Contents

R2easyR

Facilitates Mapping Experimental RNA Reactivity Data to Secondary Structures Drawn with R2R

Authors: Jacob P. Sieg, Peter C. Forstmeier, and Philip C. Bevilacqua

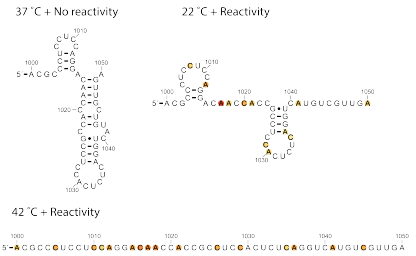
Why use R2easyR?

Manually drawing experimental RNA reactivity data as colors on a secondary structure is time consuming and error prone. R2easyR, when used with the drawing program R2R, makes processing reactivity data, and drawing the reactivity on a secondary structure much easier. With a small amount of workup in a drawing program, the output of the mostly automated R2easyR/R2R pipeline produces publication-ready figures rapidly.

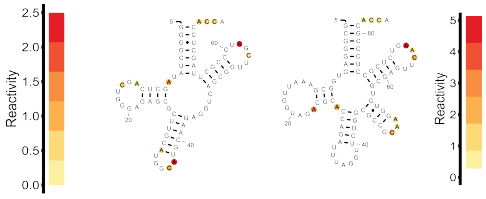
Acknowledgment: This work is supported by National Institutes of Health Grant R35-GM127064 to PCB.

Gallery of examples

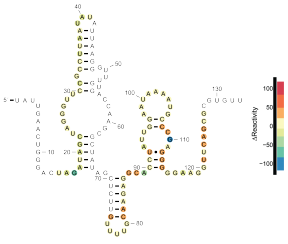
In-vivo DMS structure probing data from SructureSeq2 mapped onto predicted folds from RNAstructure for nucleotides 1000 to 1050 of the rice HOX16 mRNA.



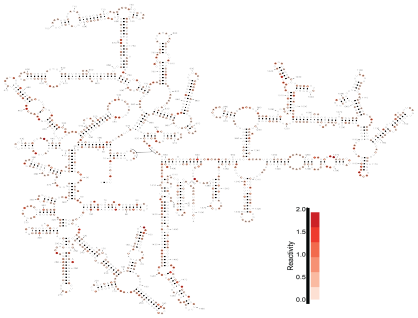
DMS-MAP data mapped onto two *E. coli* tRNA



Delta In-line probing reactivity mapped onto the *C. bot ykkC 2c* NDP binding riboswitch. Reactivity changes were induced by binding of the ADP ligand.



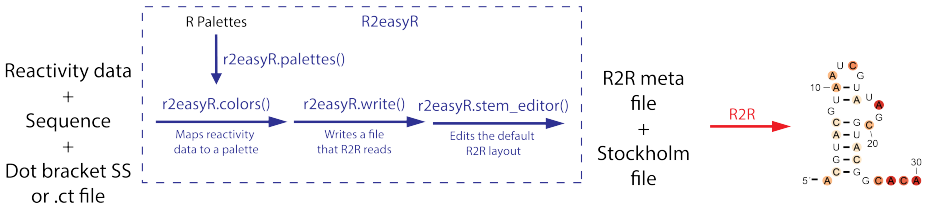
DMS structure probing data from StructureSeq2 mapped onto the consensus secondary structure of a 16S-rRNA transcript



The R2easyR package allows you to programmatically convert experimental reactivity data into input files for R2R. R2R (https://sourceforge.net/projects/weinberg-r2r/) is fantastic software for drawing RNA secondary structures. R2R structures are uniform, making secondary structures drawn with R2R intuitive and publication ready. Moreover, R2R excels at displaying conservation information for RNA motifs, the primary purpose of R2R. Unfortunately, R2R contains fewer options for drawing experimental reactivity data on RNA secondary structures. While all the code R2R needs to map continuous reactivity data to colors on a secondary structure is present, R2R does not have good programmatic options for supplying drawing instructions for reactivity data. Thus, R2easyR is designed to link reactivity data and the R2R syntax. R2easyR is a toolbox of R commands that integrate common RNA-lab data formats with appropriate color palettes, and write files that R2R uses to color nucleotides on secondary structure. This approach is trivial for short RNA (<100N), and reasonably achievable for long RNA (100–3000 N).

Overview

The core of the R2easyR is four functions: "r2easyR.pallets", "r2easyR.color", "r2easyR.write", and "r2easyR.stem_editor". "r2easy.palettes" is a palette generation function that makes 59 palettes that can be used to color reactivity data on RNA secondary structures. "r2easyR.color" is an assignment function that assigns colors to nucleotides based on the reactivity at that nucleotide. If the nucleotide reactivity is higher, the color "r2easyR.color" assigns will be more intense. "r2easyR.write" is a file writing function that writes two files (a Stockholm file and a R2R meta file) that tell R2R how to color each nucleotide on a secondary structure. "r2easyR.stem_editor" is a R2R Stockholm file editor, that supplies instructions to a stockholm file that optimizes the layout of stems in a multistem structure so that stems are less likely to clash.



The four core R2easyR functions can be used in a R script to generate generate a single RNA secondary structure in a couple minutes starting with a .csv file (Video turotial 1) or a .ct file (Video tutorial 2). Alternatively, if you are good at R scripting, one can run the four core R2easyR on a loop to print RNA secondary structures depictions in bulk. Two such bulk structure printers are built into R2easyR. The first is called "ry.tRNA" and maps reactivity data to near-"cloverleaf" tRNA structures in bulk starting with .ct and .shape formatted secondary structure/reactivity data. The second, called "r2easyR.go_fast" maps reactivity data to any RNA secondary structure in bulk starting with .ct and .shape formatted secondary structure/reactivity data (Video tutorial 3).

