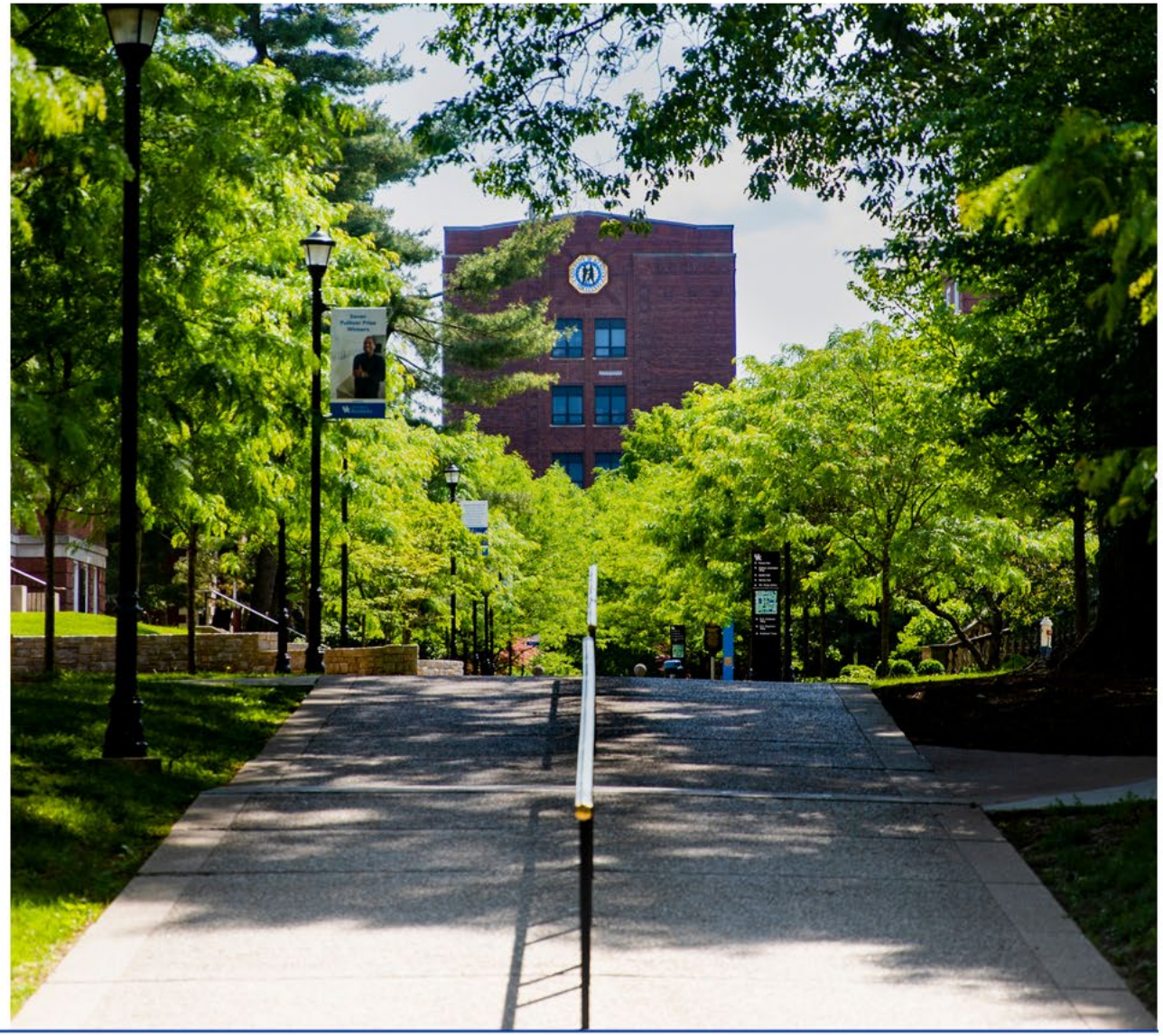
An Introduction to Generative AI in Biomedical Applications, Part 2: Using the RGAN Package in R

Anthony A. Mangino, PhD

UK AI/ML Hub Seminar Series

November 7, 2024



University of Kentucky

Outline

- What are Generative Adversarial Networks (GANs)? (Brief Recap)
- Clinical Use Case: Diagnosing Takotsubo Syndrome (Brief Recap)
- Introducing the RGAN Package and Exemplar Dataset Generation
- Closing Remarks

Acknowledgments

Taha Ahmed, MD

Vincent Sorrell, MD

Samra Haroon Lodhi, MD

Vedant Gupta, MD

Steven Leung, MD

W. Holmes Finch, PhD

Maria Hernandez Finch, PhD

Kendall Smith, MS

Axiom: As statisticians, we are also ethicists.

What are Generative Adversarial Networks (GANs)?

I might not call this AI, but you might. I am very much an AI skeptic. Caveat emptor.

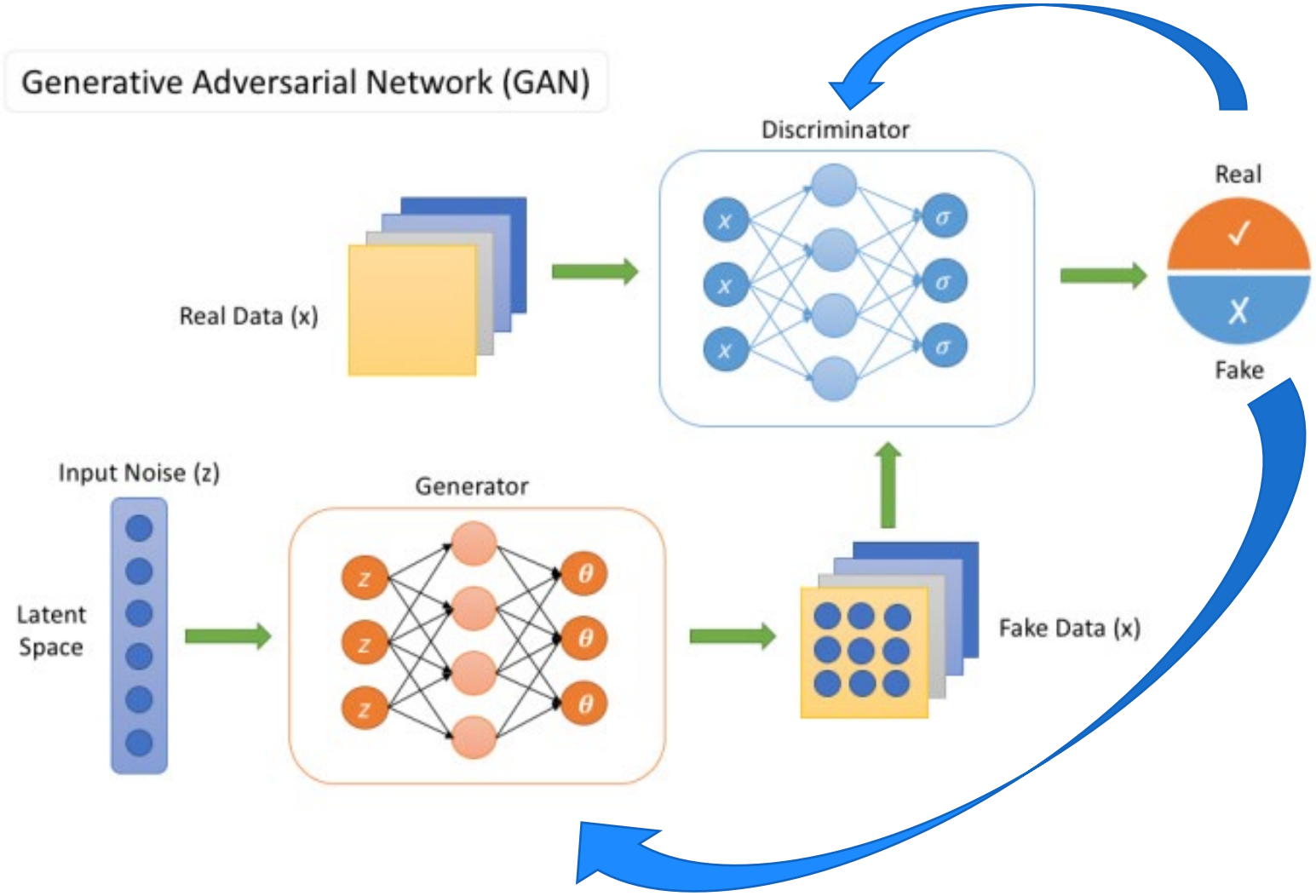
What are Generative Adversarial Networks?

Generative Adversarial Networks (GANs) were created (or at least published!) in 2014 by Ian Goodfellow while he was a student at Université de Montréal (he subsequently worked at Google Brain).

The objective of GANs is to create synthetic data that look and behave like real/source data.

Originally created for use with image data, GANs have a variety of possible architectures that are relevant to and useful for image, video, audio, and tabular data.

They can also be used in reinforcement learning or computer vision tasks.



"We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 – D(G(z)))."

Goodfellow et al., 2014

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Diagram credited to Abhishek Verma: https://www.linkedin.com/pulse/generative-adversarial-network-abhishek-verma/

Why do we care about creating *fake* data?

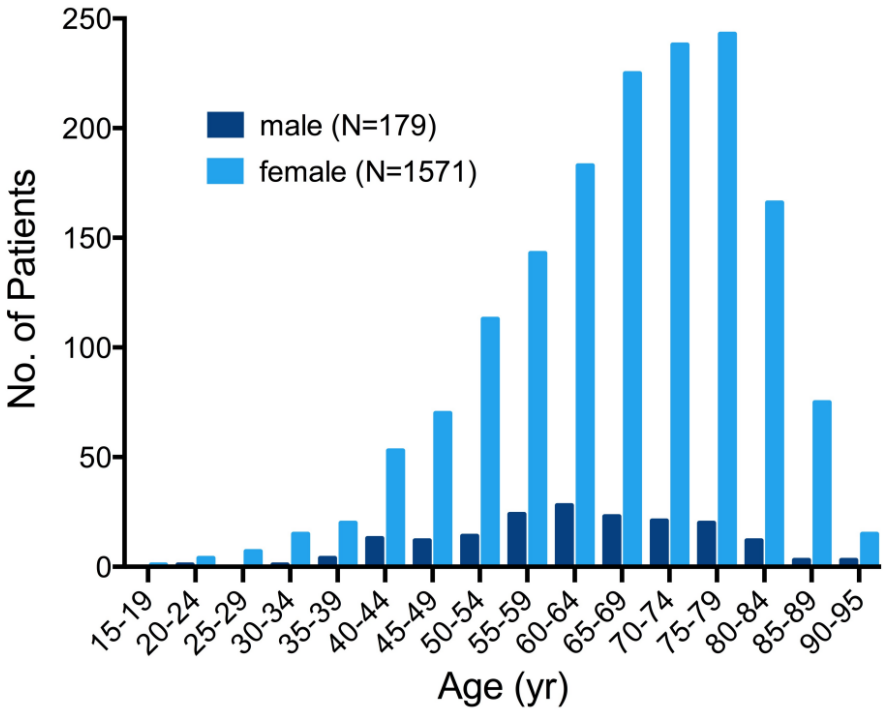
The “vanilla” GAN is a comparatively simple architecture, but because GANs are so versatile, they can be used in numerous contexts including:

- Augmenting small datasets
- Providing new datasets for internal replication/assessing stability of secondary models
- Imputing missing data*
- Creating new cases for human evaluators in training or external validation/generalization
- Style transfer (e.g., ‘create an image of a painting in the *style* of…’)

Clinical Use Case: Diagnosing Takotsubo Syndrome

Takotsubo Syndrome

- **Takotsubo Syndrome (TTS; a.k.a. 'broken heart syndrome')** is a relatively rare and **reversible** condition with symptoms that mimic specific clinical presentation of **left anterior descending acute coronary syndrome** (LAD-ACS):
  - severe pressure and/or pain in chest,
  - shortness of breath,
  - sudden onset fatigue,
  - cold sweats,
  - lightheadedness.
- Rarely reported prior to the early 2000s.
- Often preceded by great emotional and/or physical stress.
- TTS is far more prevalent in women than men.

