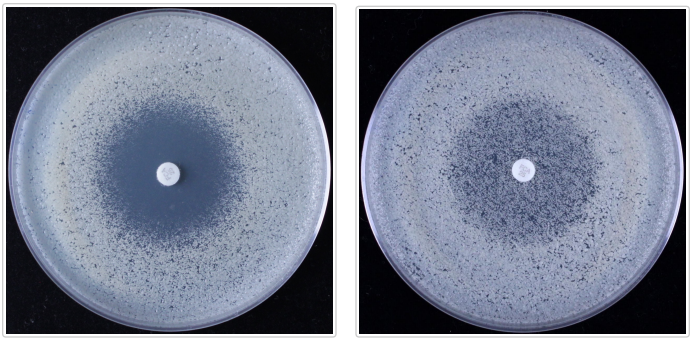
diskImageR: Quantification of resistance and tolerance to antimicrobial drugs using disk diffusion assays

Supplementary File S1. *diskImageR* tutorial

Introduction to *diskImageR*

The R package `diskImageR` provides a quantitative, unbiased method to analyze photographs of disk diffusion assays for any microbial species/microbial drug combination. This computational method measures the radius of inhibition ("RAD", i.e., resistance) at three different cutoff values (80%, 50% and 20% growth inhibition) as well as two measures of tolerance, the fraction of growth achived above RAD ("FoG"), and drug sensitivity ("slope", the rate of change from no growth to full growth).



diskImageR function overview (in the typical order of use)

- `IJMacro`: runs an ImageJ macro on the folder that contains the photograph to be analyzed [required]
- `readInExistingIJ`: used to read in existing ImageJ analyses [optional]
- `plotRaw`: used to plot the results of ImageJ analysis [optional]
- `maxLik`: maxmimum likelihood inference to fit logistic models to the data [required]
- `saveMLParam`: save the output of maximum likelihood analysis [optional]
- `createDataframe`: dataframe creation of all parameter estimates [required]
- `addType`: add a factor column to parameter estimate dataframe [optional]
- `aggregateData`: averages data from photographs of the same strain & type [optional]
- `calcMIC`: calculate the MIC from RAD values [optional]
- `readExistingDF`: read in an existing dataframe using a pop-up box [optional]
- `oneParamPlot`: plot a single resistance or tolerance parameter [optional]
- `twoParamplot`: plot resistance (RAD) and tolerance (FoG) at a specified cutoff value [optional]
- `threeParamplot`: plot resistance (RAD), tolerance (FoG), and sensitivity (slope) [optional]

See the end of this document for the skeleton walkthrough of package use, or the document "walkthrough.R" is available in the diskImageR library directory and as a supplementary file.

Required software

If you do not have R (or R Studio) installed on your computer, that is step one. The use of `diskImageR` does not require prior knowledge of R and it is the goal of this tutorial to walk through everything that is necessary to analyze photographs of disk diffusion assays. Once the package has been loaded into R, support for all built-in function can be found by typing `?functionName` in the R console.

The first step of `diskImageR` analyzes the disk diffusion photographs in ImageJ, a free, public domain image processing program available for download (http://rsb.info.nih.gov/ij/download.html). If possible ImageJ should be installed to the default location (Applications folder on a Mac, Program Files folder on a PC). If it is installed in a different location you will need to specify the path to ImageJ (see `?IJMacro` after package install). On a Mac, if you do not already have Xcode you will need to download it from the Apple Developer tools (https://developer.apple.com/xcode/download/). You may be prompted to download and install other additional programs in the R console if any are required (depends on what is already on your computer).

The primary output of `diskImageR` is CSV files that can be opened and used in any approprate program (e.g., Microsoft Excel) and a series of PDF figures that can be customized within the package functions.

Prepare plates and photographs

The analysis done by `diskImageR` will only be as good as the photographs taken of the disk assay plates. We use the Bencher Copymate II camera mounting system. In this setup there are two fluorescent lights on either side of the disk, oriented to minimize shadows on the plate in an otherwise dark room. We use the Canon Rebel T3i camera with an ISO 800, white balance "white fluorescent light", time 1/100s, picture stype "neutral", centre focused. Any camera of reasonably high quality should suffice, though the camera should always be set in manual rather than automatic mode, as the goal is to take photographs as consistently as possible. We also use the 2s timer to avoid potentially jostling the camera while taking images and/or having a hand/arm shadow in the picture. Plates should be photographed on a dark surface (we use black velvet) and plate labels are written on the side rather than the bottom of the plate. We have also used `diskImageR` with the XXX scanner on XXX settings.

Prior to analysis, images should be cropped close to the plate (as above). Because the analysis program

automatically detects the disk based on size, it is important that no other similar-sized circles be present in the image (e.g., from letters in labels).

Once you have the set of photographs that you want to be analyzed together they should be placed in the same directory, with nothing else inside that directory. **Important!** If there are any other files within the directory the script will not run properly.

The photograph file naming scheme will be be carried throughout, thus care should be taken with naming photographs in a logical manner. The general format that we use is "strain_factor1_factor2_rep.jpg". This format will allow you to use a built-in function to average across replicate pictures from the same strain. Conversely, if you intend to do this separately (or not at all) the photographs can be named anything.

Run the ImageJ macro on the set of photographs

The first function of the package is `IJMacro()`. From each photograph, an ImageJ macro that is included in `diskImageR` will automatically open each photograph from the specified directory, determine where the disk is located on the plate, find the center of the disk, and draw 40mm radial lines out from the center of the disk every 5 degrees. For each line, the pixel intensity will be determined at many points along the line using the built-in `plot profile` macro from ImageJ. This data will be stored in the folder *ImageJ_out* on your computer, with one file for each photograph.

`IJMacro()` can be run in two different ways, either through a user-interface with pop-up boxes, or directly through the R console. At this point you will specify a project name, the main project directory, and the photograph directory. The project name should ideally be fairly short (easy to type without typos!) and specific to the project. It must start with a letter, not a number or special character, but can otherwise be anything. The project name must always be specified with quotation marks around it (a surprisingly common error). Otherwise there will be a red error message (like Error in IJMacro(newProject) : object 'newProject' not found). The main project directory is the place where all files generated by the package will be saved within three directories: *ImageJ_out*, *parameter_files* and *figures*. A sub-directory will be created within each of these directories with the project name for organizational purposes, so that multiple different experiments/sets of analyses can be easily conducted from the same main project directory. The photograph directory is the one used to store photographs from above (which has nothing except the photographs to be analyzed in it).

Important! There can not be any spaces or special characters in any of the folder names that are in the path that lead to either the main project directory or the photograph directory. If there are an error box titled "Macro Error" will pop up and the script will not run (the red error message Error in tList[[i]]: subscript out of bounds will also show up in the R console).

The default assumption here and in all funtions is that the disk size is the standard 6mm. If you are using custom-sized disks you will need to specify that with the argument `diskDiam = X`, where X is the size of your disk in mm. This should also be specified in the functions `plotRaw()`, `maxLik()` and `createDataframe()`, discussed below. You will also need to change the argument `standardLoc` in `maxLik()` and `createDataframe()`. `standardLoc` is a numberic value that indicates the location (on the disk) to use to standardize intensity across photographs = TRUE). To suppress this standardization use`standardLoc = FALSE`.

