

## ABSTRACT

ZHUANG, ZIZHI. Metadata-driven Re-interpretation of Charge Tunneling Characteristics in Large Area Tunneling Junction (Under the direction of Dr. Martin Thuo).

This study is designed to analyze metadata to better understand data derived from measurement of charge tunneling across self-assembled monolayers. This thesis demonstrates that measurement with a eutectic gallium–indium alloy (EGaIn) system on n-alkanethiolate self-assembled monolayers (SAMs) is affected by the season and the time of the experiment. Higher humidity in the spring increases the probability of capillary bridge formation between the EGaIn tip and the SAM, thus increasing the proportion of thin-area defects (high conductance) junctions. When using Di-alkyl amide (DAA) as the terminal functional group of SAMs, measurement is influenced by the both season and the time that the experiment is done. We also infer that reliability of a data set can also be elucidated by mining metadata. A set of data measured with ferrocene alkanethiol ( $SC_nFc$ ) SAMs gave very noisy data and are therefore is unreliable despite. This particular dataset also did not have associated metadata, making it challenging to forensically mine.

© Copyright 2024 by Zizhi Zhuang  
All Rights Reserved

Metadata-driven Re-interpretation of Charge Tunneling Characteristics in Large Area Tunneling Junction

by  
Zizhi Zhuang

A thesis submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the degree of  
Master of Science

Material Science and Engineering

Raleigh, North Carolina  
2024

APPROVED BY:

---

Martin Thuo  
Committee Chair

---

Martin Seifrid

---

Joseph Tracy

## **BIOGRAPHY**

Zizhi Zhuang majored in Material Science and Engineering and minored in Computer Science at North Carolina State University receiving a B.S (Material Science and Engineering) in 2022. He continued as a graduate M.S student working under the tutelage of Professor Martin Thuo. His thesis work centers on mining metadata in large area tunneling junction measurements to reveal effect of adventitious contaminants or random errors.

## ACKNOWLEDGMENTS

First of all, I would like to thank my girlfriend and my parents. Without their love, enlightenment and support. There's no way I'm going to make it to the end.

Secondly, I would like to say a sincere thank you to my supervisor, Professor Martin Thuo, who introduced me to the scientific world. He opened my eyes and taught me to think critically. The most important thing was, he guided me on the way when I was at my most lost.

Thirdly, allow me to express my gratitude to my mentor, Dr. Julia Chang, who patiently taught me during this time. Additionally, I would like to thank my group members including Alana Pauls, Dr. Andrew Martin, and Dhanush U. Jamadgni.

Finally, I would like to thank all MSE faculties, who helped me and led me to graduate successfully.

## TABLE OF CONTENTS

LIST OF FIGURES .....	v
<b>Chapter 1: Introduction and Background</b> .....	<b>1</b>
<b>Chapter 2: Metadata-Driven Unveiling of Environment Effects on Large Tunnel Junctions</b>	
Introduction.....	14
Results and Discussion .....	15
Conclusion .....	24
References.....	25
<b>Chapter 3: Metadata-Driven Unveiling of Time-period Effects on Large Tunnel Junctions and the Importance of Metadata</b>	
Introduction.....	27
Results and Discussion .....	32
Conclusion .....	43
References.....	44
<b>Chapter 4: General Conclusion</b> .....	<b>46</b>
<b>Chapter 5: Future Steps</b> .....	<b>48</b>
APPENDIX: .....	49
Appendix A: Method & Python code used for mining metadata.....	50

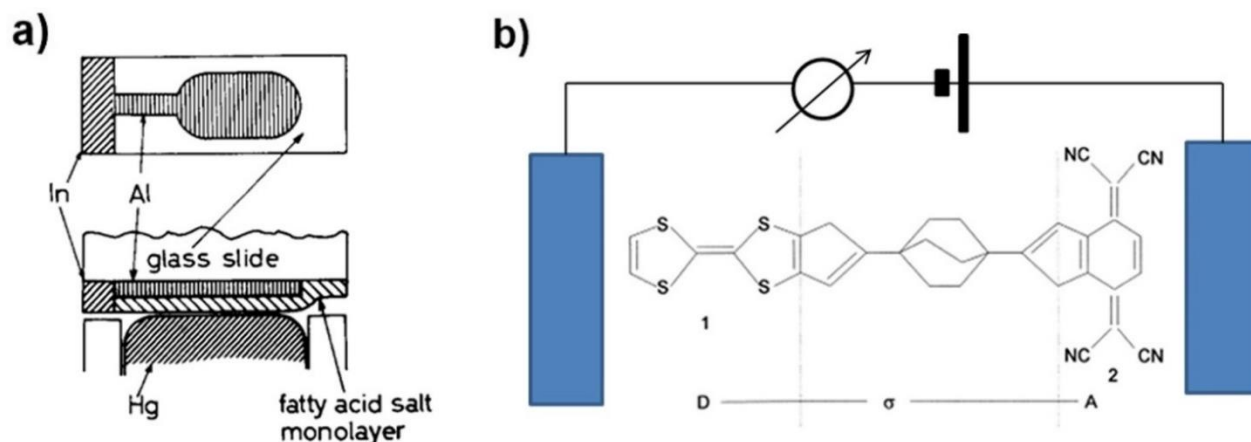
## LIST OF FIGURES

Figure 1.1 The sample used in 1971 and the first concept model of tunneling junction .....	2
Figure 1.2 Formation of self-assembled monolayer (SAMs) on gold substrate. ....	3
Figure 1.3 Schemes of several molecular tunneling junction systems .....	4
Figure 1.4 Measurement result from previous experiment and he difference between the actual measurement value and the literature value .....	7
Figure 2.1 Schematic Illustration of placement of physisorbed contaminants on a tunnel junction .....	16
Figure 2.2 Summary of the n-alkanethiolate SAM data.....	18
Figure 2.3 Effect of user, time and season on data quality.....	21
Figure 2.4 Population-independent statistical quantification of the role of different parameters .....	23
Figure 3.1 Schematic of EGaIn tunneling junctions with DAA as the functional group of SAMs and the analysis of tunneling currents through junctions .....	30
Figure 3.2 Data analysis through heat maps and three-dimensional plots with DAA as the functional group.....	33
Figure 3.3 Data analysis through dot plots with DAA as the functional group .....	36
Figure 3.4 Data analysis based on estimation plots with DAA as the functional group.....	38
Figure 3.5 Data analysis through heat maps and three-dimensional plots with SC <sub>n</sub> Fc (n = 10- 15) as SAMs .....	40
Figure 3.6 Data analysis through dot plots with SC <sub>n</sub> Fc (n = 10-15) as SAMs .....	41
Figure 3.7 Data analysis through estimation plots with SC <sub>n</sub> Fc (n = 10-15) as SAMs.....	42