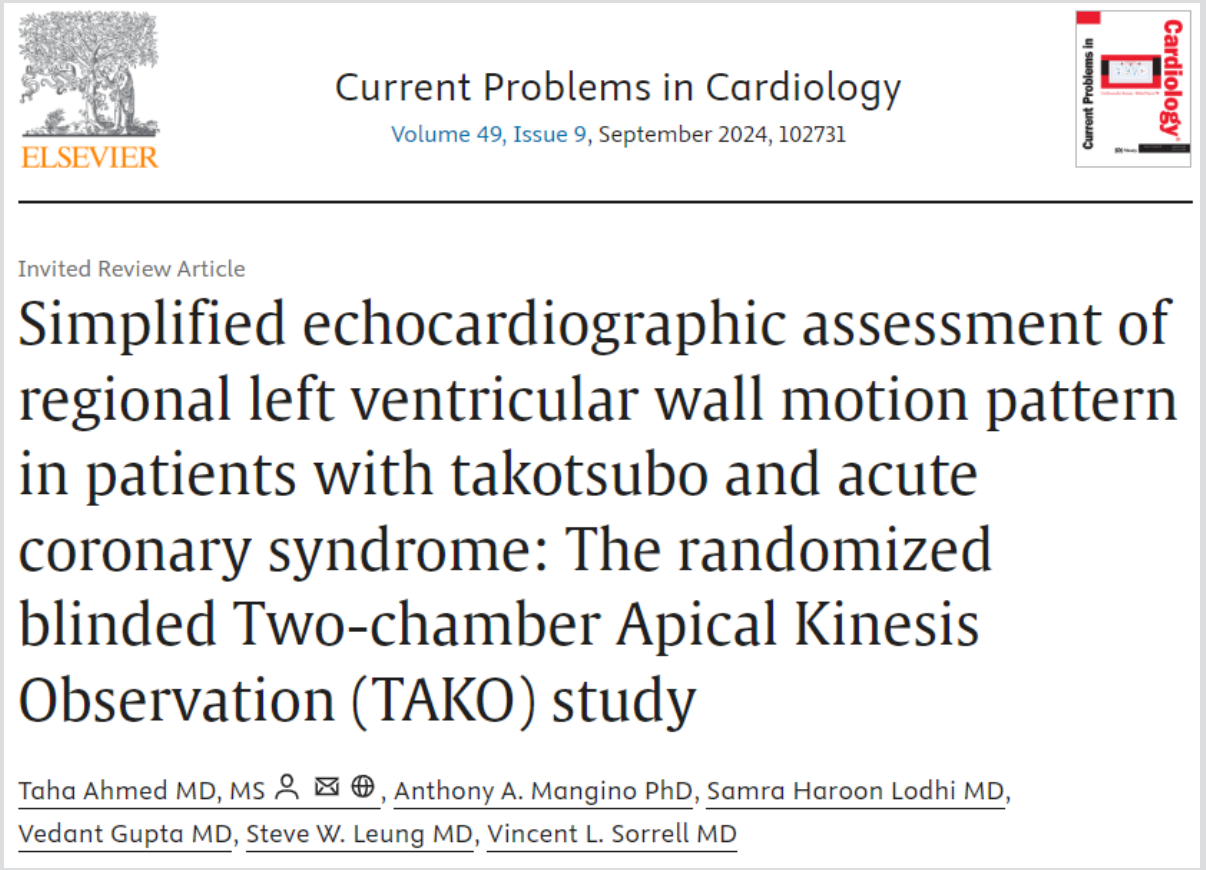


Sharkey et al., (2018)
Templin et al., (2015).

Our Original Study

- Our goal was to assess the capability of an **echocardiogram** to provide sufficient information for a clinician to diagnose TTS compared to ACS **in the absence of the conventional coronary angiography**.

- **N = 102 patients** (complete cases) fulfilling the Mayo Clinic criteria (Madhavan & Prasad, 2010) for TTS presenting to University of Kentucky Healthcare hospitals between 2011 and 2021.



ELSEVIER

Current Problems in Cardiology

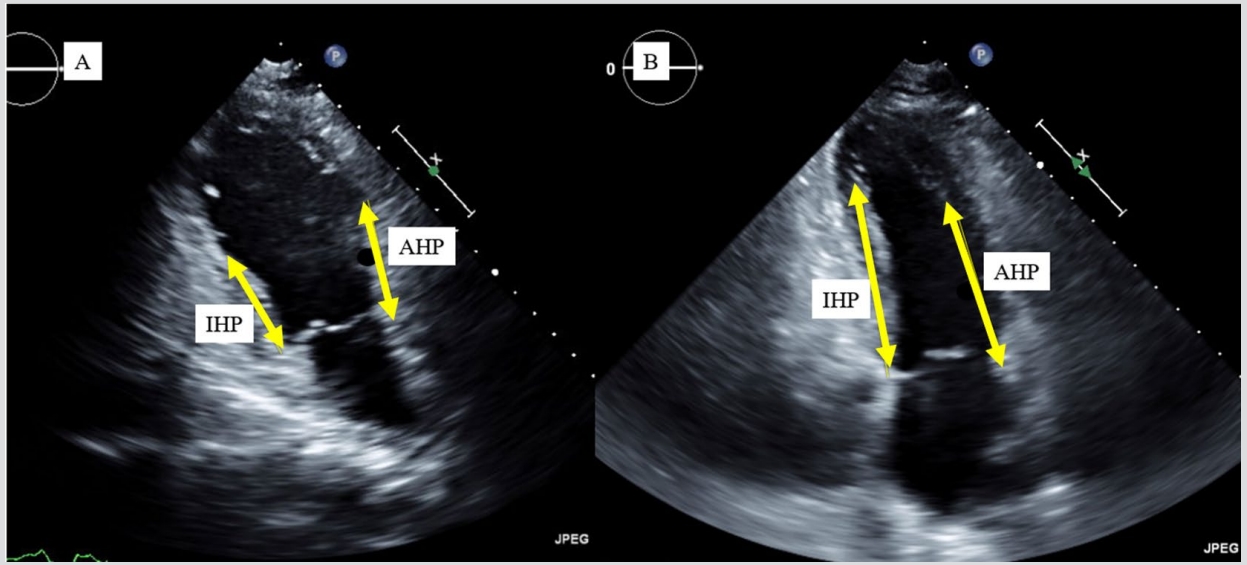
Volume 49, Issue 9, September 2024, 102731

Invited Review Article

Simplified echocardiographic assessment of regional left ventricular wall motion pattern in patients with takotsubo and acute coronary syndrome: The randomized blinded Two-chamber Apical Kinesis Observation (TAKO) study

Taha Ahmed MD, MS, Anthony A. Mangino PhD, Samra Haroon Lodhi MD, Vedant Gupta MD, Steve W. Leung MD, Vincent L. Sorrell MD

Our Original Study



- Echocardiogram used apical 2-chamber view to assess anterior and inferior wall segments to hinge points.
- Because raw hinge point measurements would be affected by patient sex, the **ratio** of the measurements was used.
- **Ratio of anterior to inferior hinge points** was hypothesized to generally be:
  - Near or greater than 1 in TTS patients
  - Less than 1 in ACS patients

Our Original Study: Model Derivation Cohort Results

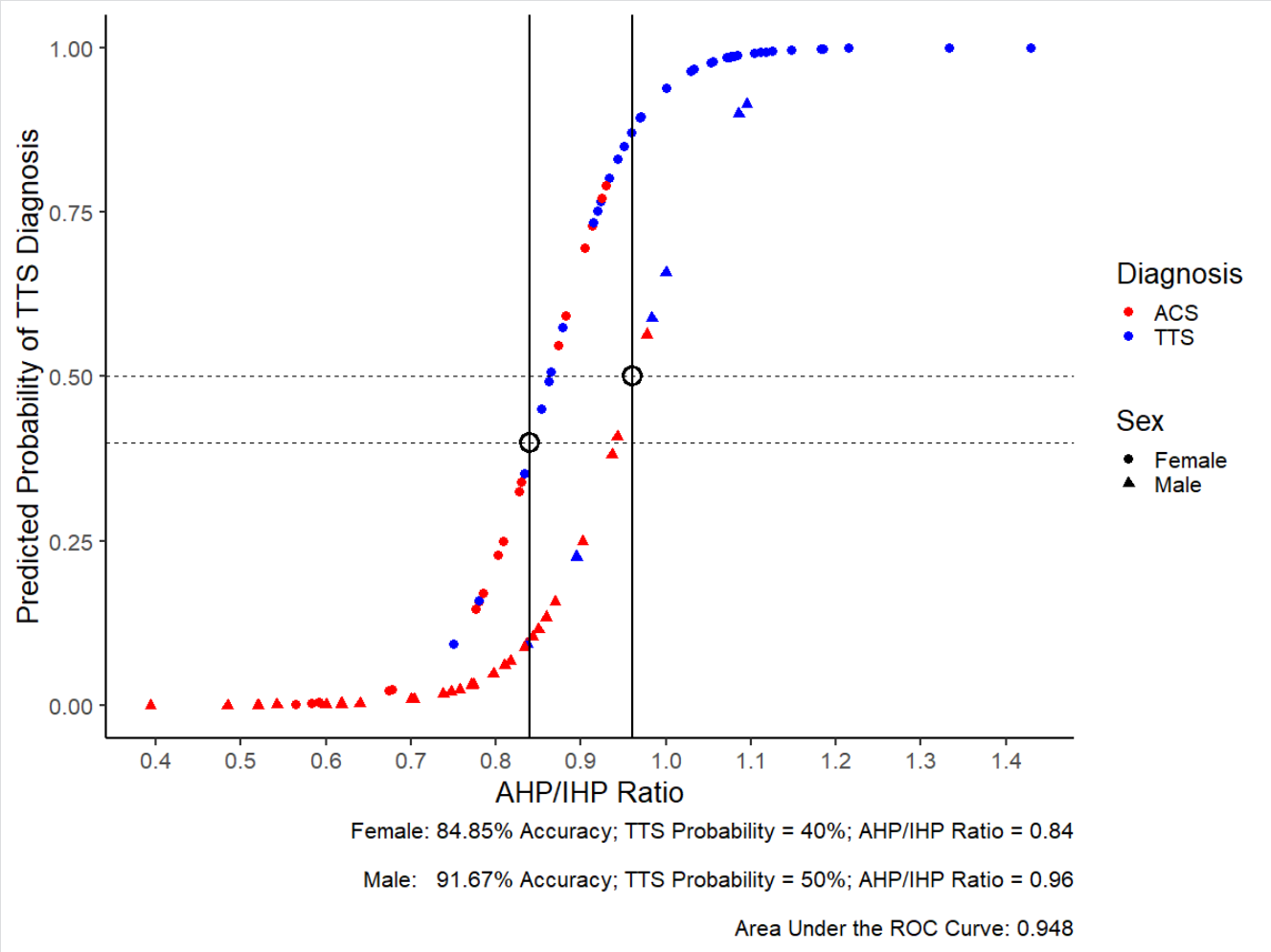
- **Logistic regression model** was fit with the following specification:

*diagnosis ~ AHP/IHP Ratio * Sex*

Sensitivity = Correct TTS Diagnosis

Specificity = Correct ACS Diagnosis

	ACS	TTS
Female	20	46
Male	30	6



Synthetic Training Cohort

Because TTS is a) a rare phenomenon, and b) a sex-imbalanced diagnosis, we used a GAN to **create a larger synthetic training sample**.