To run the ImageJ macro through a user-interface with pop-up boxes:

```
IJMacro("newProject")
```

If you would prefer to avoid pop-up boxes you can directly specify the main project and photograph directory locations:

```
IJMacro("newProject", projectDir= "/path/to/projectDir", photoDir = "/path/to/projectDir/photographs/")
```

```
#>
#> Output of imageJ analyses saved in directory:
#> /private/var/folders/v4/jvjcsmnn23nbjfjjptmpbqkr0000gn/T/RtmpxJOH0X/Rbuild27834a539e4d/diskImageR/vignettes/imageJ_out/newProject/
#>
#> Elements in list 'newProject':
#> [1] "p1_30_a" "p2_30_a"
#>
#> The average line from each phogograph has been saved:
#> /private/var/folders/v4/jvjcsmnn23nbjfjjptmpbqkr0000gn/T/RtmpxJOH0X/Rbuild27834a539e4d/diskImageR/vignettes/parameter_files/newProject/aver
```

If `IJMacro()` is unable to locate ImageJ a red error will pop-up with a message like /bin/sh: /Applications/ImageJ/ImageJ.app/Contents/MacOS/JavaApplicationStub: No such file or directory. The easiest solution is to move ImageJ to the default location or to specify the path to ImageJ with argument `ImageJLoc = "/path/to/ImageJ"`.

Important! `IJMacro()` must run completely, without error, for everything downstream. In our experience this is the most likely step for errors to occur. Errors will be indicated in the R console should they arise, and will hopefully give you clues as to what the problem is if they are different than those described above.

After `IJMacro()` has run successfully the output of the ImageJ analysis can be found in the *ImageJ_out* directory, though this is probably not particularly helpful unless you want to see the intensity calculations from each line. The information about the average line from each photograph can be found in the "averageLines.csv" file located in the *parameter_out* folder. This is the information that is used for all further analysis within `diskImageR` and may be useful for other purposes.

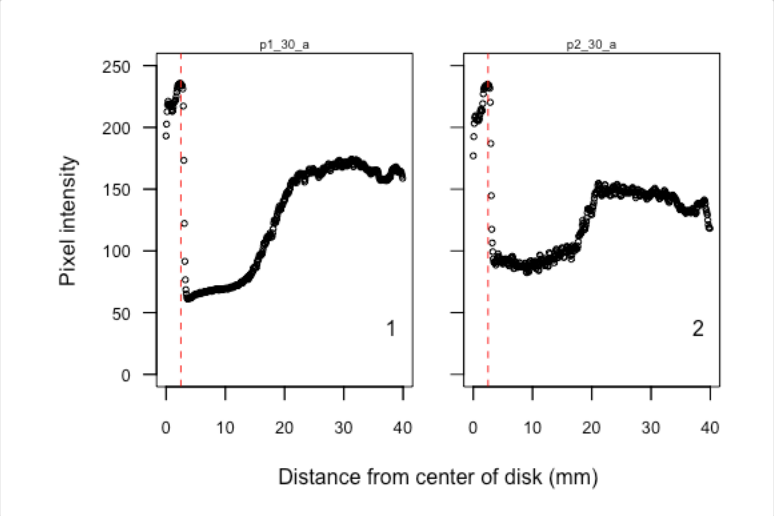
To access the output of the ImageJ analysis in a later R session use the function `readInExistingIJ()` (e.g., if you ran `IJMacro()` on a separate day then you want to conduct the downstream analyses). At this step you can also change the project name, you do not have to specify the same name that was used originally. If the name is changed new subdirectory folders will be created within the main project directory. This function will bring up a pop-up box to select the main project folder and select the directory that contains the existing ImageJ output files.

```
readInExistingIJ("betterName")
```

Plot the output of ImageJ analysis

The optional function `plotRaw()` will create a PDF file of plots saved to the *figures* directory that show the average pixel intensity across all 72 lines from each photograph (i.e., the data that can be found in the "averageLines.csv" file). This function is a good check to see whether the analysis proceeded properly and in and of itself may be useful to visualize differences between different strains or experimental factors.

```
plotRaw("newProject", showNum = TRUE, popUp = FALSE, savePDF = FALSE)
```



Many different arguments can be specified to influence the plots and the PDF that is generated, including the minimum and maximum x and y values (`xmin`, `xmax`, `ymin`, `ymax`), the number of plots in each row (`xplots`), the height and width of the PDF file (`height`, `width`), the point size (`cexPt`), and the size of the x- and y-axis font (`cexX`, `cexY`). As with all functions, you can type `?plotRaw` into the R console for all options and to see default values.

Run the maximum likelihood analysis

The next step is the function `maxLik()`, which uses maximum likelihood to find the logistic and double logistic equations that best describe the shape of the ImageJ output data. Our primary goal in curve fitting is to capture an underlying equation that fits the observed data. These data follow a characteristic "S-shape" curve, so the standard logistic equation is used where asym is the asymptote, od50 is the midpoint, and scal is the slope at od50 divided by asym/4. The midpoint from the single logistic is used to determine sensitivity.

$$y = \frac{asym * exp(scal(x - od50))}{1 + exp(scal(x - od50))} + N(0, \sigma)$$

We often observed disk assays that deviated from the single logistic, either rising more linearly than expected at low cell density, or with an intermediate asymptote around the midpoint. To fascilitate fitting these curves, we also fit a double logistic, which allows greater flexibility. In practice, as the double logistic has extra parameters, it will always provide a closer fit to the underlying data, thus the results of this model are used to determine the resistance and tolerance parameters.

$$y = \frac{asymA * exp(scalA(x - od50A))}{1 + exp(scalA(x - od50A))} + \frac{asymB * exp(scalB(x - od50B))}{1 + exp(scalB(x - od50B))} + N(0, \sigma)$$

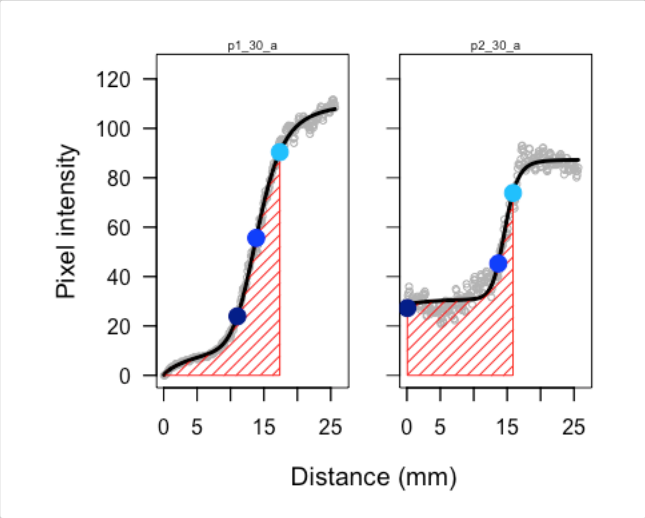
Depending on the number of photographs to be analyzed, `maxLik()` can take a fair amount of time, upwards of an hour or more. This is due to the maximum likelihood fitting procedures, which determine the best fit parameters from multiple different starting values. The status is indicated by a series of dots (".") in the R console, with one dot per photograph. This procedure is the find.mle routine from the diversitree package written by Richard Fitzjohn. If for some reason the procedure gets halted in the middle of `maxLik()` (e.g., computer is shut down) as long as R remains open it should resume where it left off when the computer is reactivated.

