Published in final edited form as:

Methods Enzymol. 2016 ; 574: 53–77. doi:10.1016/bs.mie.2016.02.002.

ArrayNinja: An Open Source Platform for Unified Planning and Analysis of Microarray Experiments

B.M. Dickson, E.M. Cornett, Z. Ramjan, and S.B. Rothbart¹

Center for Epigenetics, Van Andel Research Institute, Grand Rapids, MI, United States

Abstract

Microarray-based proteomic platforms have emerged as valuable tools for studying various aspects of protein function, particularly in the field of chromatin biochemistry. Microarray technology itself is largely unrestricted in regard to printable material and platform design, and efficient multidimensional optimization of assay parameters requires fluidity in the design and analysis of custom print layouts. This motivates the need for streamlined software infrastructure that facilitates the combined planning and analysis of custom microarray experiments. To this end, we have developed ArrayNinja as a portable, open source, and interactive application that unifies the planning and visualization of microarray experiments and provides maximum flexibility to end users. Array experiments can be planned, stored to a private database, and merged with the imaged results for a level of data interaction and centralization that is not currently attainable with available microarray informatics tools.

1. INTRODUCTION

Microarray technology is a pillar platform of functional proteomics, enabling massively parallel interrogation of macromolecular interactions in highly sensitive, reproducible, and miniaturized assay formats. Indeed, a variety of protein and peptide microarray platforms have been developed in the last decade for high-throughput analysis of protein–protein (Moore, Ajala, & Zhu, 2015), protein–peptide (Bock, Kudithipudi, et al., 2011; Bua et al., 2009; Fuchs, Krajewski, Baker, Miller, & Strahl, 2011; Garske et al., 2010; Kim et al., 2006; Nady, Min, Karetka, Chedin, & Arrowsmith, 2008; Rothbart, Krajewski, Strahl, & Fuchs, 2012), and protein–DNA (Hall et al., 2004; Hu et al., 2009) interactions. Additionally, peptide and protein microarrays have been useful for characterizing the selectivity of antibodies (Bock, Dhayalan, et al., 2011; Michaud et al., 2003; Rothbart et al., 2015; Rothbart, Lin, et al., 2012), and for profiling the substrate specificity of histone-modifying enzymes (Cornett et al., 2016; Dhayalan, Kudithipudi, Rathert, & Jeltsch, 2011; Kudithipudi, Kusevic, Weirich, & Jeltsch, 2014; Smith, Settles, Hallows, Craven, & Denu, 2011). The designs of these microarray platforms are as diverse as their applications, and taking advantage of the inherent flexibility of microarray technology requires an integrated workflow that couples the planning and analysis of microarray experiments.

¹Corresponding author: scott.rothbart@vai.org.

Microarrays are fabricated by precision robotic printing of biomaterial (eg, synthetic peptides) onto a solid surface (eg, functionalized microscope glass slides). Currently, the design and construction of custom arrays requires a rigorous understanding of the printing mechanics (ie, how the arrayer maps positions in a source plate onto the array surface). For example, if the experimentalist wants to print a specific feature in a defined coordinate on the slide, then they need to know exactly where to load that feature on the source plate so that the action of the printer results in the correct spot positioning. Additionally, at the quantification and analysis stage, the experimentalist has to repeat this process to integrate the feature identities with the postexperiment imaged slide. This latter process can be facilitated by a GAL (GenePix Associated List) file, a tab-delimited list file generated by most microarray printers that contains spot coordinates and nondescript identifiers. An issue with GAL files is that they have no unified syntax. Frequently, a GAL file will fail to load in specific densitometry software, leaving the experimentalist to implement their own workaround. Such workarounds do exist, but they are arduous and present a bottleneck to the experimentalist (Rothbart, Krajewski, et al., 2012). Moreover, for every new slide layout, complete reengineering of the analysis workflow is required. Ironically, this aspect of microarray analysis is arguably the single most time-consuming step, yet is rarely discussed with detail in publications. In addition, the critical step of converting spot identifiers into human interpretable tags is not typically addressed.

To circumvent these issues, we have developed a portable, open source, and interactive application, ArrayNinja, that unifies the planning and visualization of microarray experiments and provides maximum flexibility to the experimentalist. Specifically, this platform is aimed at eroding the barriers between array layout creation, practical implementation, and the ensuing quantification and analysis of microarray data. Herein, we detail the installation, structure, user interface, and methodologies of ArrayNinja.

2. ArrayNinja

The objective of ArrayNinja is to connect the planning and analysis stages of microarray experiments in a seamless and version-less application, thus providing maximal fluidity to the cycle of experimentation. ArrayNinja achieves this by programmatically recreating the action of the microarrayer in order to automate the mapping of biomaterial from microarray slide to source plate, or vice versa. This provides a feature that is unique to ArrayNinja in that the experimentalist can plan a custom print layout and populate the corresponding source plate by interacting with a modeled representation of the printed slide. Entering substrate identifiers for the spots on the modeled slide automatically populates a table that represents the physical layout of the source plate required for the print run.

Source plates are collected in a private database running inside ArrayNinja, which allows rapid modification of slide layouts and makes the process of merging substrate identifiers (and human interpretable tags) with imaged data completely automatic. Once a slide is scanned, the image can be loaded and paired with a source plate and slide layout to automatically populate the image with feature identifiers. The image can be viewed simultaneously with the identifiers for the highest level of data interaction and control.

The ArrayNinja platform is free to use and can be edited to create a specialized application for even the most exotic printer. The current version of ArrayNinja supports Aushon 2470 and GeneMachines OmniGrid 100 arrayers, both with a 4×4 pin configuration. Once downloaded, the computer running ArrayNinja does not need to connect to the Internet. The database is a local one, living inside ArrayNinja. This means that the experimentalist does not need to sit at a particular computer where the printer or scanner software is installed to plan experiments or to quantify data. Additionally, ArrayNinja is not tied to a particular computer operating system nor is it tied to a particular brand. Lastly, ArrayNinja does not use GAL files to map identities of coordinates. The action of the printer is recreated within ArrayNinja so that there are no versioning issues with regard to reading GAL files from different printers of different eras. This seamless, version-less, license-free unification of planning and quantification is what makes ArrayNinja unique.

