Protocol to identify functional doppelgängers and verify biomedical gene expression data using doppelgangerIdentifier

Functional doppelgangers (FDs) are independently derived sample pairs that confound machine learning model (ML) performance when assorted across training and validation sets. Here, we detail the use of doppelgangerIdentifier (DI), providing software installation, data preparation, doppelganger identification, and functional testing steps. We demonstrate examples with biomedical gene expression data. We also provide guidelines for the selection of user-defined function arguments.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.
Protocol

Protocol to identify functional doppelgängers and verify biomedical gene expression data using doppelgangerIdentifier

Li Rong Wang,1,5,6,* Xiuyi Fan,1 and Wilson Wen Bin Goh2,3,4,*

1School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
2Lee Kong Chian School of Medicine, Nanyang Technological University, Singapore, Singapore
3Centre for Biomedical Informatics, Nanyang Technological University, Singapore, Singapore
4School of Biological Sciences, Nanyang Technological University, Singapore, Singapore
5Technical contact
6Lead contact
*Correspondence: lwang039@e.ntu.edu.sg (L.R.W.), wilsongoh@ntu.edu.sg (W.W.B.G.)
https://doi.org/10.1016/j.xpro.2022.101783

SUMMARY

Functional doppelgängers (FDs) are independently derived sample pairs that confound machine learning model (ML) performance when assorted across training and validation sets. Here, we detail the use of doppelgangerIdentifier (DI), providing software installation, data preparation, doppelgänger identification, and functional testing steps. We demonstrate examples with biomedical gene expression data. We also provide guidelines for the selection of user-defined function arguments.

For complete details on the use and execution of this protocol, please refer to Wang et al. (2022).

BEFORE YOU BEGIN

Functional doppelgängers (FDs) are independently-derived samples that possess similar characteristics to each other. When FDs are split between training and validation data, validation accuracy will be overinflated regardless of how the machine learning (ML) model was trained. These inflationary effects caused by FDs are termed as doppelgänger effects (DEs).

When a data set has high numbers of FDs, it may confound model evaluation and other processes reliant on cross-validation for instance hyperparameter tuning and feature selection. To date, the most reliable method to mitigate these negative implications is to avoid assorting FDs across training and validation sets. To employ this mitigatory measure, the identification of FDs prior to model training is necessary.

Here, we present doppelgangerIdentifier (DI), an R package with 4 main functions for identifying FDs and verifying their inflationary effects on ML mode accuracy. Each function and its purpose are described in detail in Table 1.

To demonstrate how DI works, the protocol below uses a renal cell carcinoma proteomics data set as example. However, we have also applied this method to CCLE gene expression RNA-seq data sets (Ghandi et al., 2019), Duchenne muscular dystrophy and leukemia gene expression microarray data sets (Pescatori et al., 2007; Haslett et al., 2002; Armstrong et al., 2002; Golub et al., 1999; Yeoh et al., 2002; Ross et al., 2004). It is useful to note that the renal cell carcinoma data set has two technical
replicates referred to as batches (“Batch 1” and “Batch 2”) and two biomedically-relevant classes (“Normal” and “Tumor”). Table 2 describes the renal cell carcinoma data set in greater detail. Though we demonstrate this protocol on data sets with only two classes, DI also supports multi-class data sets.

Software prerequisites and data requirements

DI is supported on Windows, Linux, and Mac operating systems with R. Before installing DI, please check that the installed R version is 3.5.0 or above. Also, please ensure that the input data fulfills the following criteria: Firstly, the data should ideally be a microarray, RNA-Seq gene expression or proteomics data set. We have not thoroughly evaluated DI on other biomedical data modalities (such as metabonomics) although it should still work in principle. Secondly, since DI is heavily reliant on meta data for FD identification, check that all necessary meta data (sample label, class label, patient label and batch number) is available and accurate. Lastly, check that the data set has one or two technical replicates, or batches (data sets of more than two batches are not supported). To ensure that the data meets DI formatting requirements, please refer to the protocol below.

Table 2. Meta data for renal cell carcinoma microarray dataset

| Histological type | Patient ID | Tissue | Replicate 1 ID | Replicate 2 ID |
|-------------------|------------|--------|----------------|---------------|
| Clear Cell RCC    | 1          | Normal | 1              | 19            |
| Clear Cell RCC    | 1          | Tumor  | 2              | 20            |
| Clear Cell RCC    | 2          | Normal | 3              | 21            |
| Clear Cell RCC    | 2          | Tumor  | 4              | 22            |
| Clear Cell RCC    | 3          | Normal | 5              | 23            |
| Clear Cell RCC    | 3          | Tumor  | 6              | 24            |
| Clear Cell RCC    | 6          | Normal | 11             | 29            |
| Clear Cell RCC    | 6          | Tumor  | 12             | 30            |
| Clear Cell RCC    | 7          | Normal | 13             | 31            |
| Clear Cell RCC    | 7          | Tumor  | 14             | 32            |
| Clear Cell RCC    | 8          | Normal | 15             | 33            |
| Clear Cell RCC    | 8          | Tumor  | 16             | 34            |
| Chromophobe RCC   | 5          | Normal | 9              | 27            |
| Chromophobe RCC   | 5          | Tumor  | 10             | 28            |
| Papillary RCC     | 4          | Normal | 7              | 25            |
| Papillary RCC     | 4          | Tumor  | 8              | 26            |
| Papillary RCC     | 9          | Normal | 17             | 35            |
| Papillary RCC     | 9          | Tumor  | 18             | 36            |