From these functions the plate background intensity is substracted off the intensity from all values; this should be common across all pictures taken at the same time. If you are using plates with different coloured base medium their photographs should be analzyed separately, as there will be a different background intensity from different plates. The background intensity is determined from the observed pixel intensity right beside the disk on a plate where there are no colonies in this area (e.g., the photograph on the left above). This must be specified by the user through the argument `clearHalo = X`, where X is the numbered location of the appropriate photograph. Photographs are always analyzed and organized in alphabetical order; the order can be determined by typing `names(newProject)` (no quotation marks around the project name) in the R console. In our experiments we tend to have at least one appropriate photograph with a clear halo beside the disk. A good practice, however, would be to always take a photograph of a blank plate with just the disk in the center to use for this purpose (and save it with a name like "a" so that it is always the first photograph in the list (i.e., `clearHalo = 1`). The (non)results from this photograph can be removed in the function `createDataframe()` below.

The output of `maxLik()` is a list that is saved to the R environment and a PDF file with one plot per photograph that shows the results of the model fitting (saved to the *figures* directory). Many aspects of this figure can be

specified including the maximum y axis (`ymax`) the number of plots on the x axis (`xplots`), the height and width of the PDF file (`height`, `width`), the values of RAD to be plotted (one of `80`, `50`, `20`, or `all`) and FoG cutoff value to plot (one of `80`, `50`, or `20`). Once `maxLik()` has been run once (in a given R session), it does not need to be rerun to made adjustments to the PDF file; to make a new figure use the argument `needML = FALSE`. The default is to save only a single PDF file (i.e., to repeatedly overwrite the same file with different figure iterations), this can be supressed with the argument `overwrite = FALSE`.

```
maxLik("newProject", clearHalo = 1, RAD = "all", FoG = 20, needML = TRUE, overwrite = TRUE,
    popUp = FALSE, savePDF = FALSE)
#>
#> Status of single logistic ML: ..
#> newProject.ML has been written to the global environment
#>
#> Please note the following step may take up to an hour depending on the number of photographs being anal
#>
#> Status of double logistic ML: ..
#> newProject.ML2 has been written to the global environment
```




```
#>
```

[OPTIONAL] Save the maximum likelihood results If you are intersted in the nuts and bolts of the maximum likelihood parameters it is possible to save these results using the `saveMLParam()` function, which will save a CSV file into the *paramter_files* directory that contains parameter estimates for asym, od50, scal and sigma, as well as the log likelihood of the single and double logistic models.

```
saveMLParam("newProject")
```

Create and save a dataframe of parameter estimates

The last required step is to run the function `createDataframe()` to create and save a dataframe with the drug response parameter estimates, using the best fit parameters from the logistic equations:

- **Resistance (*RAD*)**
  asymA+asymB are added together to determine the maximum level of intensity achieved on each plate (= cell density). The level of resistance (radius of inhibition, RAD), is calculated by asking what x value (distance in mm) corresponds to the point where 80%, 50% and 20% reduction in growth occured (corresponding to *RAD80*, *RAD50*, and *RAD20*)

- **Tolerance (*FoG*)**
  the `rollmean()` function from the `zoo` package is used to calculate the area under the curve from the disk edge to each RAD cutoff value. This achieved growth is then compared to the potential growth, i.e., the area of a rectangle with length and height equal to RAD. The calculated paramaters are thus the fraction of full growth in this region (*FoG80*, *FoG50*, *FoG20*).

- **Sensitivity (*slope*)**
  the ten data points on either side of the midpoint from the single logistic equation (od50) are used to find the slope of the best fit linear model using the lm function in R, i.e., the slope at the midpoint.

If you have included a blank photograph to use for the background subtraction step in `maxLik()` this can be removed from the dataframe with the argument `removeClear = TRUE`. A CSV file is written to the *parameter_files* directory which can be opened in Microsoft Excel or any program that opens text files. The dataframe is also written and saved to the R console, should you wish to conduct further analyses in R.

```
createDataframe("newProject", clearHalo = 1, typeName="Temp")
#>
#> newProject.df has been written to the global environment
#> Saving file: /private/var/folders/v4/jvjcsmnn23nbjfjjptmpbqkr0000gn/T/RtmpxJOH0X/Rbuild27834a539e4d/dis
#> newProject_df.csv can be opened in MS Excel.
newProject.df
```

```
#>      name line Temp RAD80 RAD50 RAD20 FoG80 FoG50 FoG20 slope
#> 1 p1_30_a   p1   30    11    14    17  0.36  0.27  0.29 135.1
#> 2 p2_30_a   p2   30     0    14    16    NA  0.65  0.48 101.0
```

If you want to access this dataframe in a later R session you can do so with the function `readExistingDF("betterName")`. Any project name can be used here, not only the previous name. This file can also be loaded in standard ways (e.g., `new.df <- read.csv(file)`) though if you intend to use the functions below, you need to save it with a name that ends with ".df".

[OPTIONAL] Add additional factor columns If your photograph names contain more than one factor that is important (i.e, if your files names look like: line_factor1_factor2…") you can add extra factors into the dataframe using the function `addType()`.

```
addType("newProject", typeName="rep")
#> newProject.df has been written to the global environment
#> Saving file: /private/var/folders/v4/jvjcsmnn23nbjfjjptmpbqkr0000gn/T/RtmpxJOH0X/Rbuild27834a539e4d/dis
#> newProject_df.csv can be opened in MS Excel.
newProject.df
#>      name line Temp rep RAD80 RAD50 RAD20 FoG80 FoG50 FoG20 slope
#> 1 p1_30_a   p1   30   a    11    14    17  0.36  0.27  0.29 135.1
#> 2 p2_30_a   p2   30   a     0    14    16    NA  0.65  0.48 101.0
```

Aggregate replicate pictures

The function `aggregateData()` is used if you have done replicate disk assays on the same strain and want to calculate their average and variance. The variance function can be specified with basic R variance measures (e.g, standard deviation, `sd`), the standard error (`se`), or the coefficient of variantion (`CV`).

