xticklabels(axes[0, 0].get\_xticklabels(), rotation=45, ha='right')  
axes[0, 0].legend().set\_visible(False) # Hide the legend  
  
# Direct Subtraction Normalization  
sns.violinplot(x='genotype', y='normalized\_titer\_direct\_sub', data=df\_merged\_new, ax=axes[0, 1], inner=None, color='lightgray')  
sns.stripplot(x='genotype', y='normalized\_titer\_direct\_sub', data=df\_merged\_new, hue='unique\_id', dodge=True, marker='o', alpha=0.7, jitter=True, ax=axes[0, 1])  
axes[0, 1].set\_title('Direct Subtraction Normalization')  
axes[0, 1].set\_xticklabels(axes[0, 1].get\_xticklabels(), rotation=45, ha='right')  
axes[0, 1].legend().set\_visible(False) # Hide the legend  
  
# Ratio Normalization  
sns.violinplot(x='genotype', y='normalized\_titer\_ratio', data=df\_merged\_new, ax=axes[1, 0], inner=None, color='lightgray')  
sns.stripplot(x='genotype', y='normalized\_titer\_ratio', data=df\_merged\_new, hue='unique\_id', dodge=True, marker='o', alpha=0.7, jitter=True, ax=axes[1, 0])  
axes[1, 0].set\_title('Ratio Normalization')  
axes[1, 0].set\_xticklabels(axes[1, 0].get\_xticklabels(), rotation=45, ha='right')  
axes[1, 0].legend().set\_visible(False) # Hide the legend  
  
# Z-Score Normalization  
sns.violinplot(x='genotype', y='normalized\_titer\_zscore', data=df\_merged\_new, ax=axes[1, 1], inner=None, color='lightgray')  
sns.stripplot(x='genotype', y='normalized\_titer\_zscore', data=df\_merged\_new, hue='unique\_id', dodge=True, marker='o', alpha=0.7, jitter=True, ax=axes[1, 1])  
axes[1, 1].set\_title('Z-Score Normalization')  
axes[1, 1].set\_xticklabels(axes[1, 1].get\_xticklabels(), rotation=45, ha='right')  
axes[1, 1].legend().set\_visible(False) # Hide the legend  
  
# Add a single legend for all subplots  
handles, labels = axes[0, 0].get\_legend\_handles\_labels()  
fig.legend(handles, labels, title='Unique ID', loc='upper right')  
  
# Finalize the layout  
#plt.tight\_layout(rect=[0, 0.03, 1, 0.95]) # Adjust the rectangle in which to fit plots  
plt.show()

## Preliminary Statistical Tests

#### Shapiro-Wilk Test for Normality and Levene’s Test for Homogeneity of Variances

**Shapiro-Wilk Test**: The null hypothesis () for this test is that the data for each genotype (excluding ‘No Cell Control’) follows a normal distribution. Rejecting this hypothesis would mean that the data does not follow a normal distribution, which is an assumption for many statistical tests like ANOVA.So first we need to check for normality.

**Levene’s Test**: The null hypothesis for the Levene’s test is that all genotypes (again, excluding ‘No Cell Control’) have equal variances. Rejecting this hypothesis would imply that the variances are significantly different, which would violate the assumptions for tests like ANOVA.

# Function for Shapiro-Wilk test for normality  
def perform\_shapiro\_test(df, col\_name):  
 shapiro\_results\_list = []  
 for genotype in df['genotype'].unique():  
 subset = df[df['genotype'] == genotype][col\_name]  
 if len(subset) > 2: # Shapiro-Wilk test requires at least 3 observations  
 stat, p = shapiro(subset)  
 shapiro\_results\_list.append({'genotype': genotype, 'statistic': stat, 'p\_value': p})  
 return pd.DataFrame(shapiro\_results\_list)  
  