The first column, “Histological Type” describes the histological types of each Renal cell carcinoma (RCC) sample. The second column, “Patient ID”, denotes the patient label for the sample. The third column, “Tissue”, is the class label for the sample. The last two columns, “Replicate 1 ID” and “Replicate 2 ID”, represent the label for each of the replicates.
KEY RESOURCES TABLE

| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
|---------------------|--------|------------|
| Deposited data      |        |            |
| Renal Cell Carcinoma Data Set | Guo et al. (2015) | PXD000672 on ProteomeXchange Consortium |
| 3 Preprocessed Microarray Data Sets | Belorkar and Wong (2016) | https://doi.org/10.5281/zenodo.7080539 |
| Software and algorithms |        |            |
| doppelgangerIdentifier R Package | Wang et al. (2022) | https://doi.org/10.5281/zenodo.7080539 |
| R Version (>= 3.5.0) | R Foundation for Statistical Computing | https://cran.r-project.org/ |
| devtools | Hadley et al. (2021a) | https://cran.r-project.org/web/packages/devtools/index.html |
| sva | Leek et al. (2022) | https://bioconductor.org/packages/sva/ |
| ggplot2 | Wickham (2016) | https://cran.r-project.org/web/packages/ggplot2/index.html |
| class | Venables and Ripley (2002) | https://cran.r-project.org/web/packages/class/index.html |
| utils | R Core Team (2020) | https://www.R-project.org/ |
| dplyr | Hadley et al. (2021b) | https://cran.r-project.org/web/packages/dplyr/index.html |
| NetProt R Package | Wilson and Limsoon (2017) | https://rpubs.com/gohwils/NetProt |
| Other |        |            |
| Laptop with Windows 10 operating system, 16.0 GB RAM, 4 Cores, 3.10 GHz Processor speed | Dell | Precision 5530 |

MATERIALS AND EQUIPMENT

Table 3 describes the computational resources used in this paper. With reference to the computational resources used in this study, a minimum of 16 GB of RAM, Windows 10 operating system (Linux and Mac operating systems are also supported) and R (>= 3.5.0) is recommended.

STEP-BY-STEP METHOD DETAILS

Installing doppelgangerIdentifier (DI)

⏲ Timing: 1 min

This section describes how to install DI on a computer that has R (>= 3.5.0) installed.

1. Install all dependencies (devtools, sva, ggplot2, class, utils, dplyr) of DI.

   **Note:** If you encounter installation errors while installing any of the dependencies of DI (devtools, sva, ggplot2, class, utils, dplyr) via R for Linux, try installing the packages with apt-get (problem 1).

2. Install DI with devtools.

   ```
   > devtools::install_github("lr98769/doppelgangerIdentifier")
   ```

   △ CRITICAL: For installation error relating to serialized objects that cannot be read in older versions of R, ensure that the R version installed is >= 3.5.0 (problem 2).

Table 3. Computational resources used in this paper

| Information       | Value |
|-------------------|-------|
| Operating System  | Windows 10 |
| RAM               | 16.0 GB |
| Cores             | 4     |
| Processor speed   | 3.10 GHz |
△ CRITICAL: If met with an error message suggesting that the package is corrupted after reinstallation, restart R and check if the package is installed with the optional step below (problem 3).

Optional: To check if DI is installed, run the following code.

```r
> library(doppelgangerIdentifier)
```

Prepare data

.vol Timing: 10 min

This step prepares the gene expression data and meta data in the correct format to be passed into the two computational functions as R data frames: getPPCDoppelgangers() and verifyDoppelgangers().

3. Format the gene expression data set to match the data frame depicted in Table 4.
   a. Load the gene expression data set into R as a data frame.

   **Note:** Ensure each row of the data frame represents gene expression levels of a gene or probe while each column of the data frame represents gene expression levels from a sample. If the data frame has sample gene expression levels as rows, transpose the data frame with the following line of code.

   ```r
   > raw_data <- as.data.frame(t(raw_data))
   ```

   b. Set the sample labels as the column names and the gene labels as the row names. Remove all other redundant rows or columns.