## CHAPTER 1

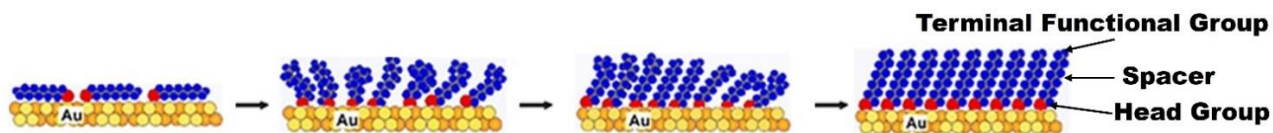
### Introduction and Background

In 1971, Mann and Kuhn [1] applied voltage to a fatty acid monolayer with electrodes at top and bottom, and defined the symmetrical characteristic of current-voltage curve of the system. This work formed the basis of molecular electronics (Figure 1.1a). Then in 1974, Aviram and Ratner [2] proposed a potential structure of molecular rectifiers, opening the door to single molecule electronics (Figure 1.1b). They discussed the possibility of applying the molecular component as electronic devices here in a diode. According to Moore's law, the number of transistors on an integrated circuit (IC) will double every 1-2 years. This has led to lengthened minimalization, but we are approaching a fabrication limit, thus molecular electronics have attracted increased attention. The transition to molecular-size electronic is inevitable since they go beyond the limits of Moore's Law [3]. There are still some fabrication challenges remaining for our total understanding of charge transport in molecular electronic, but large area junctions are promising given their stability and ability to generate reproducible data. These junctions rely on self-assembled monolayers to deduce collective behavior of similar molecules.



**Figure 1.1.** (a) Scheme of the system Mann and Kuhn experimented on 1971 [1]. (Figure reproduced from Ref. [1] with permission. Rights managed by AIP Publishing) (b) The conceptualized device of Aviram and Ratner built in 1974 [2]. (Figure reproduced from Ref. [2] with permission. Copyright © 1974 North-Holland Publishing Company)

Self-assembly is a process that components spontaneously combine into organized structures without any external intervention (Figure 1.2) [4]. Covalent and non-covalent interactions can exist in molecules at the same time, which is a very important factor to define a molecule as a part of self-assembling. [5]. If self-assembled molecules adsorb on solid or liquid surface, self-assembled monolayers (SAMs) can be formed. Three types of factors that enable self-assembled to occur on the surface, including the affinity of the head groups for the surface which decide the structure, the interaction between molecules which stabilize the structure of SAMs, and the non-covalent horizontal interaction between the chains which decide details such as tilt angle of SAMs.



**Figure 1.2.** Formation of self-assembled monolayer (SAMs) on gold substrate is an on- off- process leading to final attachment of a well order densified layer.

Typically, molecules for SAMs formation contain three parts: the head group, the organic chain, and the terminal group [6] (Figure 1.2). The head group is the end of the molecule that binds to the surface since it has a special affinity to the substrate. The choice of head group contains thiol, silane and phosphonate since they have strong affinity to the metal surface. The organic chain decides molecular ordering and can change the electrical properties [5]. The alkyl chain ( $\text{CH}_2$ ) is widely used as the organic phase of SAMs due to its simple structure, which means the thickness of the monolayer can be easily controlled. The terminal group connects the SAMs to the external environment, and determines the properties of SAMs, such as topography[7], surface energy[8], hydrophobic, and electrical conductivity [6]. The choice of terminal groups is variable. It can be a part of the organic chain, or it can be an individual part to provide the desired functionality.

Since SAMs are easy to customize and reproducible, they are widely studied in molecular electronics. Also, these characteristics of SAMs in molecular tunneling junctions provide a perfect platform to further understand the mechanism of charge transportation, and as a result, it may be possible in the future to make cheaper electronic devices to compete with or integrate into the semiconductor technology. Different types of SAMs contain different functions. The alkanethiol acts as a dielectric since the carbon chain contains a large HOMO-LUMO gap which causes low electronic conductivity [9]. The ferrocene alkanethiol contains a thiol headgroup to

constraint molecule to electrodes, an alkanethiol chain to provide the asymmetry of HOMO-LUMO level, and a ferrocene functional group to give the HOMO level that energy can reach [10]. Thus, the ferrocene alkanethiol should function as a diode.