# Function for Levene's test for homogeneity of variances  
def perform\_levene\_test(df, col\_name):  
 levene\_results\_list = []  
 levene\_stat, levene\_p = levene(\*[df[df['genotype'] == genotype][col\_name] for genotype in df['genotype'].unique()])  
 levene\_results\_list.append({'Normalization\_Type': col\_name, 'Test\_Name': 'Levene', 'P-Value': levene\_p})  
 return pd.DataFrame(levene\_results\_list)  
  
# Function for One-Way ANOVA  
def perform\_anova(df, col\_name):  
 anova\_test\_results = []  
 f\_stat, p\_value = f\_oneway(\*[df[df['genotype'] == genotype][col\_name].dropna() for genotype in df['genotype'].unique()])  
 anova\_test\_results.append({'Normalization\_Type': col\_name, 'F-Statistic': f\_stat, 'P-Value': p\_value})  
 return pd.DataFrame(anova\_test\_results)  
  
# Function for Tukey's HSD Test  
def perform\_tukey\_test(df, col\_name):  
 tukey\_test\_results = []  
 tukey\_result = pairwise\_tukeyhsd(df[col\_name].dropna(), df['genotype'])  
 tukey\_test\_results.append({'Normalization\_Type': col\_name, 'Tukey\_HSD\_Result': tukey\_result})  
 return pd.DataFrame(tukey\_test\_results)  
  
  
# Initialize empty DataFrames to store the results  
shapiro\_results\_incl\_control = pd.DataFrame()  
levene\_test\_df = pd.DataFrame()  
anova\_test\_df = pd.DataFrame()  
tukey\_test\_df = pd.DataFrame()  
  
# List of columns to perform the tests on  
test\_columns = ['titer', 'normalized\_titer\_direct\_sub', 'normalized\_titer\_ratio', 'normalized\_titer\_zscore']  
  
# Perform the tests  
for col in test\_columns:  
 # Shapiro-Wilk Test for Normality  
 shapiro\_df = perform\_shapiro\_test(df\_merged\_new, col)  
 shapiro\_results\_incl\_control = pd.concat([shapiro\_results\_incl\_control, shapiro\_df], ignore\_index=True)  
   
 # Levene's Test for Homogeneity of Variances  
 levene\_df = perform\_levene\_test(df\_merged\_new, col)  
 levene\_test\_df = pd.concat([levene\_test\_df, levene\_df], ignore\_index=True)  
   
 # One-Way ANOVA  
 anova\_df = perform\_anova(df\_merged\_new, col)  
 anova\_test\_df = pd.concat([anova\_test\_df, anova\_df], ignore\_index=True)  
   
 # Tukey's HSD Test  
 tukey\_df = perform\_tukey\_test(df\_merged\_new, col)  
 tukey\_test\_df = pd.concat([tukey\_test\_df, tukey\_df], ignore\_index=True)  
  
# Show the first few rows of each test result to confirm  
shapiro\_results\_incl\_control.head(), levene\_test\_df.head(), anova\_test\_df.head(), tukey\_test\_df.head()

## ( genotype statistic p\_value  
## 0 NA1000 0.936878 0.458711  
## 1 NA1000 ▲pilA 0.887754 0.110242  
## 2 bNY30a 0.892170 0.125683  
## 3 pilAT36C 0.955040 0.745259  
## 4 No Cell Control 0.943099 0.250764, Normalization\_Type Test\_Name P-Value  
## 0 titer Levene 0.411154  
## 1 normalized\_titer\_direct\_sub Levene 0.034100  
## 2 normalized\_titer\_ratio Levene 0.012493  
## 3 normalized\_titer\_zscore Levene 0.034100, Normalization\_Type F-Statistic P-Value  
## 0 titer 73.457536 1.247883e-22  
## 1 normalized\_titer\_direct\_sub 140.933711 4.355312e-30  
## 2 normalized\_titer\_ratio 123.840023 1.474935e-28  
## 3 normalized\_titer\_zscore 140.933711 4.355312e-30, Normalization\_Type Tukey\_HSD\_Result  
## 0 titer Multiple Comparison of Means - Tuke...  
## 1 normalized\_titer\_direct\_sub Multiple Comparison of Means - Tuk...  
## 2 normalized\_titer\_ratio Multiple Comparison of Means - Tukey ...  
## 3 normalized\_titer\_zscore Multiple Comparison of Means - Tukey ...)

