# From Data to Discovery:

# A Complete Walkthrough of Aridhia Workspaces

*Analysing the Breast Cancer Wisconsin Dataset in a Secure Research Environment*

## Introduction

In biomedical research, the challenge is rarely just having data: it's having a secure, compliant environment where you can actually work with that data. Between institutional review boards, data protection regulations, and the practical need for analytical tools, researchers often spend months navigating infrastructure before they can begin meaningful analysis.

Aridhia Project Workspaces provide a fully certified Trusted Research Environment (TRE) that you can be working in within 24 hours. No infrastructure setup, no lengthy procurement processes, no security consultants required.

In this walkthrough, we'll demonstrate a complete analytical workflow using the well-known Breast Cancer Wisconsin (Diagnostic) Dataset. You'll see every step from uploading your data through a secure airlock to exporting your research findings. This includes data ingestion, data cleansing, spreadsheet analysis, SQL querying, statistical modelling in python and R, reproducible reporting, and the governed export process that maintains data integrity throughout.

### What You'll Need

- An Aridhia Workspaces account
- The Breast Cancer Wisconsin (Diagnostic) Dataset in CSV format (download instructions below)
- About 90-120 minutes to complete the full walkthrough

### Downloading the Dataset from Kaggle:

The Breast Cancer Wisconsin (Diagnostic) Dataset is freely available on Kaggle. Follow these steps to download it:

1. Go to kaggle.com/datasets/uciml/breast-cancer-wisconsin-data in your browser
2. If you don't have a Kaggle account, click **Register** to create a free account
3. Once logged in, click the **Download** button near the top of the page
4. A ZIP file (*archive.zip*) will download to your computer
5. Extract the ZIP file. Inside you'll find *data.csv*. This is the file you'll upload to your workspace

**Top Tip**
You may wish to rename the file to something more descriptive, such as "*breast_cancer_wisconsin.csv*" before uploading.

# 1 Uploading Your Data Through the Secure Airlock

Every piece of data entering an Aridhia Workspace passes through a secure Inbound Airlock, where it's automatically scanned for malware. For maximum control over what enters your research environment, you can route files to the workspace Inbox for review before they reach your main file storage.

## To Upload Your Dataset:

1. Navigate to **Upload > Upload files or data** from the main menu. The upload wizard will open.
2. Drag and drop your **CSV file** into the upload panel, or click to browse your local files. Select your *breast_cancer_wisconsin.csv* file.
3. Select the **destination:** In the file destination options, select **INBOX**. This routes the file to your Inbox for review rather than directly to Files or Blobs.
4. Confirm **authorisation** by checking the box that confirms you are authorised to upload this data. This audit trail is maintained throughout your research.
5. Click **Upload**. Your file will pass through the Inbound Airlock for automatic malware scanning. Once cleared, it will appear in your workspace Inbox ready for you to review and approve.

## Approving Files from the Inbox:

1. Navigate to **Files > Inbox** from the Files dropdown in the menu ribbon.
2. Your uploaded file will be listed here. You can **preview** it before making your decision by selecting to open it first.
3. Select the file and click **Approve**. A dialogue box opens where you can select a destination in your workspace Files or Blobs storage.
4. Select **Files** as the destination (the root Files folder) and click **Confirm**. The file moves from the Inbox to your Files area and is now ready for analysis.

**Rejecting files:** If you accidentally uploaded the wrong file, selecting **Reject** will permanently delete it from the Inbound Airlock. Rejected files cannot be retrieved.

# 2 Inspecting Your Data in Collabora Calc

Before diving into analysis, it's good practice to inspect your data. In Aridhia Workspaces, **CSV files open directly in Collabora Calc by default**. Simply click on the file and it will open in a spreadsheet view within the secure environment.

## Opening your CSV:

1. Navigate to the **Files** tab and locate your uploaded CSV file.
2. Click on the **filename**. The file automatically opens in Collabora Calc in a new tab.
3. Use the **search bar** (Ctrl+F) to navigate to specific values, or scroll through to understand your data structure.

The Breast Cancer Wisconsin dataset contains 569 samples with 32 features, including the sample ID, diagnosis (M = malignant, B = benign), and 30 real-valued features computed from digitised images of fine needle aspirates. In Collabora, you can quickly verify the data loaded correctly and get a sense of the feature distributions.

**Note**
Files open in read-only mode by default, which protects your source data from accidental modification.

# 2B Analysing Data in Collabora Calc

While Collabora opens files in read-only mode by default, you can switch to **edit mode** to perform spreadsheet analysis, add calculations, and create new files. Collabora Calc includes data analysis tools in the **Data menu** that are familiar to Excel users.

## Switching to Edit Mode and Saving as Excel

**Recommended:** Before performing any analysis, save your CSV as an Excel file. This preserves your original data and gives you a working copy for analysis.

1. With your CSV file open, look at the **bottom right corner** of the Collabora window.
2. Click the **edit icon** (pencil) to switch from read-only to read/write mode.
3. Immediately go to **File > Save As**.
4. Save as *breast_cancer_analysis.xlsx* (Excel format).
5. You now have a working Excel file for analysis, while your original CSV remains unchanged.

***Why save as .xlsx first?*** Excel format preserves any formulas, formatting, filters, and pivot tables you create. CSV files only store raw values, so any analysis work would be lost if you saved back to CSV.

## Using AutoFilter to Explore the Data

**AutoFilter** lets you filter and sort your data by any column, which is useful for exploring patterns in your dataset. Follow these steps to try it for yourself:

1. Click anywhere in your data, then go to **Data > Autofilter**.
2. Dropdown arrows appear in each column header.
3. Click the dropdown on the **diagnosis** column and uncheck "B" to show only malignant cases, or uncheck "M" to show only benign cases.
4. Use the dropdown on numeric columns to sort by ascending/descending, helping you identify samples with extreme values.

## Sorting Your Data

To sort the entire dataset by a specific feature:

1. Select your data range (or click any cell and Calc will auto-detect the range).
2. Go to **Data > Sort**.
3. Choose **radius_mean** as the sort key, select **Descending**, and click OK.