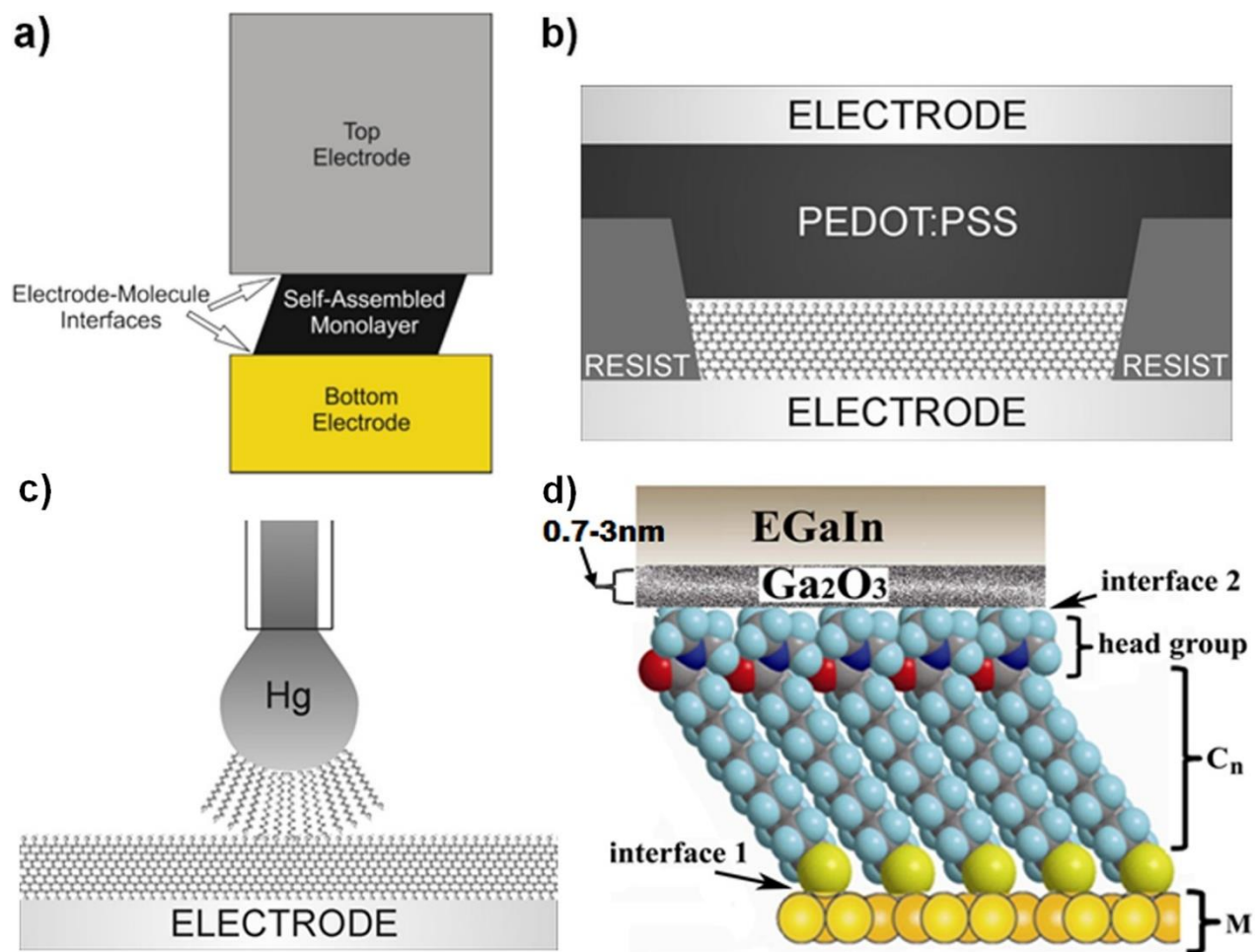


Figure 1.3. Schemes of several molecular tunneling junction systems. (a) Structure of molecular tunneling junction. (b) PEDOT: PSS tunneling junction system [9]. (c) Mercury tunneling junction system [9]. (Figure reproduced from Ref. [9] with permission. Copyright © 2008 IOP Publishing Ltd) (d) EGaIn tunneling junction system. (Figure reproduced from Ref. [11] with permission. Copyright © 2011 American Chemical Society)

The modules of a molecular tunneling junction contain: the top electrode, the bottom electrode, and the SAM placed between two electrodes (Figure 1.3a). Usually, metal will be chosen as the bottom electrode, such as Au, Ag and Pt, and should have affinity for SAM. Impurities on the surface and roughness of the bottom electrode affects orientation of the SAM, which in turn alter the properties of the SAM including charge transport [12]. In this study, ultra-flat Ag and Au were chosen as the bottom electrodes to minimize the effect of surface roughness. Over these years, many technologies have been developed that can be used as top electrodes to explore more characteristics of molecule electronic with SAMs, such as the multilayer graphene electrode [13], the mercury drop electrode (Figure 1.3c) [14], and the poly(3,4-ethylenedioxythiophene) stabilized with poly(4-styrenesulphonic acid) (PEDOT:PSS) electrode (Figure 1.3b) [15]. However, these methods are not ideal. The multilayer graphene electrode still has some limitations that have not been examined, also the solvent generated during the process is hard to remove[16]. The biggest problem of mercury drop electrode is that mercury is toxic and easy to invade in other materials [17]. For the PEDOT:PSS electrode, it is hard to reproduce the same experimental results since it requires a constant formulation [18]. Thus, the choice of top electrode is eutectic gallium–indium alloy (EGaIn) (Figure 1.3d). EGaIn were first proposed by Whitesides Group as a soft conformed top electrode [19] and has since dominated this area due to their roughness. As a liquid metal, EGaIn is easily shaped and is compliant leading to soft, non-damaging tips. Also, compared to other liquid metals, such as mercury, gallium-based liquid metals are non-toxic. The surface of EGaIn spontaneously reacts with oxygen in air to form an oxide layer with a thickness of about 0.7-3 nm (Figure 1.3d). The thickness of the oxide layer is measured during in different stage. In the early formation, the oxide layer will be liquid-like structure, and the thickness is about 0.7nm. As the time pass, the structure becomes more and

more stable, and the thickness finally reaches to 3nm. The presence of the oxide layer forms a strong shell to form a mechanically stable structure. Thus, the surface tension of the EGaIn droplet makes it a non-damaging top electrode [20]. With such a soft conformed top-electrode, charge transport (tunneling) across single molecules can be studied.

Tunneling is a temperature independent process. If the thickness of the barrier is thin, electrons have the probability to go through the barrier through tunneling since they are both particles and waves. The relation between tunneling current density  $J$  and thickness of barrier width  $d$ , is represented by the simplified Simmons' equation.