   ```r
   > rownames(raw_data) <- geneLabels
   > colnames(raw_data) <- sampleLabels
   ```

   **Note:** Ensure that sample and gene labels are unique.

4. Format the meta data to match the data frame depicted in Table 5.
   a. Load the meta data into R as a data frame.

| Table 4. Format for gene expression R data frame |
|-----------------------------------------------|
| Sample 1 | Sample 2 | ... | Sample s |
|----------|----------|-----|----------|
| Gene 1   | 0.10     | 1.34| ...      | 2.56      |
| Gene 2   | 0.11     | 0.28| ...      | 3.93      |
| ...      | ...      | ... | ...      | ...       |
| Gene g   | 1.30     | 3.29| ...      | 0.56      |

The row names of the data frame are set to the gene labels while the column names of the data frame are set to the sample labels. Ensure that the sample labels are consistent between the column names of the gene expression R data frame and the row names of the meta data R data frame (c.f. Table 5). In this example data set, there are g number of genes and s number of samples.
Note: The data frame should at least have the following 3 columns: “Class”, “Patient_ID” and “Batch”. The “Class” column describes the class label of each sample, the “Patient_ID” column denotes the patient identifier label for the sample, the “Batch” column describes the batch that the sample belongs to. The “Batch” column would be used for the batch correction steps in getPPCCDoppelgangers() and verifyDoppelgangers() functions.

b. Set the sample labels as the row names and “Class”, “Patient_ID” and “Batch” as the column names. Remove all other redundant rows or columns.

Note: Ensure that sample labels are unique and all sample labels in the gene expression data frame can also be found in this data frame.

⚠️ CRITICAL: Ensure that both data frames follow the format above strictly (all column names in the gene expression data frame are present in row names of meta data frame, vice versa) since the row and column names are used extensively for the splitting of the data set. Check that the information provided in the meta data is accurate as all meta data columns will be used in DI. The “Class” and “Patient_ID” columns would be used in getPPCCDoppelgangers() for data doppelgänger identification. The “Class” column would be used as the class label for model training in the verifyDoppelgangers() function. The “Batch” column will be used for batch correction in getPPCCDoppelgangers() and verifyDoppelgangers() functions.

Note: The DI R package also includes 4 ready-to-use data sets (rc, dmd, leuk, all; gene expression count matrix and meta data in the appropriate formats for all functions in DI). In this protocol we will be using the renal cell carcinoma data set. To load the two data frames, run the following code.

```r
>library("doppelgangerIdentifier")
>Import RC gene expression dataset
data(rc)
>Import metadata for RC gene expression dataset
data(rc_metadata)
```

### Data doppelgänger identification

⏰ Timing: ~2 s for 324 sample pairs (Depends on data set size)
This step identifies FD candidates. The identified sample pairs are defined as data doppelgängers (DDs). DDs are sample pairs (two samples) that are highly correlated with each other and also fulfill other meta data-based criteria. In this protocol, the correlation metric used is Pairwise Pearson’s correlation coefficient (PPCC; the Pearson’s correlation coefficient between two samples). The meta data-based criteria are explained in greater detail in Figure 1. A brief summary of the selection criteria for PPCC DDs would be: Sample pairs from the same class but different individuals with PPCCs greater than the maximum of all PPCCs of P2 (different class different individual).

This step identifies FD candidates. The identified sample pairs are defined as data doppelgängers (DDs). DDs are sample pairs (two samples) that are highly correlated with each other and also fulfill other meta data-based criteria. In this protocol, the correlation metric used is Pairwise Pearson’s correlation coefficient (PPCC; the Pearson’s correlation coefficient between two samples). The meta data-based criteria are explained in greater detail in Figure 1. A brief summary of the selection criteria for PPCC DDs would be: Sample pairs from the same class but different individuals with PPCCs greater than the maximum of all PPCCs from sample pairs of different class different individual. The steps for DD identification are summarized in Figure 2. The selection criterion was motivated by the goal to ensure equal variability in samples of the same class and samples between different classes. If real leakage pairs (same class same individual pairs) are also present in the data set, users can further reduce the number of DDs by removing DDs that are out of the PPCC range of real leakage pairs. The getPPCCDoppelgangers() function in DI carries out all the steps listed in Figure 2. The DDs identified by the getPPCCDoppelgangers() function will be tested for inflationary effects in the next major step.

5. Run data doppelganger identification with DI with the following line of R code.

```r
> doppel_results <- getPPCCDoppelgangers(
  >    raw_data,
  >    meta_data,
  >    do_batch_corr = TRUE,
  >    batch_corr_method = "ComBat",
  >    do_min_max = FALSE,
  >    correlation_function = cor
  >)
```

Here we list the purpose of each parameter:

a. **raw_data**: Gene expression R data frame where each column is a sample and each row is a gene. The column names are sample labels and the row names are gene labels. If the renal cell carcinoma data is used, pass in "rc" (c.f. Table 4).
b. meta_data: Meta data R data frame where each row is a sample and there are 3 columns: "Class", "Patient_ID", "Batch", indicating the class, patient and batch label of the sample respectively. The row names are sample labels and the column names are "Class", "Patient_ID", "Batch" (c.f. Table 5). If any of the 3 columns are absent, the user would encounter an error message (problem 5). Ensure that all column names in raw_data can be found in the row names of meta_data, else, an error message would be displayed (problem 6). If the renal cell carcinoma data is used, pass in "rc_metadata" (c.f. Table 4).