There are three variables that determine the print layout of a microarray: (1) the arrayer architecture, (2) the arrayer settings, and (3) the composition of the source plate(s) that will be used during printing. Given knowledge of points (2) and (3), a microarray layout can be precisely predicted by simulating the arrayer architecture. ArrayNinja simulates the motion of the arrayer and uses a database to associate, store, and retrieve arrayer settings and source plates. ArrayNinja can be modified to simulate any arrayer and is completely independent of the printed substrate and slide surface chemistry. As such, ArrayNinja is free of any company- or era-specific versioning issues. This is also what allows ArrayNinja to reverse-engineer a source plate from a micro-array layout—which is entirely unique to ArrayNinja.

The ArrayNinja platform requires a running structured query language (SQL) database and web server, so distribution via a virtual machine (VM) has been selected to reduce complexity for the end user. Additionally, this allows the user to maintain a local and private database of experiment designs. We will discuss the use of the VM distribution later, but the tools can also be obtained as a Docker application via <https://hub.docker.com/> by pulling bradleydickson/arrayninja. To facilitate the practical use of Array-Ninja, we have included a working example within the VM and Docker. The details of this example are described in Section 4.

It is worth noting that since ArrayNinja is a web server, installing the application on one accessible laboratory computer allows all privileged users access and use of ArrayNinja simultaneously without needing to install locally. While we discuss the VM here, a laboratory-wide version of ArrayNinja can be easily set up by running the Docker container (bradleydickson/arrayninja).

2.1 Obtaining and Running ArrayNinja

As a prerequisite, Oracle VM VirtualBox software should be correctly installed on the machine where ArrayNinja will run. VirtualBox is free and can be obtained from www.virtualbox.org. Once VirtualBox is installed, the ArrayNinja VM can be downloaded from <http://research.vai.org/Tools/arrayninja>.

Starting ArrayNinja

1. After downloading and uncompressing the VM image, start VirtualBox.

2. The ArrayNinja machine can be added to VirtualBox by clicking Machine → Add and selecting arrayninja.vbox from the uncompressed arrayninja folder.
3. To start ArrayNinja, select arrayninja from the list of machines in the VirtualBox interface and click the green “start” arrow. The VM will output a message stating that the ArrayNinja tools can be accessed by navigating your web browser to localhost:2080. We suggest Google Chrome or Firefox. To stop the VM after using ArrayNinja, use Machine → Close → Power Off.
4. Open your web browser and enter localhost:2080 in the URL bar. If you are using a remote computer, you would enter the IP address of that remote computer followed by :2080 to specify the correct port.

The ArrayNinja index page will load after navigating to localhost:2080, presenting a few options for the toolset to deploy. This index page also runs scripts to make certain the SQL database is initialized. Users can plan array experiments or quantify an experiment with ArrayNinja. Later we discuss each of these use cases after first discussing some details of the underlying SQL database.

2.2 Overview of the ArrayNinja Database

The ArrayNinja database depends on three types of tables: “substrate tables,” “source plate tables,” and “printer settings tables.” We will first discuss how to browse through these tables and then introduce each table’s structure. We chose phpMyAdmin as an interface for maintenance and browsing of tables. phpMyAdmin has an intuitive browser interface.

Table Management Interface

1. Enter localhost:2080/phpMyAdmin in your browser’s URL bar. All of the ArrayNinja data are stored in the “arrayplates” database, listed in the left frame of the phpMyAdmin window (Fig. 1A).
2. Click on “arrayplates” to see the tables that belong to the database. Several tables are preloaded to provide working examples.

We will next outline each of the three distinct table structures and then look at how to load/create custom tables.

SubstrateTable: The SubstrateTable (Fig. 1B) is the hub of information and must be maintained directly by users. This table contains all of the features that might be printed and links each of them to a unique numerical identifier. Additions, removals, ID changes, etc., all need to be reflected in this table.

The SubstrateTable is a running list of all features that can be printed on the microarray. To view the SubstrateTable, click on SubstrateTable in the list of tables displayed by phpMyAdmin (Fig. 1B). Column by column, the format is as follows:

ID = a unique numerical identifier for the substrate

Context = feature context (eg, H3 1–20)

Sequence = the amino acid/base/or SMILES sequence for the feature

TEXTid=simple text identifier (eg, H3)

Annotation =a human interpretable shorthand for the feature (eg, H3K9me2)

The preloaded SubstrateTable is required for the packaged data analysis example. For custom use, edit this table to incorporate unique features. Users can build upon this table or replace its contents but cannot change the table name.

We generally manage this table by keeping an up-to-date spreadsheet of all printable features. Such a table can be exported from any spreadsheet software as a comma separated values (c.s.v.) table and imported directly to the database via phpMyAdmin, as discussed later.

Source plate table: Each array layout requires a unique “source plate table,” named to reflect an experiment or layout, that defines the 384-well plate(s) used during printing. ArrayNinja is configured to use up to five 384-well source plates for a single print. Each 384-well plate has 24 columns (A through X) and 16 rows. The source plate table reflects this with 24 columns and up to 80 rows. Each entry in this table is given by a feature ID and location in the table specifies location in the 384-well plate. In other words, the table position is taken literally for the source plate position. For new experiments, we populate source plate tables by interacting with the planning tool as described later. For experiments with existing source plates, c.s.v. tables for the source plates can be imported directly via phpMyAdmin as outlined later. A working example of a source plate table is loaded in the “abEX” table, where this name indicates use of the antibody example source plates. This table holds the source plate information that was used when printing the packaged example with an Aushon 2470 microarrayer. The table represents two 384-well plates, where the loaded feature identifier is recorded for each plate well. We discuss creation of this table later. These types of tables can either be loaded through the phpMyAdmin interface or created with the ArrayNinja planning tool.