Your data is now sorted with the largest tumours at the top. You will likely notice these are predominantly malignant.

## Adding Summary Calculations

Now we'll add formulas to calculate summary statistics for your dataset.
Scroll to each cell and enter a label and a formula:

| | |
|---|---|
| Cell A575: "Total Samples" | Cell B575: *=COUNTA(A2:A570)* |
| Cell A576: "Malignant (M)" | Cell B576: *=COUNTIF(B2:B570, "M")* |
| Cell A577: "Benign (B)" | Cell B577: *=COUNTIF(B2:B570, "B")* |
| Cell A578: "Avg Radius (Mean)" | Cell B578: *=AVERAGE(C2:C570)* |
| Cell A579: "Std Dev Radius" | Cell B579: *=STDEV(C2:C570)* |

## Summary Calculations Preview

With all of those added, your newly added cells should now look like this:

| | A | B |
|---|---|---|
| 575 | Total Samples | 569 |
| 576 | Malignant (M) | 212 |
| 577 | Benign (B) | 357 |
| 578 | Avg Radius (Mean) | 14.127292 |
| 579 | Std Dev Radius | 3.5240488 |

## Creating a Pivot Table

For detailed analysis, create a Pivot Table to summarise your data by diagnosis:

1. Select any cell within your data
2. Go to **Insert > Pivot Table**
3. In the Select Source dialog, confirm **Current selection** is selected and click **OK**
4. In the Pivot Table Layout dialog, drag **diagnosis** to the **Row Fields** area
5. Drag **radius_mean** to the **Data Fields** area
6. Double-click it to change the function from Sum to **Average** and click **OK**
7. A summary table appears showing the average radius for benign vs malignant tumours

### Why this matters

Collabora Calc provides familiar spreadsheet tools for researchers who want to perform quick exploratory analysis without writing code. The AutoFilter and Pivot Table features are particularly useful for understanding data distributions before moving to more sophisticated statistical methods.

### Saving your work

Remember to save your analysis periodically using **File > Save**. Since you saved as .xlsx at the start, all your filters, formulas, and pivot tables will be preserved.

# 3 Using the Terminal for Shell Scripts

For researchers comfortable with command-line tools, Aridhia Workspaces provide a built-in **Terminal** app that supports bash shell scripts. For more details, see the Aridhia Knowledge Base.

Real-world datasets often arrive with inconsistent formatting, such as spaces in column names, mixed case, special characters, empty columns containing only spaces or null values, and Windows-style line endings (CRLF). These issues can cause problems in R, Python, and SQL.

Let's use the Terminal to create a script that automatically cleans up these common issues.

## Creating a Shell Script in the Workspace:

1. In the **Files** tab, click **New File**
2. Name the file *clean_csv.sh* and click **Create**
3. Click on the file to open it in the built-in text editor
4. Copy and paste the following script into the file (See next page)
5. Save the file using the **Save** button

## Code for the Shell Script:

```
#!/bin/bash

# clean_csv.sh
# Cleans CSV files by:
# - Converting Windows line endings (CRLF) to Unix (LF)
# - Replacing spaces in column names with underscores
# - Removing columns that are empty (contain only spaces or null values)

# Check if input file is provided
if [ -z "$1" ]; then
    echo "Usage: ./clean_csv.sh <input_file.csv> [output_file.csv]"
    echo "If output file is not specified, the input file will be modified in place."
    exit 1
fi

INPUT_FILE="$1"
OUTPUT_FILE="${2:-$1}"

# Check if input file exists
if [ ! -f "$INPUT_FILE" ]; then
    echo "Error: File '$INPUT_FILE' not found."
    exit 1
fi

# Create temporary files
TEMP_FILE=$(mktemp)
TEMP_FILE2=$(mktemp)

# Step 1: Convert CRLF to LF (remove Windows line endings from the data)
tr -d '\r' < "$INPUT_FILE" > "$TEMP_FILE"

# Step 2: Fix column names (replace spaces with underscores) - header only
head -n 1 "$TEMP_FILE" | sed 's/ /_/g' > "$TEMP_FILE2"
tail -n +2 "$TEMP_FILE" >> "$TEMP_FILE2"
mv "$TEMP_FILE2" "$TEMP_FILE"

# Step 3: Identify and remove empty columns
NUM_COLS=$(head -n 1 "$TEMP_FILE" | awk -F',' '{print NF}')

# Build list of non-empty column indices
KEEP_COLS=""
for i in $(seq 1 $NUM_COLS); do
    # Check if column has any non-empty, non-whitespace values
    COL_DATA=$(cut -d',' -f$i "$TEMP_FILE" | tail -n +2 | \
            sed 's/^[ \t]*//;s/[ \t]*$//' | grep -v '^$' | head -1)
    if [ -n "$COL_DATA" ]; then
        if [ -z "$KEEP_COLS" ]; then
            KEEP_COLS="$i"
        else
            KEEP_COLS="$KEEP_COLS,$i"
        fi
    else
        COL_NAME=$(head -n 1 "$TEMP_FILE" | cut -d',' -f$i)
        echo "Removing empty column: $COL_NAME"
    fi
done

# Extract only non-empty columns
cut -d',' -f$KEEP_COLS "$TEMP_FILE" > "$TEMP_FILE2"

# Move result to output location
mv "$TEMP_FILE2" "$OUTPUT_FILE"
rm -f "$TEMP_FILE"

echo ""
echo "CSV cleaned successfully."
echo "Output written to: $OUTPUT_FILE"

# Show the new column names
echo ""
echo "Final column names:"
head -n 1 "$OUTPUT_FILE" | tr ',' '\n' | head -20
```

## Opening the Terminal:

1. Navigate to **Tools > View apps** from the main menu
2. Find **Terminal** in the list of available applications
3. Click **Start**, a terminal window will open with a command prompt

## Running the Script:

Before running your script, you need to convert it to Unix format to make it executable.

