Magic Pie (mgcpy)

Team Captain: Sambit Panda

Sandhya Ramachandran, Bear Xiong, Richard Guo, Satish Palaniappan, Ananya Swaminathan

Date: 9/12/2018

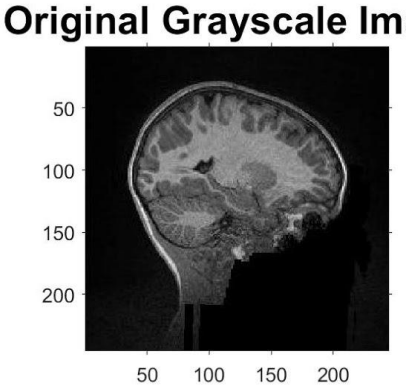
Sprint 1: Create *mgcpy* package

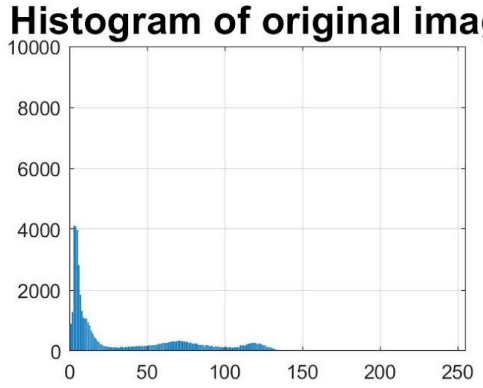
- Task 1: Implement dHSIC, HHG, Pearson, Spearman, and Mic into package (Sambit)
- Task 2: Implement MGC into the package (Satish)
- Task 3: Implement MDMR and FastMGC into package (Sandhya)
- Task 4: Implement MCORR, DCORR, and Mantel into package (Bear)
- Task 5: Implement Random Forest independence tests into package (Richard)
- Task 6: Implement 2- sample tests into package (Ananya)

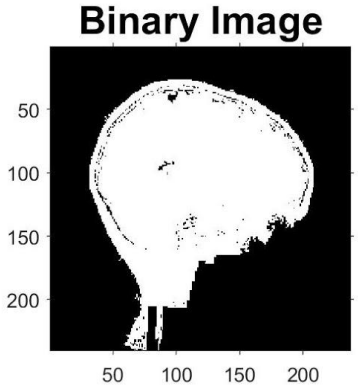
Visualization: Skull Stripping

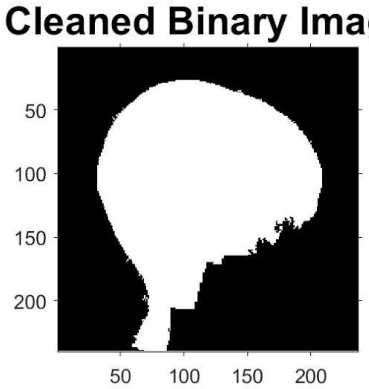
- Attempted Skull stripping of sub-NDARAA075AMK_T1w.nii.gz
- code source, edited some variables for best result
- Creates binary image, then erodes

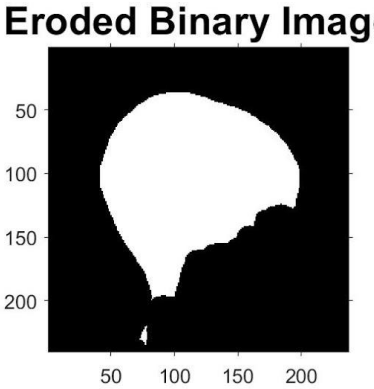
```
% Erode away 10 layers of pixels.
se = strel('disk', 10, 0);
binaryImage = imerode(binaryImage, se);
```