Printer settings table: Each array layout requires a unique printer settings table, where the table name is defined by the source plate table name, as both are required to specify a layout. This table has 1 row and 11 columns that specify scanning resolution, spot diameter, spot spacing (edge-to-edge), number of “blocks” per source plate row (number of times a 4×4 pin cluster can fit on a plate row), number of plate rows printed/to be printed, replicates (counting from 0), top margin on the microscope slide, side margin on the microscope slide (right side for Aushon and left for OmniGrid), feature limit (long direction for Aushon, short direction for OmniGrid), number of subarrays on the slide, and the spacing between subarrays. The “abEXparameters” table holds the printer settings that were used when printing the packaged example with an Aushon 2470 microarrayer. These tables are created exclusively and automatically by the ArrayNinja planning tool, as discussed later.

Table import: Here, we detail a general procedure for importing a table, and we build the example around importing a new substrate table. This procedure works for any table and could be used to import an existing source plate. It is useful to create table copies, which preserve the table structure for importing new data, or serve as risk-free templates when practicing table manipulation.

Create an Empty Table Template

1. Click “arrayplates” in the list of databases (Fig. 1, left frame)
2. Click the checkbox next to SubstrateTable
3. Select “copy table with prefix” from the “with selected” dropdown box floating under the list of tables. This dropdown box is visible in Fig. 1A. A new page will load.
4. Enter SubstrateTable in the “From” field, and enter TestTable in the “To” field. An empty table has been created, and the table format is identical to the SubstrateTable.

In the following, we use the name “TestTable” as an example and go through the steps for importing new data.

Import Table Values: Example with TestTable

1. Select the arrayplates database.
2. Click “Empty” in the TestTable row, which is in the middle of the phpMyAdmin display. This action removes all entries in the TestTable.
3. Click TestTable to view the now empty table.
4. Click the “Import” tab near the top of the page.
5. Use the “choose file” button to find your substrate table in c.s.v. format, and click “Go.” The c.s.v. format can be exported from any spreadsheet software. The required format for a SubstrateTable was specified earlier.

The new data from the imported c.s.v. file will populate the table. This procedure can be used to maintain any of the data used by ArrayNinja. This next example will look at importing a previously used source plate.

Import an Established Source Plate

1. Copy the table named “empty” into a table with your experiment name.
2. Import your source plate c.s.v. file into the new table.
3. Use the planning tool in the next section to first load your source plate and to set up your matching arrayer settings.
4. Save your work using the name specified in step 1 of this section.

The MySQL language and phpMyAdmin also offer a number of alternatives to maintain the tables. phpMyAdmin is a very robust platform with its own documentation (www.phpmyadmin.net). Tables may be directly imported from c.s.v. files and subsequently moved or renamed by interacting with the graphical phpMyAdmin interface.

3. PLANNING CUSTOM MICROARRAYS WITH ArrayNinja

The ability to easily customize microarray print formats enables experimentalists to fully capture the power of this technology. The ArrayNinja planning tool provides a simple interface for array design and source plate population to streamline custom microarray builds, and seamlessly connects the microarray fabrication step to data analysis.

Starting the Planning Tool

1. Access the planning tool by entering localhost:2080 in the browser's URL and clicking the appropriate planning option (arrayer specific). For the packaged example, select Aushon.
2. The tool first asks for source plate data, which is a table to be retrieved from the SQL database that defines the source plate for the printing job. Type the name of the source plate you want to edit in the "Load plate" text box, or if a new source plate is to be designed type "empty" and press enter to load the selected source plate. You must press the enter key before proceeding.

Once a source plate table is loaded, the array slide is drawn according to the printer settings that are associated with the source plate, and a series of printed spots appears (Fig. 2). The spots are uniquely color-coded to indicate the pin responsible for printing that spot. A 4×4 pin configuration is hard-coded. Note that all distances are in micrometers.

Planning Tool Input Fields Example

1. Resolution: Sets the resolution at which the experiment will be scanned. Enter 25.
2. Spot diameter: This is specific to the pins loaded in the printhead. Enter 275.
3. Spot spacing: Distance between spots, edge-to-edge. Enter 100.
4. Plate blocks per plate row: A 4×4 pin configuration breaks a 384-well plate into a 6×4 set of unique "blocks." This parameter sets how many of the six possible columns the printer will extract. Enter 4.
5. Total plate rows: A 4×4 pin configuration breaks a 384-well plate into a set of 6×4 blocks, each plate having four "rows." This parameter sets the total number of rows to be extracted, thus controlling the required number of 384-well plates. Enter 8 (which means two 384-well plates).
6. Replicates: Number of spot replicates counting from zero. Counting from zero has been adopted from the Aushon print control software. ArrayNinja currently dictates that replicates will be placed horizontally across the slide. Ensure this is consistent with the actual printing. Enter 1.
7. Margin side: For Aushon, this sets the distance from the right edge of the slide. For OmniGrid, this sets the distance from the left of the slide. Enter 4400.
8. Margin top: Sets the distance from the top of the slide. Enter 3900.

9. Feature in y: For Aushon, this sets the number of unique spots that can be deposited down the slide in a column before starting a new column. For OmniGrid, this sets the number of unique spots in a row (excluding replicate counting). Enter 8.
10. Super arrays: Sets the number of repeated arrays. Fig. 2A (left) has 3 super arrays and 48 subarrays, while Fig. 2A (right) has 2 super arrays and 2 subarrays. This language, and counting from 1, has been adopted from the Aushon printer control software. Enter 3
11. SuperA fudge: Distance between the repeat arrays. Enter 0.

