Part 1

In [1]:
```python
import pandas as pd
```

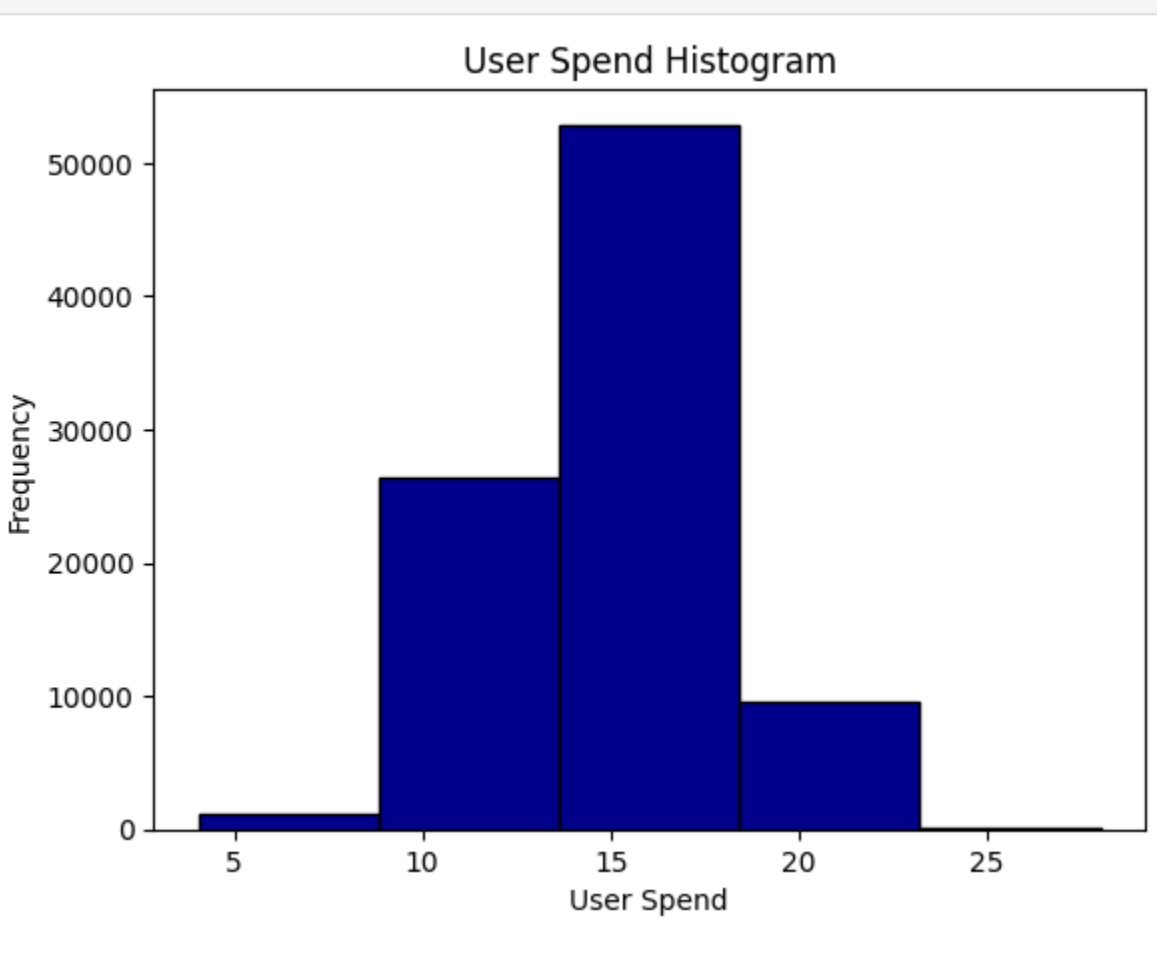
In [60]:
```python
lobster_run = pd.read_csv("lobster_run.csv")
```

In [11]:
```python
lobster_run.head(10)
```

Out[11]:

	userid	version	sum_gamerounds	retention_1	retention_7	user_spend
0	1	Gulf of Maine	3	False	False	17.09
1	2	Gulf of Maine	38	True	False	15.30
2	7	Gulf of Maine	0	False	False	13.67
3	12	Gulf of Maine	0	False	False	16.92
4	14	Gulf of Maine	39	True	False	8.42
5	15	Gulf of Maine	305	True	False	16.54
6	16	Gulf of Maine	73	True	False	16.68
7	17	Gulf of Maine	14	True	False	8.79
8	18	Gulf of Maine	204	True	False	11.03
9	19	Gulf of Maine	108	True	True	13.05