##### Analysis of Tests

1. **Shapiro-Wilk Test**: All genotypes passed the test for normality, supporting the null hypothesis. This allows us to proceed with ANOVA and other parametric tests.

#### now averages of averages for normality and variance

Up until now I believe we have been working with grouping od unique\_ids which represent a single bioreplicate of genotypes (meaning all genotype should have 3 or more bioreplicates).

However, since we have normalized presumably reducing the day to day variability of the experiment (technical) we may now think about used the mean of genotype, unique id pairings to take the mean of the genotype groupings across the days whichwould yield a single value with a singular standard deviation. Before we do this I believe now we could run levenes test for equal variances across groups (genotypes0 and check for normality of the groups (by genotype) would this make sense and be correct logic?> What would be the cons of this approach and what would be the pros ?

Pros of This Approach: Simplification: This would simplify the data structure considerably, making statistical tests less complex and easier to interpret.

Reduced Noise: Normalization followed by averaging would likely reduce both biological and technical noise, making the signal (differences between genotypes) easier to detect.

Statistical Power: With reduced noise and simpler data structure, you might gain statistical power, making it easier to detect real differences between groups.

Cons of This Approach: Loss of Information: You would be discarding information about the within-group (i.e., within each unique ID) variability. This could be problematic if that variability is of scientific interest.

Assumption of Equal Variance: By reducing the data in this manner, you’re implicitly assuming that the variance within each genotype-unique ID pairing is comparable. If this is not the case, the averaged data may be misleading.

Normality: This approach assumes that the mean of each genotype across unique IDs is normally distributed. While this is often a reasonable assumption due to the Central Limit Theorem, it should be empirically tested.

Given these considerations, you could proceed with Levene’s test for equal variances and a normality test on the reduced dataset (mean values for each genotype across unique IDs). This would give you a cleaner, simpler dataset to work with while also allowing you to test the necessary assumptions before proceeding to ANOVA.

1. **Levene’s Test**: The test for variance homogeneity also passed for all normalization types, again supporting the null hypothesis. This indicates that the variances are not significantly different across the genotypes, meeting the assumption for ANOVA.

Given that the data meet the assumptions of a normal distribution, further statistical tests like the ANOVA or t-tests to compare the means between different genotypes can be performed (parametric testing)

### One-Way ANOVA

# One-way ANOVA on original data and all three normalization methods  
anova\_test\_results = []  
for col in ['titer', 'normalized\_titer\_direct\_sub', 'normalized\_titer\_ratio', 'normalized\_titer\_zscore']:  
 f\_stat, p\_value = f\_oneway(\*[df\_merged\_new[df\_merged\_new['genotype'] == genotype][col].dropna() for genotype in df\_merged\_new['genotype'].unique()])  
 anova\_test\_results.append({'Normalization\_Type': col, 'F-Statistic': f\_stat, 'P-Value': p\_value})  
  
# Convert the list of dictionaries to a DataFrame  
anova\_test\_df = pd.DataFrame(anova\_test\_results)  
  
# Tukey's HSD Test on original data and all three normalization methods  
tukey\_test\_results = []  
for col in ['titer', 'normalized\_titer\_direct\_sub', 'normalized\_titer\_ratio', 'normalized\_titer\_zscore']:  
 tukey\_result = pairwise\_tukeyhsd(df\_merged\_new[col].dropna(), df\_merged\_new['genotype'])  
 tukey\_test\_results.append({'Normalization\_Type': col, 'Tukey\_HSD\_Result': tukey\_result})  
  