Having entered the above parameters, ArrayNinja produces the layout that we previously used to create 48-well microarrays to optimize histone meth-yltransferase and demethylase assays (Cornett et al., 2016). Many other possible print formats are permitted, limited only by the printer architecture and printhead format.

Population of a Source Plate

Once the desired layout is achieved, the underlying source plate must be populated. Ordinarily, this would require considerable mental gymnastics to deduce the mapping from slide to source plate for each spot. In ArrayNinja, substrate identities may be specified directly for each spot by interacting with the slide.

Finalize Slide Layout and Source Plate

1. Mouseover a spot. A text input field appears (Fig. 2B).
2. Without moving the mouse position, press backspace/delete to erase the existing entry.
3. Type the identifier for the substrate that should be printed on that spot.
4. Repeat for all unique spots that need to be defined.
5. Press the “Populate” button to prepare data structures for SQL.
6. Name the new experiment by typing in the text field, where prompted.
7. Press the “commit your plate” button.

The new source plate will be printed to the screen so that it may be quality-controlled or copied and pasted. We find it is helpful to print this source plate layout and use it at the bench to construct the actual source plate. Simultaneously, the source plate and print parameters have been logged in the SQL database. By navigating to phpMyAdmin, you can check that the new source plate has been created. Any unwanted tables may be deleted via phpMyAdmin, so there is no harm in practicing table operations. If a source plate was loaded and the planning tool is being used only to fix print settings, simply avoid changing spot identifiers in the preceding steps. The source plate and the new printer settings will be checked in to the SQL database, with the specified name, after pressing “Populate” and “Commit your plate.”

Notice that there is a link stating “here is image” (Fig. 2B). Clicking this link will produce a slide mock-up image. Save this as an svg for making images or a reference template. To do this, use the browser’s File → Save as dialog and enter a file name ending with the svg extension. The resulting file can be viewed in a web browser or image editing packages like Inkscape, Adobe Illustrator, etc.

4. DATA ANALYSIS FEATURES OF ArrayNinja

To facilitate the practical use of ArrayNinja, we packaged a working example in the VM (and Docker). The example experimental data (an image) can be viewed by typing localhost:2080/abEX.png in the browser’s URL. This slide contains three “Super Arrays” of 2304 spots each, and each spot contains histone peptides. Each of the three blocks was incubated with a different histone antibody. The upper super array was hybridized with an H3K9me1 antibody, the middle block with an H3K9me2 antibody, and the lower block with an H3K9me3 antibody. We will refer to this example throughout the following overview of the ArrayNinja analysis suite, where we will quantify the top panel of the slide.

4.1 Microarray Image Preparation

The first step of analysis is to prepare an image for quantification. We suggest using ImageMagick (<http://www.imagemagick.org/>) to prepare images for quantification with ArrayNinja. This approach, detailed later, allows Array-Ninja to be used for a wide range of image sources without rewriting any analysis aspects of the program.

ImageMagick is a free, open source utility that provides precise control over image preparation. In Section 5, we quantify images from a Typhoon Trio scanner, which outputs grayscale images that present the brightest spots as the darkest pixels. This image must be inverted before it can be quantified with ArrayNinja. In addition, scanners that output separate image files for each laser channel scanned must be merged. Below is a suggested workflow to invert and merge a 2-color scan with ImageMagick.

1. Install ImageMagick
2. Invert images (if needed) with the command: `convert INPUT.tif -negate OUTPUT.tif`

Use the following series of commands to merge two scans.

1. `convert -depth 16 SIGNAL_CHANNEL.TIF -clone 0 -channel GB -evaluate set 0 -delete 0 out.png 2 >error.file`
2. `convert -depth 16 CONTROL_CHANNEL.TIF -clone 0 -channel RB -evaluate set 0 -delete 0 outA.png 2 >error.file`
3. `convert -depth 16 CONTROL_CHANNEL.TIFF -clone 0 -channel RG -evaluate set 0 -delete 0 outB.png 2 >error.file`
4. `convert out.png outA.png outB.png -set colorspace RGB -combine merged.png`

These commands generate an image “merged.png” that is now compatible with ArrayNinja. This workflow is the most versatile, allowing for quantification of microarray images

acquired from any scanner that outputs a tif file. Even images acquired with X-ray film can be treated with ArrayNinja in this way. Simply invert the image and use the same image as SIGNAL_CHANNEL and CONTROL_CHANNEL in the above commands to generate the “merged.png.”

Commands (1), (2), and (3) above generate a red, green, and (arbitrary) blue channel, respectively, and command (4) merges the channels into an image named merged.png. This script assumes a 16-bit input and forces a 16-bit output by using “-depth 16”. Imagemagick will preserve the so-called quantum depth of an image by default, so these flags are not necessary. The contrast and brightness of the image can be adjusted within ArrayNinja. Contrast and brightness adjustments are for viewing only. The raw image data are quantified directly.

We do not generally find it easy to motivate use of 16-bit images over 8-bit images. Essentially, the experimental data would itself need to respect the change in precision from 8- to 16-bit for the change to make any impact on interpretation of results. Often in microarray experiments, uncertainty in mean spot intensity is far greater than the resolution of intensity in 16-bit data.

4.2 Quantification of Microarray Data

A unique feature of ArrayNinja is the ability to retrieve all the relevant information from the microarray planning stage (see Section 3) via the underlying SQL database, simplifying the process of array quantification. Below, we work step-by-step through a basic quantification example and discuss analysis details and some additional features of the analysis interface in Section 6. A working example is packaged with ArrayNinja, and the following instructions will reference that example. The example image must be saved from <http://localhost:2080/abEX.png> prior to working through the analysis example below.