For this example I am loading an existing dataset that I call `manyReps.df`. This dataset contains data for seven different lines, with twelve replicates per line, and a factor I'm interested in that has two levels. I then use `aggregateData()` to average among the 12 replicates and calculate their standard error. `aggregateData()` will save a CSV file into the *parameter_files* directory as well as a new dataframe to the console (`manyReps.ag`).

```
manyReps.df <- read.csv(file.path(getwd(), "data", "manyReps_df.csv"))
head(manyReps.df)
#>        name line   type RAD80 RAD50 RAD20 FoG80 FoG50 FoG20 slope
#> 1  A12_30_1  A12 levelA     0     1    10     1    NA  0.84  17.8
#> 2 A12_30_10  A12 levelA     1     2     8    NA  0.39  0.65  24.9
#> 3 A12_30_11  A12 levelA     0     1     9     1    NA  0.81  11.8
#> 4 A12_30_12  A12 levelA     0     1    20     1    NA  0.89  14.2
#> 5  A12_30_2  A12 levelA     0     1    12     1    NA  0.85  11.7
#> 6  A12_30_3  A12 levelA     0     1     9     1    NA  0.76  11.9
```

```
aggregateData("manyReps", replicate=c("line", "type"), varFunc="se")
#>
#> manyReps.ag has been written to the global environment
#>
#> Saving file: /private/var/folders/v4/jvjcsmnn23nbjfjjptmpbqkr0000gn/T/RtmpxJOH0X/Rbuild27834a539e4d/dis
manyReps.ag
#>    line   type RAD80 RAD50 RAD20 FoG80 FoG50 FoG20 slope se.RAD80 se.RAD50
#> 1   A12 levelA     0     1    10  0.93  0.65  0.79    18     0.08     0.18
#> 2   A13 levelA    13    18    21  0.71  0.36  0.31   181     0.62     0.37
#> 3   A14 levelA     9    14    19  0.56  0.39  0.38   130     0.40     0.28
#> 4   A15 levelA     9    15    18  0.66  0.35  0.32   174     1.45     0.26
#> 5   A16 levelA     7    14    20  0.78  0.46  0.42   117     1.04     0.31
#> 6   A17 levelA    10    15    18  0.60  0.33  0.29   193     1.10     0.31
#> 7   A18 levelA     3     9    12  0.78  0.51  0.42   120     0.68     0.25
#> 8   A12 levelB     0     2    10  0.94  0.85  0.78    26     0.00     0.14
#> 9   A13 levelB     4    16    22  0.85  0.56  0.46    74     1.59     0.59
#> 10  A14 levelB     1    15    22  0.72  0.65  0.55    52     0.42     0.32
#> 11  A15 levelB     6    15    20  0.74  0.44  0.42    83     1.47     0.45
#> 12  A16 levelB     8    15    21  0.43  0.37  0.38    99     1.70     0.37
#> 13  A17 levelB     5    13    19  0.74  0.47  0.45    85     1.99     1.73
#> 14  A18 levelB     2     8    12  0.86  0.55  0.45   122     0.58     0.22
#>    se.RAD20 se.FoG80 se.FoG50 se.FoG20 se.slope
#> 1      1.22       NA       NA   0.0235     1.98
#> 2      0.46   0.0498   0.0204   0.0095     6.52
#> 3      0.42   0.0626   0.0204   0.0123     3.08
#> 4      0.26   0.0875   0.0395   0.0215     6.61
#> 5      0.52   0.0657   0.0345   0.0174     5.52
#> 6      0.43   0.0796   0.0317   0.0157     4.98
#> 7      0.40   0.0615   0.0307   0.0181     5.18
#> 8      1.59   0.0625       NA   0.0219     2.41
#> 9      0.83   0.0627   0.0613   0.0355    15.95
#> 10     0.95   0.1254   0.0330   0.0220     7.58
#> 11     0.78       NA   0.0319   0.0201    10.59
#> 12     1.03       NA   0.0527   0.0421    12.34
#> 13     0.61   0.1205       NA   0.0545    14.41
#> 14     0.36   0.0453   0.0260   0.0169     5.30
```

Calculate MIC

The function `calcMIC` is used to convert the RAD values calculated here into the typical MIC values you would acquire with a broth microdilution assay (or an Etest strip). This conversion can be based on a) existing built-in data from a nubmer of species/drug combinations (see below), b) a user-supplied slope and intercept of the linear or quadratic relationship between RAD and log2(MIC) for the species/drug combination of interest, c) a user supplied file containing MIC information from lines previously analyzed by diskImageR for RAD, or d) a user supplied file containing both RAD and MIC information. Note that for user-supplied data (c or d) the data should not already be transformed and the file should be a .CSV file containing columns labelled “MIC” and “RAD”. If the user has supplied their own MIC data the function will first determine whether a linear or quadratic model provides a better fit. A figure that plots the standard curve will be saved in the file “RAD-MIC_standardCurve.pdf” in the *figures* directory and the calculated model parameters will be saved in the file “RAD-MIC_parameters.csv” in the *parameters_out* directory. In all cases a column containing the MIC information is added to the dataframe and the dataframe is saved.

Either a `diskImageR` dataframe (e.g., newProject.df) or aggregated dataframe (e.g., newProject.ag) can be used.

Plot parameter results

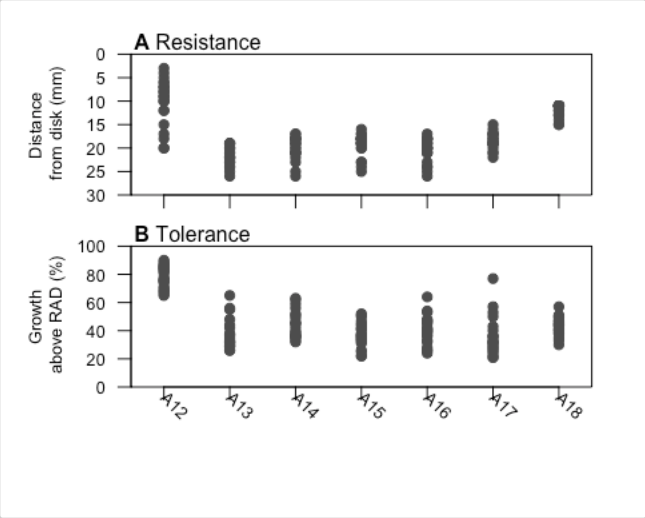
Three related plotting functions are included with `diskImageR`. The function `oneParamPlot()`will plot any of the single parameters (argument `param` supports “RAD20”, “RAD50”, “RAD80”, “FoG20”, “FoG50”, “FoG80”, “slope”, the default = “RAD20”) while `twoParamPlot()` will plot RAD and FoG at specified cutoff values (`RAD` supports “RAD20”, “RAD50”, “RAD80”; `FoG` supports “FoG20”, “FoG50”, “FoG80”), and `threeParamPlot()` will plot RAD, FoG and slope.

The required input for all three functions can be the dataframe from either `createDataframe()` (specified by argument `type="df"`, the default) or from `aggregateData()` (specified by argument `type="ag"`).

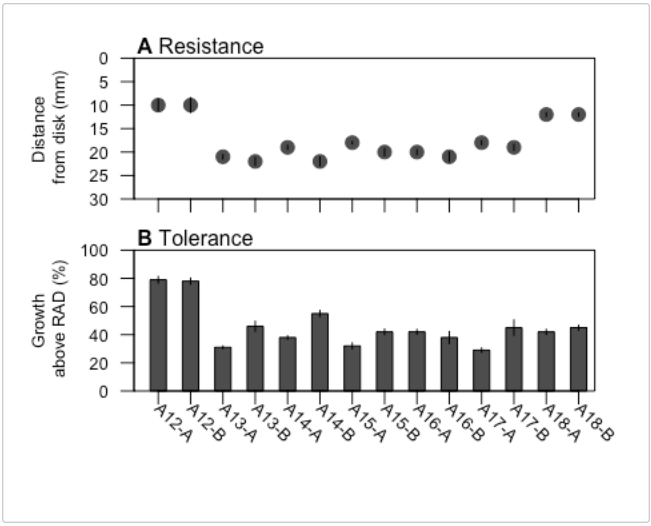
`oneParamPlot()` will plot either a barplot (argument `barplot = TRUE`) or a dotplot (argument `barplot=FALSE`). In the two and three parameter plots the default is to plot FoG as a barplot and RAD and slope as a dotplot, though FoG can also be plotted as a dotplot with argument `barplot=FALSE`.