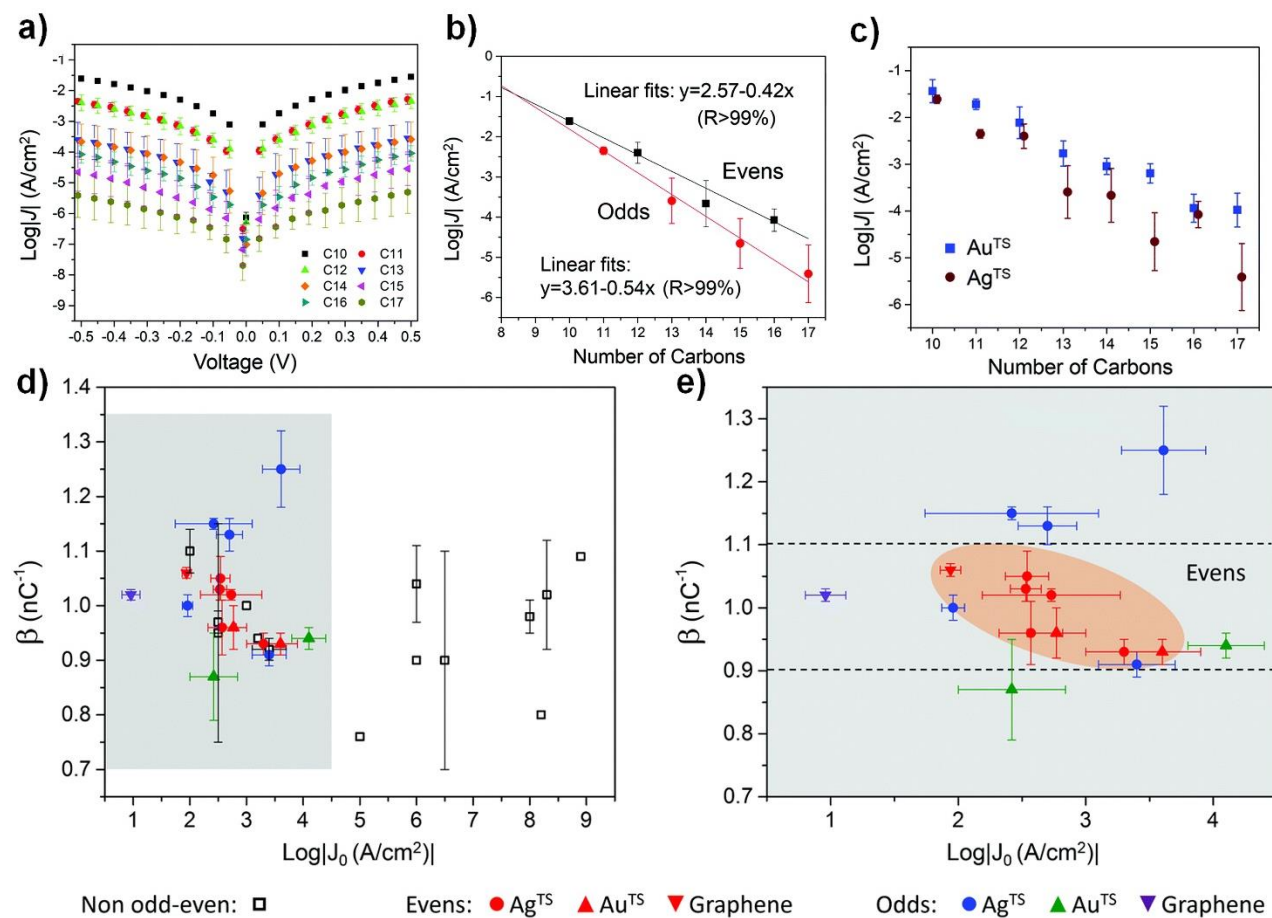
$$J = J_0 e^{-\beta d}$$

$\beta$  is the decay constant,  $J_0$  is the current density at  $d = 0 \text{ \AA}$ . Although the Simon equation explains the connection between electrical properties and physical barriers, it ignores the stereo electronic properties of the molecules that form the barrier. Since tunneling in SAMs is a through-bonding process, this property is very important. Geometrical constraints on charge transport can occur during the through-bonding process [21]. Barriers (resistance) to charge injection and the nature of the tunneling path can result in differences in tunneling probability, especially when using molecular structures such as linear hydrocarbons, which have random molecular conformations and a relatively high degree of rotational freedom. Thus, the Simmons' equation is replaced by the probabilistic Landauer formula, which can describe as:

$$G = \frac{e^2}{\pi h} T(E)$$

$$T(E) = \frac{\Gamma_1 \cdot \Gamma_2}{\left(\frac{\Gamma_1 + \Gamma_2}{2}\right)^2 + (E - E_{HOMO})^2}$$

where  $G$  is conductance,  $T(E)$  is transmission coefficient,  $\Gamma$  is molecule-electrode coupling strength, and  $E_{\text{HOMO}}$  is the Highest Occupied Molecular Orbital level of molecule. This equation focuses on capturing charge transport across molecular junctions.



**Figure 1.4.** (a) The plot of  $\text{log}|J|$  vs voltage of n-alkanethiols (n: carbon number 10-17). (b) Odd-even effect in n-alkanethiols on  $\text{Ag}^{\text{TS}}$ . (c) Comparison of tunneling rate at same voltage on Ag and Au (d) Survey on decay constant and inject current in different junctions, where the shaded area represents the effective data area. (e) Enlarged shaded area for more details [22]. (Figure reproduced from Ref. [22] with permission from the Royal Society of Chemistry. Copyright © 2018)

The motivation of this study is shown in Figure 1.4 [22]. Increasing molecular length will decrease current density (Figure 1.4a) and increase the odd-even effect (Figure 1.4b). Figure 4c shows that, as expected, SAMs with Ag substrate is more resistive than SAMs with Au substrate which are due to the difference of the molecule tilt angle and the lattice spacing [21]. Clearly even for the measurement on the same sample, there could be large difference and large deviation between each data point. Also, it can also be observed that there is a significant difference between measurements in this experiment and the literature data. Focusing on contact resistance,  $J_0$ , Figure 1.4d shows more than a 10-order difference in literature data. This implies that the contacts have more effect (variance) than the modeling sample. Similarly, the length dependent decay in conductance,  $\beta$ , shows variation from 0.7-1.3. Recent studies, however, have suggested an emerging coherence of  $J_0$  ( $\log|J_0| \approx 1.5-3.5 \text{ A/cm}^2$ ) and  $\beta$  (0.9-1.1) across Au and Ag surface (Figure 1.4e). The EGaIn top electrode and bottom electrode was made smoother, and the measurement repeated several times to remove the influence of defects in SAM. However, the observed values of  $J_0$  narrowed ( $\log|J_0| \approx 2.2-3.4 \text{ A/cm}^2$ ) but the values of  $\beta$  for Au ( $\beta^{\text{Au}} \approx 0.85$ ) and Ag ( $\beta^{\text{Ag}} \approx 1.25$ ) diverged. This data suggests that the contact point between the SAM and the electrode, as noted in Figure 1.4d, plays a significant role in charge transport characteristics of these junctions and therefore needs to be better understood.

Tunneling data from EGaIn system with different SAMs are analyzed. Because of the large amount of uncertainty and the fact that the molecular junction is a complex, dynamic system, traditional methods using Gaussian fitting are unable to capture all the details in the data. Gaussian fits, especially at a single bias, cannot demonstrate the extent of noise in the data. Moreover, when short bias ranges are used, Gaussian fits may falsely infer the reliability of the

data[23]. Thus, heat maps are used in data visualization to give a more complete picture of the measured data from EGaIn-based junctions. Even with noisy data, heat maps give a distribution of the data and capture any fluctuations with applied voltage. For single-molecule measurements, heat maps will reveal not only the quality of the contact, but also any voltage-dependent fluctuations[23].

Given the large volume of data derived from EGaIn junctions, it is challenging to draw quantitative inferences from heat maps, hence the need for statistical analysis. Population-dependent metrics like p-value or correlations (Pearson, Spearman, etc.) are likely to lead to rejection of the new hypothesis given the large volume of data. Population-independent and quantitative metrics, like unpaired mean differences and estimation plots have emerged as valuable tools to deduce and separate contradictory parameters. The estimate plot is designed to show confidence intervals for the difference between the original data and the means of all data. It uses different axis to focus on magnitude of the effect of a parameter (change), i.e., the mean difference[24]. The most important aspect of the Estimation plot is quantification. By reasoning quantitatively about the system under study, it is possible to determine whether the data is noteworthy or relevant to the topic and the correlation between data and hypothesized factors. We can also compare and pick combinations from different factors to indicate key influences.

Employing these approaches, we focus on external factors that can influence the Molecule – SAM interface. With datasets from different regions of the world, we first investigate the effect of humidity on observed charge tunneling characteristics. Second, Excessive pressure defecting in the SAM such as domain borders on the top electrode led to a large number of thin-area

defects, which causes a lot of high conductance values in the measurement result. If there exists overly humidity in the surrounding environment, it is likely that there is a layer of water attached to the top electrode. The capillary bridge will be formed between the electrode and the SAM, which causes the liquid to rise between the two surfaces and be held together by capillary forces. This phenomenon can generate the same effect as applying too much pressure, that is, a large number of high conductance values will be captured in the measurement.