Video tutorials

1.) Using R2easyR to map reactivity data to a simple RNA 2-structure using a .csv

https://youtu.be/tLZDegQ6b20

2.) Using R2easyR to map reactivity data to a simple RNA 2-structure using a CT file from RNAstructue

https://youtu.be/2wCFcz3NEiw

3.) Using R2easyR to map reactivity data to RNA 2-structures in bulk using CT files from RNAstructure

https://youtu.be/SO_8z9tvXK4

Manual

1. Installation

Prior to installing R2easyR, please install R. If you are not used to R or other command- line programs, I strongly recommend downloading and working in RStudio. To install R2easyR on your R console, open your R terminal or RStudio and type:

```
>install.packages("devtools")
```

This will take a minute. Install any dependent packages. “devtools” is a fantastic R package for developing, distributing, and downloading R packages. I cannot guarantee that your R2easyR installation will work without “devtools”. After “devtools” is installed, type/enter:

```
>devtools::install_github("JPSieg/R2easyR")
```

R2easyR should be installed in a few minutes. The command prompt may ask you to update dependent packages. I recommend skipping the update by entering an empty line. If you are using a fairly recent version of R, not much has changed and the updates can take a while. To check the installation when it finishes, type/enter.

```
>library(R2easyR)
>?r2easyR.color
```

The help files for the function "r2easyR.color" will pop up if R2easyR was installed correctly. Note the "library" function does not require quotation marks around the package. "library" loads the R2easyR package from your R library into your R memory. Thus, you will need to use "library" again after you close and restart R. You can also call R2easyR functions explicitly using the following syntax, with no need to use "library".

```
>package>::function
```

For example:

```
>?R2easyR::r2easyR.color
```

R2easyR relies on 4 common packages that do not come with base R: "ggplot2", "viridis", "RColorBrewer", and "R.utils". If you are a R user, you have probably already installed them. If not, or you are not sure, please install or reinstall using:

```
>install.packages("package")
```

For example:

```
>install.packages("ggplot2")
```

Reinstalling R2easyR on top of an existing R2easyR installation can corrupt the help files. Thus, if you need to reinstall R2easyR, please remove the current installation and then reinstall R2easyR. First, restart R so you are not removing a package that is currently loaded in the memory:

```
>q()
```

Then restart R and run:

```
>devtools::install_github("JPSieg/R2easyR")
```

2. Loading data into R and Formatting it for R2easyR

The goal of loading your data into R is to supply R2easyR the nucleotide number (N), the nucleotide sequence, the secondary structure (in dot-bracket notation (.<.>.), and the Reactivity data you want to map. R2easyR functions work on data frames, a R data format that resembles a table where each column is labeled. The order of these columns does not matter, but the columns must be labeled as exactly "N", "Nucleotide", "Dotbracket", "Reactivity".

2.1 Loading data into R with a comma separated value (.csv) text file

This is the easiest data loading strategy, and is best for reactivity data that comes from a quantified gel. Simply open up Excel, label the first four cells (A1:A4) with the header of Figure 2, and then enter the information for your RNA. Then, save as a Comma Separated Value (".csv") file. For example:

N	Nucleotide	Dotbracket	Reactivity
1	G	.	
2	A	.	0.6
3	C	<	0.2
4	G	<	
5	T	<	
6	A	<	0.1
7	C	<	0.2
8	G	.	
9	T	<	
10	A	<	0.7
11	A	.	0.9

Also note that missing reactivity values are specified by leaving the cell blank. Please do not use a text string like "NA" as a place holder for missing values, as this will mess with the data when it is read into R.

Now your ".csv" file can be read into a data frame (df) using "read.csv":

```
>df <- read.csv("file_name.csv")
```

You can check your data frame (df) using the "head" function or the "View" function.

```
>head(df)

  N Nucleotide Dotbracket Reactivity
1 1          G          .         NA
2 2          A          .        0.6
3 3          C          <        0.2
4 4          G          <         NA
5 5          T          <         NA
6 6          A          <        0.1

>View(df)
```

2.2 Loading a connectivity table (".ct") file into R and converting the ".ct" format to "dot-bracket" format in R

Specifying a secondary structure with the Dotbracket notation can be prohibitively tricky for long RNA. RNA secondary structure prediction algorithms like RNAStructure (https://rna.urmc.rochester.edu/RNAstructure.html) uses the ".ct" format to specify a RNA secondary structure. R2easyR contains a function called "read.ct" to read a ".ct" file into a R data frame and a function called "add.dot.bracket" to convert the ".ct" format to the "dot-bracket" format. The syntax for "read.ct" is the same as "read.csv":

```
>df <- read.ct("file_name.ct")
>head(df)
    N Nucleotide N-1 N+1 BP   N
1 800          G 799 801  0 800
2 801          C 800 802  0 801
3 802          A 801 803  0 802
4 803          U 802 804  0 803
5 804          C 803 805  0 804
6 805          U 804 806  0 805
```

Now simply apply "add.dot.bracket" to add a "Dotbraket" column to the data frame (df)

```
>df <- add.dot.bracket(df)
>head(df)
    N Nucleotide N-1 N+1 BP   N Dotbracket
1 800          G 799 801  0 800          .
2 801          C 800 802  0 801          .
3 802          A 801 803  0 802          .
```

```
4 803          U 802 804  0 803          .
5 804          C 803 805  0 804          .
6 805          U 804 806  0 805          .
```

2.3 Loading in reactivity data from ".react" and ".shape" files and placing reactivity data into a data frame

StructureFold2 (https://github.com/StructureFold2/StructureFold2) formats reactivity data in ".react" files. R2easyR contains a function called "read.react" that reads reactivity data into a R list that is named by the RNA. Likewise, ShapeMapper (https://weekslab.com/software/) formats reactivity data in ".shape" files. R2easyR contains a function called "read.shape" that reads reactivity data into R. Reactivity data is easily placed into a data frame after it is read into R using "read.react" or "read.shape".