Many aspects of these figure can be specified depending on type of dataframe and the number of parameters. Full details are provided in the accompanying package help files in R (e.g., `?oneParamPlot`).

```
twoParamPlot("manyReps", type= "df", popUp = TRUE, savePDF =FALSE, xlabAngle = -45)
```




```
twoParamPlot("manyReps", type = "ag", popUp = TRUE, savePDF = FALSE, xlabAngle = -45,
    order = c(1, 8, 2, 9, 3, 10, 4, 11, 5, 12, 6, 13, 7, 14), xlabels = paste(rep(manyReps.ag$line[1:7],
        each = 2), rep(c("A", "B"), 7), sep = "-"))
```


A Resistance
Distance from disk (mm)
0
5
10
15
20
25
30
B Tolerance
Growth above RAD (%)
100
80
60
40
20
0
A12-A
A12-B
A13-A
A13-B
A14-A
A14-B
A15-A
A15-B
A16-A
A16-B
A17-A
A17-B
A18-A
A18-B

***diskImageR*: Quantification of reistance and tolerance to antimicrobial drugs using disk diffusion assays**

Supplementary File S2. diskImageR walkthrough

```
################################################################
#Run the following the first time you want to use diskImageR
#to install from CRAN
################################################################

load.packages("diskImageR")

################################################################
#Run the following functions everytime to use up diskImageR
#For all functions type ?functionName to bring up a help file
################################################################

#load diskImageR
library(diskImageR)

#1. Run the ImageJ analysis component, save the output. "newProject" shhould be changed to something of your choice
 (and then the same name used throughout); note that the quotation marks are required.
#To use a pop-up box interface:
IJMacro("newProject")
#OR To specify the appropriate directories without a popup:
IJMacro("newProject",  "/path/to/projectDir", "/path/to/projectDir/photographs/")

#2. [OPTIONAL] Plot the result of ImageJ analysis (averaged among 72 lines draft outward from the center of the
 diffusion disk).
plotRaw("newProject")

#3. Use maximum likelihood to fit a bilogistic and single logistic model to the data from each photograph. "clearHalo" is
 used to specify a picture that has a clear halo; this is used to standardize all photographs and will be most effective
 when photographs are taken with equal lighting without shadows.
maxLik("newProject", clearHalo=1, RAD="all")

#4. Use the models to calculate resistance (20%, 50% and 80% reduction in growth = RAD20, RAD50, RAD80),
 tolerance (actual fraction of growth achvied above RAD relative to potential growth = FoG20, FoG50, FoG80), and
 sensitivity (slope at RAD50), which are saved in a .csv file.
createDataframe("newProject", clearHalo = 1)

#5. [OPTIONAL] Calculate the mean and error for parameter estimates across replicate pictures
aggregateData("newProject")

#6. [OPTIONAL] Calculate corresponding MIC values from RAD values using either built-in data for specific species/
 drug combinations or supply your own RAD/MIC data to use for a standard curve
calcMIC("newProject")
```

***diskImageR*: Quantification of resistance and tolerance to antimicrobial drugs using disk diffusion assays**

Supplementary File S3: Diffusion approximation to calculate MIC

The physical basis of the disk diffusion assay is the diffusion of the drug from the disk into the plate medium, generating a gradient of drug concentration along the radius of the plate (maximal at the center and minimal at the plate edge). The spreading of the drug can be due to pure diffusion or a mixture of additional processes, such as dissipation, degradation or active transport/advection. Naively, one could use the measured RAD to assess the MIC by calculation of the diffusion profile, but this demands a specific knowledge of the exact spreading mechanisms and all the relevant rates. In this section we demonstrate a method that allowed us to assess the MIC based on the measurements of RAD, without *a-priori* knowledge of the diffusion rate. We compare our results to the broth microdilution (BMD) assay, and found good agreement with a pure diffusion model.

Following Bonev *et al*., (2008), Delignette-Muller & Flandrois (1994), and Kavanaugh (1963) we assumed pure diffusion of the drug in the plate, without any additional mechanisms that might influence the drug profile, and a 2D circular geometry with negligible height (semi-2D). The drug concentration as a function of time t and distance r from the center of the plate, $C(r,t)$, is thus given by:

$$C(r,t) = \frac{M}{4\pi HDt} e^{-r^2/4Dt} \qquad \text{(S1)}$$

where M is the amount of drug on the disk (5-500 μg), H is the agar height (2 mm, which corresponds to 15mL liquid in a 8.5cm plate) and D is the unknown diffusion coefficient. Taking r to be the measured inhibition zone radius RAD and σ_{MIC} the MIC concentration (which we are seeking), we get:

$$RAD^2 = BlnM + A \qquad \text{(S2)}$$

where $A = -4Dt\ln(4\pi HDt) - 4Dt\ln\sigma_{MIC}$ and $B = 4Dt$. Hence, σ_{MIC} can be calculated by (Bednár, 2000):

$$\sigma_{MIC} = \frac{1}{BH\pi} e^{-\frac{A}{B}} \qquad \text{(S3)}$$

RAD was measured for four different strains (1, 3, 11, 18; Table 1) in a series of experiments with different fluconazole (FLC) concentrations by plating 2×10^5 cells on casitone medium and placing a filter-paper disk containing FLC in the center of each plate. The total amount of FLC

varied between 25μg and 500 μg (stock: 25mg/ml in ethanol) The plates were incubated at 30ºC for 48 hours. RAD was measured using *diskImageR* and RAD^2 was plotted against $\log_2(FLC)$, to obtain *A* and *B* by linear regression.

σ_{MIC} calculated using eq.S3 was compared with MIC measured by the broth micro-dilution (BMD) assay. The σ_{MIC} values provided a good approximation of the BMD MIC (Table S1), with a typical bias towards slightly higher values, which reflects the spatial restriction in the 2D disk assay. A test of the linear model (Bonev *et al*., 2008) revealed significant deviation from the BMD MIC values (data not shown), supporting the choice of the pure diffusion quadratic model. The good agreement of *diskImageR* and traditional BMD MIC measurements supports *diskImageR* as a convenient and powerful method for assessing MIC.

REFERENCES

Bednár, M. (2000). New formula for calculating antibiotic critical concentration by the disk diffusion method. *Med Sci Monit* **6**, 168–170.

Bonev, B., Hooper, J. & Parisot, J. (2008). Principles of assessing bacterial susceptibility to antibiotics using the agar diffusion method. *J of Antimicrob Chemother* **61**, 1295–1301.

Delignette-Muller, M. L. & Flandrois, J. P. (1994). An accurate diffusion method for determining bacterial sensitivity to antibiotics. *J Antimicrob Chemother* **34**, 73–81.

Kavanagh, F. (1963) Analytical chemistry. Academic Press Inc., New York.

Supporting Table S1: MIC values approximated by the diffusion equation (σ_{MIC}) using parameters inferred from disk diffusion measurements are in agreement with direct MIC measurements from broth microdilution assays (BMD MIC).

Strain	σ_{MIC}	BMD MIC
1	1.82	2
3	0.2	0.125
11	0.81	0.5
18	8.8	4

Supporting Table S2. Calculated parameters from the literature for the model between $\log_2$(MIC) and RAD. Papers included in this dataset included only studies that reported results from a single species or genus. We converted ZOI to RAD (by subtracting the disk diameter, 6mm, and dividing by 2). We then determined the average RAD value for each MIC value that had at least two measurements in the dataset to avoid skewing the relationship by the overabundance of sensitive strains assessed. The best fit relationship (linear vs. quadratic, i.e., RAD vs. RAD^2) was determined by comparison of R^2.