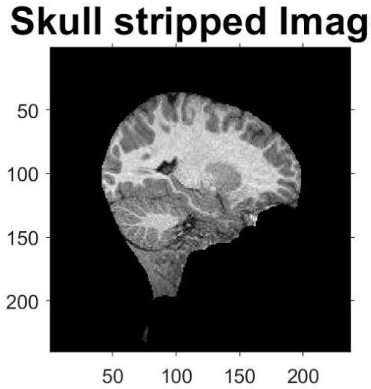












- Trade off: some edge of brain lost, but all skull gone

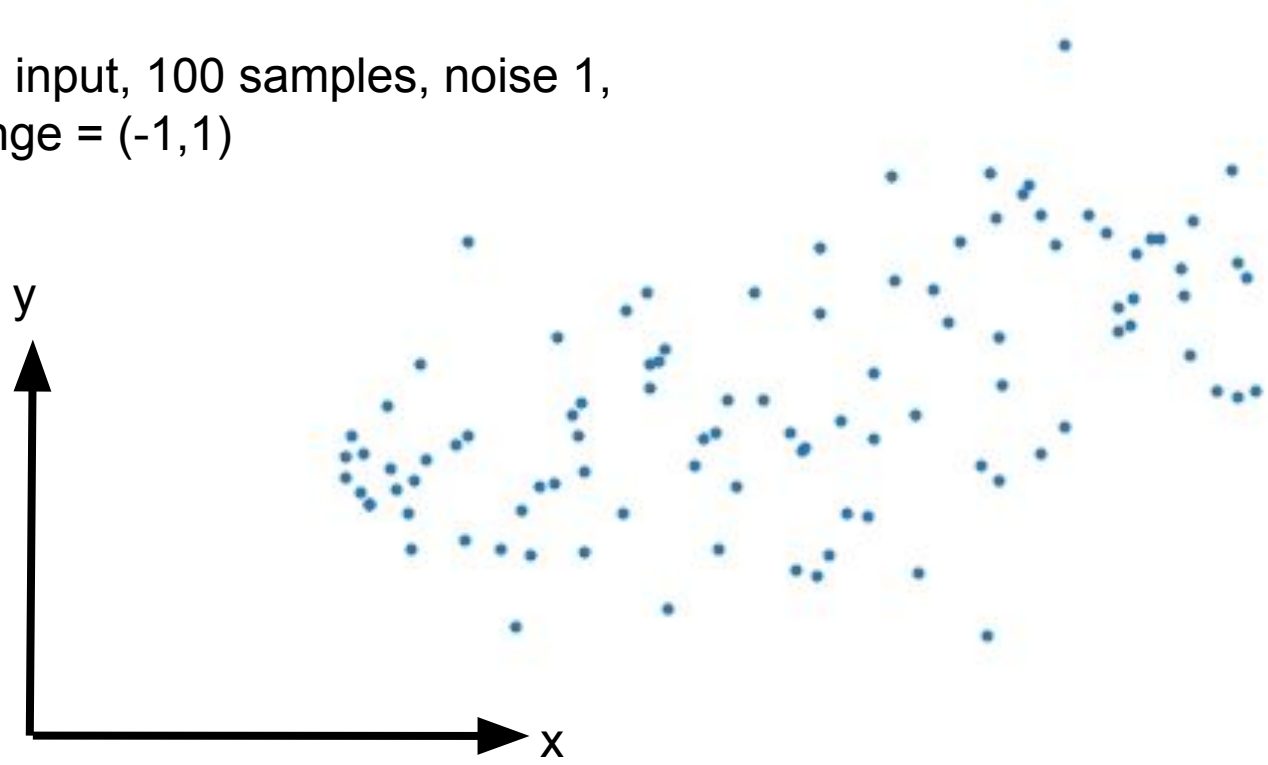
Task 1 (Sambit): Implement dHSIC, HHG, Pearson, Spearman, and Mic into package

Last Week's Accomplishments:

- More robust testing on higher dimensional data sets for rv_corr
- Begin working on dHSIC function (little bit more involved)
- Finalize PR with rv_corr function and merge with development branch with full coverage
- Make rv_corr compliant with documentation guidelines
- Begin generating linear, cubic, exponential etc. for testing by our functions - started with linear

Linear Distribution Plot

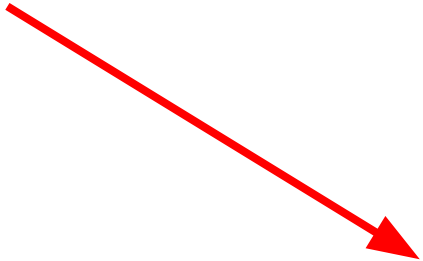
1D input, 100 samples, noise 1,
range = (-1,1)