Step-By-Step Quantification

1. Navigate the web browser to localhost:2080 and choose to quantify data. For the packaged example, choose the Aushon option.
2. Load plate: Enter the name that was used to store the source plate in the planning phase (Section 3) and press enter. The SQL data have been fetched and loaded. To work-up the packaged example, type abEX and press enter.
3. Use the “Choose File” button to select an image to load. Load abEX. png to work through the example.
4. Change the resolution to match your image. For the example, enter 20.
5. Adjust the margin side and margin top values until the circles are closely aligned to the spots. For the example, set the margin side value to 5350, and margin top to 6900.
6. Set the circle diameter slightly larger than the spot diameter. This sets the area that will be analyzed for each spot. For the example, enter 320.

7. If working with the packaged example, set Super Arrays to 1. Each of the three Super Arrays in this example was hybridized with a different antibody, so they should be analyzed individually. Default settings aggregate all active spots by ID. Otherwise, set Super Arrays according to your layout.
8. Click the “Toggle zoom” button to activate the zoom window. Mouseover a spot to show the spot in zoom. Toggle zoom applies a magnification to the spot and should only be activated after the resolution is set. Saturation in this zoom window does not imply saturation in the image.
9. Inspect the image for spots that have been compromised by handling of the slide or print artifacts. Printer drips, smudges, or debris can result in artifactual spots. ArrayNinja allows these spots to be manually excluded from analysis. To do this, click on the image anywhere (to be sure the cursor is not active in any input field) and mouse over a spot. Press the “a” key on the keyboard to inactivate a spot. Press “a” again to activate the spot. Any inactive spot will be white and will not be quantified. In the example data, two of the H2AS1pCit3K5ac spots would be considered artifactual. Use the “Find” function to locate these features by typing the peptide name (case sensitive) in the Find field and pressing enter. The peptide spots will be highlighted with a red circle. Inactivate the two artifactual spots by mousing over and pressing “a” (Fig. 3A).
10. Click the “Spot Seek” button to automatically fine-tune the circle positions. Each circle will move toward maximum overlap with the spot on the image. This is achieved by calculating the “mass-weighted” average x and y positions of the pixels within each circle, where the “mass” of a pixel is taken to be the pixel (r,g,b) values. The center of each circle is updated to coincide with this average x and y , so that the center of the circle is iteratively driven to the “intensity-center” of the spot. This requires the earlier step of adjusting margins. Click the “Spot Seek” button three times for the example image.
11. The last step before quantification is the specification of a reference strategy. To set a reference spot, mouse over any spot without a printed feature and press the “r” key. The spot will turn orange. Press “r” again to de-reference the spot. For the packaged example, set several (at least 3) references across the slide where the spots are bright green with no red color, and have a “null” or “none” identity. Use the Find field to find all of the “none” spots.
12. Click “populate.” Once the pixels are counted, mousing over a circle will display the cumulative pixel counts for each color (Fig. 3A).
13. Enter a name for the experiment in the text field, and click “submit.” An interactive bar graph will appear and a table of results will be shown (Fig. 3B and C). This data summary web page can be saved by right clicking on the page and choosing “view source.” The resulting text can be pasted in a file named with the html extension. The file can then be viewed in a web browser at any time. The data table (Fig. 3C), containing averages and standard errors in the mean (s.e.m.), can be pasted into a spreadsheet by copying and pasting “as html.” Additionally,

a table of all spot intensities and identifiers is listed with the spots organized by k-means clustering. The clustering results provide all individual spot signals and help identify outliers.

5. BENCHMARKING ArrayNinja AGAINST ImageQuant TL

In this section, we validate ArrayNinja results by benchmarking against ImageQuant TL (GE Lifesciences) quantification for an image that has mostly homogeneous background and an image that has inhomogeneous background. These peptide microarray slides were printed on an OmniGrid 100 printer and imaged with a Typhoon Trio scanner. The raw images were processed as described earlier for ArrayNinja. For ImageQuant TL, images were directly loaded into the array analysis suite using the ds file generated by the Typhoon scanner. The ImageQuant dataset from Typhoon scans were deposited on www.histoneantibodies.com (Rothbart et al., 2015). For the ArrayNinja quantification, we did not inactivate any spots, so there is no quality control being asserted at the user level. It can be seen by the error bars (or lack thereof) on some datapoints that the ImageQuant data have benefitted from user control. While the scans that produced this ImageQuant data are available, the user asserted controls are not. ArrayNinja makes recording these controls very simple. A screenshot of the user interface (as shown in Fig. 3A) before data submission records all details except how many times “spot seek” was used.

5.1 Homogeneous Noise

The ArrayNinja default mode requires the user to set references so that a nonlocal (ie, global) noise threshold can be computed. ImageQuant TL uses a local noise correction by default. In Section 5.3, we discuss why we chose a nonlocal noise threshold as the default for ArrayNinja.

Fig. 4A shows a comparison of ArrayNinja using a nonlocal noise correction to ImageQuant using a local noise correction. The local and nonlocal corrections should be identical when the noise is homogeneous over the image. Indeed, the results are strikingly similar, and all the same conclusions about the antibody behavior follow directly. In this case, with uniform background, the methods are nearly identical.

5.2 Inhomogeneous Noise

Fig. 4B shows quantification results for a noisy image where both ArrayNinja and ImageQuant use local corrections. The overall error estimates are larger than in Fig. 4A for both platforms, but both still lead to the same conclusions regarding the antibody behavior. We note that on the set of data points in Fig. 4B, where ArrayNinja estimates lower signals than ImageQuant, the ImageQuant results do not have error bars. In these data points, only one spot contributed to the ImageQuant results, reflecting some user-level data control (like spot inactivation) that is omitted from the ArrayNinja results.