1. First, convert the script file to Unix format using **dos2unix**. This ensures the script's own line endings are correct for Linux:
   *dos2unix /home/workspace/files/clean_csv.sh*
2. Make the script executable:
   *chmod +x /home/workspace/files/clean_csv.sh*
3. Run the script on your CSV file when in the files folder:
   *./clean_csv.sh ./breast_cancer_wisconsin.csv*

The script has performed three cleaning operations: it converted any Windows-style line endings (CRLF) to Unix-style (LF), replaced spaces in column names with underscores (e.g., concave points_mean became concave_points_mean), and removed the empty Unnamed: 32 column that Kaggle datasets often include.

**The resulting CSV file uses Unix line endings throughout and is ready for use in R, Python, and SQL without any further conversion.**

### Why use the Terminal?

While you could rename columns manually in Collabora Calc or programmatically in Python/R, shell scripts are ideal for repeatable data preprocessing tasks. You can version control your scripts, apply them to multiple files, and integrate them into automated data pipelines.

# 4 Running a Python Background Job

For reproducible data preprocessing, Aridhia Workspaces support running Python and R scripts as **background jobs**. This is particularly useful for data cleaning, transformation, or any processing that you want to run without keeping an interactive session open. You can run a long term background job and log out of the workspace. The maximum length of a job run is 24 hours. This background job will process the clean CSV and add a new binary column equivalent to diagnosis.

## Creating a Preprocessing Script:

1. Navigate to **Files > New File** and create a new file called *preprocess_data.py*.
2. Enter a preprocessing script that loads the CSV, performs any necessary cleaning (such as handling the ID column or encoding the diagnosis variable), and saves a processed version.
3. Save the file, then close the text editor tab.

## Example Script:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('/home/workspace/files/breast_cancer_wisconsin.csv')

# Encode diagnosis: M=1 (malignant), B=0 (benign)
df['diagnosis_binary'] = (df['diagnosis'] == 'M').astype(int)

# Save processed data
df.to_csv('/home/workspace/files/breast_cancer_processed.csv', index=False)
print(f'Processed {len(df)} records. Malignant: {df.diagnosis_binary.sum()}')
```

### Running as a Background Job:

1. In the Files list, click the **menu button** next to your Python script.
2. Select **Run in background**.
3. Configure the working directory in the dialog that appears.
4. Note the **output directory**: results will be stored in: *files/background_job_output/[filename][timestamp]*.
5. Click Run. This job will complete almost instantly.

## 5 Creating a Database Table

For more sophisticated querying and integration with Aridhia's analytical tools, you'll want to convert your CSV to a database table. Each workspace includes a dedicated PostgreSQL database that integrates with all built-in tools.

### Converting Your CSV to a Database Table

1. In the **Files** tab, locate your processed CSV file.
2. Click the **menu button** and select **Convert to table**.
3. The **New database table wizard** opens.
4. Confirm or modify the table name (e.g. *breast_cancer_data*).
5. Click **Next**. The wizard will auto-detect column data types using a sophisticated type guesser. This will take a few seconds to complete. When complete point you could review the suggested types and adjust if needed. In this case, they will all be correct.
6. Select a **Primary Key**: Choose the id column as your primary key to ensure unique row identification.
7. Click **Confirm** to create the table. A conversion report will be generated and visible in the Activity tab.

Your data is now stored in the workspace's PostgreSQL database and can be queried using SQL, accessed from R and Python, or analysed using the built-in statistical modules.

## 6 Querying Your Data with the SQL Editor

Now that your data lives in a PostgreSQL database, you can leverage the capabilities of SQL for data exploration, aggregation, and complex queries. The built-in **SQL Editor** provides an interactive environment for writing and executing queries directly against your data.

### Opening the SQL Editor:

1. Navigate to **Files > New File**.
2. Create a new file with a **.sql** extension (e.g. *analysis_queries.sql*). This automatically opens the SQL Editor.
3. The **Data panel** on the right shows all available tables. You can drag and drop table names directly into your query.

### Example Query 1:

**Basic data exploration** to get an overview of your dataset:

```
-- Count records by diagnosis
SELECT
   diagnosis,
   COUNT(*) as count,
   ROUND(AVG(radius_mean)::numeric, 2) as avg_radius,
   ROUND(AVG(texture_mean)::numeric, 2) as avg_texture,
   ROUND(AVG(perimeter_mean)::numeric, 2) as avg_perimeter
FROM public.breast_cancer_data
GROUP BY diagnosis
ORDER BY diagnosis;
```

## Example Query 2:

**Identifying high-risk features** to find samples with extreme values:

```
-- Find samples with radius_mean above 2 standard deviations
SELECT
    id,
    diagnosis,
    radius_mean,
    texture_mean,
    perimeter_mean
FROM public.breast_cancer_data
WHERE radius_mean > (
    SELECT AVG(radius_mean) + 2 * STDDEV(radius_mean)
    FROM public.breast_cancer_data
)
ORDER BY radius_mean DESC;
```

## Running and Saving Results

1. Click **Run file** to execute all queries, or highlight specific statements and click **Run selected**.
2. Query results display at the bottom of the editor.
3. Use the toolbar to **export results** to a **new file**, or open in a new tab for further analysis.

### Top Tip

The SQL Editor supports PostgreSQL syntax including window functions, CTEs (WITH clauses), and JSON operations. Table names must be prefixed with *public*. when querying.

# 7 Point-and-Click Analysis with Data Table Analytics

One of Aridhia Workspace's standout features is **Data Table Analytics**: over 20 statistical and visualisation modules that let you analyse data without writing any code. Each module generates R code that you can view, copy, and build upon.

## Launching Data Table Analytics:

1. Navigate to the **Database** tab and select your *breast_cancer_data* table.
2. Click the **Analyse Data** button in the menu bar. *(You can access 'Analyse Data' from .csv file menu as well).*
3. The Data Table Analytics interface opens, showing your data with an analytics sidebar.

## Available Modules:

- **Statistical Modelling:** Linear Regression, Binomial Logistic Regression, ANOVA
- **Statistical Testing:** Student's T-test, Mann-Whitney U test, Pearson's Chi-squared test
- **Descriptive Statistics:** Single mean, Compare means, Correlation, Cross tabulation
- **Data Visualisation:** Histogram, Scatter plot, Box plot, Heatmap, and more

## Example: Running a Logistic Regression

For our breast cancer dataset, let's build a simple predictive model:

1. Select **Binomial Logistic Regression** from the Statistical Modelling section.
2. Set ***diagnosis_binary*** as your outcome variable.
3. Select predictor model variables such as *radius_mean*, *texture_mean*, and *perimeter_mean*.
4. Click **Run**. The results display coefficients, odds ratios, and model diagnostics.
5. Click **</> Show code** to see the underlying R code. You can copy this to build more sophisticated analyses.

# 8 Reproducible Analysis with Jupyter Lab

For more sophisticated analyses that combine code, visualisations, and narrative documentation, the built-in **Jupyter Lab App** provides an ideal environment. Jupyter notebooks are particularly valuable for creating reproducible research workflows that can be shared with collaborators or included as supplementary materials in publications.

## Launching Jupyter:

1. Navigate to the **Apps** tab (Tools > View Apps)
2. Select **Jupyter Lab** and click **Start**
3. Create a new notebook with a **Python 3** kernel

### Pasting code into Jupyter

Jupyter Notebook is a containerised app with its own clipboard for security. To paste code from outside the app, look for the small sidebar on the left edge of the Jupyter window. Click it to expand, then use the paste icon to transfer text from your system clipboard into the app. Once pasted via the sidebar, the text will appear where your cursor is positioned. Within Jupyter itself, standard Ctrl+V works normally.

## Example Notebook

The following example demonstrates a complete machine learning workflow.

Create cells in Jupyter by clicking the **+** button, then **use the dropdown** to switch between **Code** and **Markdown** cell types. The Markdown cells are typically used to provide context to the visuals.

## Markdown Cell 1

```
# Breast Cancer Classification Analysis

This notebook demonstrates a complete machine learning pipeline for predicting breast cancer
diagnosis using the Wisconsin dataset.
```

## Code Cell 1 - Setup and Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

# Configure plotting style
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette('husl')

# Load data from workspace files
df = pd.read_csv('/home/workspace/files/breast_cancer_processed.csv')
print(f'Dataset shape: {df.shape}')
df.head()
```

## Markdown Cell 2

```
## Exploratory Data Analysis

Before building our model, let's visualise the feature distributions and correlations to understand
which characteristics best distinguish malignant from benign tumours.
```

## Code Cell 2 - Feature Correlation Heatmap

```
# Select mean features for correlation analysis
mean_features = [c for c in df.columns if '_mean' in c]
correlation_matrix = df[mean_features].corr()

# Create correlation heatmap
plt.figure(figsize=(12, 10))
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
sns.heatmap(correlation_matrix, mask=mask, annot=True, fmt='.2f',
            cmap='RdBu_r', center=0, square=True, linewidths=0.5,
            cbar_kws={'shrink': 0.8, 'label': 'Correlation'})
plt.title('Feature Correlation Matrix (Mean Values)', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

## Code Cell 3 - Feature Distribution by Diagnosis

```
# Create distribution plots for key features
key_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
                'smoothness_mean', 'compactness_mean']

fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.ravel()

for idx, feature in enumerate(key_features):
    ax = axes[idx]
    for diagnosis, color, label in [('B', '#00C5B5', 'Benign'),
                                    ('M', '#0B2341', 'Malignant')]:
        subset = df[df['diagnosis'] == diagnosis][feature]
        ax.hist(subset, bins=25, alpha=0.6, color=color, label=label, density=True)
        ax.axvline(subset.mean(), color=color, linestyle='--', linewidth=2)
    ax.set_xlabel(feature.replace('_', ' ').title())
    ax.set_ylabel('Density')
    ax.legend()

fig.suptitle('Feature Distributions by Diagnosis', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

## Markdown Cell 3

```
## Model Training and Evaluation

We'll train a Random Forest classifier with cross-validation, then evaluate performance using
confusion matrix, ROC curve, and feature importance analysis.
```

## Code Cell 4 - Model Training

```
# Prepare features and target
feature_cols = [c for c in df.columns if '_mean' in c]
X = df[feature_cols]
y = df['diagnosis_binary']

# Split and scale
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Random Forest with cross-validation
clf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
cv_scores = cross_val_score(clf, X_train_scaled, y_train, cv=5)

print(f'Cross-validation scores: {cv_scores}')
print(f'Mean CV accuracy: {cv_scores.mean():.3f} (+/- {cv_scores.std()*2:.3f})')

# Fit final model and predict
clf.fit(X_train_scaled, y_train)
y_pred = clf.predict(X_test_scaled)
y_pred_proba = clf.predict_proba(X_test_scaled)[:, 1]
```

## Code Cell 5 - Confusion Matrix Visualisation

```
# Create confusion matrix heatmap
cm = confusion_matrix(y_test, y_pred)

fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax,
            xticklabels=['Benign', 'Malignant'],
            yticklabels=['Benign', 'Malignant'],
            annot_kws={'size': 16})
ax.set_xlabel('Predicted Diagnosis', fontsize=12)
ax.set_ylabel('Actual Diagnosis', fontsize=12)
ax.set_title('Confusion Matrix', fontsize=14, fontweight='bold')

# Add accuracy annotation
accuracy = (cm[0,0] + cm[1,1]) / cm.sum()
ax.text(0.5, -0.15, f'Accuracy: {accuracy:.1%}', transform=ax.transAxes,
        ha='center', fontsize=12)
plt.tight_layout()
plt.show()
```

## Code Cell 6 - ROC Curve Analysis

```
# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(fpr, tpr, color='#0B2341', lw=2,
        label=f'ROC curve (AUC = {roc_auc:.3f})')
ax.plot([0, 1], [0, 1], color='#00C5B5', lw=2, linestyle='--',
        label='Random classifier')
ax.fill_between(fpr, tpr, alpha=0.2, color='#0B2341')

ax.set_xlim([0.0, 1.0])
ax.set_ylim([0.0, 1.05])
ax.set_xlabel('False Positive Rate', fontsize=12)
ax.set_ylabel('True Positive Rate', fontsize=12)
ax.set_title('Receiver Operating Characteristic (ROC) Curve',
             fontsize=14, fontweight='bold')
ax.legend(loc='lower right')
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

## Code Cell 7 - Feature Importance Analysis