The GAN was fit with a batch size of 50 across 1000 epochs with a Wasserstein value/loss function to yield 1020 cases.

The GAN was fit using the RGAN package in R (Neunhoeffer, 2022).

Significant differences between datasets found only for patient sex (p = 0.006).

	Source Data	GAN-Generated Data
Female, n (%)	66 (64.7%)	510 (50.0%)
Male, n (%)	36 (35.3%)	510 (50.0%)
Age, M (s)	59.6 (12.7)	58.8 (6.09
Inferior Hinge Point, M (s)	4.71 (1.47)	4.58 (1.95)
Anterior Hinge Point, M (s)	4.02 (0.995)	4.10 (1.28)
AHP/IHP Ratio, M (s)	0.891 (0.184)	0.889 (0.215)
ACS Diagnosis, n (%)	50 (49.0%)	510 (50.0%)
TTS Diagnosis, n (%)	52 (51.0%)	510 (50.0%)
Male TTS Cases, n (%)	6 (5.9%)	23 (2.2%)

Synthetic Cohort: Concordance with Source Training Cohort

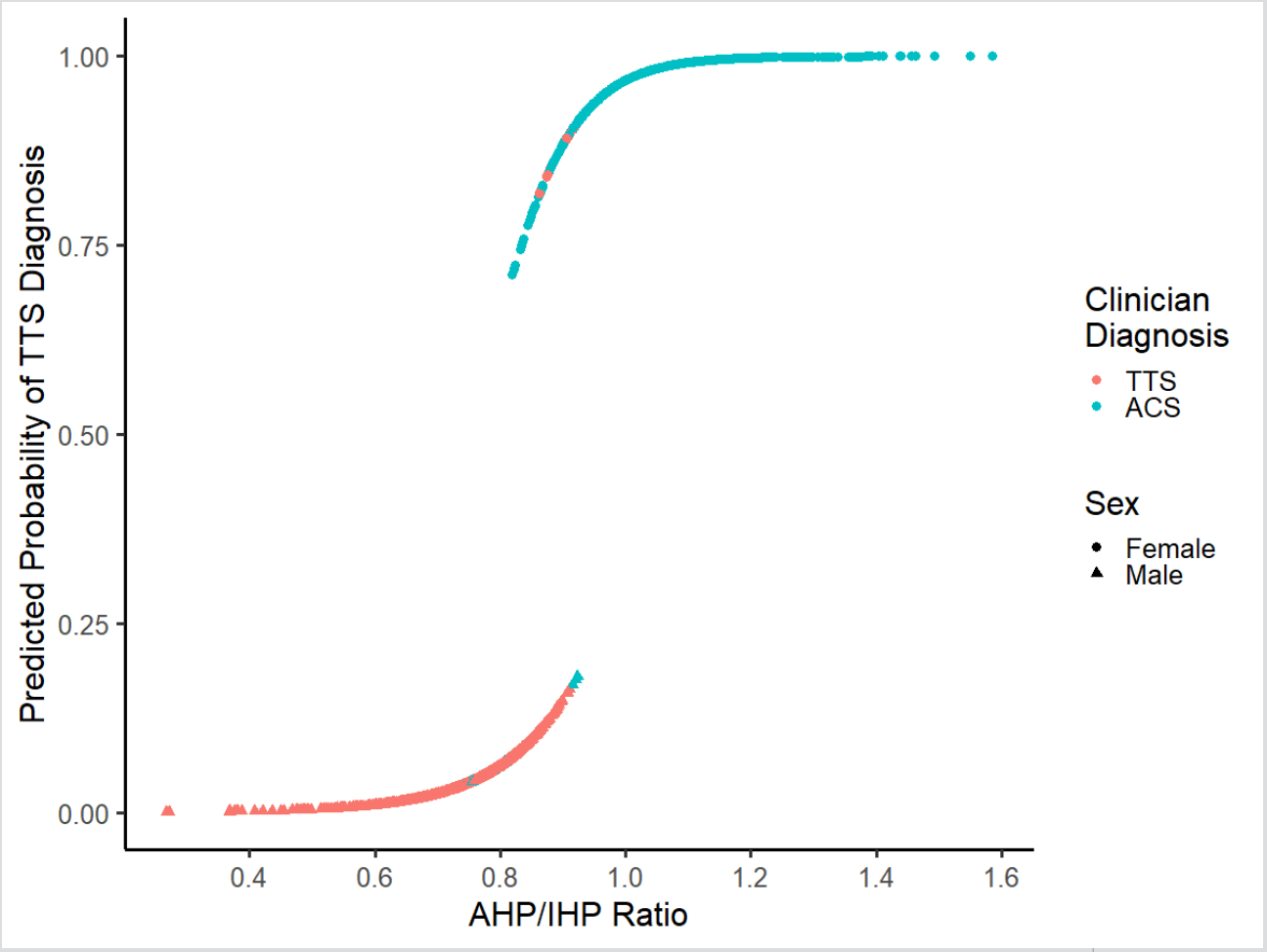
Training Set		Source	Synthetic
Source Validation Set	Training		X
	Validation		

We fit a logistic regression model with the same specification, but with **only the 1020 synthetic cases**.

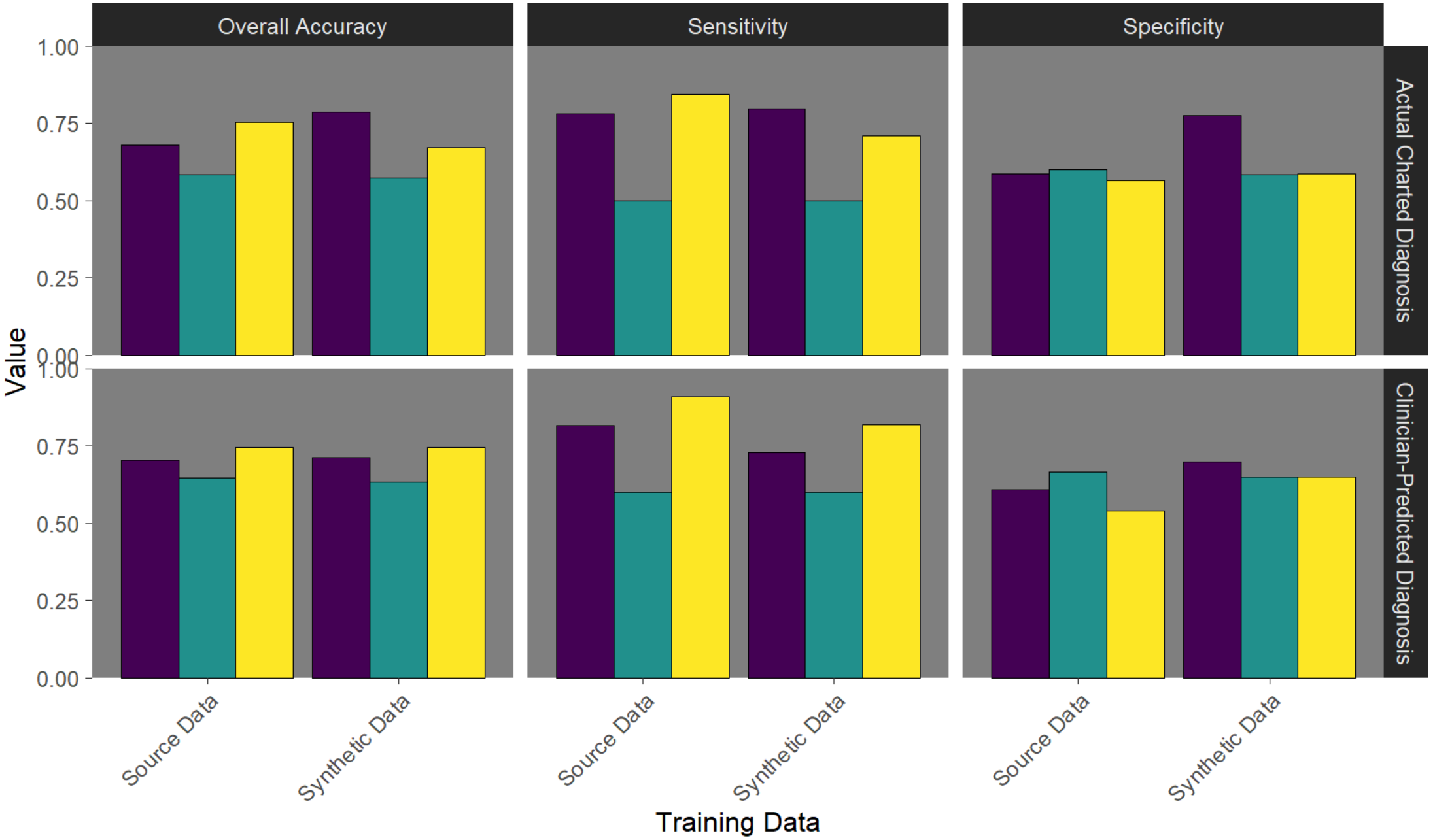
*diagnosis ~ AHP/IHP Ratio * Sex*

Agreement with source training cohort was assessed as if the source cohort were a novel set of cases (i.e., the validation set).

Cutoff = 0.6
Overall Accuracy = 83.33%
Sensitivity = 86.00%
Specificity = 80.77%



Validation Cohort: Results with Synthetic Training Set



Cutpoints

Overall = 0.7

Male = 0.2

Female = 0.9

Conclusions

- Using a logistic regression trained on GAN-generated data **did not yield more precise predictions** in our validation cohort than from a logistic regression trained on our source data.