To read in data, simply type:

```
>Reactivity <- read.shape("file_name.ct")
>shape
 [1]         NA 0.14494805 0.75081005         NA 1.70183611         NA         NA 0.0771665
[12] 0.37853340         NA 0.78105101         NA         NA 0.15641876         NA 0.0104279
[23]         NA 0.64235971         NA         NA 0.00000000         NA         NA 0.1626755
[34] 0.51826749 1.28576221 0.50888237         NA 0.50888237         NA         NA         NA
[45] 1.01880752         NA         NA 1.06781873         NA 1.01880752         NA
```

Now add a new column to your data frame (df):

```
>df$Reactivity <- Reactivity
>head(df)
    N Nucleotide N-1 N+1 BP   N Dotbracket Reactivity
1 800          G 799 801  0 800          .         NA
2 801          C 800 802  0 801          .  0.1449481
3 802          A 801 803  0 802          .  0.7508100
4 803          U 802 804  0 803          .         NA
5 804          C 803 805  0 804          .  1.7018361
6 805          U 804 806  0 805          .         NA
```

3. Generating Color Palettes with R2easyR

3.1 Generating color palettes that are built into R2easyR

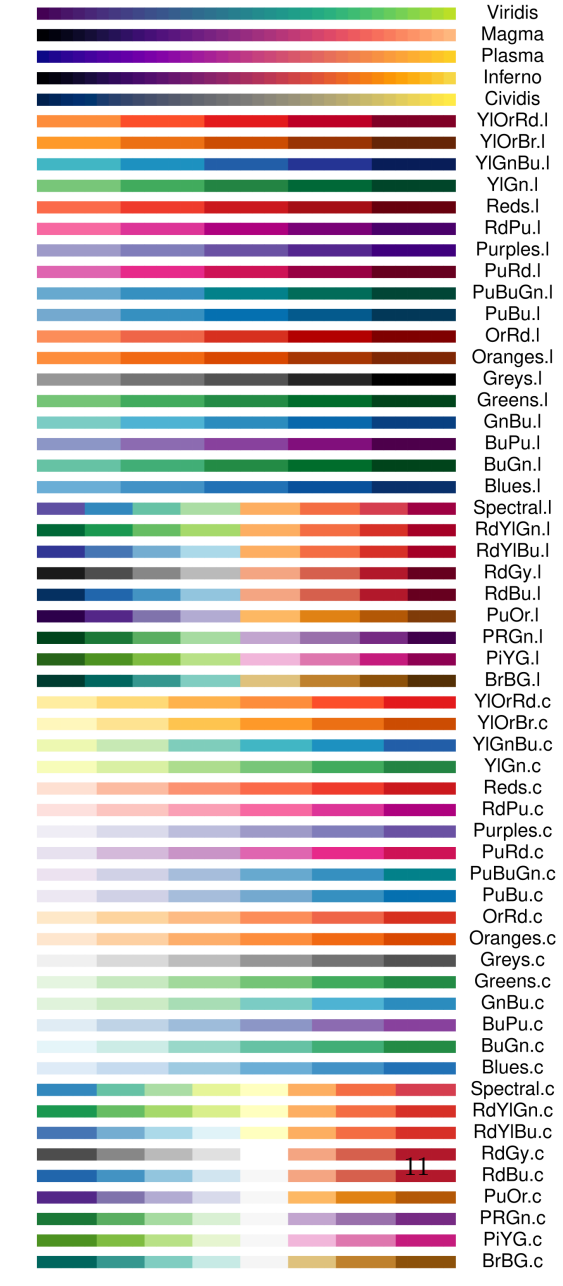
R2easyR uses pallets consisting of vectors containing 35 R colors, arranged from coolest/least-intense to warmest/most-intense. R2easyR contains a function called "r2easyR.palettes" that generates a curated list of 59 palettes using RColorBrewer (https://cran.r-project.org/web/packages/RColorBrewer/index.html) and Viridis palettes (https://mran.revolutionanalytics.com/web/packages/viridis/vignettes/intro-to-viridis.html). These palettes can be fed into the function "r2easyR.color" to color nucleotides. The following code will generate that list in an object called "palettes":

```
>palettes <- r2easyR.palettes()
```

We can call an individual pallet from that list using the $ syntax:

```
>palettes$Reds.c
"#FEE0D2" "#FEE0D2" "#FEE0D2" "#FEE0D2" "#FEE0D2" "#FCBBA1" "#FCBBA1" "#FCBBA1" "#FCBBA1" "#
[13] "#FC9272" "#FC9272" "#FC9272" "#FC9272" "#FC9272" "#FB6A4A" "#FB6A4A" "#FB6A4A" "#FB6A4
[25] "#EF3B2C" "#EF3B2C" "#EF3B2C" "#EF3B2C" "#EF3B2C" "#CB181D" "#CB181D" "#CB181D" "#CB181
```

A PDF depiction of all 59 palettes made by "r2easyR.pallets" is written in your working directory when you run "r2easyR.pallettes", to help you choose a palette that works for your application. ".c" palettes like "Reds.C" are recommended for coloring reactivity data as circles behind nucleotides because they avoid very dark shades that would obscure the nucleotide letter. ".l" palettes are recommended for coloring reactivity data as the color of the nucleotide letter because they avoid light shades that would be hard to see against a white background.



3.2 Creating a custom color palette

R2easyR contains a function called "r2easyR.custom.palette" for creating your own palette. "r2easyR.custom.palette" converts a vector containing 35 or less R colors to a format that "r2easyR.color" can use. To make a vector("a") containing R colors we use the "c()" syntax. First we have to make a vector containing the colors we want.

```
>a <- c("green", "yellow", "Red")
>a
[1] "green"  "yellow" "Red"
```

Now we can use "r2easyR.custom.palettes" to make a R2easyR palette out of "a":

```
> test.pal <- r2easyR.custom.palette(a)
 [1] "green"  "green"  "green"  "green"  "green"  "green"  "green"  "green"  "green"  "green"
[14] "yellow" "yellow" "yellow" "yellow" "yellow" "yellow" "yellow" "yellow" "yellow" "yello
[27] "Red"    "Red"    "Red"    "Red"    "Red"    "Red"    "Red"    "Red"    "Red"
```

"testpall" is now an R object that can be fed into "r2easyR.color" to assign colors to reactivity data.

4 Mapping Reactivity Data to Color Pallets with R2easyR

4.1 Choosing a color pallet

"r2easyR.pallettes" writes a PDF depiction of all 59 palettes to help the user choose a palette. You can reference this PDF, or Figure 3 when you are choosing a palette. I find the following rules get aesthetically pleasing results:

1. ".l" palettes are recommended for coloring letters because they exclude light shades, which would be hard to see against a white background.

2. ".c" palettes are recommended for coloring circles behind letters because they exclude dark colors, which obscure the letter inside the circle.

3. Palettes that transition from one color two another are good for mapping change in reactivity data.

4. Viridis palettes, Viridis, Magma, Plasma, Inferno, and cividis are not recommended because they result in cluttered secondary structures. Viridis makes great palettes, but they don't look good on RNA secondary structures.