As noted earlier, ArrayNinja outputs all individual nonzero spot intensities organized by the k-means clustering algorithm. This allows the experimentalist to check how spots of the same ID present signal. Inspecting the clustering results helps find replicate spots that present disparate signals, and going back to the image to find these spots can help identify

the experimental conditions that cause the disparate results. Typical causes of this are printing errors or debris on the slide. The clustering results from the data in Fig. 4B suggest that local noise correction over-penalizes spots in noisier regions of the image, where we noted that replicate spots in high noise regions were clustered in low signal groups, and replicates in low noise regions were clustered in high signal groups. This suggests that the local correction may over-penalize spots in high noise areas. Using a nonlocal correction led to much less disparate clustering results for the same set of spots (eg, spots highlighted in Fig. 4B). In the next subsection, we present some formal details to explain these observations.

5.3 Implicit Assumption of Local Noise Correction

We can think of two reasons to support use of a local background correction: (1) the experiment was planned without incorporating reference spots, or (2) the belief that local noise corrections can “fix” inhomogeneous noise. Here, we provide mathematical reasoning to show that the local correction cannot “fix” inhomogeneous noise, in any sense, unless a strict assumption is met. This assumption was not met by the experiment in Fig. 4B.

The observed signal $\hat{S}(x, y)$ at any point (x, y) on the detector face can be written as $\hat{S}(x, Y) = \int_X \int_Y [S(X, Y) + N(X, Y)] \text{PSF}(x - X, y - Y) dXdY$. The point spread function (PSF) of the detector is $\text{PSF}(x, y)$, the signal emitted by the target fluorophore is $S(x, y)$, and the signal due to contaminant is $N(x, y)$. With a local correction, a given spot intensity is found by averaging this observed signal over the (x, y) points on the interior of a spot and subtracting the average observed signal found by averaging over the (x, y) points in the local exterior of the spot. For the interior of the spot, we have the average $\langle \hat{S} \rangle_{\text{int}} = \langle S * \text{PSF} \rangle_{\text{int}} + \langle N * \text{PSF} \rangle_{\text{int}}$ and for the exterior region we have $\langle \hat{S} \rangle_{\text{ext}} = \langle N * \text{PSF} \rangle_{\text{ext}}$, where there is no fluorophore on the exterior and all signal is due to noise. We use $*$ to indicate convolution with the PSF. If a local correction will “fix” inhomogeneous noise, then $\langle \hat{S} \rangle_{\text{int}} - \langle \hat{S} \rangle_{\text{ext}} = \langle S * \text{PSF} \rangle_{\text{int}}$. In order for this to work out correctly, the distribution $N(x, y)$ must be the same on the exterior boundary of the spot and on the interior of the spot in the mean sense, or else $\langle N * \text{PSF} \rangle_{\text{int}}$ and $\langle N * \text{PSF} \rangle_{\text{ext}}$ are not equal. Indeed, this assumption on $N(x, y)$ is implicit to the use of the local correction, owing to the idea that the correction should be stronger in regions of higher local noise. This basic assumption is not valid for this experiment, as can clearly be seen in Fig. 4B, where there are many dark spots where antibody (fluorophore) did not bind but are surrounded by high levels of background noise—the contaminant did not penetrate the spots, so $N(x, y)$ cannot be assumed similar inside and outside of the spots. This results in an unjustified coordinate-dependent bias in spot intensity.

The behavior of local corrections should be characterized case by case to assess the validity of the underlying assumptions. The experiment in Fig. 4B does not conform to those assumptions.

6. METHODOLOGICAL DETAILS

6.1 Default (Whole Spot) Quantification

Once the spot centers are found, the image can be quantified. Spots are quantified as follows: First, a spot-bounding domain is found by moving away from the spot center by a distance of c_r (plus a pixel) and then moving up a distance of c_r (plus a pixel), where c_r is the radius of the circle ($c_r = 320/2 \mu\text{m}$ in the step-by-step example earlier). Second, every pixel inside the bounding domain is tested to find those pixels, no more than the circle radius c_r from the circle center. In other words, the interior of the circle is determined. Third, the color values of each interior pixel are summed into a red, blue, and green tally. Finally, the noise threshold value is subtracted from each spot value.

6.2 Nonlocal Noise Thresholding

The user must define reference spots, and each reference spot is quantified as described earlier. The noise threshold value is determined by averaging the cumulative tally from each reference spot over all user-defined references. The s.e.m. in this average is also computed. The noise threshold is defined by adding three times the s.e.m. to the mean intensity of the reference spots. The noise threshold is subtracted from all spots, and spots with negative signal are set to zero.

6.3 Local Noise Thresholding

ArrayNinja also supports local noise corrections. If the “Local noise” box is checked, all the pixels in the bounding domain, but outside the interior of the spot, are averaged and the s.e.m. is computed. The average plus three times the s.e.m. is taken as the noise threshold. The noise threshold is subtracted from all spots and spots with negative signal are set to zero. No references need to be set in this mode. When using this mode, keep in mind the implicit assumptions stated in Section 5.3. This is best for uniform or homogeneous noise.

6.4 Variegated Spot Morphology

ArrayNinja also includes a checkbox called “Morph.” This checkbox enables a quantification method that corrects for variation in morphology, as shown in Fig. 5. This mode can only be used with nonlocal noise thresholding. The user defines several reference spots, and they are quantified with uniform circle size as described. Each reference spot’s signal intensity is averaged over the number of pixels in that spot to define the average pixel intensity $\langle \pi \rangle$. Each reference spot’s average pixel intensity is then averaged over the set of references and the s.e.m. of this value is also computed. The noise threshold is set to $\langle \pi \rangle + 3$ s.e.m. The more references, the smaller the s.e.m. will be. Likewise, fewer reference spots will lead to a larger background cutoff. This is the same noise threshold as in the default nonlocal method, except the default nonlocal method does not average over spot pixels. All spots have the same number of pixels in the default nonlocal method, so it amounts to division by a constant.