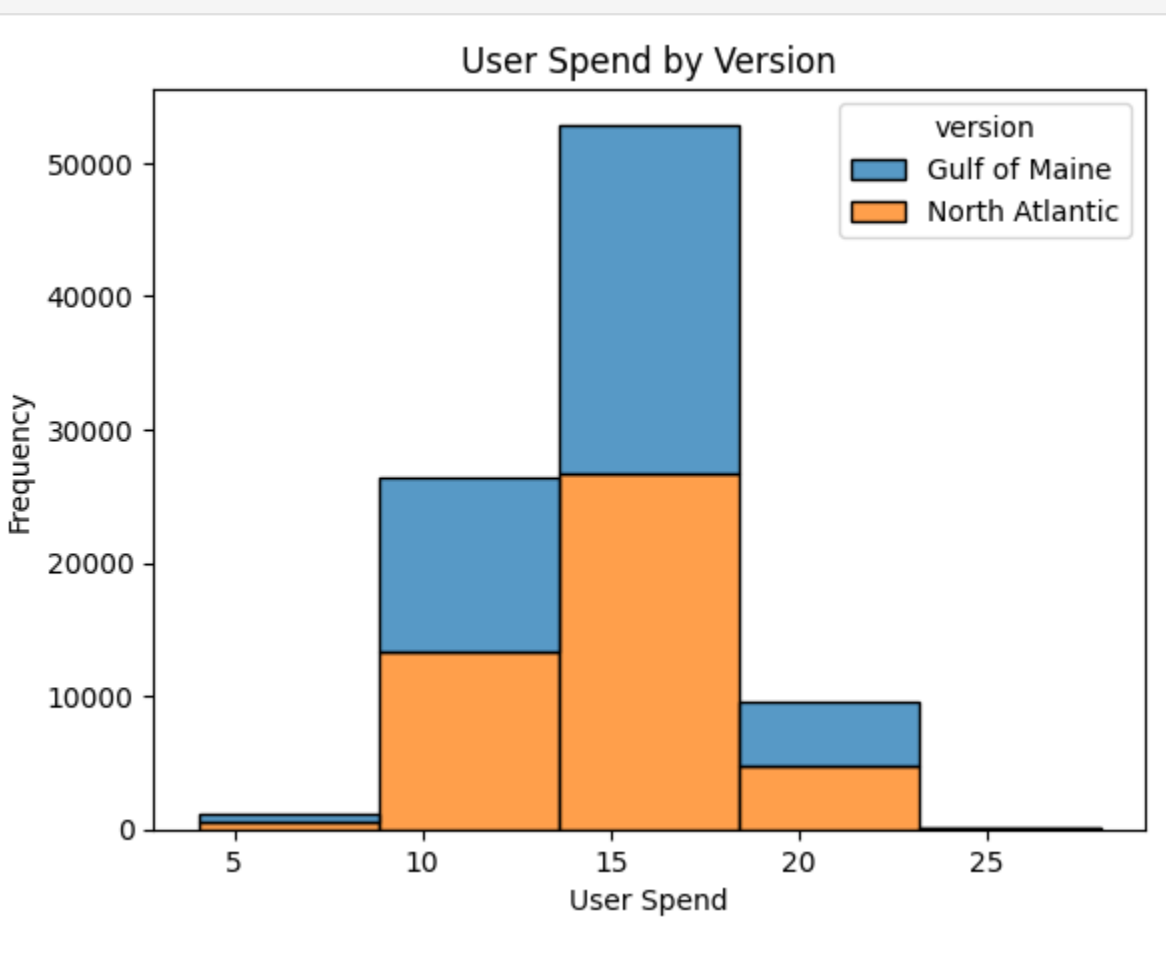
In [46]:
```python
import matplotlib.pyplot as plt

# Define the data
data = lobster_run

user_spend = data['user_spend']

# Create histogram
plt.hist(user_spend, bins=5, color='darkblue', edgecolor='black')

# Add labels and title
plt.title('User Spend Histogram')
plt.xlabel('User Spend')
plt.ylabel('Frequency')

# Show histogram
plt.show()
```



Describe your plot in 1-2 sentences?