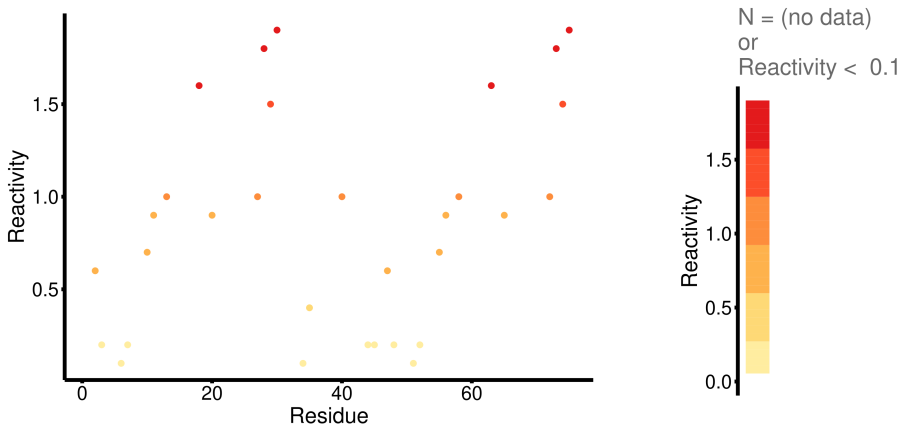
4.2 Mapping reactivity data to color palettes with "r2easyR.colors"

R2easyR contains a function called "r2easyR.color" that assigns colors based on reactivities. "r2easyR.color" uses the following syntax to add color information to a data frame containing a "Reactivity" column:

```
df <- r2easyR.color(df,
                    palettes$Reds.c)
head(df)
    N Nucleotide N-1 N+1 BP   N Dotbracket Reactivity Labels  Colors
1 800          G 799 801  0 800          .         NA      0 dimgrey
2 801          C 800 802  0 801          .  0.1449481      1 #FEE0D2
3 802          A 801 803  0 802          .  0.7508100      1 #FC9272
4 803          U 802 804  0 803          .         NA      0 dimgrey
5 804          C 803 805  0 804          .  1.7018361      1 #CB181D
6 805          U 804 806  0 805          .         NA      0 dimgrey
```

This will add two columns to the data frame "(df), a "Label" column that is used by R2easyR to determine if there is data or not and a "Colors" column that contains R formatted colors. After the "Label" and "Colors" columns are made with "r2easyR.colors", R2R input files are written with the R2easyR writing function "r2easyR" (see section 5.1), and R2R is used to write a PDF depiction of the secondary structure (see section 5.3).

"r2easyR.color" also prints two helpful PDFs when it runs. The first, labeled "Rxn_plot.pdf" is a plot of reactivity versus nucleotide number, with data points colored as they will appear in the final PDF. Looking at "Rxn_plot.pdf" is a quick way to know how a given palette will make your secondary structure look. The second PDF, labeled "Legend.pdf" is a legend you can use for your secondary structure. One can simply open "Legend.pdf" in Illustrator and copy and paste it where it needs to go.



4.3 Using a reactivity threshold to remove low reactivity data that cause clutter

Low reactivity data that is almost 0 can make a RNA secondary structure look very cluttered, especially if you have quantified a gel because every nucleotide will have some baseline level of reactivity, which can make the resulting secondary structure look cluttered. "r2easyR.color" has an argument called "abs_reactivity_threshold" that will have "r2easyR.color" treat any reactivity data below this threshold like there is no data present, so that it does not clutter up the figure. For example:

```
df <- r2easyR.color(df,
                    palettes$Reds.c,
                    abs_reactivity_threshold = 0.2)
```

```
head(df)
    N Nucleotide N-1 N+1 BP   N Dotbracket Reactivity Labels  Colors
1 800          G 799 801  0 800          .         NA      0 dimgrey
2 801          C 800 802  0 801          .  0.1449481      0 dimgrey
3 802          A 801 803  0 802          .  0.7508100      1 #FC9272
4 803          U 802 804  0 803          .         NA      0 dimgrey
5 804          C 803 805  0 804          .  1.7018361      1 #CB181D
6 805          U 804 806  0 805          .         NA      0 dimgrey
```

Note: reactivity values below 0.2 are treated like missing reactivity values in the "Label" and "colors" column

4.4 Specifying a manual scale

A "r2easyR.color" argument called "manual.scale" can be used to specify the scale that "r2easyR.color" should map reactivity data too. "manual.scale" is useful in two situations. The first is when you want to map reactivity to more than one secondary structure using the same scale. The second is when one nucleotide is much more reactive than the rest of the nucleotides in the data set, so that reactivity of most of the nucleotides are bleached out by the highly reactive nucleotide in the final secondary structure. In the second case, you can set the manual scale so that it fits the more numerous less reactive data and maps the extremly reactive nucleotide to the strongest color in the palette. Setting a manual scale is simple:

```
>df <- r2easyR.color(df,
                     palettes$Reds.c,
                     abs_reactivity_threshold = 0.2,
                     manual.scale = c(0, 2))
>head(df)
    N Nucleotide N-1 N+1 BP   N Dotbracket Reactivity Labels  Colors
1 800          G 799 801  0 800          .         NA      0 dimgrey
2 801          C 800 802  0 801          .  0.1449481      0 dimgrey
3 802          A 801 803  0 802          .  0.7508100      1 #FC9272
4 803          U 802 804  0 803          .         NA      0 dimgrey
5 804          C 803 805  0 804          .  1.7018361      1 #CB181D
6 805          U 804 806  0 805          .         NA      0 dimgrey
```

4.5 Other r2easyR.color arguments

The purpose of other r2easyR.color arguments are explained by the help file. Pulling up the help file is simple:

```
>?r2easyR.colot
```

5 Writing R2R input files with R2easyR and making figures with R2R

R2easyR contains the function "r2easyR.write", which writes the files R2R uses to draw a secondary structure. The syntax is simple:

```
>r2easyR.write("Example", df, colors = "circles")
```

The first argument is the prefix that you want on the files you are writing. "r2easyR.write" will add the file suffixes ".sto" and ".r2r_meta" for you. The second argument is a data frame that was made with "r2easyR.colors". The third argument is the name of the RNA you are drawing. You don't have to set this argument. The RNA we are currently drawing was made up, so I called it "made up". Other arguments are explained in the help file:

```
>?r2easyR.write
```

Note that the argument "colors" is how you want R2R to draw the reactivity data. The options are, "NA" to not draw any reactivity data, "letters" to color the letters according to their reactivity, and "circles" to draw reactivity data as circles behind the nucleotides. I like "circles" the best.