Species	Drug	Intercept	Slope	Relationship	R^2	Isolates	Reference
Acinetobacter spp	tigecycline (15ug)	5.06	-0.11	quadratic	0.92	102	Jones et al., 2007 UCM 45:227-230
Aspergillus spp	voriconazole (1ug)	3.97	-0.67	linear	0.89	75	Serrano et al. 2004, JAC 53:739-742
Bifidobacterium spp	clindamycin (2ug)	4.05	-0.48	linear	0.94	141	Domig et al., 2007, Int J Food Microbio 120:191-195
Bifidobacterium spp	erythromycin (15ug)	5.96	-0.45	linear	0.99	162	Domig et al., 2007, Int J Food Microbio 120:191-195
Bifidobacterium spp	tetracycline (30ug)	6.62	-0.36	linear	0.92	142	Domig et al., 2007, Int J Food Microbio 120:191-195
Candida glabrata	fluconazole (25ug)	6.52	-0.028	quadratic	0.93	234	Pfaller et al. 2003, JCM 41:1875-1880
Candida glabrata	voriconazole (1ug)	2.22	-0.32	quadratic	0.94	234	Pfaller et al. 2003, JCM 41:1875-1880
Candida spp	fluconazole (25ug)	6.83	-0.04	quadratic	0.98	2069	Barry et al., 2002, AAC 46:1781-1784 & Pfaller et al., 2003 JCM 41: 1440-1446
Candida spp	posaconazole (5ug)	3.93	-0.07	quadratic	0.89	2090	Diekema et al., 2007, JCM 45:1974-1977
Candida spp	voriconazole (1ug)	2.23	-0.035	quadratic	0.95	1496	Barry et al., 2002, AAC 46:1781-1784 & Pfaller et al., 2003 JCM 41: 1440-1446
Cryptococcus neoformans	fluconazole (25ug)	6.36	-0.46	linear	0.98	285	Pfaller et al., 2004 JCM 42: 380-383
Haemophilus influenzae	ampicillin (10ug)	6.92	-0.68	linear	0.91	159	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	cefaclor (30ug)	14.26	-1.23	linear	0.83	166	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	cefamandole (30ug)	12.9	-1.16	linear	0.95	143	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	cefuroxime (30ug)	6.43	-0.047	quadratic	0.86	144	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	chloramphenicol (30)	4.57	-0.022	quadratic	0.92	145	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	rifampin (5ug)	5.85	-0.77	linear	0.92	135	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Haemophilus influenzae	tetracycline (30ug)	11.9	-0.96	linear	0.88	150	Doern et al., 1990, Eur J Clin Microbiol Infect Dis 9:329-336
Neisseria gonorrhoeae	ceftriaxone (30ug)	2.08	-0.02	quadratic	0.97	99	Jones et al., 1989, JCM 27:2758-2766
Neisseria gonorrhoeae	penicillin (10U)	5.37	-0.49	linear	0.99	100	Jones et al., 1989, JCM 27:2758-2766
Neisseria gonorrhoeae	tetracycline (30ug)	5.16	-0.02	quadratic	0.95	97	Jones et al., 1989, JCM 27:2758-2766
Staphylococcus aureus	mupirocin (200ug)	8.74	-0.066	quadratic	0.93	434	Swenson et al., 2010 UCM 48:2469-2475
Streptococcus pneumoniae	cefditoren(5ug)	10	-1.43	linear	0.85	197	Kelly et al. 1999, JCM 37:3296-3299
Streptococcus pneumoniae	chloramphenicol (30)	6.33	-0.52	linear	0.92	244	Jorgensen et al., 1994, JCM 32:2448-2459
Streptococcus pneumoniae	erythromycin (15)	4.08	-0.74	linear	0.98	247	Jorgensen et al., 1994, JCM 32:2448-2459
Streptococcus pneumoniae	meropenem (10)	4.11	-0.04	quadratic	0.93	248	Jorgensen et al., 1994, JCM 32:2448-2459
Streptococcus pneumoniae	penicillin	0.27	-0.72	linear	0.86	239	Jorgensen et al., 1994, JCM 32:2448-2459
Streptococcus pneumoniae	tetracycline (30ug)	4.15	-0.4	quadratic	0.72	63	Jorgensen et al., 1994, JCM 32:2448-2459

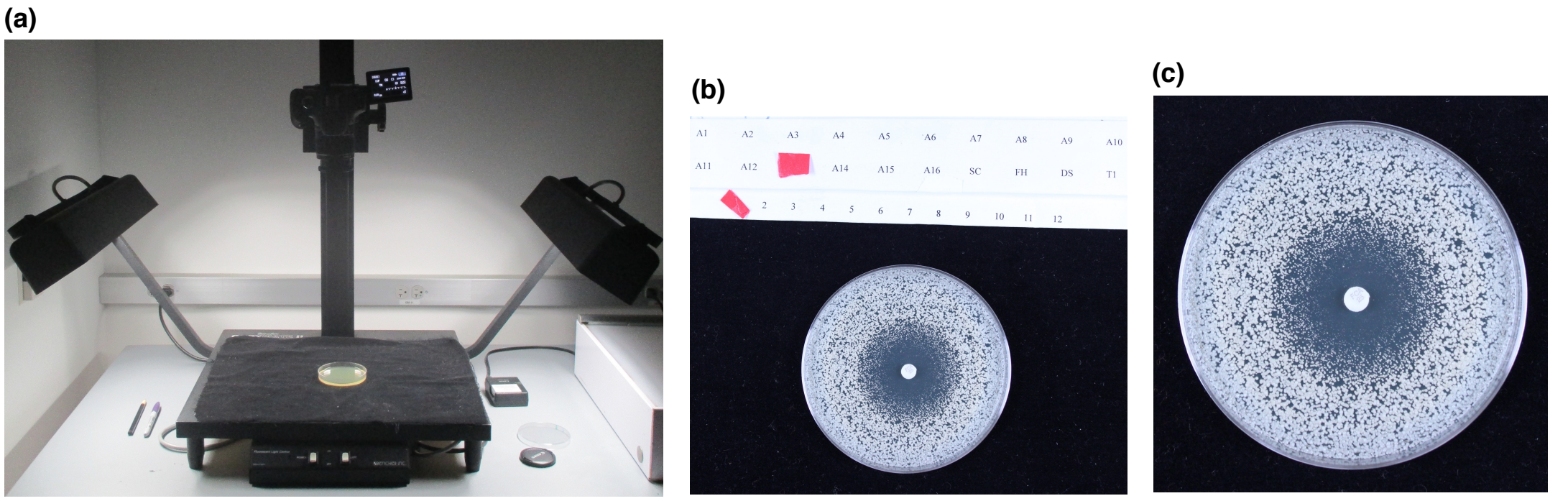


Figure S1. Photography set-up. A) The camera is mounted at a fixed distance above the plate (Bencher Copymate II). Two fluorescent lights on either site of the disk are mounted to minimize shadows in an otherwise dark room. A Canon Rebel T3i camera is used in manual mode with an ISO 800, white balance "white fluorescence light", time 1/100s, picture style "neutral", centre focused. We also use the 2s timer to avoid potentially jostling the camera while taking images and/or having a hand/arm shadow in the picture. B) Plates are photographed on a surface covered with black velvet and plate labels are written on the side rather than the bottom of the plate; a labeling scheme is included in the original image. C) Prior to analysis, images are cropped close to the plate. Because the analysis program automatically detects the disk based on size, it is important that no other similar-sized circles be present in the image (e.g., from letters in labels).