The histogram shows that users across both the Gulf of Maine and North Atlantic app versions have spent a significant amount on the game, with a high frequency of users having spent at least $15. This suggests that both versions of the game are engaging and enjoyable for users, which could be a positive indicator for the game's overall success and user retention.

In [49]:
```python
import pandas as pd
import seaborn as sns


user_spend = data['user_spend']
version = data['version']

# Create histogram with hue variable
sns.histplot(data, x=user_spend, hue=version, bins=5, multiple='stack')

# Add labels and title
plt.title('User Spend by Version')
plt.xlabel('User Spend')
plt.ylabel('Frequency')

# Show histogram
plt.show()
```



Describe your plot in 1-2 sentences?

This histogram provides detailed information about which app version users have spent the most money on in in-app purchases. The data suggests that the Gulf of Maine version has a significantly higher number of observations compared to the North Atlantic version, indicating that more users are making in-app purchases in the Gulf of Maine version.

This observation implies that the Gulf of Maine version may be more enticing to users, or that the in-app purchases offered in that version may be more appealing or relevant to users compared to the North Atlantic version. This information can be valuable for game developers and marketers looking to optimize in-app purchases and monetization strategies for their app.

In [61]:
```python
from scipy.stats import shapiro
import pandas as pd

# extract user_spend variable
user_spend = lobster_run['user_spend']

# perform Shapiro-Wilk test
stat, p = shapiro(user_spend)

# print test statistic and p-value
print('Test statistic:', stat)
print('p-value:', p)

# interpret the result
alpha = 0.05
if p > alpha:
    print('User spend variable is normally distributed (fail to reject H0)')
else:
    print('User spend variable is not normally distributed (reject H0)')
```

```
Test statistic: 0.9999650120735168
p-value: 0.5058879852294922
User spend variable is normally distributed (fail to reject H0)
/usr/local/lib/python3.9/dist-packages/scipy/stats/_morestats.py:1816: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

a. The null hypothesis of the test is that the user spend variable is normally distributed.

b. The p-value obtained from the normality test is 0.5058879852294922, which is greater than the commonly used significance level of 0.05. Therefore, we fail to reject the null hypothesis and conclude that the user spend variable is normally distributed.

In [17]:
```python
from scipy.stats import ttest_ind

gulf_of_maine = lobster_run[lobster_run['version'] == 'Gulf of Maine']['user_spend']
north_atlantic = lobster_run[lobster_run['version'] == 'North Atlantic']['user_spend']

t_stat, p_val = ttest_ind(gulf_of_maine, north_atlantic, equal_var=False)

alpha = 0.05
if p_val < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")

print("Test statistic:", t_stat)
print("p-value:", p_val)
```

```
Fail to reject the null hypothesis
Test statistic: 1.5874056946017774
p-value: 0.1124242973851463
```

a. The null hypothesis was that there is no significant difference in user spending between the Gulf of Maine and North Atlantic versions.

b. Based on the test, with a p-value of 0.112, we fail to reject the null hypothesis. This suggests that there is not enough evidence to conclude that there is a significant difference in user spending between the two versions.

Part 2

In [19]:
```python
# Assuming your DataFrame is named df
summary_stats = lobster_run.groupby('version')[['retention_1', 'retention_7']].agg(['mean', 'std', 'count'])

print(summary_stats)
```

```
               retention_1                  retention_7
                      mean       std  count        mean       std  count
version
Gulf of Maine     0.448188  0.497314  44700    0.195615  0.396678  44700
North Atlantic    0.442283  0.496663  45489    0.182000  0.385849  45489
```

Based on the table, we can see that the mean retention rates for both versions are relatively similar, with Gulf of Maine having slightly higher retention rates at both Day 1 and Day 7 compared to North Atlantic. However, the standard deviation values are quite high for both versions, indicating that there is a large variation in the retention rates among users.

Without conducting a statistical test, it's difficult to definitively conclude whether the version is impacting user behavior. However, based on the summary stats, it seems that there may be some differences in retention rates between the two versions, with Gulf of Maine having slightly higher retention rates.

F

a. What will be the null hypothesis of your test?

version doesnt impact the likehood of retention

In [62]:
```python
gulf_of_maine = lobster_run[lobster_run['version'] == 'Gulf of Maine']
num_users_gulf = (gulf_of_maine['retention_1'] == 1).sum()
print("Number of users who stayed with the game for one day in Gulf of Maine version:", num_users_gulf)
```