When the nonreference spots are quantified in the variegated method, the interior of each spot is identified as earlier, but only those pixels with a signal intensity greater than $\langle \pi \rangle + 3$ s.e.m. will be included in this quantification. Each pixel with intensity less than this

threshold is discarded, and the average pixel intensity of each nonreference spot minus the threshold is returned as the quantized spot value. Thus, only spot pixels brighter than the noise threshold are contributing to the spot intensity, and spots are no longer assumed to be circular.

The variegated morphology approach is attractive, because it implies that artifacts in spot morphology should be accounted for in the quantification. In Fig. 5, we show the results of the variegated morphology approach and the default approach for the packaged example image. In this quantification, no spots were discarded, and 11 references were set. Exactly the same references were used in each case. The “Morph” toggle allows for quantification with both methods without changing any other parameters.

Fig. 5 shows that for bright, well-resolved spots, the two methods agree. However, for spots with little to no structure that contain comet tails or artifacts (eg, H2AS1pCit3K5ac in the example), the variegated morphology method has the potential to produce different signals. This is because the average pixel intensity was calculated using only the brightest pixels. In the default method, all pixels on the interior of the circle are counted, resulting in a much lower effective average intensity. Thus, the variegated morphology method is not a standalone replacement for user-level quality control. The user should remain diligent in examining images for artifactual spots.

With modifications to the ArrayNinja core, the variegated methodology also opens avenues for quantifying cell and tissue microarrays, or other platforms presenting inherently irregular features.

7. LIMITATIONS, ASSUMPTIONS, OTHER FEATURES, AND FUTURE DEVELOPMENT

7.1 Limitations and Assumptions

Our microarray printer has a 4×4 pin configuration, so this is built into the code. Our tools also mandate that the source plates be read by the printer in a left-to-right, top-to-bottom pattern. We routinely use 384-well plates for our source plates. Accordingly, ArrayNinja assumes 24 columns by 16 rows for each source plate. ArrayNinja is built to talk to an SQL database, which allows integration of the planning and analysis stages, and forces certain formats on the tables where the feature identifiers, source plate layouts, and printing parameters are stored. The source code for all tools is freely available, so customization is permissible.

Lastly, we note that the OmniGrid 100 version of ArrayNinja may not count replicates in the same way as the OmniGrid printer. We have successfully tested the tools against a 48×48 spot layout that is commercially available through Epicypher and Millipore but not against any other formats. Robust stress testing of ArrayNinja for microarrays other than the Aushon 2470 is recommended.

7.2 Other Features

The ArrayNinja quantification tool also includes a “Find” function and a set of inputs that activate spots that are specifically deposited by a given pin in a particular subarray. ArrayNinja also supports contrast and brightness adjustments to aid in spot finding. (Only raw image data are quantified.)

The “Find” function is straightforward. Enter a case-sensitive string and ArrayNinja will highlight all substrates that contain that string with a red circle. For the packaged example, try searching for K9me2. All peptides containing dimethylation at lysine 9 will be highlighted by a red boundary.

In the Aushon-specific quantization tool, the iPin, jPin, and subA fields allow pin isolation, which is useful when analyzing slides in which many subarrays were printed (eg, Fig. 2 in Cornett et al., 2016). Our printer uses a 4×4 pin configuration, which is assumed throughout ArrayNinja. Thus, the pins may be uniquely identified by specification of an (i,j) pair. Enter the *i*-th pin in the iPin field and the *j*-th pin in the jPin field and the requested subarray(s) in the subA field to activate only that pin. Lists of pins or subarrays can be selected by entering a space delimited series like “1 2 3 4” for both the iPin and jPin, to select the diagonal pins (1,1), (2,2), (3,3), (4,4).

7.3 Future Development

ArrayNinja can be extended to arbitrary printer architecture, and expansion to better cover existing commercial printers is a leading avenue of development. ArrayNinja could be extended to liquid handlers as well to allow the planning and use of custom source plates from master source plates that hold an entire substrate library. Where most other developments have come at the level of processing data, ArrayNinja represents development of the infrastructure that strings together all aspects of experiment planning and quantification. This unification and development of infrastructure is what makes ArrayNinja unique.

8. SUMMARY

ArrayNinja unifies the planning and analysis stages of microarray experimentation. Once the print layout is determined, and the source plate is populated, both the layout and plate are logged to a private database. After experimentation and slide imaging, the print parameters and source plate can be retrieved for automatic pairing of the scanned image with the print substrate identifiers. The quantification automatically aggregates spots with the same identifier and computes the average intensity along with s.e.m. (taking into account the number of spots for each identifier, which is not assumed constant). Spots that fall on regions of the slide that are damaged or confounded by obvious artifacts (see the included example) can be discarded easily by interacting with the imaged slide. The quantified data are displayed so that it can be inserted into the platform of choice (text files, spreadsheets, plotting softwares, etc.).

ArrayNinja is free to download (<http://research.vai.org/Tools/arrayninja>) and can also be obtained as a Docker app (bradleydickson/arrayninja). The tool is flexible, portable, and

platform independent, and we have validated its quantification against commercially available software.

Acknowledgments

We thank Dr Rochelle Tiedemann for piloting ArrayNinja and for providing useful comments on the chapter. This work was supported in part by grant CA181343 from the National Institutes of Health to S.B.R.