c. do_batch_corr: An optional boolean parameter. If set to FALSE, the batch correction step will be skipped. The default value for do_batch_corr is "TRUE".

d. batch_corr_method: An optional string parameter. Users can toggle between "ComBat" or "ComBat_seq" to change the batch correction method used. The "ComBat" setting should be toggled if the raw_data supplied is a microarray gene expression data. The "ComBat_seq" setting should be toggled if the raw_data supplied is an RNA-Seq gene expression data set. If an invalid batch correction method is specified (Not "ComBat" or "ComBat_seq"), the user will encounter an error message (problem 7). This parameter will be ignored if "do_batch_corr" is set to FALSE. The default value for batch_corr_method is "ComBat".

e. do_min_max: An optional Boolean parameter. If set to FALSE, min-max normalization before correlation calculation is skipped. We generally recommend setting do_min_max to "TRUE" if there exist variables in raw_data that have disproportionately large magnitudes compared to other variables. The default value for do_min_max is "FALSE".

f. correlation_function: An optional function parameter. Users can pass in a user-defined function that accepts 2 vector parameters: x, y and returns a single float value. The default value for correlation_function is "cor".

**Optional:** Though only DD identification with PPCC was demonstrated here, DI also allows users to experiment with other correlation metrics with the correlation_function parameter in getPPCCDoppelgangers()..

**Optional:** Though DI only provides "ComBat" and "ComBat_Seq" batch correction methods, the user can also utilize other batch correction methods by setting "do_batch_corr" to FALSE and passing in the batch corrected gene expression R data frame into the raw_data parameter.

**Note:** If the meta data R data frame passed into getPPCCDoppelgangers() contains only 1 batch, this function identifies DDs within the batch. The batch correction steps will hence be skipped by default. If the meta data R data frame passed into getPPCCDoppelgangers() contains 2 batches, this function will only identify DDs between the batches. Note that this function does not support data sets with more than 2 batches. If a data set of more than 2 batches are passed into the function, an error message will be shown (problem 8). getPPCCDoppelgangers() returns an R list. This list contains the following: 1) Gene expression matrix just before the calculation of sample pair correlations after the preprocessing steps, 2) Matrix of PPCC float values, 3)
R data frame of PPCC values with labeled sample pairs (Labeled “Not Doppelgänger” or “Doppelgänger”), 4) The PPCC cut_off float value. The third object in the list would be used in the subsequent major step for the generation of a CSV experiment plan.

6. Visualize the data doppelgänger identification results on a scatter plot with the following line of code.

```r
>visualisePPCCDoppelgangers(ppcc_doppelganger_results=doppel_results)
```

a. `ppcc_doppelganger_results`: List returned from `getPPCCDoppelgangers`. If the list returned from `getPPCCDoppelgangers` is incomplete (does not include `PPCC_df` or `PPCC_df` does not have all the necessary columns), the user will observe an error message (problem 9 and problem 10).

Optional: Graphs generated by `visualisePPCCDoppelgangers()` were made with the `ggplot2` R package, hence, users can customize the graphs from the function (E.g., Changing the graph’s title) using any `ggplot2` functions.

**Functional doppelgänger testing**

- Timing: ~3 s Training Size: 28, Testing Size: 8, 6 Training-Validation sets, 10 random features (Depends on training, test size, number of training and validation sets and random features)

In the DD identification step, DDs, potential FDs were identified in the data set. In this step, we will be verifying that the DDs found overinflates random model accuracies. To test for this inflationary effect, the user will construct an experiment plan specifying the sample labels in each training-validation set. Ideally, the training-validation sets should contain increasing numbers of identified DDs between the training and validation set for easier observation of DEs. Next, the user invokes the `verifyDoppelgangers()` function to train and validate KNN models with random feature sets (Example of a random feature set; 10 variables sampled randomly without replacement from all variables in the data set) on the training-validation sets in the experiment plan (Steps carried out by `verifyDoppelgangers()` are described in greater detail in Figure 3). Once all KNN models are validated, the results of FD testing can be viewed using the `visualiseVerificationResults()` function. During visualization, we compare the performances of random KNN models between the different training-validation sets and with the binomial negative control (also generated in the `verifyDoppelgangers()` method).