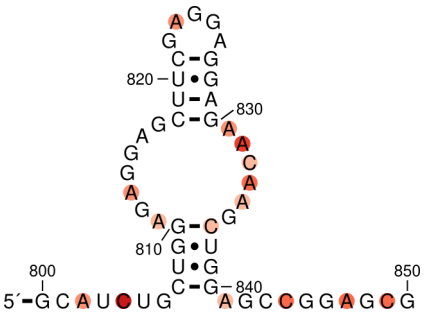
The writing function "r2easyR" writes two R2R input files, a stockholm formatted alignment file (suffix ".sto") and a R2R meta file (suffix ".r2r_meta"). The Stockholm file contains all of the raw data R2R needs to draw a structure. The R2R meta file contains settings for drawing a specific RNA. If your RNA is short or simple, you will likely not need to edit the R2R inputs provided by R2easyR to get a publication quality figure. If your RNA is larger and/or complex, you may need to specify the layout of a few secondary structure elements. There is a layout editor called "r2easyR.stem_editor" built into R2easyR that will provide a nice layout for most RNA (see Section 6). Use "list.files" to see the stockholm and R2R meta file.

```
>list.files()
 [7] "Example.r2r_meta"                "Example.sto"
```

Now you can generate your secondary structure using R2R. First you need to switch to a bash terminal with R2R installed. The terminal provided by RStudio works just fine. In your bash terminal, run:

```
$ cd ~/Path/to/directory
$r2r --disable-usage-warning Example.r2r_meta Example.pdf
```

R2R will print a figure that looks like this:

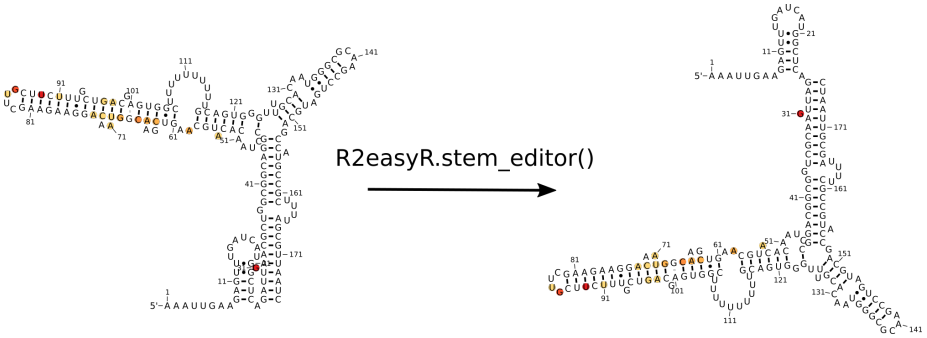


6. Programmatic customization in R2easyR

The default layout provided by R2R may not be perfect, where stems are clashing and falling in on themselves. For large structures, this can look like a pile of spaghetti. R2R has place explicit commands that allow you to customize a layout, but these can be tricky to learn and implement. Likewise, you may want to have different regions of the RNA have different nucleotide colors for a number of reasons. The most common reason is because you are trying to draw the interaction of two different RNA, or parts of an RNA that are very far apart. The easy way to handle this is to add some filler sequence, for example a string of Us, then have R2R color them white so that they disappear. Another common reason is because no data is available for a certain region, and you want it to be grey. This section will introduce a functions that edit the R2R stockholm files to optimize the stem layouts and color large swaths of nucleotides.

6.1 Optimized stem layouts in R2easyR

A common customization to figures produced by R2R is using place explicit commands to flip adjacent helices across the line of the backbone in order to keep helices that are close together from clashing in 2D space. R2easyR contains a built in stem editor called "r2easyR.stem_editor" that will write the place explicit commands for you, resulting in a more optimum layout for most RNA that are less than 200 nucleotides long. As an example, see this depiction of part of the E. coli 16S-rRNA:



The "r2easyR.stem_editor" edits the stockholm file directly, meaning that you should apply the stem editor after you have written R2R inputs. For, example:

```
>list.files()
 [7] "demo.r2r_meta"                "demo.sto"
>r2easyR.stem_editor("demo.sto")
[1] "# STOCKHOLM 1.0"
```

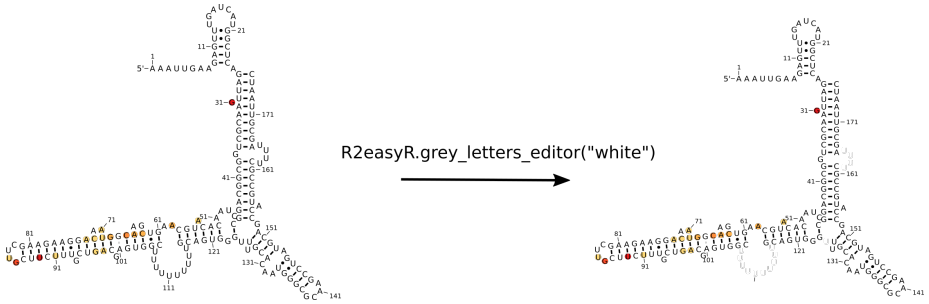
```
 [2] "default\tAAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAUGCAAGUCGAACGGUAACAGGA
 [3] "#=GC SS_cons\t........<<<<<.......>>>>>.<<<<.<<<<<<.<<<<<<<<<....<<<.<<<..<<<..<<<<<..
 [4] "#=GC R2R_LABEL\t.........................A.......................................
 [5] "#=GC cons\tAAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAUGCAAGUCGAACGGUAACAC
 [6] "#=GC conss\t22222222222222222222222222222222222222222222222222222222222222222222222222
 [7] "#=GC cov_SS_cons\t33333333333333333333333333333333333333333333333333333333333333333333333333
 [8] "#=GF R2R circle_nuc #:30 rgb:227,26,28"
 [9] "#=GF R2R circle_nuc #:54 rgb:254,217,118"
[10] "#=GF R2R circle_nuc #:58 rgb:254,178,76"
[11] "#=GF R2R circle_nuc #:62 rgb:254,178,76"
[12] "#=GF R2R circle_nuc #:65 rgb:254,178,76"
[13] "#=GF R2R circle_nuc #:66 rgb:253,141,60"
[14] "#=GF R2R circle_nuc #:68 rgb:254,178,76"
[15] "#=GF R2R circle_nuc #:69 rgb:254,217,118"
[16] "#=GF R2R circle_nuc #:70 rgb:254,217,118"
[17] "#=GF R2R circle_nuc #:72 rgb:254,217,118"
[18] "#=GF R2R circle_nuc #:73 rgb:254,217,118"
[19] "#=GF R2R circle_nuc #:84 rgb:254,217,118"
[20] "#=GF R2R circle_nuc #:85 rgb:252,78,42"
[21] "#=GF R2R circle_nuc #:88 rgb:227,26,28"
[22] "#=GF R2R circle_nuc #:90 rgb:254,217,118"
[23] "#=GF R2R circle_nuc #:96 rgb:254,217,118"
[24] "#=GF R2R circle_nuc #:97 rgb:254,217,118"
[25] "#=GF R2R SetDrawingParam nucShrinkWithCircleNuc 1 pairBondScaleWithCircleNuc 1"
[26] "#=GF R2R tick_label_regular_numbering 1 10 firstNucNum 1"
[27] "#=GF R2R place_explicit A A-- 45 1 0 0 0 90 f"
[28] "//"
```