Environment effect should be considered as a parameter in the probabilistic Landauer formula, which can be described as:

$$T(E) = \frac{\Gamma_1 \cdot \Gamma_2}{\left(\frac{\Gamma_1 + \Gamma_2}{2}\right)^2 + (E - E_{HOMO})^2} + \phi(E)$$

Where  $\phi(E)$  is the added environmental factor, and it should be the sum of possibilities of external factors affecting the measurement, including possibility of time period affects the measurement, the possibility of user affects the measurement, etc. The reason that environmental effect is an added factor but not an effect on  $\Gamma$  is that, while changes in molecule-electrode coupling strength are explained in terms of classical physics, the possibility of environmental influences is explained in terms of quantum physics, thus environmental factors cannot be defined as an effect on the coupling strength.

Another external factor is the user who completes the measurement. Even if the measurement environment and the molecular junction are identical, a researcher with a lot of experimental experience is not the same as a novice one. Inexperienced users are likely to have operational oversights and record keeping deficiencies. For example, a poor grasp of the timing of the

experiment or the pressure applied to the top electrode might produce a large amount of anomalous data. Finally, the time period and season in which the experiment was done could also play important roles. Since measurements are often done in the open, high traffic near the measuring storage may affect the quality of the data. During seasons when the temperature changes, such as spring, the fluctuating temperature leads to changes in the building HVAC systems and thus could affect the adventitious contaminations on the SAM. It is therefore important that all aspects of the ambient environment be considered given that: i) very low currents are measured (often picoAmps), ii) SAMs are very thin and any perturbation to the size (connect path) can significantly skew the data, iii) to minimize variabilities, it is essential that all environmental factors are mapped and quantified.

## REFERENCES

1. Mann, B. and H. Kuhn, *Tunneling through Fatty Acid Salt Monolayers*. Journal of Applied Physics, 1971. **42**(11): p. 4398-4405.
2. Aviram, A. and M.A. Ratner, *Molecular rectifiers*. Chemical Physics Letters, 1974. **29**(2): p. 277-283.
3. Razumov, V.F., *Molecular electronics: Problems and prospects*. Bulletin of the Russian Academy of Sciences: Physics, 2012. **76**(2): p. 194-197.
4. Whitesides, G.M. and M. Boncheva, *Beyond molecules: self-assembly of mesoscopic and macroscopic components*. Proc Natl Acad Sci U S A, 2002. **99**(8): p. 4769-74.
5. Casalini, S., et al., *Self-assembled monolayers in organic electronics*. Chem Soc Rev, 2017. **46**(1): p. 40-71.
6. Love, J.C., et al., *Self-assembled monolayers of thiolates on metals as a form of nanotechnology*. Chem Rev, 2005. **105**(4): p. 1103-69.
7. Sagiv, J., *Organized monolayers by adsorption. 1. Formation and structure of oleophobic mixed monolayers on solid surfaces*. Journal of the American Chemical Society, 2002. **102**(1): p. 92-98.
8. Bain, C.D. and G.M. Whitesides, *Formation of Monolayers by the Coadsorption of Thiols on Gold - Variation in the Length of the Alkyl Chain*. Journal of the American Chemical Society, 1989. **111**(18): p. 7164-7175.
9. Akkerman, H.B. and B. de Boer, *Electrical conduction through single molecules and self-assembled monolayers*. Journal of Physics: Condensed Matter, 2008. **20**(1).
10. Liu, R., et al., *Organometallic molecular rectification*. J Chem Phys, 2006. **124**(2): p. 024718.
11. Thuo, M.M., et al., *Odd-even effects in charge transport across self-assembled monolayers*. J Am Chem Soc, 2011. **133**(9): p. 2962-75.
12. Chen, J., et al., *Limits to the Effect of Substrate Roughness or Smoothness on the Odd-Even Effect in Wetting Properties of n-Alkanethiolate Monolayers*. Langmuir, 2015. **31**(25): p. 7047-54.
13. Wang, G., et al., *A new approach for molecular electronic junctions with a multilayer graphene electrode*. Adv Mater, 2011. **23**(6): p. 755-60.
14. Slowinski, K., et al., *Through-Bond and Chain-to-Chain Coupling. Two Pathways in*

- Electron Tunneling through Liquid Alkanethiol Monolayers on Mercury Electrodes.* Journal of the American Chemical Society, 1997. **119**(49): p. 11910-11919.
15. Akkerman, H.B., et al., *Towards molecular electronics with large-area molecular junctions.* Nature, 2006. **441**(7089): p. 69-72.
  16. Seo, S., et al., *Solution-processed reduced graphene oxide films as electronic contacts for molecular monolayer junctions.* Angew Chem Int Ed Engl, 2012. **51**(1): p. 108-12.
  17. Rampi, M.A. and G.M. Whitesides, *A versatile experimental approach for understanding electron transport through organic materials.* Chemical Physics, 2002. **281**(2-3): p. 373-391.
  18. Van Hal, P.A., et al., *Upscaling, integration and electrical characterization of molecular junctions.* Nat Nanotechnol, 2008. **3**(12): p. 749-54.
  19. Chiechi, R.C., et al., *Eutectic gallium-indium (EGaIn): a moldable liquid metal for electrical characterization of self-assembled monolayers.* Angew Chem Int Ed Engl, 2008. **47**(1): p. 142-4.
  20. Zhao, Z., et al., *Smart Eutectic Gallium-Indium: From Properties to Applications.* Adv Mater, 2023. **35**(1): p. e2203391.
  21. Belding, L., et al., *Conformation, and Charge Tunneling through Molecules in SAMs.* J Am Chem Soc, 2021. **143**(9): p. 3481-3493.
  22. Chen, J., et al., *Understanding interface (odd-even) effects in charge tunneling using a polished EGaIn electrode.* Phys Chem Chem Phys, 2018. **20**(7): p. 4864-4878.
  23. Sporrer, J., et al., *Revealing the Nature of Molecule-Electrode Contact in Tunneling Junctions Using Raw Data Heat Maps.* J Phys Chem Lett, 2015. **6**(24): p. 4952-8.
  24. Ho, J., et al., *Moving beyond P values: data analysis with estimation graphics.* Nat Methods, 2019. **16**(7): p. 565-566.

## Chapter 2

### Metadata-Driven Unveiling of Environment Effects on Large Tunnel Junctions

(Part of this work is submitted to the J. Phys. Chem. Letts)

#### Introduction

Theory and experiment have established that charge transport through alkyl chains occurs predominantly via a tunneling mechanism.[1-13]. We use a rectangular potential barrier as a first-order model of tunneling through a Metal-Insulator-Metal (MIM) junction [2, 4, 6-9]. In this model, the current density ( $J$ ; A/cm<sup>2</sup>) decreases exponentially with the increasing length of the path that a charge follows between the electrodes ( $d$ ; nm) as summarized in equation 1. The question of the trajectory of the electron from one electrode to the other—its origin at a step edge, or an impurity in the surface; whether it takes the shortest distance (through-space tunneling), or follows the path through orbitals of organic chains (through-bond tunneling); or effect of physisorbed impurities, and how it responds to discontinuities in the film—remains incompletely understood [14, 15]. Several researches [2, 4, 6] have found that through-bond tunneling dominates charge transport through SAMs of n-alkanethiolate.