7. Test the DDs found in the previous major step with `verifyDoppelgangers()`.

```r
>veri_results <- verifyDoppelgangers(
>  experiment_plan_filename,
>  raw_data,
>  meta_data,
>  separator = "\\.*",
>  feature_set_portion = 0.1,
>  num_random_feature_sets = 10,
```
a. experiment_plan_filename: String file path to user-defined CSV experiment plan. Refer to Table 6 for the format of the CSV experiment plan. If the renal cell carcinoma data is used, download its experiment plan from https://github.com/lr98769/doppelgangerIdentifier/blob/main/tutorial/experiment_plans/rc_ex_plan.csv and pass in the string file path to this CSV file.
b. raw_data: Gene expression R data frame where each column is a sample and each row is a gene. The column names are sample labels and the row names are gene labels. If the renal cell carcinoma data is used, pass in "rc" (c.f. Table 4).
c. meta_data: Meta data R data frame where each row is a sample and there are 3 columns: "Class", "Patient_ID", "Batch", indicating the class, patient, and batch label of the sample respectively. The row names are sample labels and the column names are "Class", "Patient_ID", "Batch" (c.f. Table 5). If any of the 3 columns are absent, the user would encounter an error message (problem 5). If the renal cell carcinoma data is used, pass in "rc_metadata".
d. separator: An optional character parameter, the character passed will be used as the separator for the header names. For instance, the header name for the training data in the training-validation set "0_Doppel" would be "0_Doppel.train" with the separator "\". The default separator is ".
e. feature_set_portion: An optional float parameter between 0 and 1 representing the proportion of variables to be used for feature set generation. For instance, a proportion of 0.1 means 10% of all features are used during random feature set generation. If an invalid feature set proportion is chosen, an error message will be displayed (problem 14). The default feature_set_portion is 0.1.
f. num_random_feature_sets: An optional integer parameter for the number of random feature sets generated for each training-validation set. The default num_random_feature_sets is 10.
g. seed_num: An optional integer parameter denoting the seed number for random feature set generation. The default seed_num is 2021.
h. do_batch_corr: An optional boolean parameter. If set to FALSE, the batch correction step will be skipped. The default setting for do_batch_corr is TRUE.
i. batch_corr_method: An optional string parameter. Users can toggle between "ComBat" or "ComBat_seq" to change the batch correction method used. The "ComBat" setting should be toggled if the raw_data supplied is a microarray gene expression data. The "ComBat_seq"

```
> seed_num = 2021,
> do_batch_corr = TRUE,
> batch_corr_method = "ComBat",
> k = 5,
> size_of_val_set = 8,
> neg_con_seed = 10
>
```

Figure 3. Steps for functional doppelgänger (FD) verification

All the steps listed in this figure are carried out by the function verifyDoppelgangers() in doppelgangerIdentifier.
setting should be toggled if the raw_data supplied is an RNA-Seq gene expression data set. If an invalid batch correction method is specified (Not “ComBat” or “ComBat_seq”), the user will encounter an error message (problem 7). This parameter will be ignored if “do_batch_corr” is set to FALSE. The default batch_corr_method is “ComBat”.

j. k: An optional integer parameter representing the k hyperparameter for KNN classification models. The default k value is “5”.

k. size_of_val_set: An optional integer parameter for the size of each validation set (We assume the size of each validation set is the same. This value would be used as the n parameter of the binomial negative control). The default size_of_val_set is “8”.

l. neg_con_seed: An optional integer parameter for the seed number of the binomial negative control. The default negative control seed is “10”.

8. Visualize the FD testing results from the above step on a scatter plot with the visualiseVerificationResults() function:

```r
visualiseVerificationResults(
  verification_results = veri_results,
  original_train_valid_names = c(),
  new_train_valid_names = c()
)
```

a. verification_results: List returned from verifyDoppelgangers().

b. original_train_valid_names: Vector of strings containing the original names of training and validation pairs. This parameter is used with new_train_valid_names to change the names of the training-validation sets in the graph. The names in original_train_valid_names and new_train_valid_names should be aligned (same order). E.g., original_train_valid_names = c(“old_1”, “old_2”, “old_3”) and new_train_valid_names = c(“new_1”, “new_2”, “new_3”). The default value is an empty vector.

c. new_train_valid_names: Vector of strings containing the new names of training and validation pairs. This parameter is used with original_train_valid_names to change the names of the training-validation sets in the graph. The names in original_train_valid_names and new_train_valid_names should be aligned (same order). E.g., original_train_valid_names = c(“old_1”, “old_2”, “old_3”) and new_train_valid_names = c(“new_1”, “new_2”, “new_3”). The default value is an empty vector.
Optional: Graphs generated by `visualiseVerificationResults()` were made with the `ggplot2` R package, hence, users can customize the graphs from the function (E.g., Changing the graph’s title) using any `ggplot2` functions.

EXPECTED OUTCOMES

In step “data doppelganger identification”, we expect 2 outputs. Firstly, after running step 5, the `getPPCCDoppelgangers()` function returns a R list with 4 objects, 1) `Processed_data`, 2) `PPCC_matrix`, 3) `PPCC_df`, 4) `cut_off`. Each object is described in greater detail in Table 7. The `PPCC_df` labels sample pairs that are DD as “Doppelganger”. The user will refer to this data frame to generate a CSV experiment plan for step 7.

Secondly, after invoking `visualisePPCCDoppelgangers()` in step 6, a `ggplot2` correlation scatter plot similar to Figure 4 will be returned. Refer to Figure 4 for guidelines on how to interpret the scatter plot.