The stem editor will print the lines for the new stockholm file in your R console. Note the place explicit command that was added on line 27. Rerun R2R on the R2R meta file to get a new figure with an optimized layout.

6.2 Coloring nucleotides in a structure

The "r2easyR.grey_letters_editor" allows you to specify the color for a large swath of nucleotides and runs on a syntax and functionality similar to the "R2easyR.stem_editor". This section will cover two common applications- removing intervening artificial filler regions in a secondary structure and coloring nucleotides where no data is available grey.

It is often informative to map reactivity data from a small portion of a large RNA onto a secondary structure. It is not necessary to draw the whole RNA, just a small subset. In the case of 16S-rRNA, the 5-prime region base pairs to nucleotides 500 residues ustream, and it is unnecessary and difficult to draw the intervening 300 nucleotide region. In this case, I like to replace the intervening region with a short string of Us in the original .csv file to simplify drawing. These Us can be removed by coloring them white using the grey letters editor. For example:



Once again, the grey letters editor is applied directly to the R2R.sto. The second argument specifies what regions should be colored and the third argument specifies the color.

```
r2easyR.grey_letters_editor(R2R.sto = "demo.sto", Nucleotides = c(107:116, 125, 163:166), "w
 [1] "# STOCKHOLM 1.0"
 [2] "default\tAAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAUGCAAGUCGAACGGUAACAGGA
 [3] "#=GC SS_cons\t........<<<<<.......>>>>>.<<<<.<<<<<<.<<<<<<<<<....<<<.<<<..<<<..<<<<<..
 [4] "#=GC R2R_LABEL\t............................A.....................................
 [5] "#=GC cons\tAAAUUGAAGAGUUUGAUCAUGGCUCAGAUUGAACGCUGGCGGCAGGCCUAACACAUGCAAGUCGAACGGUAACAC
 [6] "#=GC conss\t22222222222222222222222222222222222222222222222222222222222222222222222222
 [7] "#=GC cov_SS_cons\t33333333333333333333333333333333333333333333333333333333333333333333333
 [8] "#=GF R2R circle_nuc #:30 rgb:227,26,28"
```

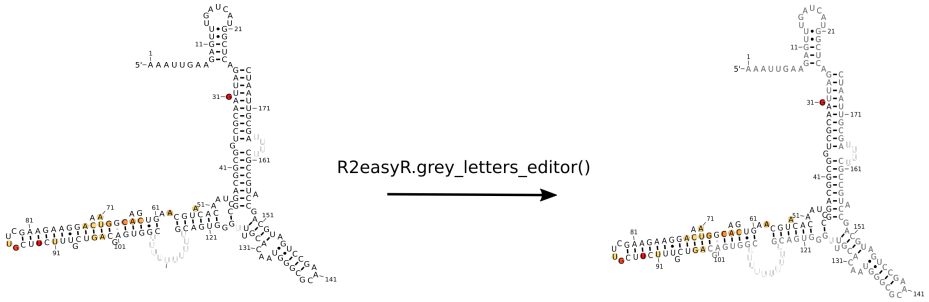
```
 [9] "#=GF R2R circle_nuc #:54 rgb:254,217,118"
[10] "#=GF R2R circle_nuc #:58 rgb:254,178,76"
[11] "#=GF R2R circle_nuc #:62 rgb:254,178,76"
[12] "#=GF R2R circle_nuc #:65 rgb:254,178,76"
[13] "#=GF R2R circle_nuc #:66 rgb:253,141,60"
[14] "#=GF R2R circle_nuc #:68 rgb:254,178,76"
[15] "#=GF R2R circle_nuc #:69 rgb:254,217,118"
[16] "#=GF R2R circle_nuc #:70 rgb:254,217,118"
[17] "#=GF R2R circle_nuc #:72 rgb:254,217,118"
[18] "#=GF R2R circle_nuc #:73 rgb:254,217,118"
[19] "#=GF R2R circle_nuc #:84 rgb:254,217,118"
[20] "#=GF R2R circle_nuc #:85 rgb:252,78,42"
[21] "#=GF R2R circle_nuc #:88 rgb:227,26,28"
[22] "#=GF R2R circle_nuc #:90 rgb:254,217,118"
[23] "#=GF R2R circle_nuc #:96 rgb:254,217,118"
[24] "#=GF R2R circle_nuc #:97 rgb:254,217,118"
[25] "#=GF R2R SetDrawingParam nucShrinkWithCircleNuc 1 pairBondScaleWithCircleNuc 1"
[26] "#=GF R2R tick_label_regular_numbering 1 10 firstNucNum 1"
[27] "#=GF R2R place_explicit A A-- 45 1 0 0 0 90 f"
[28] "#=GF R2R nuc_color #:106 rgb:255,255,255"
[29] "#=GF R2R nuc_color #:107 rgb:255,255,255"
[30] "#=GF R2R nuc_color #:108 rgb:255,255,255"
[31] "#=GF R2R nuc_color #:109 rgb:255,255,255"
[32] "#=GF R2R nuc_color #:110 rgb:255,255,255"
[33] "#=GF R2R nuc_color #:111 rgb:255,255,255"
[34] "#=GF R2R nuc_color #:112 rgb:255,255,255"
[35] "#=GF R2R nuc_color #:113 rgb:255,255,255"
[36] "#=GF R2R nuc_color #:114 rgb:255,255,255"
[37] "#=GF R2R nuc_color #:115 rgb:255,255,255"
[38] "#=GF R2R nuc_color #:124 rgb:255,255,255"
[39] "#=GF R2R nuc_color #:162 rgb:255,255,255"
[40] "#=GF R2R nuc_color #:163 rgb:255,255,255"
[41] "#=GF R2R nuc_color #:164 rgb:255,255,255"
[42] "#=GF R2R nuc_color #:165 rgb:255,255,255"
[43] "//"
```