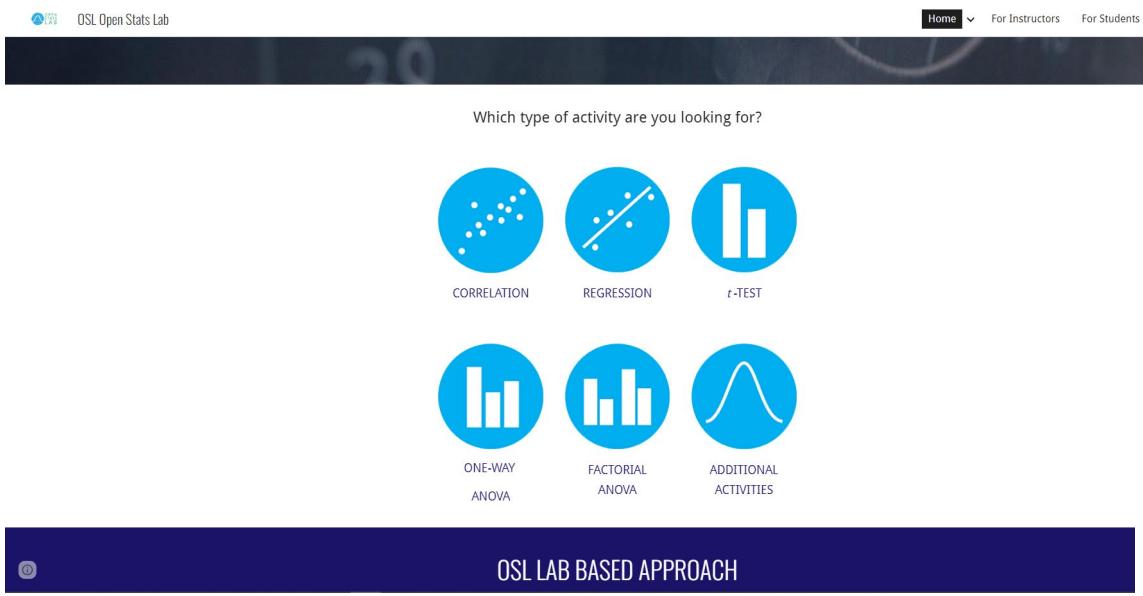
- This **contradicts our previous work** using both clinical (Mangino, 2023) and educational (Mangino et al., 2021) data.

- Previous research indicates **the classifier itself** (LR vs RF vs Boosting vs etc.) has an appreciable effect on the utility of GAN-generated data (Mangino et al., 2021).

- It is possible that as the GAN creates data that more closely match the source data, our secondary model results more closely match those obtained from source data.

Introducing the RGAN Package

If you would like to follow along or test this method afterward, there are several very clean, very clear datasets available on OpenStatsLab: **https://sites.google.com/view/openstatslab/home**.



DO OUTBREAKS OF INFECTIOUS DISEASES INFLUENCE VOTING BEHAVIOR?

Original article: Beall, A. T., Hofer, M. K., & Shaller, M. (2016). Infections and elections: Did an Ebola outbreak influencethe 2014 U.S. federal elections (and if so, how)? *Psychological Science, 27,* 595-605.

LEARNING OBJECTIVES

1. Conduct bivariate correlations
2. Restrict analyses to only certain cases
3. Generate a correlation matrix of the results

ARTICLE
ACTIVITY
DATA (SPSS)
DATA (.CSV)
R SCRIPT

Introducing the RGAN Package

- The **R Statistical Software** package is a commonly used open-source programming language for data analysis and visualization.
  - https://www.r-project.org/
  - I prefer using the **RStudio** IDE for improved functionality: https://posit.co/download/rstudio-desktop/

- The **RGAN** Package (Neunhoeffer, 2022) is an addition to R that facilitates a simple and intuitive syntax for training GANs.
  - https://cran.r-project.org/web/packages/RGAN/index.html
  - https://github.com/mneunhoe/RGAN

- RGAN requires the following R packages as dependencies: `torch`, `viridis`, `cli`, and `devtools`.
  - Installation can be tricky depending on whether you have `torch` through your Python installation.

Biostat CIRCL - CU2_0007 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Biostat CIRCL - CU2_0007

00_project-settings.R | 01_GAN-fitting.R | 04_fitting-final-classifiers.R | 06_plotting-results.R | 02_analysis.R

Source on Save | Run | Source

```
1
2
3
4  ####################
5  ## Load Packages ####
6  ####################
7
8  ## Load packages
9  source("code/00_project-settings.R")
10
11 ## Load the clean validation data
12 load("output/clean-data.rds")
13
14
15 ## Load the usable source and GAN data
16 #load("output/GAN Study/gan-data_source-data_large-sample2.rds")
17
18
19
20
21
22 ####################################
23 ## Removing Unnecessary Variables ####
24 ####################################
25
26 dat_source = dat %>%
27   select(
28     mrn,
29     sex,
30     age,
31     #lad_stenosis_pct,
```

21:1 (Untitled) R Script

Console | Background Jobs

R 4.1.0 · C:/Users/aama241/University of Kentucky/Biostat CIRCL - CU2_0007/

```
        ACS TTS
  Female 178 586
  Male   225  31
> predict(mod_gan, newdata = dat_gan, "response") %>% hist()
> predictions_gan = predict(mod_gan, newdata = dat_gan, "response")
> levels(dat_gan$sex) = c("Female", "Male")
> dat_gan %>%
+   ggplot() +
+   geom_point(aes(x = ahp_ihp_ratio2, y = predictions_gan, color = diagnosis, shape = sex),
+              size = 3) +
+   #geom_vline(xintercept = c(0.892, 0.914), size = 1) +
+   #geom_hline(yintercept = c(0.50, 0.60), linetype = "dashed") +
+   #geom_point(aes(x = 0.84, y = 0.40), pch = 4, size = 3, stroke = 2, color = "black") +
+   #geom_point(aes(x = 0.96, y = 0.50), pch = 4, size = 3, stroke = 2, color = "black") +
+   labs(x = "AHP/IHP Ratio",
+        y = "Predicted Probability of TTS Diagnosis",
+        color = "Clinician\nDiagnosis",
+        shape = "Sex") +
+   scale_color_manual(labels = c("TTS", "ACS"), values = c("#F8766D", "#00BFC4")) +
+   scale_x_continuous(n.breaks = 10) +
+   theme_classic(base_size = 24)
>
```

Environment | History | Connections | Tutorial

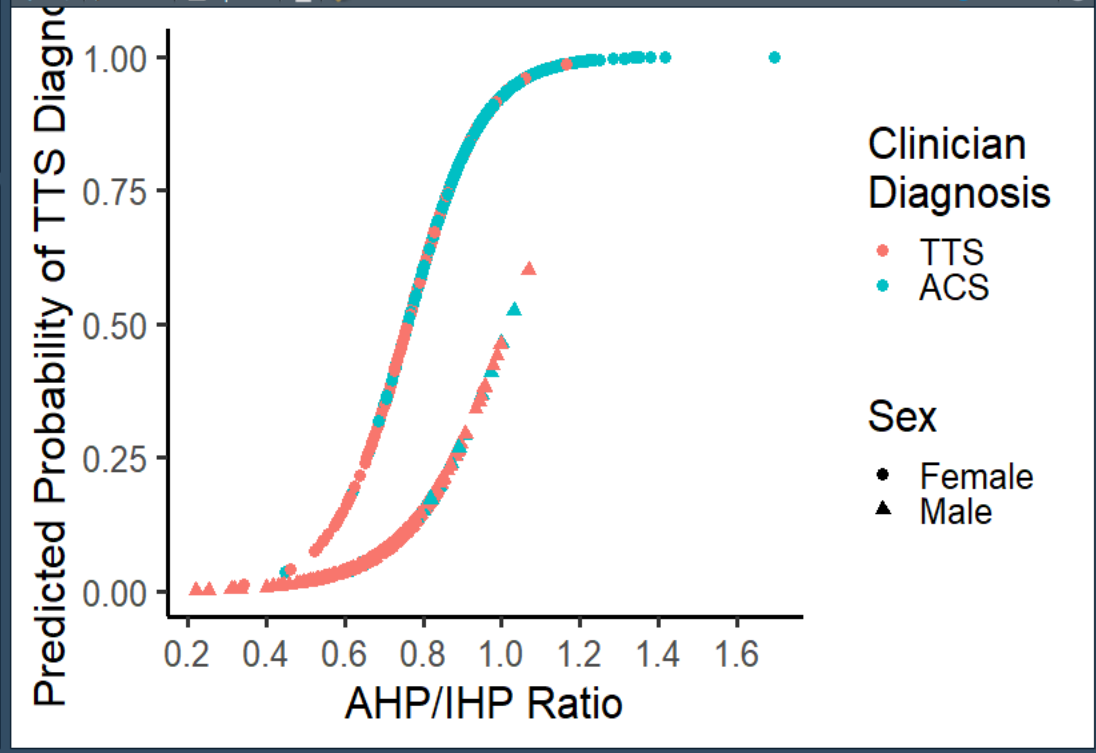
Import Dataset | 956 MiB | List

R | Global Environment

dat_gan	1020 obs. of 8 variables
dat_source	102 obs. of 8 variables
dat_source2	num [1:102, 1:7] 83816 120238 1005446 1081520 1331206 ...
dat_validation	225 obs. of 15 variables
gan	List of 5
mod_gan	List of 30
mod_source	List of 30
tmp	143 obs. of 15 variables
transformer	Environment
transformer_dat	num [1:102, 1:9] -1.6 -1.6 -1.5 -1.49 -1.46 ...
Values	
d_network	function (input)
g_network	function (input)
predictions_gan	Named num [1:1020] 0.9149 0.0695 0.8804 0.2253 0.0978 ...

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish



Introducing the RGAN Package

The basic process of fitting a GAN for **tabular data** is as follows:

1. Import your dataset, specify relevant variable types (e.g., numeric, factor), and perform any data cleaning necessary.

2. Create and fit a transformer to your dataset to standardize all variables, specifying categorical and continuous variables as needed.

3. Specify the Generator and Discriminator architectures (# hidden nodes, # hidden layers, activation function, loss function, etc.) **This step is optional.**

4. Fit the GAN to your source data (4a), evaluating periodically (4b).

5. Evaluate your final dataset and secondary models for similarity to source data.

6. Repeat steps 3-5 as necessary.

Basic “toy” example can be found at:
https://github.com/mneunhoe/RGAN

Step 1 & 2: Importing Data and Fitting Transformer

1)

```
dat_source = rio::import("data/dataset_source.csv")

str(dat_source[2:ncol(dat_source)],)
```

```
> str(dat_source[2:ncol(dat_source)],)
tibble [102 x 6] (S3: tbl_df/tbl/data.frame)
 $ sex           : num [1:102]
 $ age           : num [1:102]
 $ inferior_hinge: num [1:102]
 $ anterior_hinge: num [1:102]
 $ ahp_ihp_ratio2: num [1:102]
 $ diagnosis     : num [1:102]
```

All variables are treated as numeric.

The RGAN package will only work with complete data. No missingness!

2)

```
## Initialize new transformer
transformer = data_transformer$new()
```

```
transformer$fit(as.data.frame(dat_source),
                discrete_columns = c("sex",
                                     "diagnosis"))

transformer_dat = transformer$transform(dat_source)
```

```
> head(transformer_dat)
                0 1                                                 0 1
[1,] -1.599805 0 1 -0.9885911  0.8821658  1.0828349 -0.2234850 1 0
[2,] -1.595694 1 0  0.9762626 -0.8233102 -0.4246605  0.7488413 0 1
[3,] -1.495780 0 1 -0.5170262  1.2914801  0.9823352 -0.7267375 1 0
[4,] -1.487194 0 1  0.5832919  0.4046325  0.0778380 -0.6395672 1 0
[5,] -1.459012 1 0  0.8976685  1.0868230  1.6858331  0.0746951 1 0
[6,] -1.455679 1 0  1.6050158 -0.8233102 -0.7261595  0.2821247 0 1
```

Side Note: Working with Categorical Variables

- Numeric variables do not require any special treatment prior to training the transformer and the GAN.
- Categorical variables must be **one-hot encoded**, which is part of the transformer process.
- Each **category** gets its own binary vector (1 = category present and 0 = category absent).