In major step “functional doppelganger testing”, we also expect 2 outputs. Firstly, after running step 7, the `verifyDoppelgangers()` function returns a R list with 5 objects, 1) `experimentPlanList`, 2) `combat_minmax`, 3) `feature_sets`, 4) `accuracy_mat`, 5) `accuracy_df`. Refer to Table 8 for detailed descriptions of each object.

Secondly, after running step 8, the `visualiseVerificationResults()` function returns a `ggplot2` accuracy scatter plot similar to Figure 5. Refer to Figure 5 for guidelines on how to interpret the scatter plot.

QUANTIFICATION AND STATISTICAL ANALYSIS

To aid in the analysis of functional doppelganger (FD) test results, the following statistical models and statistics were used:

1. A statistical model was used as a negative control in FD testing (labeled as “Binomial” in Figure 5). We model the number of correct predictions made by a random ML model not influenced by doppelganger effects (DEs) as a binomial model with \( p = 0.5 \) and \( n \) = number of validation samples. This statistical model was used under the assumption that a binary classification model trained with a random feature set would perform similarly to a series of coin tosses (Ho et al., 2020). The accuracy of each random model under the “Binomial” case is hence obtained from dividing the number of correct predictions (modeled by the binomial statistical model) by the number of validation samples (n).

2. For easier observation of a positive relationship between number of DD samples (validation samples that are DDs with at least 1 training sample) and random model accuracy, we calculate the
average validation accuracy of all ML models trained with random feature sets for each training-validation set and display it as a crossbar in Figure 5.

LIMITATIONS
As mentioned in earlier sections, this protocol does not support the identification of PPCC DDs between more than 2 batches. However, the user could potentially circumvent this limitation by applying this protocol to all possible pairwise combination of batches.

This protocol may not identify all FDs in certain data sets since non-PPCC DD FDs (sample pairs that are not identified as PPCC DDs but are FDs) could also be present. There are 2 plausible explanations for the existence of non-PPCC DD FDs. Firstly, the identification threshold used in step 5 could be inflated due to the presence of outlier sample pairs in the “Different Class Different Patient” case.

Table 8. List returned by verifyDoppelgangers()

| Key                | Value                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| experimentPlanList | R list of lists containing vectors of sample names in each training-validation set. The sample names in each set corresponds to the CSV experiment plan inputted by the user. |
| combat_minmax      | Gene expression float data frame with columns of genes and rows of samples (transpose of raw_data) just before model training.         |
| feature_sets       | R list of random feature sets and feature sets of highest and lowest variance. Each feature set is a vector of gene labels.               |
| accuracy_mat       | Float accuracy matrix. Each row shows model accuracies for a feature set. Each column shows model accuracies for a training-validation set. |
| accuracy_df        | Float accuracy data frame. Shows the same information as accuracy_mat. There are 3 columns in this data frame: “FeatureSet”, “Train_Valid”, “Accuracy”. Each row shows the accuracy of a model trained and evaluated on a specific training-validation set with a specific feature set. |

The column “Key” lists the keys used in the list. The column “Value” describes the objects attached to each key value in the “Key” column. There are 5 objects in the returned list.
This reduces the sensitivity of the protocol, thereby, resulting in lesser identified FDs. Secondly, FDs which are not linearly correlated may not be detected if PPCC is used as the correlation metric for DD identification. This is due to Pearson’s correlation coefficient’s inability to detect non-linear relationships. Users can experiment with other correlation coefficients capable of capturing non-linear relationships such as dCor (Clark, 2013).

DI verifies the inflationary effects of detected PPCC DDs through training and validating multiple random K-nearest neighbors’ classifiers. However, DEs can affect other ML models other than KNN. To make DI simple and easy to use, the verifyDoppelgangers() function does not provide a parameter to change the ML models used. However, a discerning user who is also interested to check for DEs on other models is free to modify the protocol.

Though DI provides functions to identify and verify FDs, it does not provide any interventive methods to mitigate DEs in the data set. Simply, lay users could attempt to avoid DEs by preventing the assortment of FDs across training-validation sets once they have identified the FDs with DI. More advanced users may wish to study the dataset more thoroughly to identify the sources or factors driving similarities between FD pairs. For example, if the high mutual correlations observed are only due to housekeeping genes which are irrelevant or of low interest to the study, they may be systematically removed without incurring information loss to the study. Alternatively, users may also use