```
# Extract feature importances
importance_df = pd.DataFrame({
    'feature': feature_cols,
    'importance': clf.feature_importances_
}).sort_values('importance', ascending=True)

# Create horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 8))
colors = ['#00C5B5' if x < importance_df['importance'].median()
          else '#0B2341' for x in importance_df['importance']]
ax.barh(importance_df['feature'], importance_df['importance'], color=colors)

ax.set_xlabel('Feature Importance', fontsize=12)
ax.set_ylabel('Feature', fontsize=12)
ax.set_title('Random Forest Feature Importance',
             fontsize=14, fontweight='bold')

# Add value labels
for i, (v, feature) in enumerate(zip(importance_df['importance'],
                                     importance_df['feature'])):
    ax.text(v + 0.005, i, f'{v:.3f}', va='center', fontsize=9)

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

## Markdown Cell 4

```
## Summary

Our Random Forest classifier achieved strong performance:
- **Cross-validation accuracy**: ~96% (±2%)
- **AUC-ROC**: >0.99 indicating excellent discrimination
- **Key predictive features**: Concave points, perimeter, and area

The model demonstrates that cell nucleus morphology features are highly predictive of malignancy,
consistent with clinical understanding of breast cancer pathology.
```

**Run the notebook by clicking the ▶▶ button.**

### Pre-installed Packages

The Python kernel includes NumPy, pandas, sklearn, scipy, seaborn, matplotlib, and more. Run "*pip list*" to see the full list.

# 9 Working with R in the Console

For researchers who prefer R, the built-in R Console provides direct command-line access with pre-installed biostatistics packages.

## Launching the R Console:

1. Navigate to **Tools > New R Console**.
2. Select your preferred R version from the sidebar (Workspaces support R versions 4.3.2, 4.4.1 and 4.5.1).
3. The console opens and you can begin typing commands.

## Working with the Database Using XAP

XAP is Aridhia's R package for interfacing with the workspace. It includes functions for reading and writing to database tables and files:

```
# View available XAP functions
library(xaputils)
ls('package:xaputils')

# Read from the database table
df <- xap.read_table('breast_cancer_data')

# Run a quick summary
summary(df)

# Create a visualisation
library(ggplot2)
ggplot(df, aes(x=radius_mean, y=texture_mean, color=diagnosis)) +
  geom_point(alpha=0.7) +
  theme_minimal()
```

Visual outputs appear in the **Plots** tab on the right-hand panel. Click any plot to enlarge it.

## Saving a Plot to Blob Storage

**Blobs** provide storage optimised for large files such as images, genomic data, and video.

1. In the **Plots** tab, click on your visualisation to enlarge it.
2. Click the **Save** icon to save the plot directly to workspace Blobs.
3. Choose a filename (e.g. *diagnosis_scatter_plot.png*) and select your destination folder in Blobs. Click **Save**.
4. The plot is now stored in **Files > Blobs** and can be accessed from the Files tab, exported through the airlock, or referenced in reports.

***Why Blobs?*** While the Files folder is ideal for documents and scripts, Blob storage is optimised for large unstructured data. Saved plots, images, and large outputs should be stored in Blobs. We'll use this saved plot later when exploring the Activity tab.

# 9B Building an Interactive R Web App

Aridhia Workspaces support **R web apps** (built with Shiny), allowing you to create interactive dashboards and data exploration tools that run entirely within the secure environment. This is useful for sharing analysis with collaborators who may not be comfortable writing R code.

Let's create an interactive statistics dashboard for our breast cancer dataset that uses the **xaputils** package to connect to the workspace database.

## Creating the Web App from a Template

1. From the main menu, select **Tools > New R Web App**.
2. Give your app a name, such as *breast_cancer_explorer*.
3. Choose **Blank single file** from the template dropdown (this creates an empty app.R script).
4. Select your preferred **R image version** and click **Create R Web App**.
5. A new tile appears on the **R Web Apps** tab (Tools > View R Web Apps), and a new folder is created in **Files** named after your app.

## Customising the App Code

1. Navigate to **Files > breast_cancer_explorer** folder.
2. Click on **app.R** to open it in the editor.
3. Replace the contents with the following code:

### Top Tip

Since this is the built-in file editor rather than a containerised app, you can paste directly using Ctrl+V.

```
library(shiny)
library(ggplot2)
library(xaputils)
library(dplyr)

# Load data from workspace database using xaputils
df <- xaputils::xap.read_table('breast_cancer_data')

# Get numeric columns for selection
numeric_cols <- names(df)[sapply(df, is.numeric)]
numeric_cols <- numeric_cols[!numeric_cols %in% c('id', 'X')]

ui <- fluidPage(
  titlePanel('Breast Cancer Wisconsin - Interactive Explorer'),
  sidebarLayout(
    sidebarPanel(
      h4('Data Summary'),
      verbatimTextOutput('summary_stats'),
      hr(),
      selectInput('x_var', 'X-axis Variable:',
                  choices = numeric_cols, selected = 'radius_mean'),
      selectInput('y_var', 'Y-axis Variable:',
                  choices = numeric_cols, selected = 'texture_mean'),
      checkboxGroupInput('diagnosis_filter', 'Diagnosis:',
                         choices = c('Malignant' = 'M', 'Benign' = 'B'),
                         selected = c('M', 'B')),
      hr(),
      h4('Selected Feature Statistics'),
      tableOutput('feature_stats')
    ),
    mainPanel(
      tabsetPanel(
        tabPanel('Scatter Plot', plotOutput('scatter_plot', height = '500px')),
        tabPanel('Distribution', plotOutput('distribution_plot', height = '500px')),
```

```
      tabPanel('Box Plot', plotOutput('box_plot', height = '500px')),
      tabPanel('Correlation', plotOutput('correlation_plot', height = '500px'))
    )
  )
 )
)

server <- function(input, output, session) {
 # Reactive filtered data
 filtered_data <- reactive({
  df %>% filter(diagnosis %in% input$diagnosis_filter)
 })

 # Summary statistics
 output$summary_stats <- renderPrint({
  data <- filtered_data()
  cat('Total samples:', nrow(data), '\n')
  cat('Malignant:', sum(data$diagnosis == 'M'), '\n')
  cat('Benign:', sum(data$diagnosis == 'B'), '\n')
 })

 # Feature statistics table
 output$feature_stats <- renderTable({
  data <- filtered_data()
  data %>%
   group_by(diagnosis) %>%
   summarise(
    Mean = round(mean(get(input$x_var)), 2),
    SD = round(sd(get(input$x_var)), 2),
    Min = round(min(get(input$x_var)), 2),
    Max = round(max(get(input$x_var)), 2)
   )
 })

 # Scatter plot
 output$scatter_plot <- renderPlot({
  ggplot(filtered_data(), aes_string(x = input$x_var, y = input$y_var,
                                     color = 'diagnosis')) +
   geom_point(alpha = 0.7, size = 3) +
   scale_color_manual(values = c('B' = '#00C5B5', 'M' = '#0B2341'),
                      labels = c('B' = 'Benign', 'M' = 'Malignant')) +
   theme_minimal(base_size = 14) +
   labs(color = 'Diagnosis')
 })

 # Distribution plot
 output$distribution_plot <- renderPlot({
  ggplot(filtered_data(), aes_string(x = input$x_var, fill = 'diagnosis')) +
   geom_density(alpha = 0.6) +
   scale_fill_manual(values = c('B' = '#00C5B5', 'M' = '#0B2341'),
                     labels = c('B' = 'Benign', 'M' = 'Malignant')) +
   theme_minimal(base_size = 14) +
   labs(fill = 'Diagnosis')
 })

 # Box plot
 output$box_plot <- renderPlot({
  ggplot(filtered_data(), aes_string(x = 'diagnosis', y = input$x_var,
                                     fill = 'diagnosis')) +
   geom_boxplot(alpha = 0.7) +
   scale_fill_manual(values = c('B' = '#00C5B5', 'M' = '#0B2341'),
                     labels = c('B' = 'Benign', 'M' = 'Malignant')) +
   scale_x_discrete(labels = c('B' = 'Benign', 'M' = 'Malignant')) +
   theme_minimal(base_size = 14) +
   labs(fill = 'Diagnosis', x = 'Diagnosis')
 })

 # Correlation heatmap
 output$correlation_plot <- renderPlot({
  cor_vars <- c('radius_mean', 'texture_mean', 'perimeter_mean',
                'area_mean', 'smoothness_mean', 'compactness_mean')
  cor_matrix <- cor(filtered_data()[cor_vars])
  cor_df <- as.data.frame(as.table(cor_matrix))
  ggplot(cor_df, aes(Var1, Var2, fill = Freq)) +
   geom_tile() +
   geom_text(aes(label = round(Freq, 2)), size = 4) +
   scale_fill_gradient2(low = '#0B2341', mid = 'white', high = '#00C5B5',
                        midpoint = 0, limits = c(-1, 1)) +
   theme_minimal(base_size = 12) +
   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
   labs(x = '', y = '', fill = 'Correlation')
 })
}

shinyApp(ui = ui, server = server)
```

4. Save the file by pressing the **Save** button.

## Running the Web App

1. Navigate to the **R Web Apps** tab.
2. Find your *breast_cancer_explorer* tile and click **Refresh** to load your updated code.
3. Click **Run App** to launch the app in a new tab.
4. Use the dropdown menus to select different features for comparison, and toggle between diagnosis groups to explore the data interactively.
5. Navigate between tabs to view scatter plots, distributions, box plots, and correlation heatmaps.

## Key Features of this App:

- **XAP database integration:**
  Uses the xaputils package to read data directly from the workspace database with a single function call: *xap.read_table()*
- **Simplified connection:**
  No need for manual database credentials or environment variables; xaputils handles the connection automatically.
- **Interactive filtering:**
  Compare malignant vs benign samples dynamically.
- **Multiple visualisations:**
  Scatter plots, density distributions, box plots, and correlation heatmaps.
- **Secure by design:**
  All analysis runs within the TRE boundary. No data export functionality is included.

### Why this matters

R web apps are ideal for creating analysis tools that colleagues can use without needing R expertise. By using the xaputils package, the app connects to the workspace database with minimal code, eliminating the need to manage database credentials manually.

The *xap.read_table()* function provides a simple, secure interface to workspace data, ensuring that any updates to the underlying data are immediately reflected in the interactive visualisations.

For more information on xaputils functions, see the Aridhia Knowledge Base.

### Note

Some apps may require libraries that are not included by default by Aridhia. Simply use "*include.packages()*" in the R-console to add the packages you need. The default list of packages can be found in this Knowledge Base article.

# 10 Collaborating Through the Activity Tab

The **Activity** tab provides a summary of recent workspace events (e.g. file uploads, deletions, analysis runs) and serves as a central hub for team communication. You can add **Notes** and **Insights** to highlight important findings or start discussions about your research.

## Adding a Note to a Research Asset

Let's add a note to the scatter plot we saved to Blobs earlier in **Step 8**:

1. Navigate to the **Activity** tab in your workspace.
2. Click the **Add a note** button at the top right of the activity feed.
3. In the note editor, type your observation. For example: ***"The scatter plot shows clear separation between benign and malignant diagnoses based on radius and texture. This confirms our hypothesis that morphological features are predictive."***
4. Optionally, link the note to the file by selecting *diagnosis_scatter_plot.png* from your Blobs. This attaches the note to that specific asset.
5. Click **Save**. Your note appears in the workspace Activity feed, visible to all workspace members.

## Commenting on Notes

Other team members can add **Comments** to your note, creating a threaded discussion. This is useful for asynchronous collaboration, allowing colleagues in different time zones to contribute to ongoing research conversations.