Many groups have reported values of  $\beta$  of 0.8-1.0 Å<sup>-1</sup> for junctions with alkanethiolate SAMs, usually in systems with one covalent bond and one van der Waals interface (M-SAM//M)[4-7], or with two covalent bonds and one van der Waals interface (M-SAM//SAM-M) [1, 2, 7, 16].

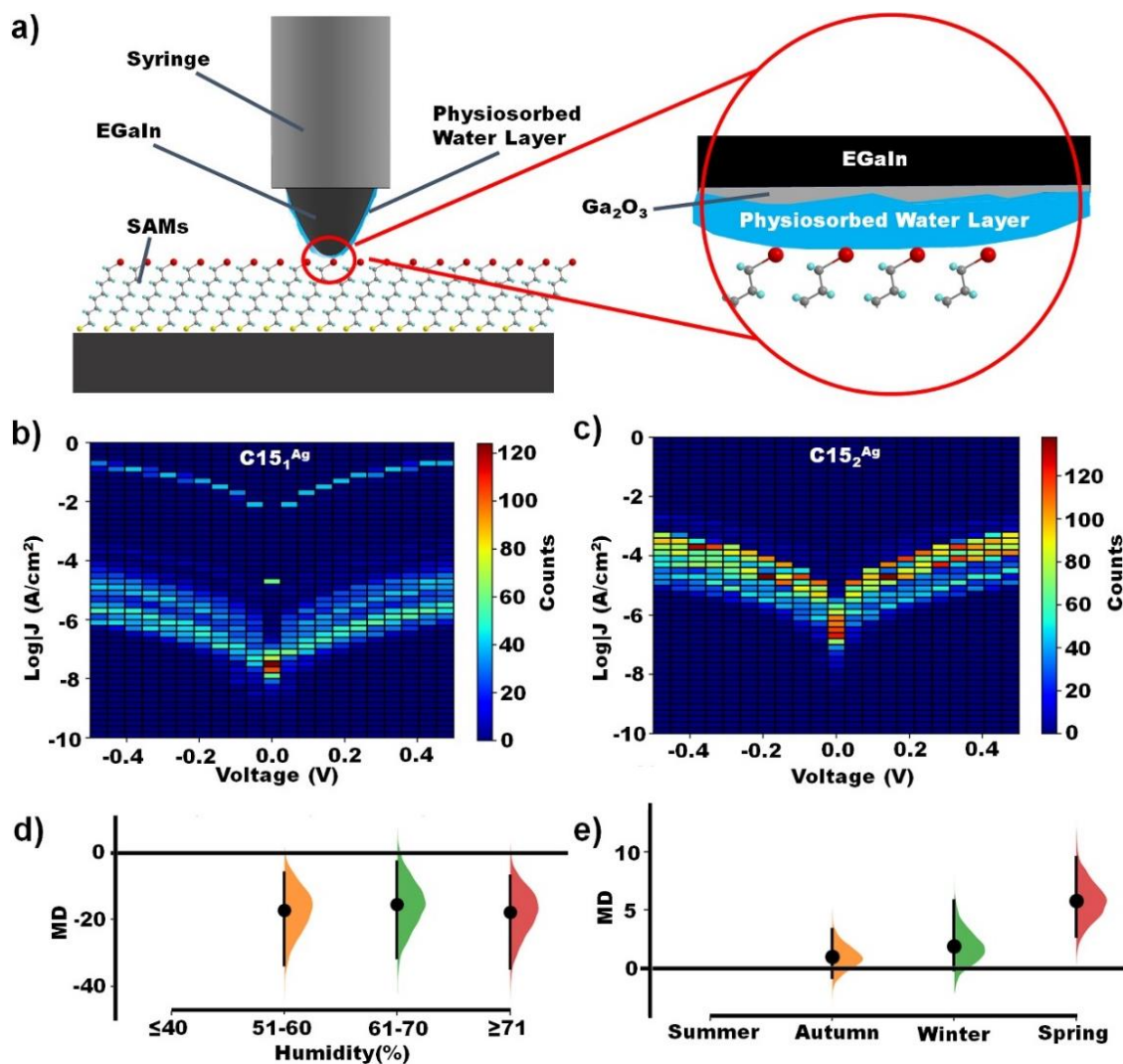
There is some controversy over the role of the metal-molecule interface in the determination of  $\beta$ . Frisbie and co-workers used conducting atomic force microscopy (cp-AFM) to compare the junctions M-SAM//M and M-SAM//SAM-M, and found that they had the same  $\beta$  (1.0 Å<sup>-1</sup>) [5].

Slowinsky and co-workers assembled SAMs formed from  $\alpha,\omega$ -dialkanethiol between silver and

mercury in an electrochemical cell, and changed one bond in situ from physisorbed ( $\beta=1.3 \text{ \AA}^{-1}$ ) to chemisorbed ( $\beta=1.0 \text{ \AA}^{-1}$ ) [17]. Cui et. al used scanning tunneling microscopy (STM) to show that junctions with a singly-bound alkanethiolate ( $\beta=0.8 \text{ \AA}^{-1}$ ) had values of  $\beta$  that were 75% higher than the  $\beta$  for junctions with a doubly bound  $\alpha,\omega$ -dialkanethiolate ( $\beta\sim 0.46 \text{ \AA}^{-1}$ ) [4]. By comparing transport through physisorbed SAMs of dialkyl sulfide to transport through chemisorbed SAMs of alkanethiolate, we simultaneously investigate the effects of the quality of the metal-molecule interface and the disorder in the SAM on the magnitude and distance-dependence of the current density through MIM junctions. Despite a somewhat coherence in the value of  $b$  and  $J_0$ , we showed that when the top and bottom electrodes are well polished to minimize surface asperities, these values shift with a concomitant increase in the difference between Au and Ag [18].

Nomenclature. The notation RSH is for alkanethiols denoted as HSC<sub>n</sub>. As previously [1-3], we denote the MIM junctions M1-SAM//M2, where M1 and M2 denote the metals of the bottom and top electrodes, SAM is the self-assembled monolayers supported on the bottom electrode, “-” indicates a chemical bond while “//” indicates a van der Waals (physisorbed) interface.