```
Number of users who stayed with the game for one day in Gulf of Maine version: 20034
```

In [82]:
```python
pd.value_counts(lobster_run['version'],normalize=True)
```

Out[82]:
```
North Atlantic    0.504374
Gulf of Maine     0.495626
Name: version, dtype: float64
```

In [83]:
```python
pd.value_counts(lobster_run['retention_1'])
```

Out[83]:
```
False    50036
True     40153
Name: retention_1, dtype: int64
```

In [85]:
```python
expected_Gulfmaine= 40153*0.495626
print(expected_Gulfmaine)
```

```
19900.870778
```

What is this expected value? How did you arrive at this value? Expected value is 19900.870778

The expected value can be calculated by multiplying the total number of users who showed continued engagement with the app one day after the download by the expected retention rate under the null hypothesis.

To calculate the expected retention rate under the null hypothesis, we can take the overall retention rate across both versions of the game as the expected rate. This is calculated as the total number of users who showed continued engagement with the app one day after the download across both versions divided by the total number of users.

In [100…
```python
from scipy import stats

# Sample size and retention rate for Gulf of Maine version
n1 = 44700
x1 = int(0.448188 * n1)


# Expected retention rate under the null hypothesis
p0 = 19900.870778 / n1

# Perform binomial test
p_value = stats.binom_test(x1, n=n1, p=p0, alternative='two-sided');

# Hypothesis testing
alpha = 0.05
if p_value <= alpha:
    print(f"p-value = {p_value:.4f} is less than or equal to the significance level of {alpha}, so we reject the null hypothesis.")
else:
    print(f"p-value = {p_value:.4f} is greater than the significance level of {alpha}, so we fail to reject the null hypothesis.")
```

```
p-value = 0.2056 is greater than the significance level of 0.05, so we fail to reject the null hypothesis.
<ipython-input-100-125cebc887f5>:12: DeprecationWarning: 'binom_test' is deprecated in favour of 'binomtest' from version 1.7.0 and will be removed in Scipy 1.12.0.
  p_value = stats.binom_test(x1, n=n1, p=p0, alternative='two-sided');
```

Based on the test, we fail to reject the null hypothesis. version doesnt impact the likehood of retention

G

What will be the null hypothesis of your test?

version doesnt impact the likehood of retention

In [101…
```python
gulf_of_maine = lobster_run[lobster_run['version'] == 'Gulf of Maine']
num_users_gulf = (gulf_of_maine['retention_7'] == 1).sum()
print("Number of users who stayed with the game for one day in Gulf of Maine version:", num_users_gulf)
```

```
Number of users who stayed with the game for one day in Gulf of Maine version: 8744
```

In [102…
```python
pd.value_counts(lobster_run['version'],normalize=True)
```

Out[102]:
```
North Atlantic    0.504374
Gulf of Maine     0.495626
Name: version, dtype: float64
```

In [103…
```python
pd.value_counts(lobster_run['retention_7'])
```

Out[103]:
```
False    73166
True     17023
Name: retention_7, dtype: int64
```

In [104…
```python
expected_Gulfmaine= 17023*0.495626
print(expected_Gulfmaine)
```

```
8437.041398
```

What is this expected value? How did you arrive at this value? Expected value is 8437.041398

The expected value can be calculated by multiplying the total number of users who showed continued engagement with the app 7 days after the download by the expected retention rate under the null hypothesis.

To calculate the expected retention rate under the null hypothesis, we can take the overall retention rate across both versions of the game as the expected rate. This is calculated as the total number of users who showed continued engagement with the app 7 days after the download across both versions divided by the total number of users.

In [105…
```python
from scipy import stats

# Sample size and retention rate for Gulf of Maine version
n1 = 44700
x1 = int(0.195615 * n1)


# Expected retention rate under the null hypothesis
p0 = 8437.041398/ n1

# Perform binomial test
p_value = stats.binom_test(x1, n=n1, p=p0, alternative='two-sided');

# Hypothesis testing
alpha = 0.05
if p_value <= alpha:
    print(f"p-value = {p_value:.4f} is less than or equal to the significance level of {alpha}, so we reject the null hypothesis.")
else:
    print(f"p-value = {p_value:.4f} is greater than the significance level of {alpha}, so we fail to reject the null hypothesis.")
```