## Promoting a Note to an Insight

When a note contains a particularly important conclusion or observation, you can promote it to an **Insight** for greater emphasis:

1. Find your note in the **Activity** feed.
2. Click the dropdown menu on the note and select **Promote to insight**.
3. The note is now marked as an **Insight** and will appear with a distinctive badge in the feed.

## Filtering the Activity Feed

Use the filter controls at the top of the Activity tab to show only **Insights** (for a summary of key findings) or **Airlock requests** awaiting approval. This helps you quickly find important information in a busy workspace.

### Why this matters

The Activity tab creates a research journal within your secure environment. Notes and Insights become part of the workspace's audit trail, documenting your analytical decisions and conclusions.

When collaborators join the project or when you return after time away, this history provides valuable context for understanding how the research progressed.

### Need to add collaborators?

If your research requires additional team members, contact Aridhia to add more users to your workspace. Additional users can view the Activity feed, add their own Notes and Comments, and participate in collaborative review workflows.

# 11 Creating Professional Reports with R Markdown

One of the most useful features for biomedical researchers is the ability to create reproducible research reports directly within the secure environment. The **R Development Environment** lets you use **R Markdown** to combine narrative text, R code, statistical outputs, and visualisations into professional documents. You can then review and share them using **Collabora Writer**.

## Launching the R Development Environment:

1. Navigate to the **Apps** tab (Tools > View Apps)
2. Select **R Development Environment (RDE)** and click **Start**
3. Once the RDE loads, go to **File > New File > R Markdown**
4. In the dialogue box, set the title to "*Breast Cancer Analysis Report*"
5. Put your name as the **author** and select **Word** as the output format

### Pasting code into the R Development Environment

Like Jupyter, the R Development Environment is a containerised app with its own secure clipboard. To paste code from outside the app, use the small sidebar on the left edge of the window. Click it to expand, then use the paste icon to transfer text from your system clipboard. The text will appear at your cursor position. Within the R Development Environment itself, standard Ctrl+V works normally.

## Example R Markdown Document

Replace the default content with the following template:

```
---
title: "Breast Cancer Wisconsin: Diagnostic Analysis Report"
author: "Research Team"
date: "`r Sys.Date()`"
output:
  word_document:
    toc: true
    toc_depth: 2
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)
library(xaputils)
library(ggplot2)
library(dplyr)
library(knitr)
```

# Executive Summary

This report presents an analysis of the Breast Cancer Wisconsin
(Diagnostic) dataset, examining the relationship between cell
nucleus features and cancer diagnosis.

# Data Overview

```{r load-data}
df <- xap.read_table('breast_cancer_data')
```

The dataset contains `r nrow(df)` samples with
`r sum(df$diagnosis == 'M')` malignant and
`r sum(df$diagnosis == 'B')` benign cases.

## Summary Statistics

```{r summary-table}
summary_df <- df %>%
  group_by(diagnosis) %>%
  summarise(
```
```

```
    N = n(),
    `Mean Radius` = round(mean(radius_mean), 2),
    `Mean Texture` = round(mean(texture_mean), 2),
    `Mean Area` = round(mean(area_mean), 1)
  )
kable(summary_df, caption = 'Summary by Diagnosis')
```

```
# Key Findings

## Feature Distributions

```{r radius-plot, fig.width=8, fig.height=4}
ggplot(df, aes(x = radius_mean, fill = diagnosis)) +
  geom_density(alpha = 0.6) +
  scale_fill_manual(values = c('B' = '#00C5B5', 'M' = '#0B2341'),
                    labels = c('Benign', 'Malignant')) +
  labs(title = 'Distribution of Mean Radius by Diagnosis',
       x = 'Mean Radius', y = 'Density', fill = 'Diagnosis') +
  theme_minimal()
```

Malignant tumours show significantly larger mean radius values
compared to benign tumours, with minimal overlap in distributions.

## Statistical Testing

```{r t-test}
t_result <- t.test(radius_mean ~ diagnosis, data = df)
```

A Welch's t-test confirms the difference is statistically
significant (t = `r round(t_result$statistic, 2)`,
p < 0.001).

# Conclusions

The analysis confirms that cell nucleus morphology features,
particularly radius and area measurements, are strong predictors
of malignancy in breast tissue samples.