Figure 5. Scatter plot generated by the visualiseVerificationResults() function to visualize the functional doppelgänger (FD) identification results for renal cell carcinoma data
X-axis: The names of each training-validation set. “i Doppel” where \(i = 0, 2, 4, 6, 8\) are training-validation sets with \(i\) number of data doppelgänger (DD) samples, validation samples that are DDs with at least one training sample.
“Perfect Leakage” refers to the training-validation set where all validation samples can be found in the training set.
“Binomial” refers to the statistical model used as a negative control (cf. quantification and statistical analysis). Y-axis: The validation accuracies of each model. Models represented with a gray dot are models with random feature sets. Models represented by pink or green dots are models trained with feature sets of highest and lowest variance, respectively. These models serve as the upper and lower bounds of model accuracies for each training-validation set (feature set of highest variance features are expected to be better feature sets therefore they are expected to have higher accuracies, vice versa). The violin plots show the distribution of random model accuracies (excludes the feature sets of highest and lowest variance) and the horizontal cross bars show the average of all random model accuracies for that training-validation set. For the renal cell carcinoma data, a positive relationship between the number of DD samples and random model accuracies is evident from the scatter plot (from observing the increasing trend in horizontal cross bars). This proves that the DDs that were tested in this experiment are capable of inflating random model performance and are hence FDs. If no clear positive relationship can be observed, check the following: 1) If random model performance is high even in the “0 Doppel” case, this suggests the existence of non-PPCC DD FDs in the data set. This could be due to the presence of outliers during DD identification (c.f. Figure 4). 2) If certain training-validation sets have higher random model accuracies than the “0 Doppel” case, the DDs added to that training-validation set are FDs. Note that the accuracies shown on this graph are not exact since the accuracies have been jittered to reduce the number of overlapping dots.
feature engineering, data normalization and transformation approaches to mitigate DEs. At this point, we are still studying which approaches are effective.

TROUBLESHOOTING

Problem 1
Installation is unsuccessful due to errors during the installation of dependency packages (step 2).

Potential solution
If the operating system used is Linux, install the dependency package with apt-get instead. For example, to install devtools run the following line in the command line:

```
> sudo apt-get install -y r-cran-devtools
```

Problem 2
Installation errors due to outdated R versions (step 2).

Potential solution
Update R to the newest version (Newest version of R can be installed from https://cran.r-project.org/).

Problem 3
Corrupted package from reinstallation (step 2).

Potential solution
Restart R (step 2).

Problem 4
User is shown an error message that suggests the absence of a package named “locfit” when using the getPPCCDoppelgangers() function. (step 5).

Potential solution
Install the “locfit” package and rerun the line of code.

Problem 5
User is shown this error message “Error: The following columns are not found in meta_data:” (steps 5 and 7).

Potential solution
Check that the meta_data R data frame contains all 3 columns with the exact column names (Same capitalization and spelling): “Class”, “Patient_ID”, “Batch”.

Problem 6
User is shown this error message “Error: Not all samples (colnames) in raw_data are found in (rownames of) meta_data” (step 5).

Potential solution
Ensure that all column names in raw_data can be found in the rownames of meta_data. Ensure that there are no changes in spelling or capitalization between the two R data frames.

Problem 7
User is shown this error message “Error: Invalid batch correction method is specified.” (steps 5 and 7).
Potential solution
DI only incorporates ComBat and ComBat-Seq batch correction methods. Hence, only "ComBat" or "ComBat_seq" are accepted for the batch_corr_method parameter. If either one of these methods were specified but this error is still observed, ensure there are no spelling or capitalization mistakes.

Problem 8
User is shown this error message "Error: There should only be 1 or 2 batches in the 'Batch' column of meta_data" (step 5).

Potential solution
If the data set only has 2 batches, check that there are only 2 types of batch labels in the meta_data R data frame. If the data set has more than 2 batches, it is not compatible with DI.

Problem 9
User is shown this error message "Error: PPCC_df not found in ppcc_doppelganger_results" (step 6).

Potential solution
The list returned from getPPCCDoppelgangers() is incomplete (does not include PPCC_df). Rerun getPPCCDoppelgangers() and observe for error messages and retry step 6.

Problem 10
User is shown this error message "Error: The following columns are not found in PPCC_df." (step 6).

Potential solution
The list returned from getPPCCDoppelgangers() is incomplete (PPCC_df does not include all 3 columns "ClassPatient", "PPCC", "DoppelgangerLabel"). Rerun getPPCCDoppelgangers() and observe for error messages and retry step 6.

Problem 11
User is shown this error message "_ is not a valid label e.g., 'train', 'valid'" (step 7).

Potential solution
Ensure that all header columns are of the correct format of training-validation set name and “train” or “valid” label separated by the separator (default separator is "."). For example, “Doppel_0.train” is a valid column name. Check that training-validation set names do not contain any spaces since they may be converted to "." after conversion. Do not include the separator (default separator is ".") in the training-validation set name. Make sure that the exact labels “train” or “valid” are used (No changes in spelling or capitalization). Avoid leaving empty columns between the data set columns. If excel is used for planning, save the spreadsheet as "CSV (MS-DOS) (*.csv)".

Problem 12
User is shown this error message "Not all sample names in the plan can be found in the columns of raw_data" (step 7).

Potential solution
Ensure that all sample names in the experiment plan can be found in the columns of the raw_data data frame.
Problem 13
User is shown this error message "Not all sample names in the plan can be found in the rows of meta_data" (step 7).

Potential solution
Ensure that all sample names in the experiment plan can be found in the rows of the meta_data data frame.