## Results and Discussion:

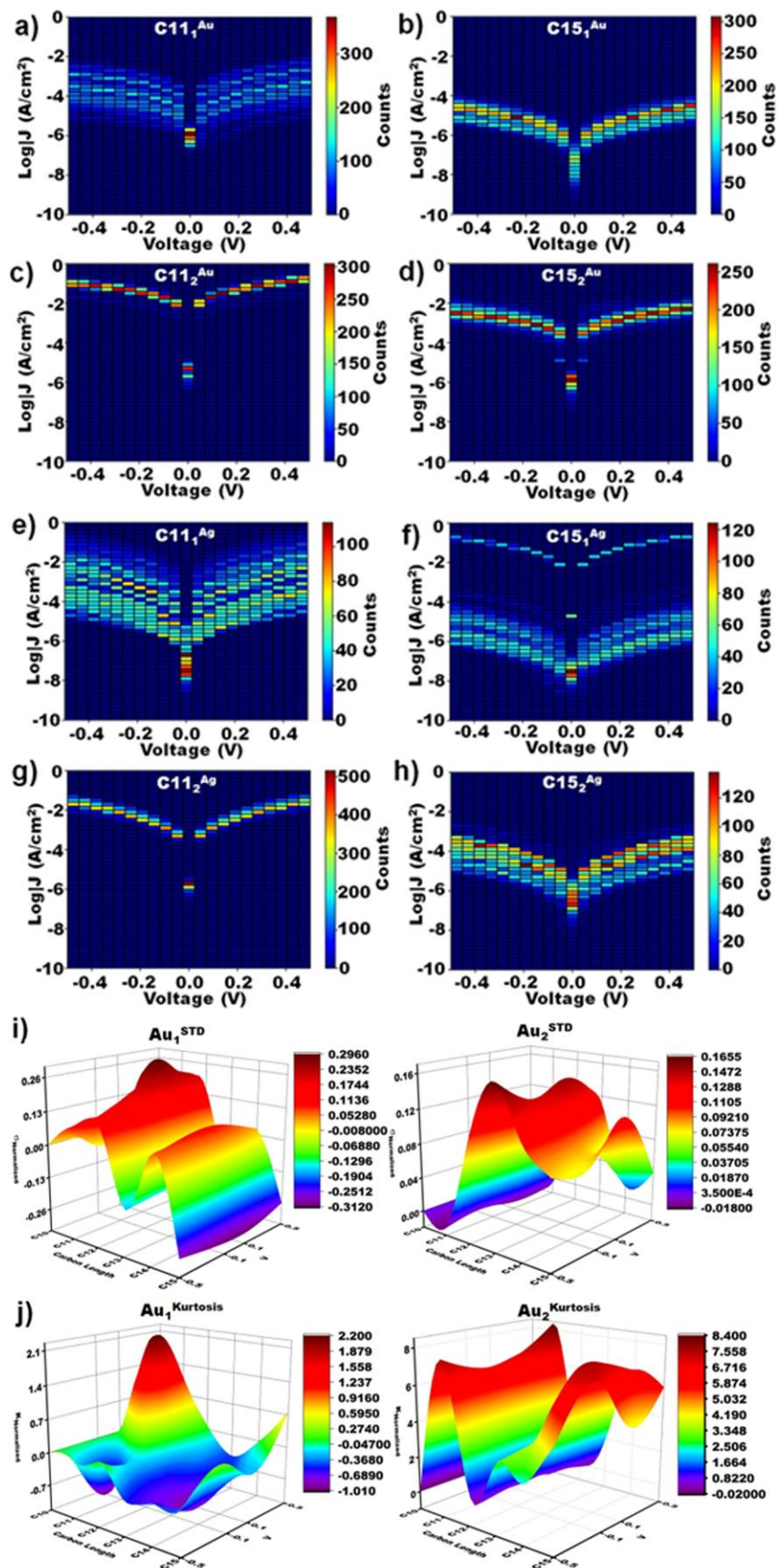


**Figure 2.1:** Schematic illustration of placement of physisorbed contaminants on a tunnel junction.

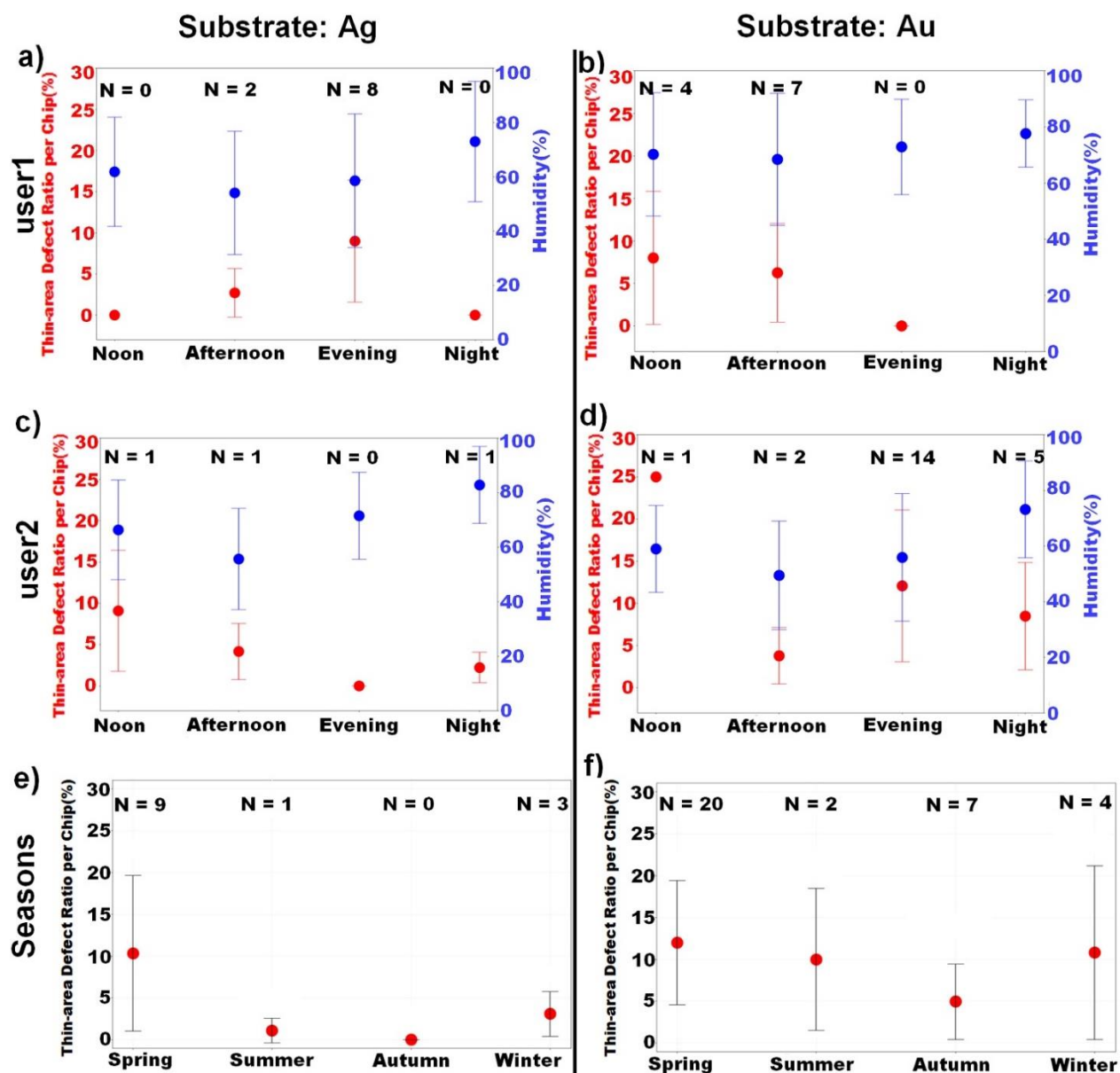
(a) Scheme of adopting EGaIn tunneling junction constructed with different carbon numbers of alkanethiols as the system for environmental factor investigation. (b-c) Heat maps constructed on data measured by the same sample (EGaIn system with carbon number 15 alkanethiol as SAM and silver as substrate), but with different users. (d-e) Estimation plot constructed from of all experimental data of the EGaIn tunneling junction measured in different humidity and different seasons.