Original Categorical Variables

```
  sex    diagnosis
  <fct>  <fct>
1 Male   tts
2 Female tts
3 Female tts
4 Female tts
5 Female tts
6 Female tts
> table(dat$sex, dat$diagnosis)

        acs tts
  Female  20  46
  Male    30   6
```

Binary-Coded Categorical Variables

```
    sex diagnosis
  <dbl>     <dbl>
1     1         1
2     0         1
3     0         1
4     0         1
5     0         1
6     0         1
> table(dat_source$sex, dat_source$diagnosis)

    0  1
0  20 46
1  30  6
```

One-Hot Encoded Categorical Variables

```
      0 1 0 1
 [97,] 0 1 0 1
 [98,] 1 0 0 1
 [99,] 1 0 0 1
[100,] 1 0 0 1
[101,] 1 0 0 1
[102,] 1 0 0 1
> table(transformer_dat[,3], transformer_dat[,9])

    0  1
0  20 46
1  30  6
> table(transformer_dat[,2], transformer_dat[,8])

    0  1
0   6 30
1  46 20
```

Step 3 & 4: Specifying G & D Networks and Fitting GAN

3) Manually specifying the G and D architectures was not done here. The default specifications were used within the `gan_trainer` function.

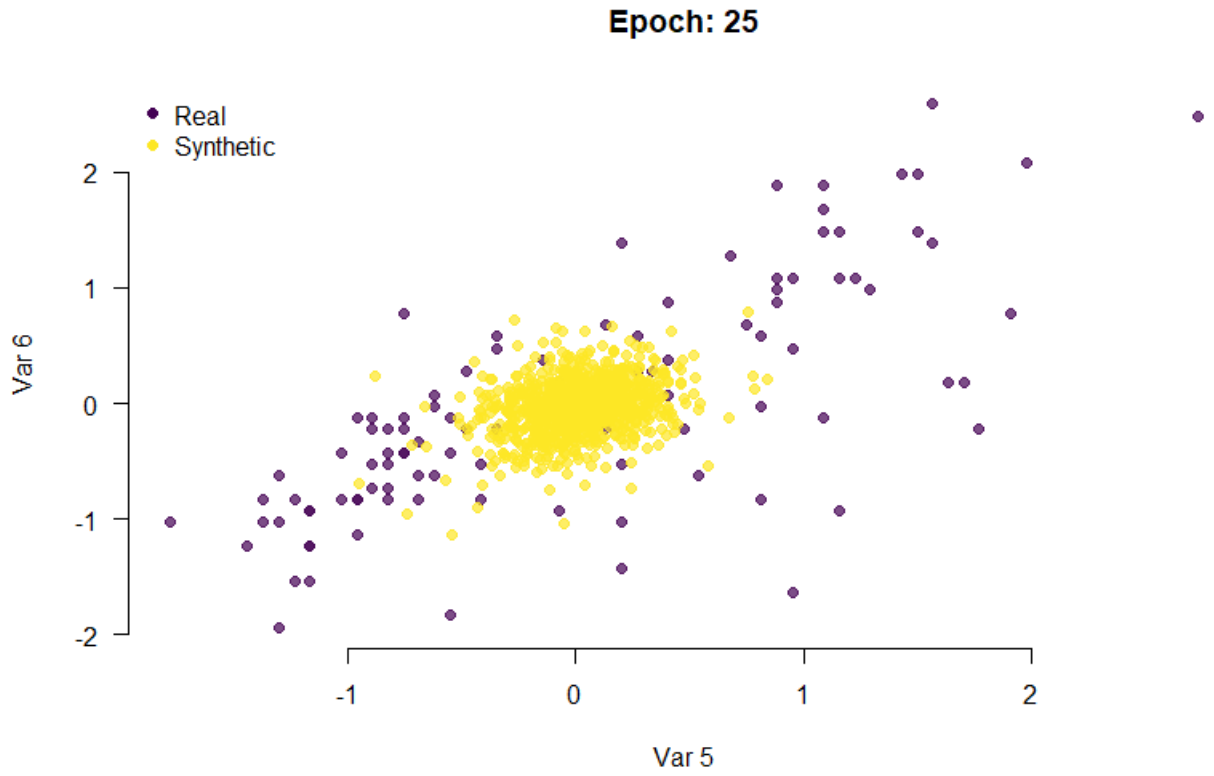
```
# Generator architecture
g_network =
  Generator(noise_dim = 10,
            data_dim = ncol(dat_source),
            hidden_units = list(256, 8, 64),
            dropout_rate = 0.01)
```

4a)

```
gan = gan_trainer(data = transformer_dat,                # transformed dataset
                  noise_dim = 5,                         # dimensions of "noise" data/random error
                  noise_distribution = "normal",         # distribution of "noise" data/random error
                  value_function = "wasserstein",        # type of loss function
                  data_type = "tabular",                 # type of data
                  base_lr = 0.0001,                      # GAN learning rate
                  ttur_factor = 5,                       # Multiplier for learning rate
                  weight_clipper = NULL,                 # Wasserstein GAN limits on D network weights
                  batch_size = 35,                       # Number of training samples included in minibatch for training
                  epochs = 400,                          # Total number of training cycles
                  plot_progress = TRUE,                  # Plot data points periodically
                  plot_interval = 50,                    # How often to plot data points
                  eval_dropout = FALSE,                  # Drop cases when sampling from synthetic data?
                  synthetic_examples = nrow(dat)*10,     # Number of synthetic cases to generate
                  plot_dimensions = c(5, 6),             # Columns in data to plot
                  device = "cpu")                        # On which device training should be done
```

Step 4: Training & Evaluating GAN

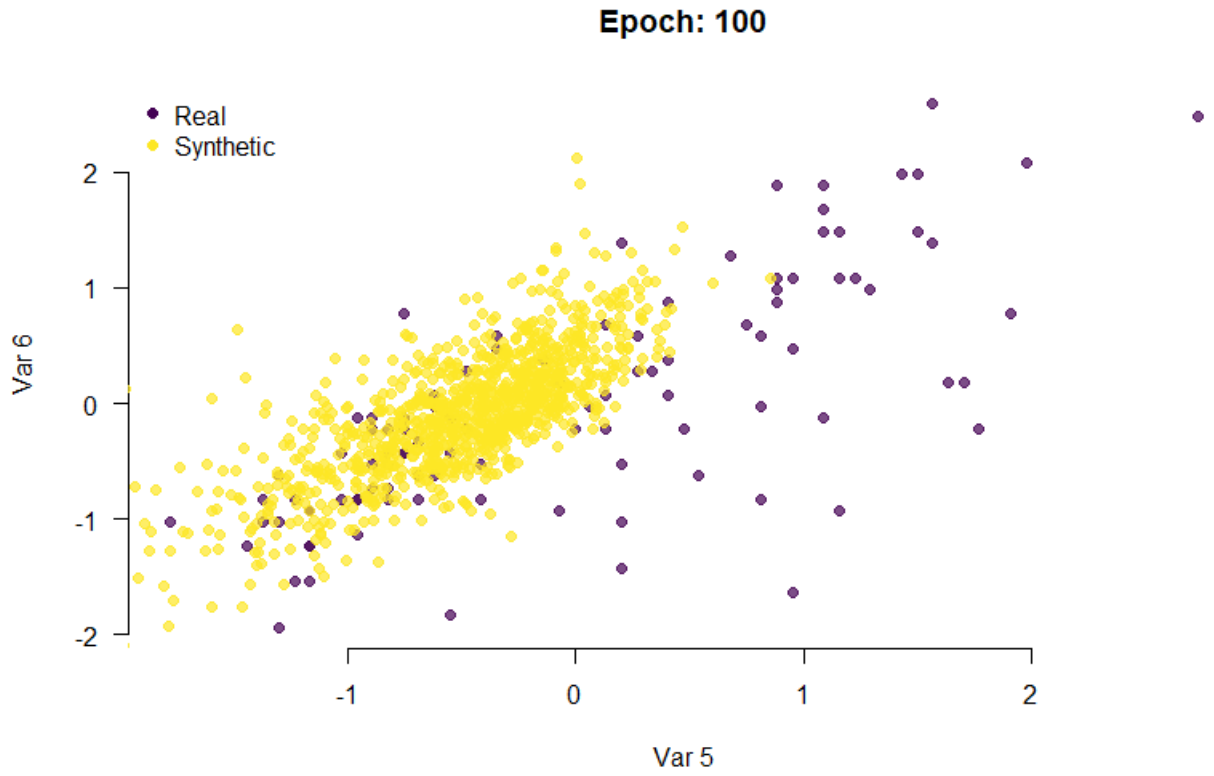
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

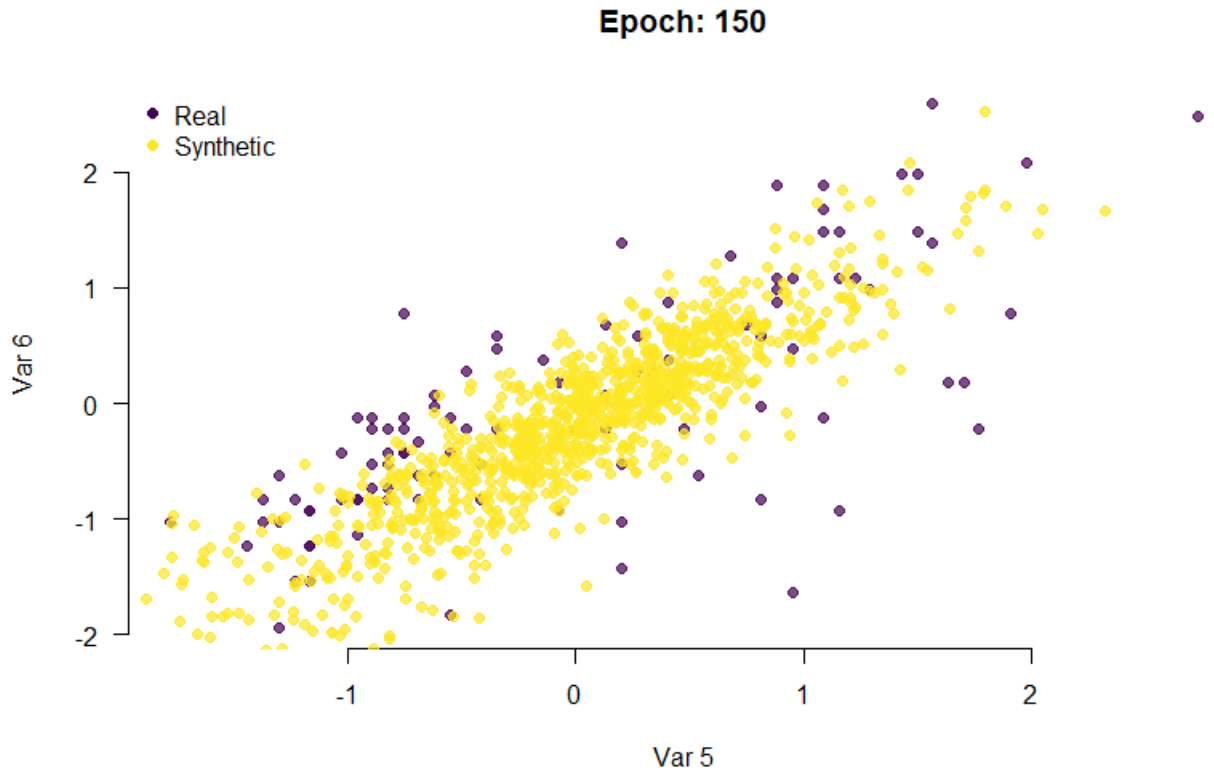
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