The grey letters editor will print the lines for the new stockholm file in your R console. Note that lines 28 to 42, are showing that the specified nucleotides will be colored white (RGB 225, 225, 225).

Likewise, regions where no data is available can be set to grey using the default settings of the grey letters editor:

```
>r2easyR.grey_letters_editor(R2R.sto = "demo.sto", Nucleotides =c(1:30, 99:106, 117:124, 126
```

This will have the following effect:



Minor improvements to this final figure can be achieved by work up in a drawing program like Adobe Illustrator.

7. Drawing pseudoknots

7.1 Removing then annotating a pseudoknot manually

Drawing a pseudoknot is a common problem for depicting secondary structure. A good strategy is to draw the secondary structure of the RNA in the absence of pseudoknots and label those pseudoknots after the core R2R.sto file is written. R2easyR contains a function called "r2easyR.pknot_drawer" which will edit the R2R.sto file to label certain sequences as a pseudoknot.

For example, first read in a pseudoknotted secondary structure as a text file with CT or dotbracket formatted secondary structure information as you would for a normal secondary structure. Then find your pseudoknotted sequence. In the case of our sample RNA, the pseudoknot occurs at nucleotides 14 to 18 and nucleotides 64 to 68.

```
>df[c(14:18, 64:68),]
   N Nucleotide N-1 N+1 BP  N Dotbracket
14 14          A  13  15 68 14          <
15 15          G  14  16 67 15          <
16 16          C  15  17 66 16          <
17 17          C  16  18 65 17          <
18 18          C  17  19 64 18          <
64 64          G  63  65 18 64          >
65 65          G  64  66 17 65          >
66 66          G  65  67 16 66          >
67 67          C  66  68 15 67          >
68 68          U  67   0 14 68          >
```

Remove the pseudoknot by changing the Dotbracket column so that no secondary structure is specified at the pseudoknot sequence.

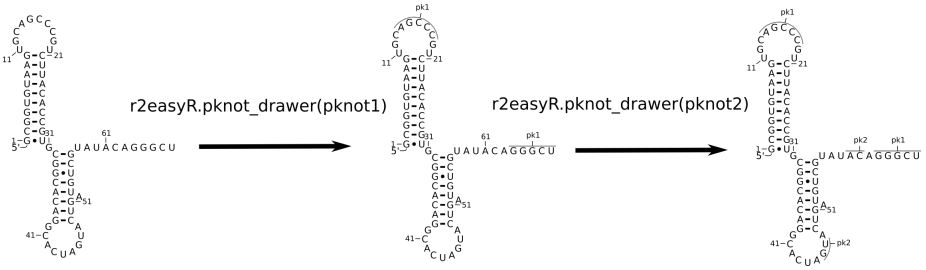
```
>pknot <- c(14:18, 64:68)
>df$Dotbracket[pknot] <- "."
>df[c(14:18, 64:68),]
    N Nucleotide N-1 N+1 BP  N Dotbracket
14 14          A  13  15 68 14          .
15 15          G  14  16 67 15          .
16 16          C  15  17 66 16          .
17 17          C  16  18 65 17          .
18 18          C  17  19 64 18          .
64 64          G  63  65 18 64          .
65 65          G  64  66 17 65          .
66 66          G  65  67 16 66          .
67 67          C  66  68 15 67          .
68 68          U  67   0 14 68          .
```

The R2R input files can be written and edited normally.

```
>r2easyR.write("demo", df)
>r2easyR.stem_editor("demo.sto")
```

Running R2R on the inputs you just generated will result in a secondary structure with no pseudoknot specified. You can add the pseudoknot label with the pseudoknot drawer, which edits the R2R.sto file directly. The stem editor will print the lines it edited in the console.

```
>r2easyR.pknot_drawer(pknot = pknot,  R2R.sto = "demo.sto")
[1] "#=GC R2R_LABEL\t.............aaaaa.............A.......................N......bbbbb"
[1] "#=GC R2R_XLABEL_pk\t..............a.............................................b.
[2] "#=GF R2R outline_nuc a"
[3] "#=GF R2R outline_nuc a"
[4] "#=GF R2R tick_label pk:a pk1"
[5] "#=GF R2R outline_nuc b"
[6] "#=GF R2R outline_nuc b"
[7] "#=GF R2R tick_label pk:b pk1"
```



You can add a second pseudoknot to the drawing by rerunning the pseudoknot labeler on a new sequence.

```
>r2easyR.pknot_drawer(pknot = c(46:47, 61:62),  R2R.sto = "demo.sto")
[1] "#=GC R2R_LABEL\t.............aaaaa.............A.............cc.........N...dd.bbbbb"
[1] "#=GC R2R_XLABEL_pk\t.............................................c.............d.....
[2] "#=GF R2R outline_nuc c"
[3] "#=GF R2R outline_nuc c"
[4] "#=GF R2R tick_label pk:c pk2"
[5] "#=GF R2R outline_nuc d"
[6] "#=GF R2R outline_nuc d"
[7] "#=GF R2R tick_label pk:d pk2"
```

Note: the pseudoknot labeler will count how many other pseudoknots have been added by the labler and automatically update the index. For example the second pseudoknot we labeled is called pk2.