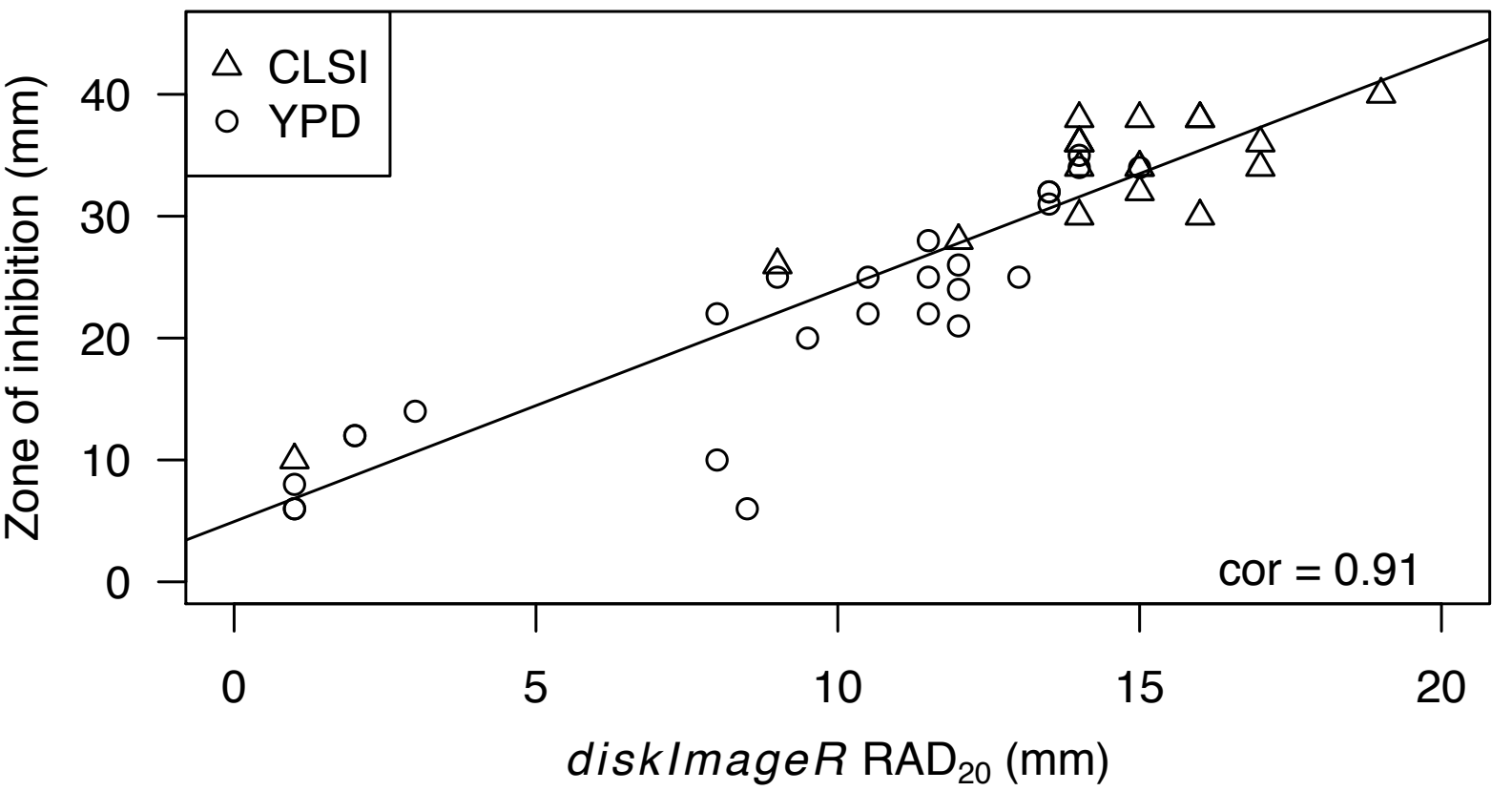


Figure S2. Resistance measured with *diskImageR* correlates with zone of inhibition measured with a ruler. The zone of inhibition was measured with a ruler from disk assay plates from each strain growth under CLSI (triangles) and YPD (circles) conditions and compared to the same plates measured for RAD_{20} with *diskImageR*.

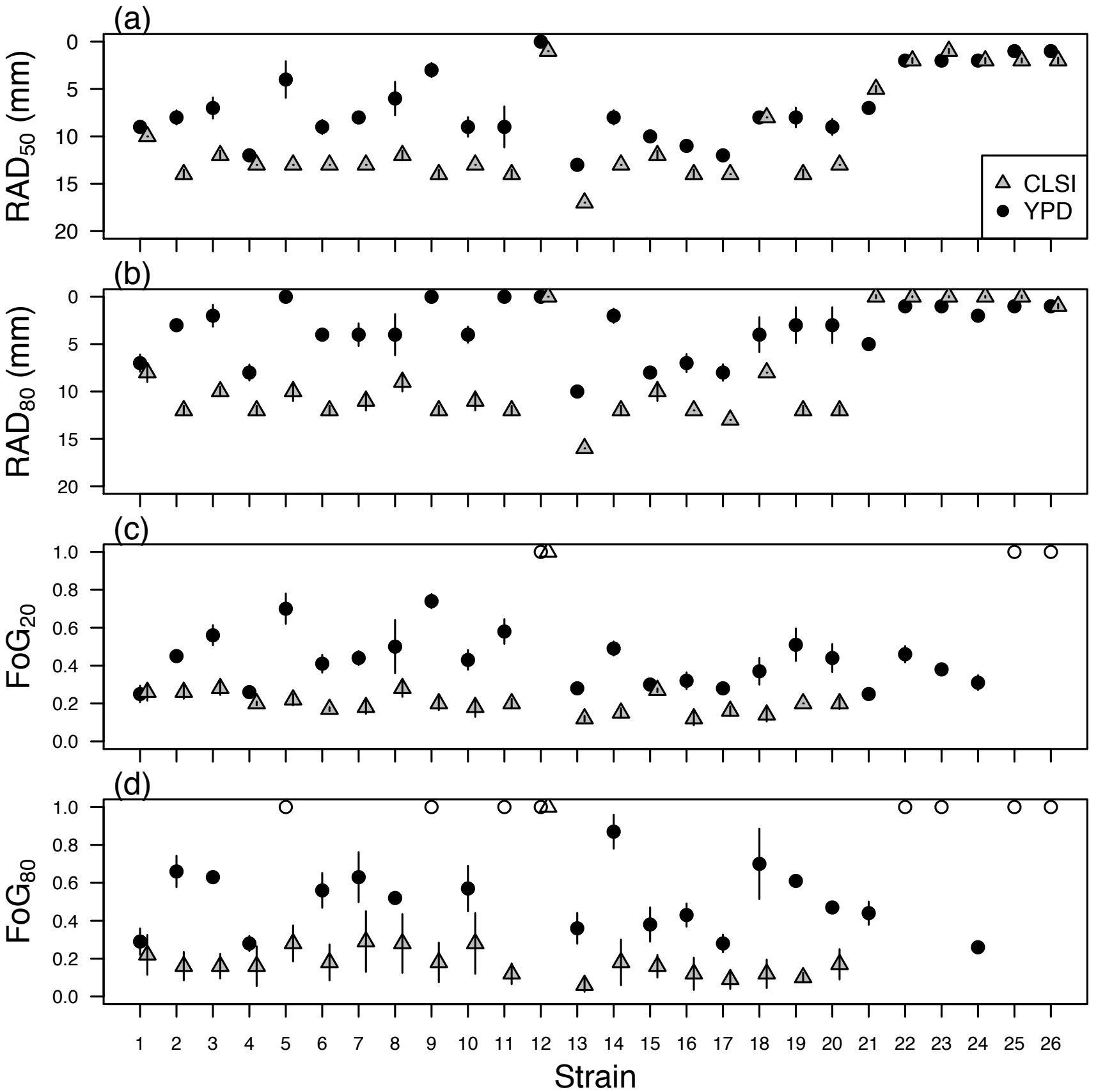


Figure S3. RAD and FoG determined for 26 clinical isolates of *C. albicans* for fluconazole. Disk assays were conducted based on CLSI conditions (grey triangles) and YPD conditions (black circles). RAD was calculated as the point at 20% (in the main text), a) 50% (RAD_{50}) or b) 80% inhibition relative to full growth. FoG was calculated at the point where growth is reduced by c) 20% (FoG_{20}), 50% (FoG_{50}, in the main text), or d) 80% (FoG_{80}). Note that FoG cannot be accurately measured when the radius of inhibition (RAD) is < 2mm (open points) and these data were not included for statistical purposes.