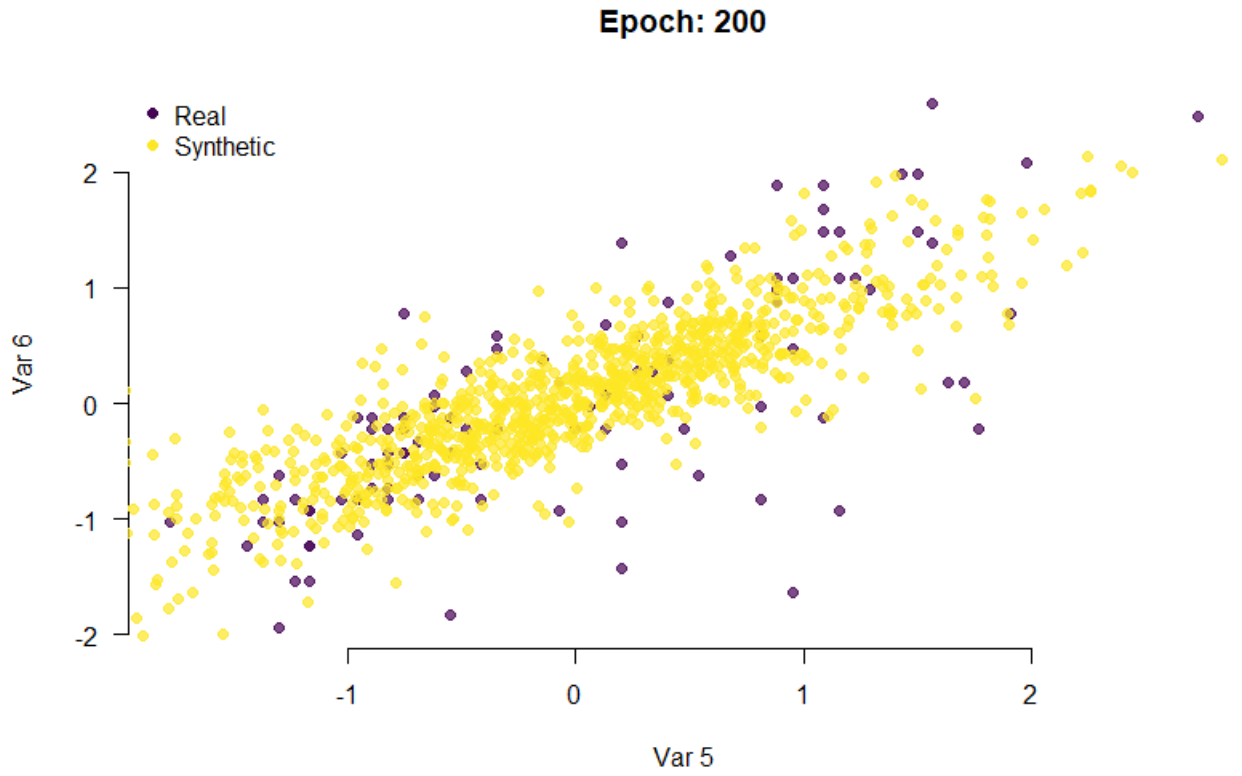
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

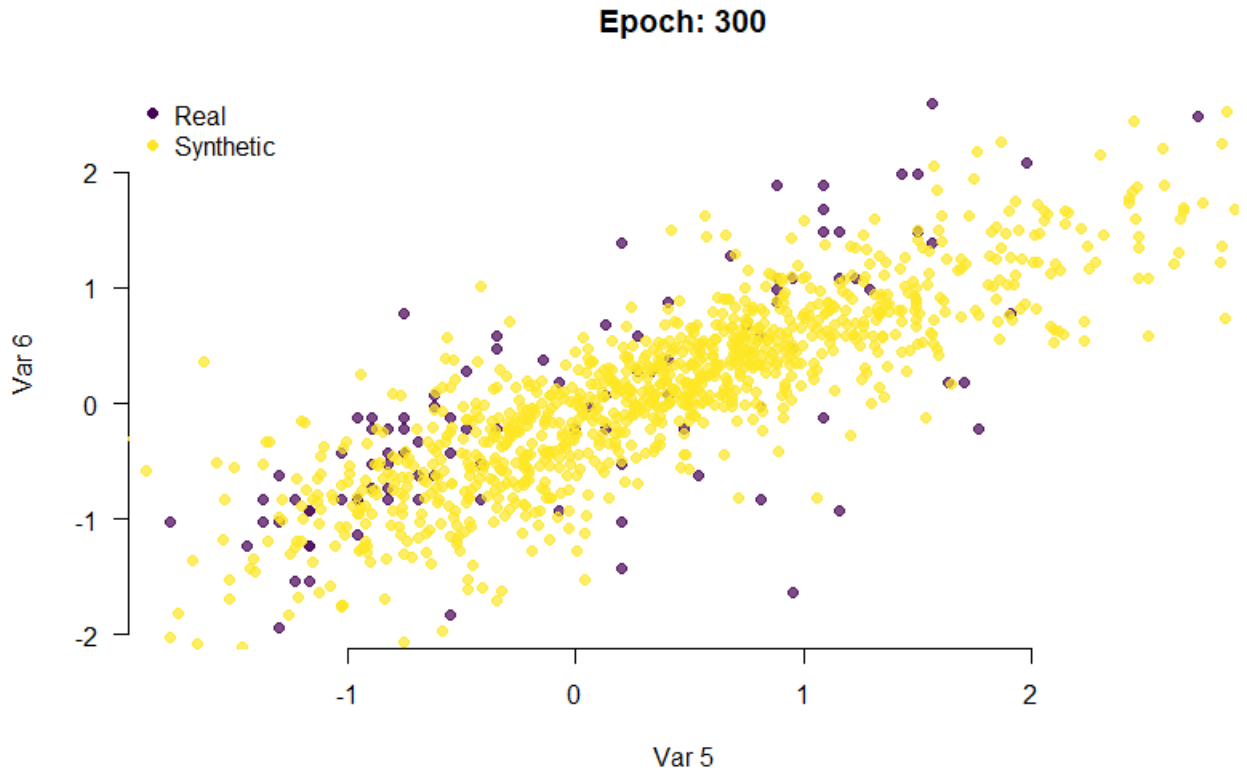
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

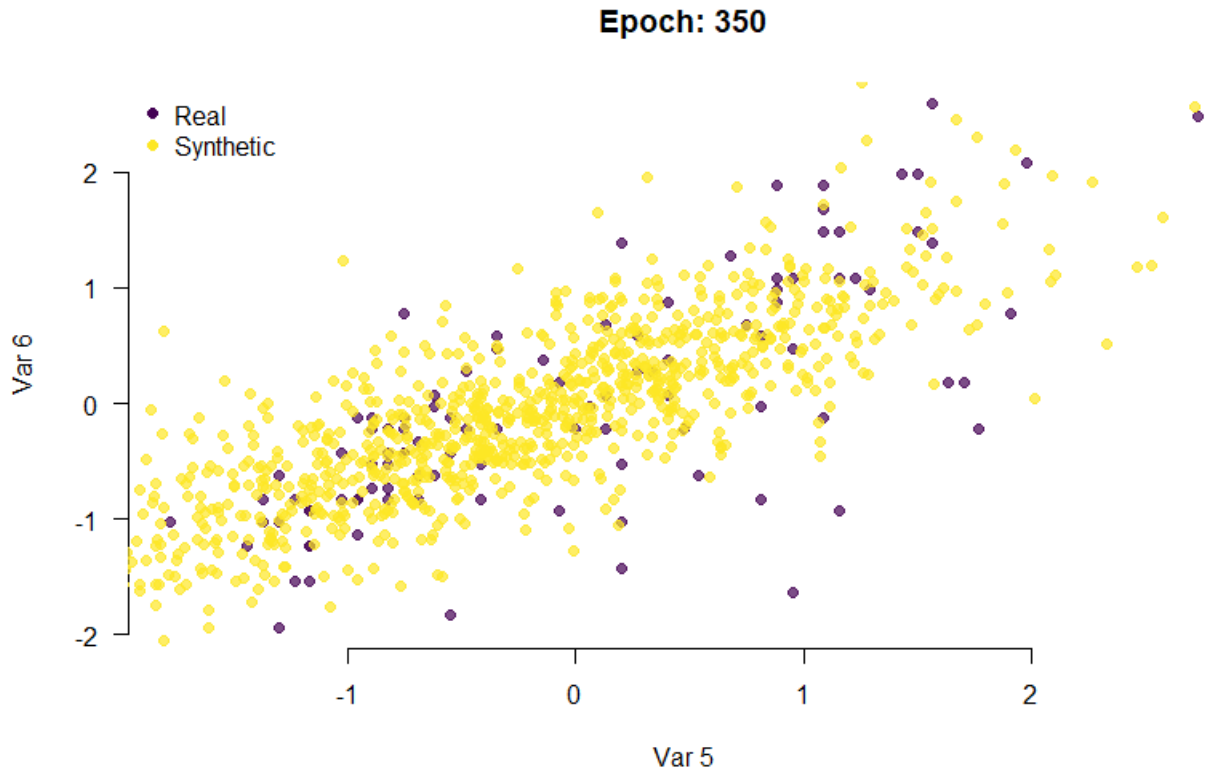
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