# Convert the list of dictionaries to a DataFrame  
tukey\_test\_df = pd.DataFrame(tukey\_test\_results)  
  
# Display One-Way ANOVA and Tukey's HSD test results  
anova\_test\_df, tukey\_test\_df

## ( Normalization\_Type F-Statistic P-Value  
## 0 titer 73.457536 1.247883e-22  
## 1 normalized\_titer\_direct\_sub 140.933711 4.355312e-30  
## 2 normalized\_titer\_ratio 123.840023 1.474935e-28  
## 3 normalized\_titer\_zscore 140.933711 4.355312e-30, Normalization\_Type Tukey\_HSD\_Result  
## 0 titer Multiple Comparison of Means - Tuke...  
## 1 normalized\_titer\_direct\_sub Multiple Comparison of Means - Tuk...  
## 2 normalized\_titer\_ratio Multiple Comparison of Means - Tukey ...  
## 3 normalized\_titer\_zscore Multiple Comparison of Means - Tukey ...)

##### Analysis of Tests

**ANOVA**: For all types of titer values (original, direct subtraction normalized, and ratio normalized), the p-values were less than 0.001. This strongly suggests rejecting the null hypothesis, indicating significant differences in means across genotypes.

**Tukey’s HSD**: Similar to ANOVA, the results indicate that there are pairs of genotypes that have significantly different means, leading to the rejection of the null hypothesis for those pairs.

## Interpretations

1. **Original Titer Values**: Significant differences were observed between the genotype NA1000 and the other genotypes (bNY30a, pilAT36C, and No Cell Control). The p-values were below 0.001, suggesting strong evidence against the null hypothesis of equal means.
2. **Direct Subtraction Normalized Values**: Similar trends were observed as with the original titer values. The normalization did not drastically change the significance levels between different genotypes.
3. **Ratio Normalized Values**: Again, the normalization seems consistent with the original titer values in terms of statistical significance.

In summary, the statistical tests indicate that the different genotypes have significantly different titer levels. The assumptions for normality and homogeneity of variances were met, allowing for a robust statistical analysis. These findings are consistent across different types of normalization, suggesting that the observed differences are not artifacts of data transformation but are indicative of real biological variations.

#### Averages of Averages

# Calculate the "average of averages" for each genotype for each unique\_id and each normalization method  
average\_of\_averages\_by\_id\_df = df\_merged\_new.groupby(['unique\_id', 'genotype']).agg({  
 'titer': 'mean',  
 'normalized\_titer\_direct\_sub': 'mean',  
 'normalized\_titer\_ratio': 'mean',  
 'normalized\_titer\_zscore': 'mean'  
}).reset\_index()  
  
# Calculate the overall "average of averages" for each genotype and each normalization method  
average\_of\_averages\_overall\_df = average\_of\_averages\_by\_id\_df.groupby('genotype').agg({  
 'titer': ['mean', 'std'],  
 'normalized\_titer\_direct\_sub': ['mean', 'std'],  
 'normalized\_titer\_ratio': ['mean', 'std'],  
 'normalized\_titer\_zscore': ['mean', 'std']  
}).reset\_index()  
  
# Flatten the multi-index for easier handling  
average\_of\_averages\_overall\_df.columns = ['\_'.join(col).rstrip('\_') for col in average\_of\_averages\_overall\_df.columns.values]  
  
# Show the new DataFrame to confirm  
average\_of\_averages\_overall\_df

## genotype ... normalized\_titer\_zscore\_std  
## 0 NA1000 ... 8.331942e-01  
## 1 NA1000 ▲pilA ... 1.115869e+00  
## 2 No Cell Control ... 3.778134e-16  
## 3 bNY30a ... 7.065548e-01  
## 4 pilAT36C ... 9.019879e-01  
##   
## [5 rows x 9 columns]

#### Statistics of Averages

from scipy.stats import ttest\_ind  
  
# Perform statistical tests on the "average of averages" data  
  
# Initialize empty DataFrames to store the results for "average of averages"  
shapiro\_avg\_of\_avg\_results = pd.DataFrame()  
anova\_avg\_of\_avg\_test\_df = pd.DataFrame()  
tukey\_avg\_of\_avg\_test\_df = pd.DataFrame()  
  