---
*Report generated within Aridhia Trusted Research Environment*
```

## Rendering the Document

1. Save your R Markdown file as *analysis_report.Rmd* in */home/workspace/files/*.
2. Click the **Knit** button (or press Ctrl+Shift+K) to render the document.
3. The RDE will execute all R code and generate *analysis_report.docx* in the same directory.
4. The rendering process may take a minute as it executes the code and formats the output.

## Reviewing and Collaborating in Collabora Writer

1. Navigate to the **Files** tab in your workspace.
2. Locate *analysis_report.docx* and click to **Open in Collabora**.
3. The Word document opens in Collabora Writer, displaying your formatted report with the table of contents, summary statistics table, and embedded visualisations.
4. The document opens in **editable mode** and supports **real-time collaborative editing**. This means that multiple team members can work on the same document simultaneously, with changes appearing in real time.
5. Use the **navigation pane** to jump between sections, review the density plot visualisation, and verify that all statistical outputs rendered correctly.

## Adding Comments for Peer Review

Collabora Writer includes full commenting functionality, making it ideal for peer review within the secure workspace. To add a comment:

1. Select the text you want to comment on (for example, highlight the phrase “statistically significant” in the Statistical Testing section).
2. Choose **Insert > Comment** from the menu bar, or use the keyboard shortcut **Ctrl+Alt+C** (Cmd+Option+C on Mac).
3. A coloured comment box appears in the page margin, connected to your selected text by a line. The commented text range is highlighted.
4. Type your comment, for example: “***Should we include the confidence interval here for completeness?***”
5. The comment box displays your name (from your workspace user settings) and a timestamp.
6. Collaborators can reply to your comment using the **Reply** option in the comment’s dropdown menu, creating threaded discussions directly in the document.

## Resolving Comments

When feedback has been addressed, click the dropdown arrow in the comment box and mark it as **Resolved**. Resolved comments can be hidden from view while remaining part of the document's history.

### Need additional reviewers?

If your research requires peer review from additional team members, contact Aridhia to add more users to your secure workspace. Each user can access shared documents, add comments, and participate in the collaborative review process.

Once your review is complete, you can export the document as a PDF for final distribution:

1. In Collabora Writer, go to **File > Export As > PDF**.
2. A PDF export dialogue appears. Accept the defaults or adjust settings as needed, then click **Export**.
3. The PDF (analysis_report.pdf) is saved to your workspace Files.
4. Navigate to the **Files** tab and click on your PDF. It opens automatically in the **Collabora PDF Viewer**, no external software is required.

### PDF as the final output

The PDF format is ideal for sharing finalised reports as it preserves formatting across all systems and cannot be accidentally edited. We’ll export this PDF through the airlock in Step 12.

## The Reproducible Research Workflow

This workflow demonstrates a key principle of modern research:

- **Source document (.Rmd):** Contains both the analysis code and narrative text. Re-running this document will regenerate all results.
- **Output document (.docx):** A polished, shareable report suitable for collaborators, supervisors, or publication supplementary materials.
- **Review in Collabora:** Verify formatting and content without leaving the secure environment or needing Microsoft Office installed.

### Why this matters

Reviewers and collaborators often expect reports in Word format. This workflow lets you create professional documents programmatically while maintaining full reproducibility. If the underlying data changes, simply re-knit the document to update all tables, figures, and in-line statistics.

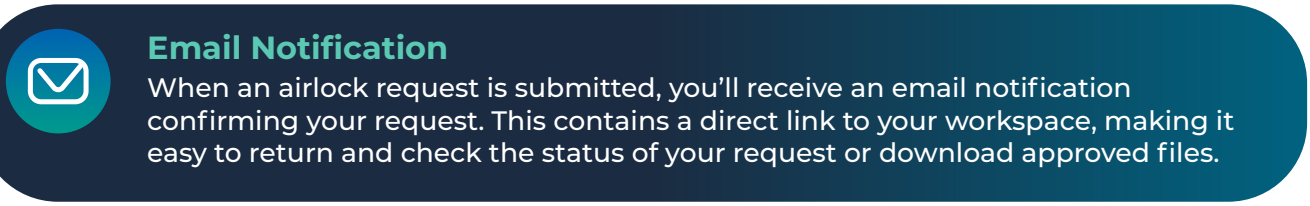
# 12 Exporting Results Through the Airlock

Just as data entering a TRE must pass through a secure inbound airlock, data leaving must go through a governed **outbound airlock** process. This ensures accountability and traceability for all data movements.

Let's export the PDF report we created in **Step 11** ("*analysis_report.pdf* ") so it can be shared with collaborators or submitted for publication.

## Submitting an Airlock Request

1. Navigate to the **Files** tab and locate *analysis_report.pdf*
2. Select the file by checking the box next to it.
3. Click the **Airlock** button in the toolbar.
4. Choose whether to Download to your local machine or Copy to another workspace.
5. Provide a **justification:** Enter a comment describing why the data needs to be exported, for example: "*Final analysis report for journal submission. Contains only aggregate statistics and visualisations, no individual patient data.*" (This is **mandatory** and forms part of the audit trail.)
6. Submit the request.




If you are a **Workspace Administrator**, you can approve airlock requests (including your own). Standard users will need to wait for an administrator to review their request.

The process is as follows:

1. In the **Activity** tab, Airlock requests and their status appear in the workspace activity stream.
2. To view only pending requests, click **Awaiting authorisation** in the top right of the page to filter the activity stream.
3. Click on the airlock request to review the details, including the files requested and the justification provided.
4. Choose to **Approve** or **Reject** the request. If rejecting, provide a reason.

## Downloading Approved Files

Once approved, you can download the files from the **Approved** section of the airlock. The **complete audit trail** (who requested, who approved, what was exported, and when) is preserved for compliance and governance purposes.

# 13 Reviewing the Workspace Audit

For governance, compliance, and reproducibility, Aridhia Workspaces maintain a complete audit trail of all activity. Workspace Administrators and Managers can access this via the **Audit** tab.

## Accessing the Audit Tab

1. From the **Administer** dropdown menu, select **Audit**.
2. The Audit table displays a record of all workspace events: user logins, file uploads and deletions, database table modifications, airlock requests, and more.
3. Use the **filter icon** in the table header to filter by user, action type, or date range. Multiple filters can be applied at once.
4. Click any column header to sort the table (except the "Who" column). The default sort is by date ("When").
5. To export the audit log, click the **Download** button in the top right. This exports the data as a CSV file for offline analysis or reporting.

### Note

The last 30 days of audit data are shown in the workspace interface, but all activity is stored in the background for the lifetime of the workspace.

### Why this matters

The audit trail provides evidence for regulatory compliance, supports reproducibility by documenting exactly what was done and when, and enables administrators to monitor workspace usage. For research involving sensitive data, this level of traceability is often required by institutional review boards and data governance committees.

## What We've Accomplished

Congratulations, that's the end of this walkthrough.

We've carried out a complete research workflow in an Aridhia Project Workspace:

- **Secure data upload** through the malware-scanning inbound airlock
- **Data inspection** using Collabora Calc without exporting files
- **Spreadsheet analysis** using AutoFilter, Pivot Tables, and formulas in Collabora Calc
- **Automated preprocessing** via Python background jobs
- **Command-line processing** using the Terminal app with bash shell scripts
- **Database integration** with automatic type detection
- **SQL querying** with the built-in SQL Editor for data exploration and aggregation
- **Point-and-click statistical analysis** with transparent R code generation
- **Reproducible notebook-based analysis** with Jupyter
- **R programming** with the built-in console and XAP package
- **Interactive R web app** directly connected to the database for dynamic data exploration
- **Professional report generation** using R Markdown in the RDE, with review in Collabora Writer
- **Governed data export** via the audited outbound airlock with Activity tab approval workflow
- **Complete audit trail** for compliance and reproducibility via the Audit tab