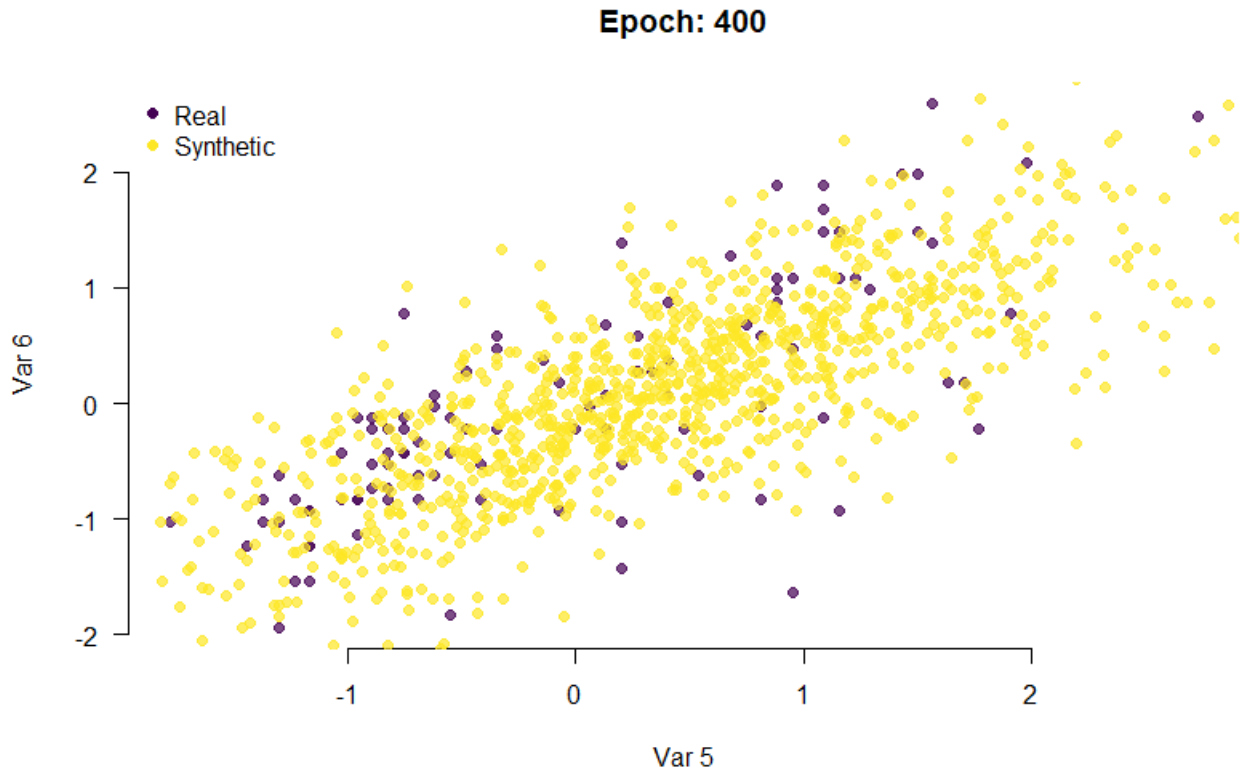
4b)



X-Axis = AHP
Y-Axis = IHP

Step 4: Training & Evaluating GAN

4b)

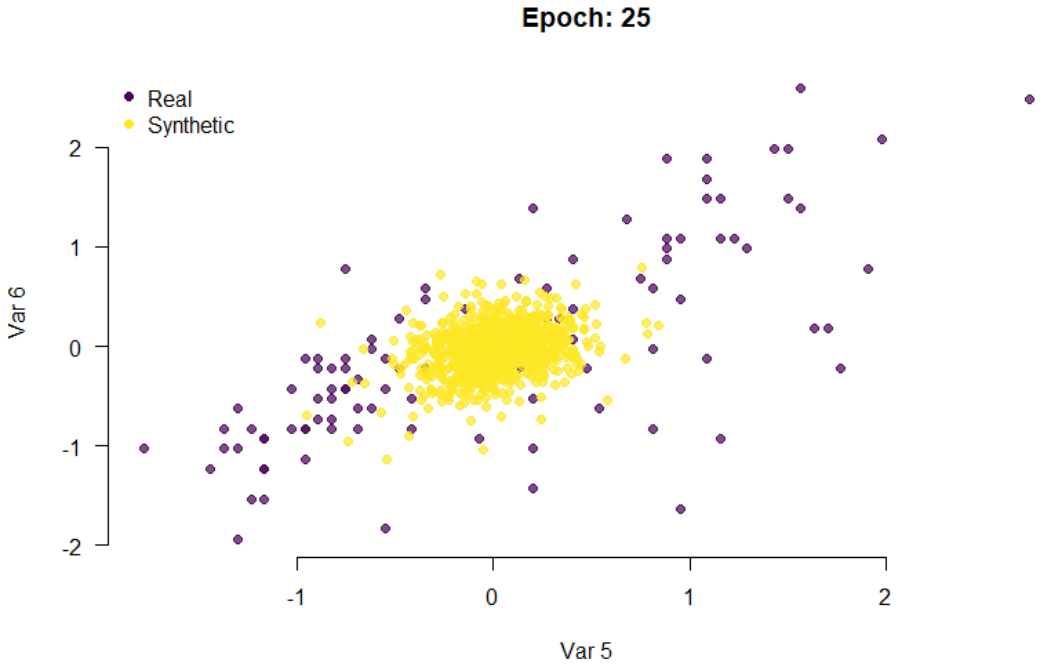


X-Axis = AHP
Y-Axis = IHP

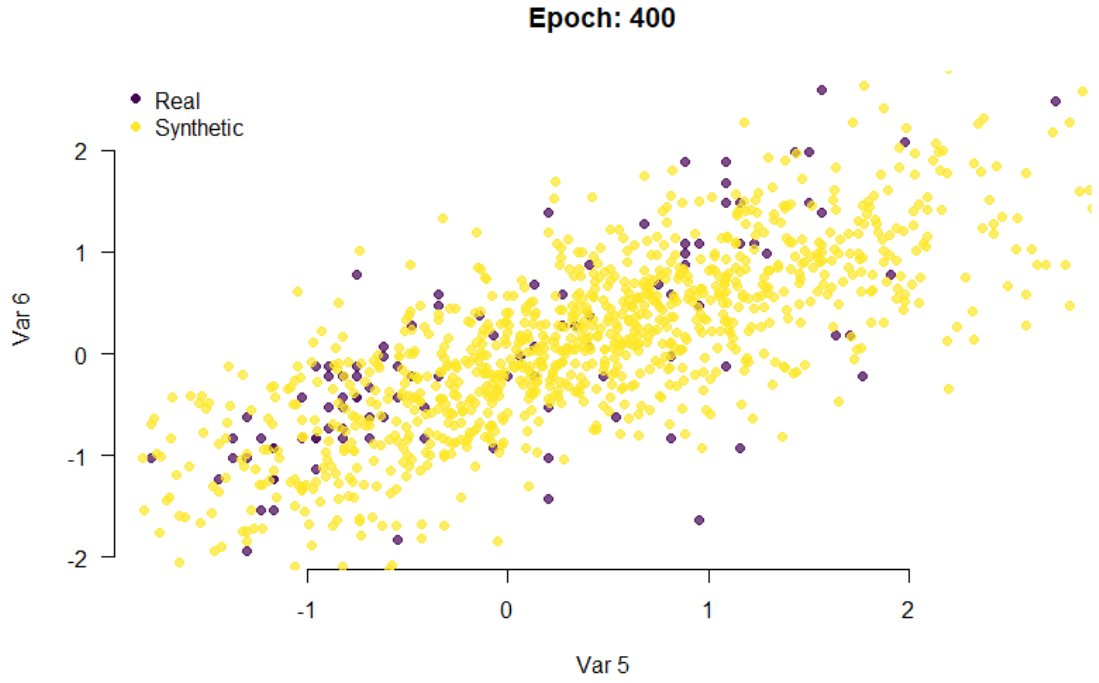
Step 4: Training & Evaluating GAN

4b)

Beginning of Training



End of Training



X-Axis = AHP
Y-Axis = IHP

Step 5: Evaluating Synthetic Data & Secondary Models

- Extracting synthetic cases and back-transforming to original metrics

```
dat_gan = sample_synthetic_data(gan,
                                transformer = transformer)
```

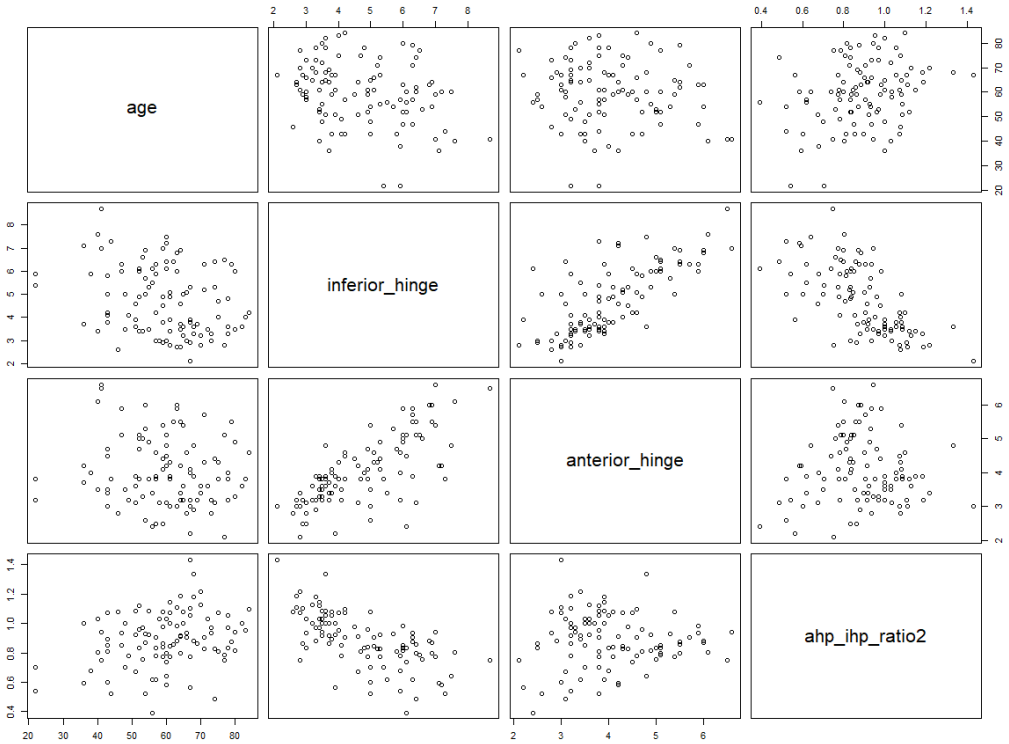
- Assessing statistical properties of the synthetic data: Central tendency and variability, univariate cell counts/proportions

	GAN (N=1020)	Real (N=102)	Overall (N=1122)	p-value
sex				
Female	764 (74.9%)	66 (64.7%)	830 (74.0%)	0.034
Male	256 (25.1%)	36 (35.3%)	292 (26.0%)	
age				
Mean (SD)	59.0 (9.23)	59.6 (12.7)	59.0 (9.60)	0.634
Median [Min, Max]	59.0 [21.7, 97.7]	60.0 [22.0, 84.0]	59.2 [21.7, 97.7]	
inferior_hinge				
Mean (SD)	5.30 (1.59)	4.71 (1.47)	5.25 (1.58)	<0.001
Median [Min, Max]	5.33 [-1.75, 10.3]	4.55 [2.10, 8.70]	5.30 [-1.75, 10.3]	
anterior_hinge				
Mean (SD)	4.10 (0.915)	4.02 (0.995)	4.10 (0.922)	0.432
Median [Min, Max]	4.17 [0.818, 7.07]	3.80 [2.10, 6.60]	4.16 [0.818, 7.07]	
ahp_ihp_ratio2				
Mean (SD)	0.866 (0.168)	0.891 (0.184)	0.868 (0.169)	0.187
Median [Min, Max]	0.872 [0.221, 1.70]	0.903 [0.393, 1.43]	0.874 [0.221, 1.70]	
diagnosis				
ACS	403 (39.5%)	50 (49.0%)	453 (40.4%)	0.078
TTS	617 (60.5%)	52 (51.0%)	669 (59.6%)	