# Perform the tests on "average of averages" data  
for col in test\_columns:  
 # Shapiro-Wilk Test for Normality on average of averages  
 shapiro\_avg\_of\_avg\_df = perform\_shapiro\_test(average\_of\_averages\_by\_id\_df, col)  
 shapiro\_avg\_of\_avg\_results = pd.concat([shapiro\_avg\_of\_avg\_results, shapiro\_avg\_of\_avg\_df], ignore\_index=True)  
   
 # One-Way ANOVA on average of averages  
 anova\_avg\_of\_avg\_df = perform\_anova(average\_of\_averages\_by\_id\_df, col)  
 anova\_avg\_of\_avg\_test\_df = pd.concat([anova\_avg\_of\_avg\_test\_df, anova\_avg\_of\_avg\_df], ignore\_index=True)

## C:\Users\MicrobeJ\ANACON~1\envs\RStudio\Lib\site-packages\scipy\stats\\_morestats.py:1879: UserWarning: Input data for shapiro has range zero. The results may not be accurate.  
## warnings.warn("Input data for shapiro has range zero. The results "

# Show the first few rows of each test result to confirm  
shapiro\_avg\_of\_avg\_results, anova\_avg\_of\_avg\_test\_df

## ( genotype statistic p\_value  
## 0 NA1000 0.881508 0.345116  
## 1 NA1000 ▲pilA 0.937362 0.638309  
## 2 No Cell Control 0.875009 0.205199  
## 3 bNY30a 0.913168 0.499346  
## 4 pilAT36C 0.873916 0.306715  
## 5 NA1000 0.910718 0.486212  
## 6 NA1000 ▲pilA 0.857273 0.250596  
## 7 No Cell Control 0.857713 0.144397  
## 8 bNY30a 0.802703 0.107157  
## 9 pilAT36C 0.996737 0.890848  
## 10 NA1000 0.950829 0.721298  
## 11 NA1000 ▲pilA 0.827550 0.161503  
## 12 No Cell Control 1.000000 1.000000  
## 13 bNY30a 0.874456 0.315500  
## 14 pilAT36C 0.969040 0.662195  
## 15 NA1000 0.910718 0.486213  
## 16 NA1000 ▲pilA 0.857273 0.250596  
## 17 No Cell Control 0.869115 0.182344  
## 18 bNY30a 0.802703 0.107157  
## 19 pilAT36C 0.996737 0.890848, Normalization\_Type F-Statistic P-Value  
## 0 titer 24.866752 6.501814e-07  
## 1 normalized\_titer\_direct\_sub 56.775188 1.300948e-09  
## 2 normalized\_titer\_ratio 46.500455 6.151280e-09  
## 3 normalized\_titer\_zscore 56.775188 1.300948e-09)

# Create an empty list to store t-test results  
ttest\_results = []  
  
# Calculate the t-test for each genotype against the "No Cell Control" for each normalization method  
for col in ['titer', 'normalized\_titer\_direct\_sub', 'normalized\_titer\_ratio', 'normalized\_titer\_zscore']:  
 # Extract the "average of averages" data for the "No Cell Control"  
 no\_cell\_control\_data = average\_of\_averages\_by\_id\_df[average\_of\_averages\_by\_id\_df['genotype'] == 'No Cell Control'][col]  
   
 # Loop through each genotype to perform the t-test  
 for genotype in average\_of\_averages\_by\_id\_df['genotype'].unique():  
 if genotype != 'No Cell Control':  
 # Extract the "average of averages" data for the current genotype  
 genotype\_data = average\_of\_averages\_by\_id\_df[average\_of\_averages\_by\_id\_df['genotype'] == genotype][col]  
   
 # Perform the t-test  
 t\_stat, p\_value = ttest\_ind(no\_cell\_control\_data, genotype\_data)  
   
 # Store the results  
 ttest\_results.append({  
 'Normalization\_Type': col,  
 'Genotype': genotype,  
 'T-Statistic': t\_stat,  
 'P-Value': p\_value  
 })