```
p-value = 0.0002 is less than or equal to the significance level of 0.05, so we reject the null hypothesis.
<ipython-input-105-73c987b2046f>:12: DeprecationWarning: 'binom_test' is deprecated in favour of 'binomtest' from version 1.7.0 and will be removed in Scipy 1.12.0.
  p_value = stats.binom_test(x1, n=n1, p=p0, alternative='two-sided');
```

Based on the test, we reject the null hypothesis. version does impact the likehood of retention.

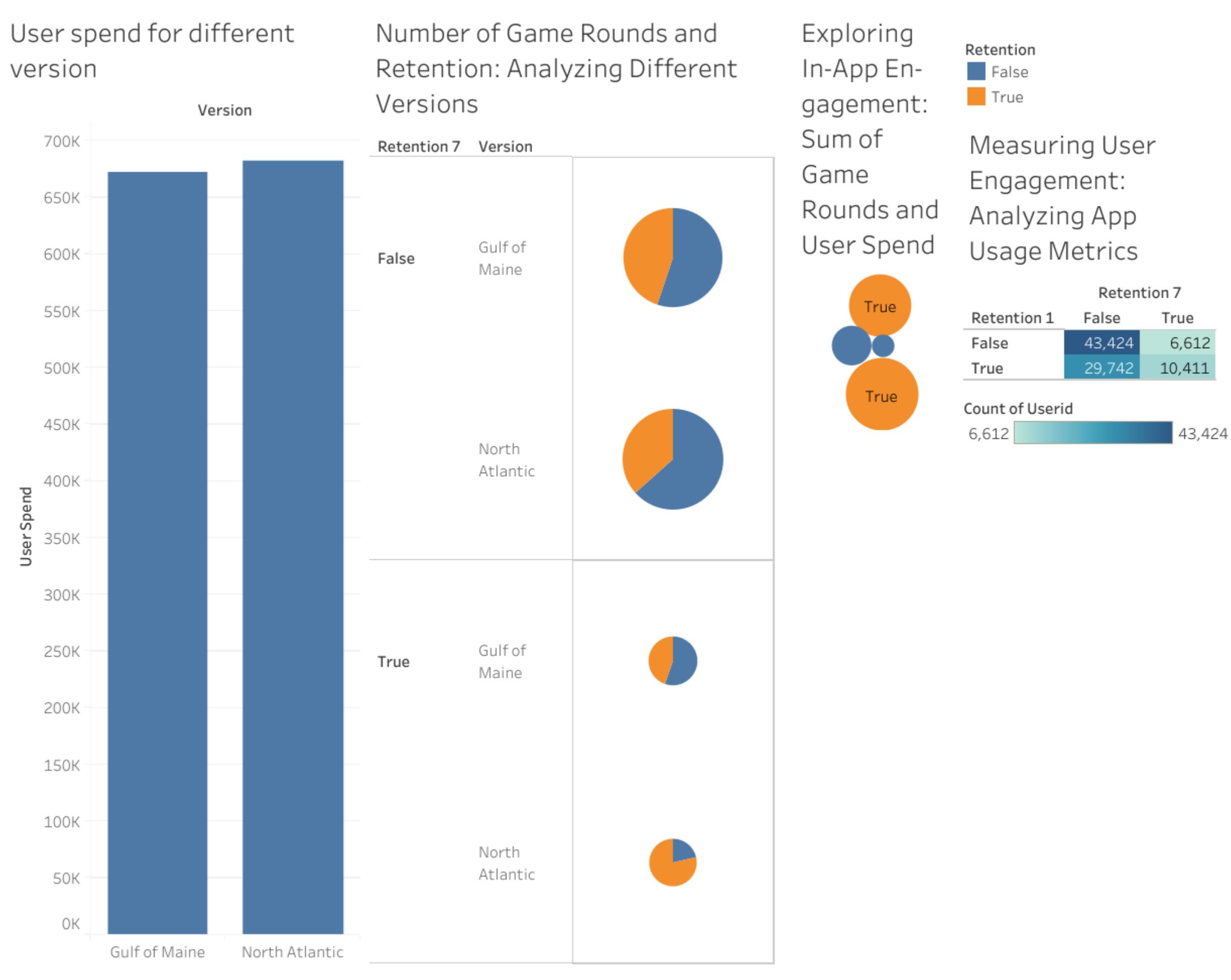
H

Based on the results of the statistical tests, it appears that the version of the game does not have a significant impact on retention one day after download, but it may have a significant impact on retention seven days after download. Therefore, Lobster Land may want to consider further investigating the factors that affect retention after seven days and make changes to the game based on those findings. Additionally, it may be helpful to gather more data on user behavior and demographics to gain a more comprehensive understanding of how to improve retention rates.

To put these results into context, it would be helpful to gather more information about the user demographic, their behavior within the app, and the features of the app itself. Understanding the reasons why users may be more likely to retain in one version of the app versus the other can inform future development and marketing efforts. Additionally, it would be beneficial to conduct further testing and analysis to validate these results and ensure that they are not due to chance or bias.

Part III: Using Tableau to Build a Dashboard

Image of my dashboard created of different plots



My dashboard

Tableau dashboard link

Descriptions of the different plots-

1)Bar plot- **User spend for different version**

bar plot displays the user spend for different versions of the app, providing valuable insights into the amount spent by users while playing the game. The Gulf of Maine version of the app shows a total spend of 671,866, while the North Atlantic version shows a slightly higher total spend of 682,403.

2)Pie chart - **Number of Game Rounds and Retention: Analyzing Different Versions**

This visualization consists of two pie charts, with the first column representing retention 7 and the second column representing version. The first half of the pie charts show retention 1 when retention 7 is false for both versions, while the second half shows retention 1 when retention 7 is true.

This visualization helps in analyzing user engagement by calculating the sum of gamerounds for each version of the game. Notably, the North Atlantic version has the highest number of gamerounds when retention is zero for both day 1 and day 7, with 23,597 gamerounds. This suggests that users of the North Atlantic version are highly engaged with the game, even if they haven't returned to play for multiple days continuously.

3)Bubble-chart -**Exploring In-App Engagement: Sum of Game Rounds and User Spend**

This bubble chart provides a comprehensive summary of in-app engagement by displaying the total number of gamerounds and the amount of money spent by users for each combination of engagement on the first and seventh day after downloading the app.

Notably, users who engaged with the app for one day have played the highest number of gamerounds, with 2,197,029 gamerounds, and have spent $446,615. This indicates that early engagement is a key factor in driving user engagement and in-app spending.