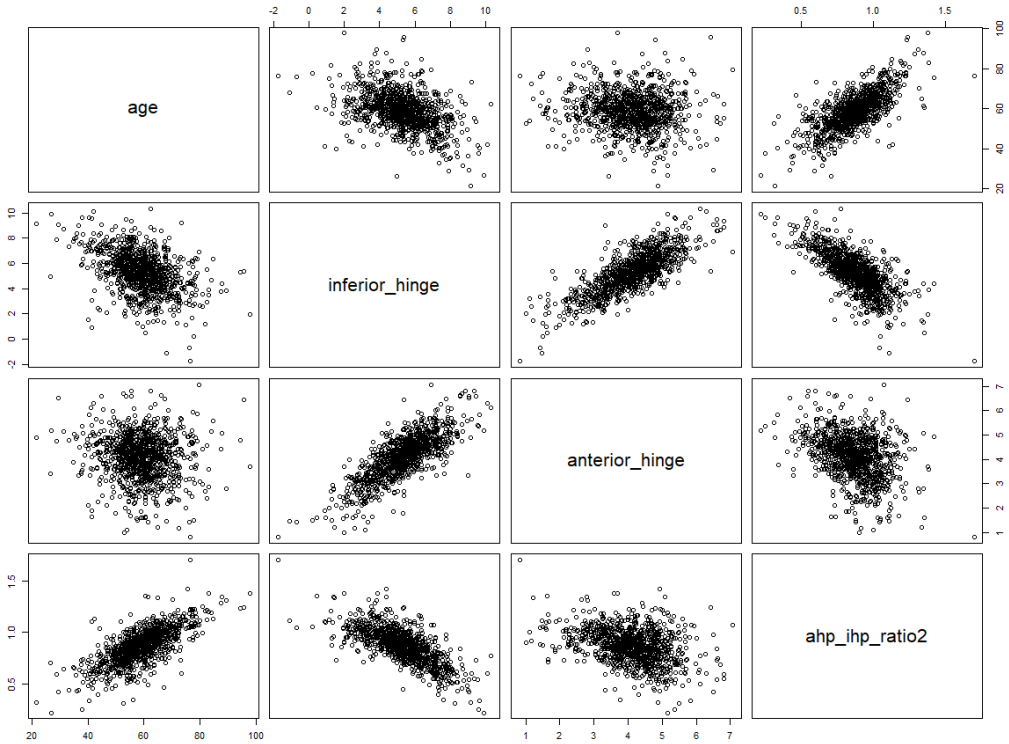
Step 5: Evaluating Synthetic Data & Secondary Models

- Assess correlations among numeric variables. The **topography** of the associations, not necessarily the raw correlation coefficient.

Source Data



Synthetic Data



Step 5: Evaluating Synthetic Data & Secondary Models

- Assess bivariate cell counts among relevant categorical variables.

Source Data

	diagnosis		
	ACS	TTS	Total
sex			
Female	20 (20%)	46 (45%)	66 (65%)
Male	30 (29%)	6 (5.9%)	36 (35%)
Total	50 (49%)	52 (51%)	102 (100%)

Synthetic Data

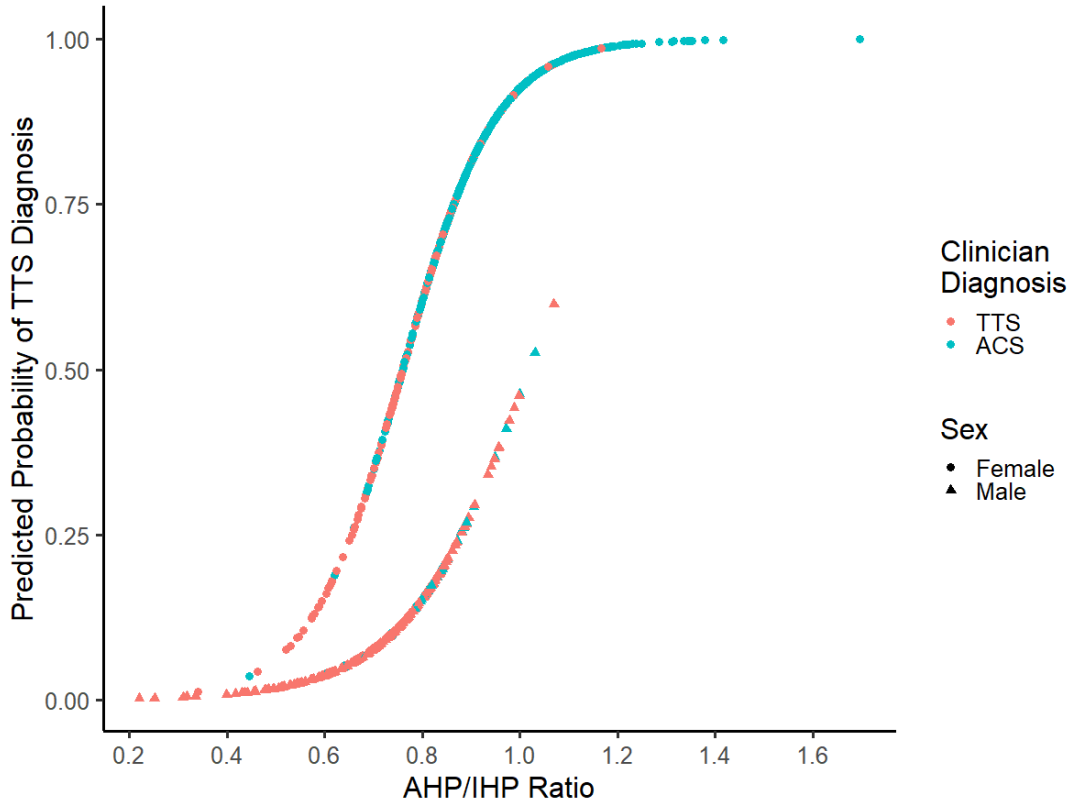
	diagnosis		
	ACS	TTS	Total
sex			
Female	178 (17%)	586 (57%)	764 (75%)
Male	225 (22%)	31 (3.0%)	256 (25%)
Total	403 (40%)	617 (60%)	1,020 (100%)

Step 5: Evaluating Synthetic Data & Secondary Models

- Fit logistic regression model on synthetic data and obtain predictions for source data. Plot results.

```
mod_gan = glm(
  diagnosis ~
    sex : ahp_ihp_ratio,
  data = dat_gan,
  family = binomial(link = "logit")
)
```

```
predictions_gan = predict(
  mod_gan,
  newdata = dat_source,
  type = "response"
)
```



Step 5: Evaluating Synthetic Data & Secondary Models

In evaluating secondary models fit to synthetic data, it is important to obtain:

- Model fit statistics (R-squared, AIC, BIC, log-likelihood)
- Coefficients/parameter estimates
- Standard errors & confidence intervals
- Fitted and residual values
- Any other model-specific evidence for evaluating fit and assessing assumptions of the model

Step 5: Evaluating Synthetic Data & Secondary Models

- We have already discussed the model results from our source and synthetic data (see Slides 12 and 15), so I won't repeat that here.

- **Overall conclusion:** The synthetic data **might** look similar to the source data, but with enough discrepancies that synthetic cases could be identified by even a non-clinician.

- **Next Steps:** Modify the GAN hyperparameters. Try different batch sizes, learning rates, number of epochs, loss function, optimizers, and/or G & D network architectures.

What can we say about the quality of this synthetic dataset?

- The measures of central tendency and variability are **similar**, but with some impossible values (e.g., negative IHP measurement).
- Bivariate scatterplots for numeric variables are **similar**, but some relationships are too close to a 1:1 correspondence (e.g., age x AHP/IHP ratio).
- Crosstables have some key discrepancies in proportions (e.g., 57% female TTS patients in synthetic data vs 45% in source data).
- **Verdict:** The GAN needs more tuning before synthetic data can be used.

Closing Remarks

Axiom: As statisticians, we are also ethicists.

Conclusions

Synthetic data are only as good as our source data.

- If your dataset is small and non-comprehensive, your GAN might not effectively learn the dataset.
- Your synthetic data might not faithfully represent your source data.

Good generated data do not automatically beget better clinical decision-making.

- Even with a well-tuned generative model, your synthetic cases may or may not be believable.
- Your synthetic data may have impossible values, but still get good answers from your secondary analysis models and/or predictions.

Future Directions

- More fully examine synthetic datasets to determine optimal GAN hyperparameters.
- Assess whether these results hold in small, complete datasets or if we can obtain a similar level of precision in other, more complex datasets.
- Determine whether second-order bias is introduced in synthetic data, whether through mechanisms like imputation or through the very process of generating synthetic data.
- Devise consistent methods for quantifying the synchrony between source and synthetic data. Identifying interpretable metrics for generative models.

References

Ahmed, T., Mangino, A.A., Lodhi, S.H., Gupta, V., Leung, S.W., & Sorrell, V.L. (2024). Simplified echocardiographic assessment of regional left ventricular wall motion pattern in patients with takotsubo and acute coronary syndrome: The Randomized Blinded Two-chamber Apical Kinesis Observation (TAKO) Study. Current Problems in Cardiology, 102731. DOI: 10.1016/j.cpcardiol.2024.102731

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27-36*.

Madhavan, M., & Prasad, A. (2010). Proposed Mayo Clinic criteria for the diagnosis of Tako-Tsubo cardiomyopathy and long-term prognosis. Herz, 35(4), 240–243. https://doi.org/10.1007/s00059-010-3339-x

Mangino, A.A. (2023, August). Case Studies in Data Emulation and Augmentation Using Generative Adversarial Networks in Psychoeducational Data. Presented at the 2023 Joint Statistical Meetings; Toronto, ON, CA.

Mangino, A.A., Smith, K.A., Finch, W.H., & Hernández-Finch, M. E., (2021). Improving Predictive Classification Models Using Generative Adversarial Networks in the Prediction of Suicide Attempts. Measurement and Evaluation in Counseling and Development, 55(2), 116-135. https://doi.org/10.1080/07481756.2021.1906156

Neunhoeffer, N., (2022). RGAN: Generative Adversarial Nets (GAN) in R. R package version 0.1.1. https://CRAN.R-project.org/package=RGAN

Sharkey SW, Maron BJ. Survival After Takotsubo, Revisited. J Am Coll Cardiol. 2018 Aug 21;72(8):883-884. doi: 10.1016/j.jacc.2018.06.022. PMID: 30115227.

Templin, C., Ghadri, J. R., Diekmann, J., Napp, L. C., Bataiosu, D. R., Jaguszewski, M., ... & Lüscher, T. F. (2015). Clinical features and outcomes of takotsubo (stress) cardiomyopathy. New England Journal of Medicine, 373(10), 929-938.

Verma, A. (2019, July). Generative adversarial network. *Linkedin*. Https://www.Linkedin.Com/pulse/generative-adversarial-network-abhishek-verma/

Thank you!

Contact Information

Anthony A. Mangino, PhD

Department of Biostatistics

Biostatistics Consulting and Interdisciplinary Research Collaboration Lab (Biostat CIRCL)

Anthony.Mangino@uky.edu

Any questions?