Problem 14
User is shown this error message "Error: Invalid feature_set_portion. Please choose a value between 0 and 1" (step 7).

Potential solution
The feature_set_portion passed into the function is out of the range of 0 and 1 e.g., -2. Ensure that the feature_set_portion value passed into the function is an integer value between 0 and 1 (excluding 0) e.g., 0.1.

RESOURCE AVAILABILITY

Lead contact
Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Wilson Wen Bin Goh (wilsongoh@ntu.edu.sg).

Materials availability
This study did not generate new unique reagents.

Data and code availability
The renal cell carcinoma dataset used in this study is publicly available at GitHub: https://doi.org/10.5281/zenodo.7080539.

The DI package and any code used in this study are also publicly available at GitHub: https://doi.org/10.5281/zenodo.7080539 as of the date of publication.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS
This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. W.W.B.G. also acknowledges support from a Ministry of Education (MOE), Singapore Tier 1 grant (grant no. RG35/20).

AUTHOR CONTRIBUTIONS
W.L.R. implemented analyses and wrote the manuscript. W.W.B.G. and F.X. supervised and co-wrote the manuscript.

DECLARATION OF INTERESTS
The authors declare no competing interests.
REFERENCES

Armstrong, S.A., Staunton, J.E., Silverman, L.B., Pieters, R., Den Boer, M.L., Minden, M.D., Sailan, S.E., Lander, E.S., Golub, T.R., and Korsmeyer, S.J. (2002). MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nat. Genet. 30, 41–47.

Belorkar, A., and Wong, L. (2016). GFS: fuzzy preprocessing for effective gene expression analysis. BMC Bioinf. 17, 540.

Clark, M. (2013). A Comparison of Correlation Measures, 4 (Center for Social Research, University of Notre Dame).

Ghandi, M., Huang, F.W., Jané-Valbuena, J., Kryukov, G.V., Lo, C.C., Mcdonald, E.R., Barretina, J., Gelfand, E.T., Bielski, C.M., Li, H., et al. (2019). Next-generation characterization of the cancer cell line encyclopedia. Nature 569, 503–508.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531–537.

Guo, T., Kouvonen, P., Koh, C.C., Gillet, L.C., Wolski, W.E., Rost, H.L., Rosenberger, G., Collins, B.C., Blum, L.C., Gillessen, S., et al. (2013). Rapid mass spectrometric conversion of tissue biopsy samples into permanent quantitative digital proteome maps. Nat. Med. 21, 407–413.

Hadley, W., Jim, H., Winston, C., and Jennifer, B. (2021a). Devtools: Tools to Make Developing R Packages Easier, 3 ed (CRAN). R package version 2.4.

Hadley, W., Romain, F., Lionel, H., and Kirill, M. (2021b). Dplyr: A Grammar of Data Manipulation, 3 ed (CRAN). R package version 1.0.

Haslett, J.N., Sanoudou, D., Kho, A.T., Bennett, R.R., Greenberg, S.A., Kohane, I.S., Beggs, A.H., and Kunkel, L.M. (2002). Gene expression comparison of biopsies from Duchenne muscular dystrophy (DMD) and normal skeletal muscle. Proc. Natl. Acad. Sci. USA 99, 15000–15005.

Ho, S.Y., Phua, K., Wong, L., and Bin Goh, W.W. (2020). Extensions of the external validation for checking learned model interpretability and generalizability. Patterns 1, 100129.

Leek, J.T., Johnson, W.E., Parker, H., Fertig, E., Jaffe, A., Zhang, Y., and Storey, J. (2022). Sva: Surrogate Variable Analysis (Biocductor). R package version 3.36.0.

Pescatori, M., Broccolini, A., Minetti, C., Bertini, E., Bruno, C., D’Amico, A., Bernardini, C., Mirabella, M., Silvestri, G., Giglio, V., et al. (2007). Gene expression profiling in the early phases of DMD: a constant molecular signature characterizes DMD muscle from early postnatal life throughout disease progression. FASEB. J. 21, 1210–1226.

R Core Team (2020). R: A Language and Environment for Statistical Computing (CRAN).

Ross, M.E., Mahfouz, R., Onciu, M., Liu, H.-C., Zhou, X., Song, G., Shurtleff, S.A., Pounds, S., Cheng, C., Ma, J., et al. (2004). Gene expression profiling of pediatric acute myelogenous leukemia. Blood 104, 3679–3687.

Venables, W.N., and Ripley, B.D. (2002). Modern Applied Statistics with S (Springer).

Wang, L.R., Choy, X.Y., and Goh, W.W.B. (2022). Doppelganger spotting in biomedical gene expression data. iScience 25, 104788.

Wilson, W.B.G., and Limsoon, W. (2017). NetProt: complex-based feature selection. J. Proteome Res. 16, 3102–3112.

Yech, E.-J., Ross, M.E., Shurtleff, S.A., Williams, W.K., Patel, D., Mahfouz, R., Behm, F.G., Raimondi, S.C., Relling, M.V., Patel, A., et al. (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1, 133–143.