References

- Bock I, Dhayalan A, Kudithipudi S, Brandt O, Rathert P, Jeltsch A. Detailed specificity analysis of antibodies binding to modified histone tails with peptide arrays. *Epigenetics*. 2011; 6(2):256–263. [PubMed: 20962581]
- Bock I, Kudithipudi S, Tamas R, Kungulovski G, Dhayalan A, Jeltsch A. Application of Celluspot peptide arrays for the analysis of the binding specificity of epi-genetic reading domains to modified histone tails. *BMC Biochemistry*. 2011; 12:48. [PubMed: 21884582]
- Bua DJ, Kuo AJ, Cheung P, Liu CL, Migliori V, Espejo A, ... Gozani O. Epigenome microarray platform for proteome-wide dissection of chromatin-signaling networks. *PLoS ONE*. 2009; 4(8):e6789. [PubMed: 19956676]
- Cornett EM, Dickson BM, Vaughan RM, Krishnan S, Trievel RC, Strahl BD, Rothbart SB. Substrate specificity profiling of histone-modifying enzymes by peptide microarray. *Methods in Enzymology*. 2016; 574:31–52. [PubMed: 27423856]
- Dhayalan A, Kudithipudi S, Rathert P, Jeltsch A. Specificity analysis-based identification of new methylation targets of the SET7/9 protein lysine methyltransferase. *Chemistry & Biology*. 2011; 18(1):111–120. [PubMed: 21276944]
- Fuchs SM, Krajewski K, Baker RW, Miller VL, Strahl BD. Influence of combinatorial histone modifications on antibody and effector protein recognition. *Current Biology*. 2011; 21(1):53–58. [PubMed: 21167713]
- Garske AL, Oliver SS, Wagner EK, Musselman CA, LeRoy G, Garcia BA, ... Denu JM. Combinatorial profiling of chromatin binding modules reveals multisite discrimination. *Nature Chemical Biology*. 2010; 6(4):283–290. [PubMed: 20190764]
- Hall DA, Zhu H, Zhu X, Royce T, Gerstein M, Snyder M. Regulation of gene expression by a metabolic enzyme. *Science*. 2004; 306(5695):482–484. [PubMed: 15486299]
- Hu S, Xie Z, Onishi A, Yu X, Jiang L, Lin J, ... Zhu H. Profiling the human protein-DNA interactome reveals ERK2 as a transcriptional repressor of interferon signaling. *Cell*. 2009; 139(3):610–622. [PubMed: 19879846]
- Kim J, Daniel J, Espejo A, Lake A, Krishna M, Xia L, ... Bedford MT. Tudor, MBT and chromo domains gauge the degree of lysine methylation. *EMBO Reports*. 2006; 7(4):397–403. [PubMed: 16415788]
- Kudithipudi S, Kusevic D, Weirich S, Jeltsch A. Specificity analysis of protein lysine methyltransferases using SPOT peptide arrays. *Journal of Visualized Experiments*. 2014; 93:e52203.
- Michaud GA, Salcius M, Zhou F, Bangham R, Bonin J, Guo H, ... Schweitzer BI. Analyzing antibody specificity with whole proteome micro-arrays. *Nature Biotechnology*. 2003; 21(12):1509–1512.
- Moore CD, Ajala OZ, Zhu H. Applications in high-content functional protein microarrays. *Current Opinion in Chemical Biology*. 2015; 30:21–27. [PubMed: 26599287]
- Nady N, Min J, Kareta MS, Chedin F, Arrowsmith CH. A SPOT on the chromatin landscape? Histone peptide arrays as a tool for epigenetic research. *Trends in Biochemical Sciences*. 2008; 33(7):305–313. [PubMed: 18538573]
- Rothbart SB, Dickson BM, Raab JR, Grzybowski AT, Krajewski K, Guo AH, ... Strahl BD. An interactive database for the assessment of histone antibody specificity. *Molecular Cell*. 2015; 59(3):502–511. [PubMed: 26212453]
- Rothbart SB, Krajewski K, Strahl BD, Fuchs SM. Peptide microarrays to interrogate the “histone code”. *Methods in Enzymology*. 2012; 512:107–135. [PubMed: 22910205]

- Rothbart SB, Lin S, Britton LM, Krajewski K, Keogh MC, Garcia BA, Strahl BD. Poly-acetylated chromatin signatures are preferred epitopes for site-specific histone H4 acetyl antibodies. *Scientific Reports*. 2012; 2:489. [PubMed: 22761995]
- Smith BC, Settles B, Hallows WC, Craven MW, Denu JM. SIRT3 substrate specificity determined by peptide arrays and machine learning. *ACS Chemical Biology*. 2011; 6(2):146–157. [PubMed: 20945913]

Fig. 1. Screenshots of the ArrayNinja database manager. (A) The phpMyAdmin interface and tables preloaded in the “arrayplates” database. (B) The preloaded “SubstrateTable.”

Fig. 2. Screenshots of the ArrayNinja planning tool. (A) A microarray layout consisting of 48 subarrays and three super arrays (*left*), and a microarray layout of two super arrays (*right*). (B) The ArrayNinja planning interface for designing array layouts, populating source plate tables, and naming experiments.

Fig. 3. Data analysis with ArrayNinja. (A) Screenshot of the ArrayNinja quantification interface. Interactive bar graphs (B) and tables (C) displaying quantified array results are automatically generated after clicking the “submit your data” button.

Fig. 4.

Benchmarking ArrayNinja against ImageQuant TL. (A) Scatter plot comparing quantification methods of arrays with homogeneous noise between ImageQuant TL using local noise thresholding and ArrayNinja using nonlocal noise thresholding. Arrays were hybridized with an H3K4me3 antibody (Epicyphe #13-0004). Correlation coefficient (R^2) was calculated by linear regression using gnuplot. (B) Scatter plot comparing quantification methods of arrays with inhomogeneous noise between ImageQuant TL using local noise thresholding and ArrayNinja using local noise thresholding. Arrays were hybridized with an H3K27ac antibody (Diagenode #C15410196). *Red arrow* points to a quantized data point from average intensity of spots marked by *white boxes*. R^2 was calculated as above.

Fig. 5.

Comparison of whole-spot and variegated quantification methods with ArrayNinja. Scatter plot comparison of whole-spot and variegated morphology quantification methods. Inset images illustrate how spot morphology impacts correlation between the two methods.