In response to last week's question

- After looking through scipy.stats.pearsonr documentation and debugging through the function, it appears that it only works on 1D arrays while rv_corr needs to work on multi-dimensional so that it can be used in Dcorr, Mantel, and Mcorr when calculating powers of the tests

```
# Presumably, if abs(r) > 1, then it is only some small artifact of
# floating point arithmetic.
r = max(min(r, 1.0), -1.0)
df = n - 2
if abs(r) == 1.0:
    prob = 0.0
```



```
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

Stuff to do this week

Since it’s a shorter week:

- Finish creating linear, cubic, etc. distributions to help test future data
- Submit PR for simulations

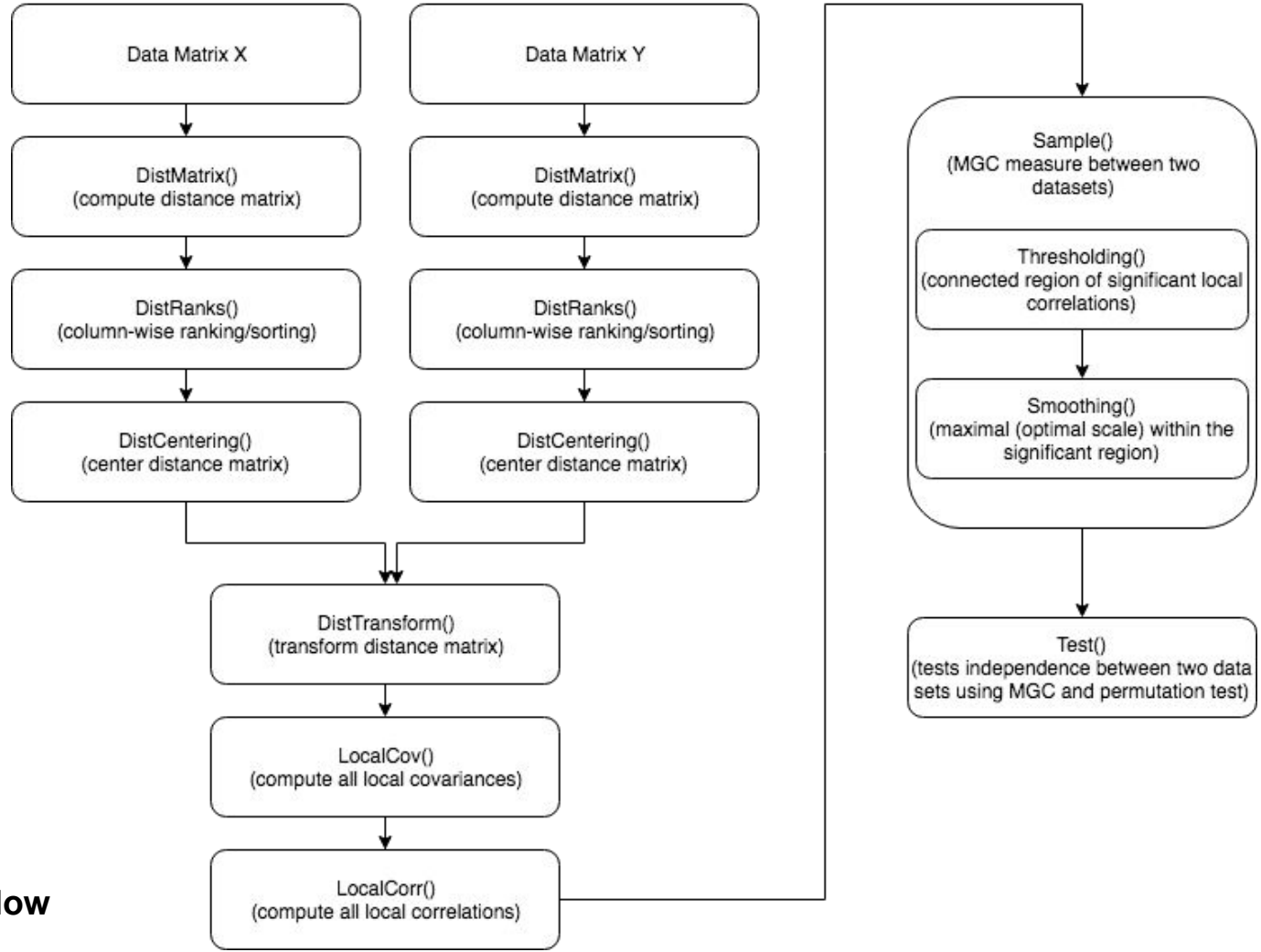
If there is more time:

- Test rv_corr with created distributions
- Figure out if I can integrate pearsonr from scipy.stats into rv_corr

Task 2 (Satish): Implement MGC into the package

Last Week Accomplishments:

- Study the pseudo code in the MGC paper and the R code, and list down all the functions and its dependencies required to port to Python
- Create stubbed versions of all the above functions and add proper docstrings to define the inputs and outputs
- Implement one of the above function with tests in Python and raise a PR
  - Implemented DistCentering and mgc.distTransform methods from R with tests



Algorithm Flow

draw.io

- **Dependencies**
  - Core Python package dependencies
    - NumPy
    - SciPy
  - Other Python package dependencies
    - Pytest
    - Pep8
- **Caveats while porting from R**
  - Array indices run from 0 to n-1 in Python, while in R it runs from 1 to n
  - A dot product between a 1D matrix() of size N and t(matrix) of size M in R automatically broadcasts to a N x M 2D matrix
  - Use np.sqrt for square root of -ve numbers

Example:

```
np.sqrt(local_variance_A.reshape(-1, 1) @ local_variance_B.reshape(-1, 1).T).real
```

```r
DistCentering<-function(X,option,optionRk){
  n=nrow(X)
  if (optionRk!=0){
    RX=DistRanks(X) # the column ranks for X
  } else {
    RX=matrix(0,n,n)
  }

  if (option=='rank'){
    X=RX
  }
  # Default mgc transform
  EX=t(matrix(rep(colMeans(X)*n/(n-1),n), ncol = n))

  if (option=='dcor'){ # unbiased dcor transform
    EX=t(matrix(rep(colMeans(X)*n/(n-2),n), ncol = n))
     +matrix(rep(rowMeans(X)*n/(n-2),n), ncol = n)
      -sum(X)/(n-1)/(n-2)
    EX=EX+X/n
  }
  if (option=='mantel'){ # mantel transform
    EX=sum(X)/n/(n-1)
  }

  A=X-EX

  # The diagonal entries are always excluded
  for (j in (1:n)){
    A[j,j]=0
  }

  result=list(A=A, RX=RX)
  return(result)
}
```

```python
def center_distance_matrix(distance_matrix, base_global_correlation="mgc", is_ranked=True):

    n = distance_matrix.shape[0]
    ranked_distance_matrix = np.zeros(distance_matrix.shape)

    if is_ranked:
        ranked_distance_matrix = rank_distance_matrix(distance_matrix)

    if base_global_correlation == "rank":
        distance_matrix = ranked_distance_matrix

    # 'mgc' distance transform (col-wise mean) - default
    expected_distance_matrix = np.repeat(
        ((distance_matrix.mean(axis=0) * n) / (n-1)), n).reshape(-1, n).T

    # unbiased version of dcor distance transform (col-wise mean + row-wise mean - mean)
    if base_global_correlation == "dcor":
        expected_distance_matrix = np.repeat(((distance_matrix.mean(axis=0) * n) / (n-2)), n).reshape(-1, n).T \
                                   + np.repeat(((distance_matrix.mean(axis=1) * n) / (n-2)), n).reshape(-1, n) \
                                   - (distance_matrix.sum() / ((n-1) * (n-2)))
        expected_distance_matrix += distance_matrix / n

    # mantel distance transform
    elif base_global_correlation == "mantel":
        expected_distance_matrix = distance_matrix.sum() / (n * (n-1))