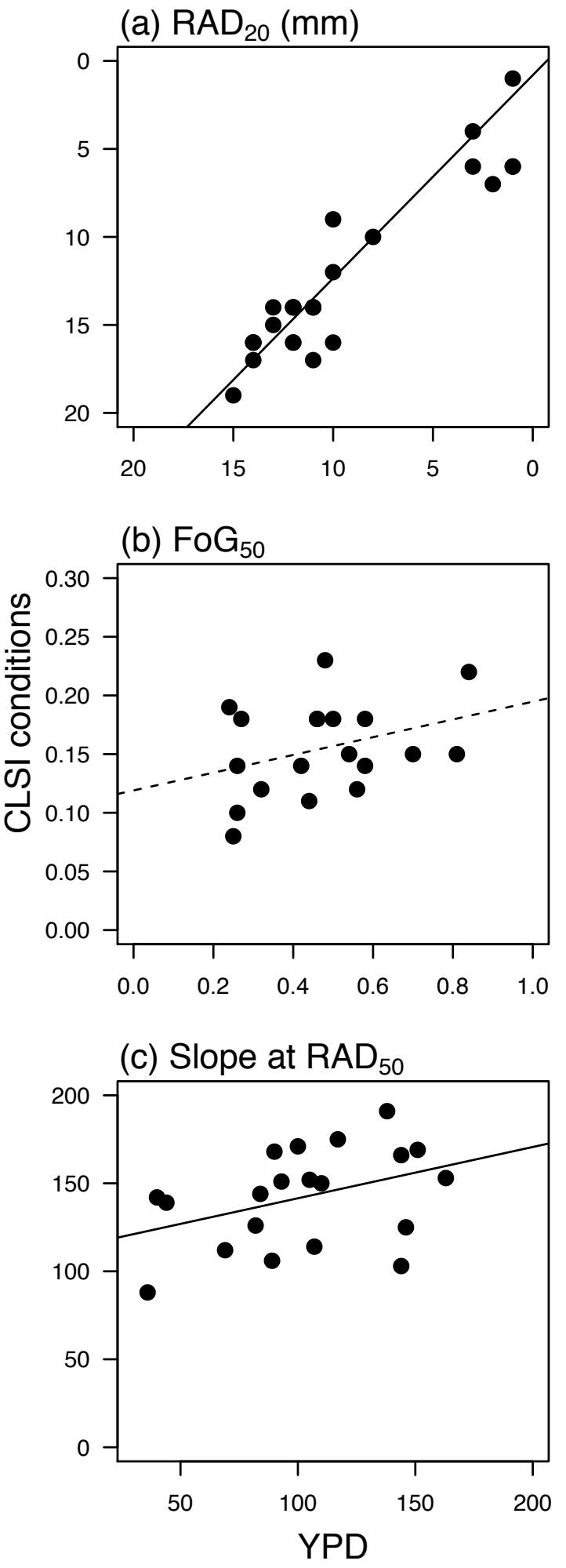


Figure S4. Correlation for resistance, tolerance and sensitivity across different assay conditions. Disk assays conducted on CLSI conditions or YPD conditions were significantly correlated for a) RAD_{20}, and c) SLOPE at $RAD_{50,}$ but not for b) $FoG_{50..}$

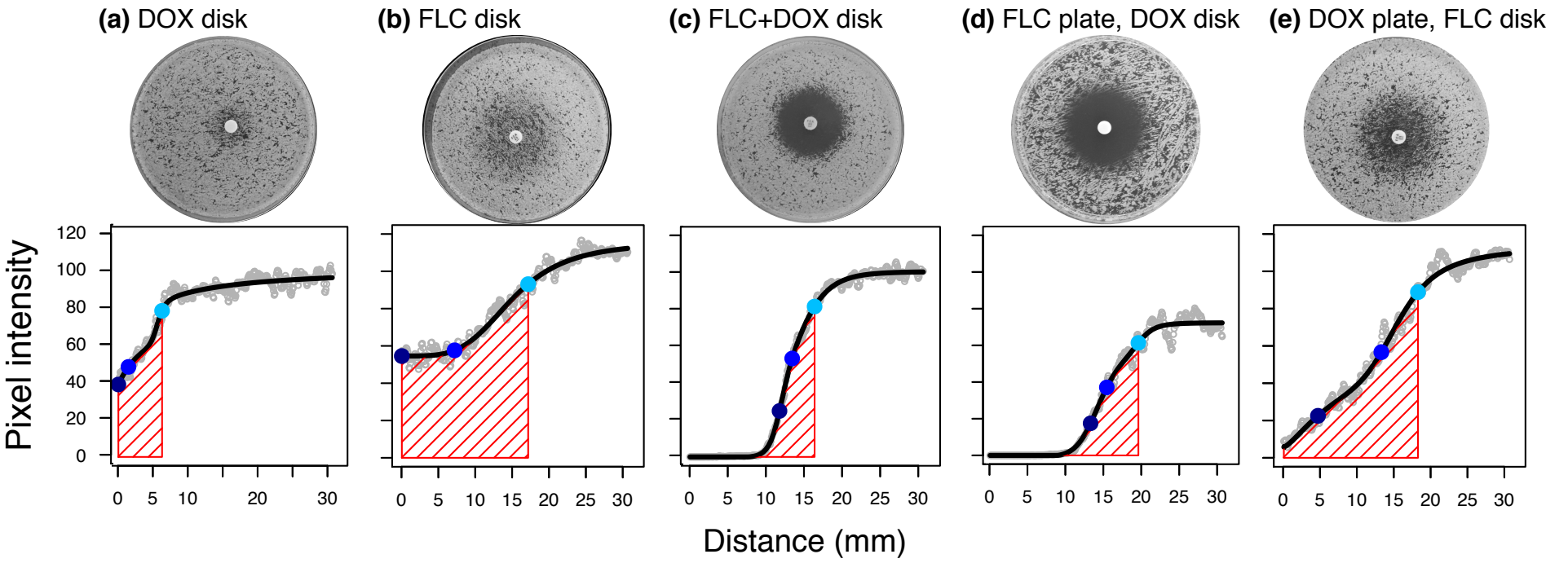


Figure S5. Drug combinations quantified using *diskImageR* for fluconazole (FLC) and doxycycline (DOX). The interaction between drugs was compared to a single drug alone in the disk (25μg/ml for each drug) a) DOX disk only; b) FLC disk only; c) for both drugs added to the same disk; d) where FLC was in the disk and DOX was present in the plate (25μg/ml); or e) where DOX was in the disk and fluconazole (1μg/ml) was in the plate.