## C:\Users\MicrobeJ\ANACON~1\envs\RStudio\Lib\site-packages\scipy\stats\\_axis\_nan\_policy.py:523: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.  
## res = hypotest\_fun\_out(\*samples, \*\*kwds)

# Convert the list of dictionaries to a DataFrame  
ttest\_results\_df = pd.DataFrame(ttest\_results)  
  
# Show the means for each genotype used in the t-test for each normalization type  
mean\_values = average\_of\_averages\_by\_id\_df.groupby(['genotype']).agg({  
 'titer': 'mean',  
 'normalized\_titer\_direct\_sub': 'mean',  
 'normalized\_titer\_ratio': 'mean',  
 'normalized\_titer\_zscore': 'mean'  
}).reset\_index()  
  
ttest\_results\_df, mean\_values

## ( Normalization\_Type Genotype T-Statistic P-Value  
## 0 titer NA1000 2.725152 2.340955e-02  
## 1 titer NA1000 ▲pilA 2.226827 5.297050e-02  
## 2 titer bNY30a 8.335814 1.591339e-05  
## 3 titer pilAT36C 6.429573 2.026553e-04  
## 4 normalized\_titer\_direct\_sub NA1000 3.188478 1.103350e-02  
## 5 normalized\_titer\_direct\_sub NA1000 ▲pilA 1.868448 9.453223e-02  
## 6 normalized\_titer\_direct\_sub bNY30a 19.688258 1.042844e-08  
## 7 normalized\_titer\_direct\_sub pilAT36C 18.751550 6.756801e-08  
## 8 normalized\_titer\_ratio NA1000 3.078837 1.316467e-02  
## 9 normalized\_titer\_ratio NA1000 ▲pilA 1.841564 9.866593e-02  
## 10 normalized\_titer\_ratio bNY30a 16.129822 5.994737e-08  
## 11 normalized\_titer\_ratio pilAT36C 21.972496 1.943053e-08  
## 12 normalized\_titer\_zscore NA1000 3.188478 1.103350e-02  
## 13 normalized\_titer\_zscore NA1000 ▲pilA 1.868448 9.453223e-02  
## 14 normalized\_titer\_zscore bNY30a 19.688258 1.042844e-08  
## 15 normalized\_titer\_zscore pilAT36C 18.751550 6.756801e-08, genotype titer ... normalized\_titer\_ratio normalized\_titer\_zscore  
## 0 NA1000 166.250000 ... 0.894686 -9.613608e-01  
## 1 NA1000 ▲pilA 170.500000 ... 0.917339 -7.544857e-01  
## 2 No Cell Control 203.714286 ... 1.000000 -5.286776e-18  
## 3 bNY30a 82.583333 ... 0.440919 -5.033961e+00  
## 4 pilAT36C 107.444444 ... 0.473664 -5.835771e+00  
##   
## [5 rows x 5 columns])

#### Barplots

# Creating the barplot using the "average of averages" data  
  
# Initialize the figure with modified dimensions  
fig, axes = plt.subplots(2, 2, figsize=(20, 20))  
  
# List of normalization types  
norm\_types = ['titer', 'normalized\_titer\_direct\_sub', 'normalized\_titer\_ratio', 'normalized\_titer\_zscore']  
  
# Titles for subplots  
titles = ['Original Data', 'Direct Subtraction Normalization', 'Ratio Normalization', 'Z-Score Normalization']  
  
# Create barplots for each normalization type  
for ax, col, title in zip(axes.flatten(), norm\_types, titles):  
 sns.barplot(x='genotype', y=f"{col}\_mean", data=average\_of\_averages\_overall\_df, ax=ax, yerr=average\_of\_averages\_overall\_df[f"{col}\_std"], capsize=.2)  
 ax.set\_title(title)  
 ax.set\_ylabel('Mean Titer')  
 ax.set\_xticklabels(ax.get\_xticklabels(), rotation=45, ha='right')  
  
# Finalize the layout  
plt.tight\_layout()  
plt.show()