7.2 Using an automated pk finder to draw pseodoknots from a connectivity table (CT)

Pseudoknots are troublesome for structure drawing programs because their non-nested nature fools the drawing program into pairing the incorrect bases. One way to deal with this is to eliminate pseudoknots in the dot bracket specified secondary structure line, as outlined in section 7.1. Alternatively, Peter C. Forstmeier has written some creative code to identify pseudoknotted residues from a connectivity table (CT) and compiled it into the "r2easyR.pk_finder" algorithm. The pk finder analyzes a CT and identifies nests of base pairs that are non-nested (pseudoknotted) in relation to the dominant (longest) secondary structure elements. To implement the pk finder, first read the CT into a R dataframe using "read.ct" for RNAStructure formatted connectivity files or generic R text file parsers like read.ct. The data frame should have the following column names in order for the pk finder to recognize the data structure:

```
>df <- read.ct("PKB335(1).ct")
>head(df)
  N Nucleotide N-1 N+1 BP N
1 1          G   0   2 36 1
2 2          G   1   3 35 2
3 3          U   2   4 34 3
4 4          U   3   5 33 4
5 5          U   4   6 32 5
6 6          G   5   7 31 6
```

Then you can use the pk finder to identify pknots from this connectivity table.

```
>list.pk <- r2easyR.pk_finder(df)
[[1]]
[1]  10  11 103 104

[[2]]
[1]  13  14  15  16  99 100 101 102

[[3]]
[1] 21 22 37 38
```

The pk finder found 3 pseudoknots in this connectivity table. Note the output of the pk finder is a list of R objects that can be passed to other functions in R2easyR. The first element is a data frame with the pknots removed from the dot bracket column and can be passed to "r2easyR.write". The second element is a list of pseudoknotted residues and can be passed to "r2easyR.pknot_drawer". To see what is in the list, run:

```
>list.pk
$r2easyR.dataframe
      N Nucleotide N-1 N+1  BP   N Dotbracket
1     1          G   0   2  36   1          <
2     2          G   1   3  35   2          <
3     3          U   2   4  34   3          <
4     4          U   3   5  33   4          <
5     5          U   4   6  32   5          <
6     6          G   5   7  31   6          <
7     7          C   6   8  30   7          <
8     8          U   7   9   0   8          .
9     9          U   8  10   0   9          .
10   10          G   9  11 104  10          .
11   11          U  10  12 103  11          .
12   12          U  11  13   0  12          .
13   13          G  12  14 102  13          .

$pknot.list
$pknot.list[[1]]
[1]  10  11 103 104

$pknot.list[[2]]
[1]  13  14  15  16  99 100 101 102

$pknot.list[[3]]
```

```
[1] 21 22 37 38
```

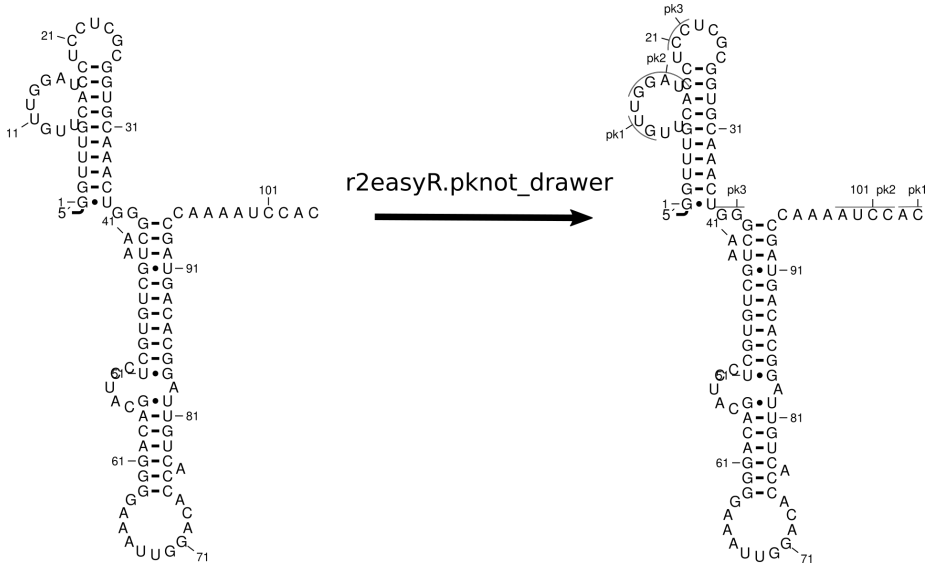
The R2R inputs can be written using the data frame ($r2easyR.dataframe), then R2R inputs can be edited normally.

```
>r2easyR.write("demo", list.pk$r2easyR.dataframe)
>r2easyR.stem_editor("demo.sto")
```

Then pknot IDs can be passed to the "r2easyR.pknot_drawer".

```
>r2easyR.pknot_drawer("demo.sto", list.pk$pknot.list)
```

Output R2R meta and Stockholm files can be passed to r2r normally. The result of each step is shown here:



8. Patching R2R to remove black lines around reactivity circles

The current version of R2R draws black lines around nucleotide reactivity circles, which do not look crisp. The creator of R2R, Zasha Weinburg, has provided a quick patch to remove the lines around circles.

Step 1 In your bash terminal, use "cd" and "ls" to Navigate to the "R2R-1.0.6" folder

Step 2 Open the RnaDrawer.cpp script in the src folder using the vim text editor

```
$vim R2R-1.0.6/src/RnaDrawer.cpp
```

Step 3 Jump to line 1595 in vim

```
1595 [shift] + g
```

Step 4 Press [i] to go into editing mode. Delete the lines:

```
pdf.SetLineWidth(posInfoVector[i].circleNucPenWidth);
pdf.EdgeCircle(AdobeGraphics::Color_Black(),posInfoVector[i].pos,radius);
```

Step 5 Press [Esc] to leave editing mode

Step 6 Save the changes and quit vim by typing:

```
:wq
```

Step 7 Remake R2R and Reinstall R2R

```
$make
$make install
```

I will edit R2easyR to custom circle lines when a new version of R2R is released that supports this functionality. Until then, this patch will do.