    centered_distance_matrix = distance_matrix - expected_distance_matrix

    # the diagonal entries are always excluded
    np.fill_diagonal(centered_distance_matrix, 0)

    return {"centered_distance_matrix": centered_distance_matrix,
            "ranked_distance_matrix": ranked_distance_matrix}
```

Stuff to do this week

- Implement *MGCLocalCorr* module with *LocalCov* (local covariances computations) and *mgc.localcorr* (local correlations computations) functions from R in Python, with tests and raise PR
- Merge the DistTransform PR to development.
- Write *setup.py,* for making the “mgcpy” package easy to install

Task 3 (Sandhya): Implement MDMR and FastMGC into package

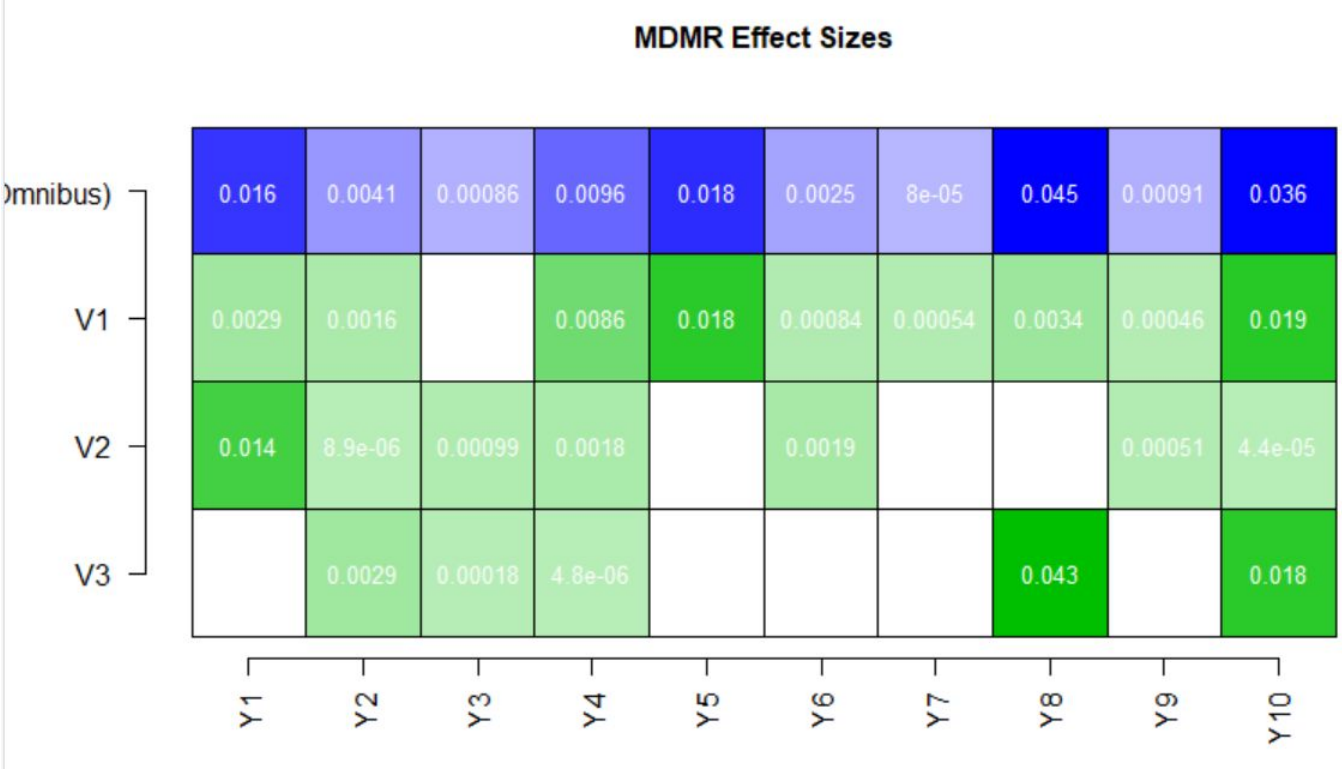
Last Week's Accomplishments:

- Convert remaining MDMR code to python: link
- Contact author of MDMR code for help
- If successful, run MDMR on spiral data, linear data
  - Need clarification on columns input
- If successful, convert MDMR code to our coding structure/documentation
- Ran MDMR on sample data in R: link

```r
library(MDMR)
vignette("mdmr-vignette")

# Load data
data(mdmrdata)

# Compute distance matrix
D <- dist(Y.mdmr, method = "euclidean")

# Conduct MDMR
mdmr.res <- mdmr(X = X.mdmr, D = D)

# Check results
summary(mdmr.res)
```

```
           Statistic Numer.DF Pseudo.R2 Analytic.p.value
(Omnibus)     0.1602         3    0.1380           < 1e-20 ***
X1            0.0652         1    0.0562           < 1e-14 ***
X2            0.0242         1    0.0208       5.2361e-07 ***
X3            0.0751         1    0.0648           < 1e-14 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1
```

```
15
16  # Study univariate effect sizes
17  delta.res <- delta(X.mdmr, Y = Y.mdmr, dtype = "euclidean",
18                     niter = 10, plot.res = T)
```



Next Week:

- Test python version on same dataset, recreate results
- Reach: Work with Anibal's commented code to set default values in python code for inputs other than X and D

```
def mdmr(D, X, columns, permutations):
```

Collective Team Task:

Last week: Completed the AWS proposal and submitted for AWS research credits

This week: Visualization of data, Make more PRs in the repo