Conversely, users who did not engage with the app on the first day but showed continued engagement on the seventh day have the lowest number of gamerounds and user spend. This suggests that early engagement is critical for long-term user engagement and in-app spending.

4) **Measuring User Engagement: Analyzing App Usage Metrics**

This table displays the number of users for each combination of engagement on the first and seventh day after downloading the app. Users are classified as "continued engagement" if they played the game on both the first and seventh day, and "no continued engagement" if they did not play on both days.

Overall, the majority of users did not show continued engagement with the app one day after downloading and seven days after downloading. Specifically, 43,424 users did not show continued engagement on both days, while only 10,411 users continued engagement on both days.

In [1]:
```
!pip install nbconvert
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (6.5.4)
Requirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages (from nbconvert) (6.0.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.8.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (from nbconvert) (23.0)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.7.3)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.3.0)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.2.2)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (1.2.1)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbconvert) (4.9.2)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (3.1.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (4.11.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (2.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (2.14.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (5.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert) (0.7.1)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.9/dist-packages (from jupyter-core>=4.7->nbconvert) (3.2.0)
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.9/dist-packages (from nbclient>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-packages (from nbformat>=5.1->nbconvert) (4.3.3)
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packages (from nbformat>=5.1->nbconvert) (2.16.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-packages (from beautifulsoup4->nbconvert) (2.4)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/dist-packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (22.2.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.9/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.19.3)
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.9/dist-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.2)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.9/dist-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (23.2.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.9/dist-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
```

In [2]:
```
!jupyter nbconvert --to html assignment4.ipynb
```

```
[NbConvertApp] WARNING | pattern 'assignment4.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the cell output (the default behaviour is to abort conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook. To recover
                                  previous default behaviour (outputting to the current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```