Figure 2.1a schematically illustrates the position of the adsorbed adventitious contaminant – here using water as the main culprit. The insert illustrates perturbation of the barrier width due to the presence of the water layer. From this simple schematic, and from the simplified Simmons equation (equation 1), we can infer a reduced tunnel current from this barrier compared to one without the physisorbed contaminant. The caveat, however, is the likelihood that the adlayer can be electrostatically removed once a bias has been applied. The repercussion of a field-driven adlayer removal is that this would render the junction asymmetric since an applied forward and reverse bias would generate dissimilar fields on the adlayer. To better illustrate this, all data were analyzed and displayed as histograms (Figure 2.1b-c) with the user identified as a subscript. One thing needs to mention is that user 1 is actually a set of users who work together in the lab to complete data collection. Figure 2.1b for example shows the data from C<sub>15</sub> (pentadecane thiol) on Ag substrate from user 1 while Figure 2.1c shows similar data from user 2. From the histogram, user 1 had hard junctions (high voltage trace) and bimodal data while user 2 had a coherent data set albeit skewed towards the higher bias. From these data, effect of a variable on tunneling current is quantified using estimation plots (Figure 2.1d-e), a low unpaired mean difference implies that the variable under investigation is decreasing then the current density is decreasing and vice versa.

**Figure 2.2:** Summary of the n-alkanethiolate SAM data. (a-b) Heat maps constructed from carbon number 11 and 15 alkanethiol SAM with gold as substrate which collected by USER1 at different data collection times. (c-d) Heat maps constructed from carbon number 11 and 15 alkanethiol SAM and gold as substrate which collected by USER 2 at different data collection times. (e-f) Heat maps constructed from carbon number 11 and 15 alkanethiol SAM with silver as substrate which collected by USER1 at different data collection times. (g-h) Heat maps constructed from carbon number 11 and 15 alkanethiol SAM with silver as substrate which collected by USER2 at different data collection times. (i) Three-dimensional (3D) standard deviation plots comparing data measured when used different carbon number alkanethiol as SAM and gold as substrate from USER1 and USER2 (j) Three-dimensional (3D) kurtosis plots comparing data measured when used different carbon number alkanethiol as SAM and gold as substrate from USER1 and US



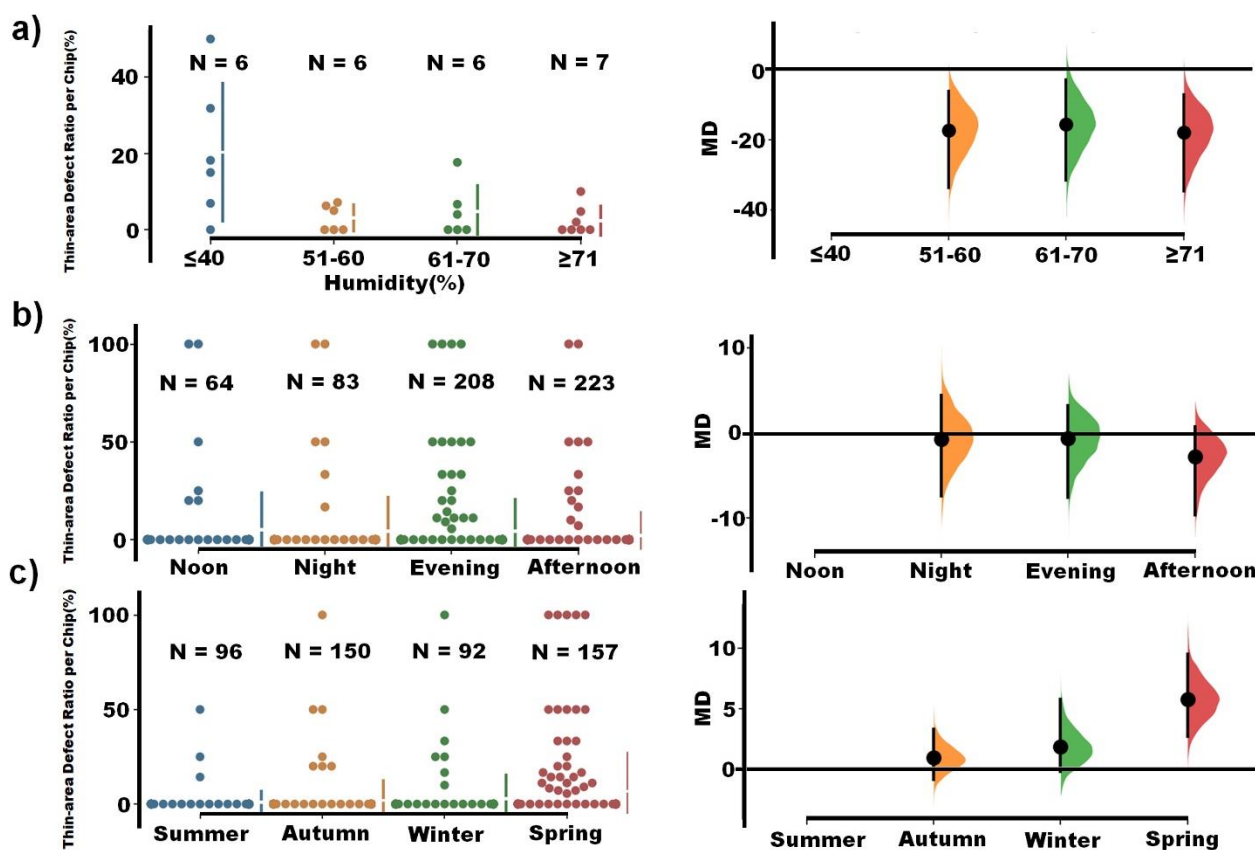
To further understand the effect of user-to-user we compare  $C_{15}$  and  $C_{11}$  SAMs on Ag and Au (Figure 2.2a-h). On Au, user 1 had a larger spread in their data than user 2. On Ag, however, we observe that user 2 had a better spread for  $C_{11}$  SAMs but the width of the spread was comparable across  $C_{15}$  on Ag although user 1 data shows hard contact. We can therefore infer that fluctuations across users are highly random. To further ascertain this, we look at the distributing of standard deviations (Figure 2.2i) and kurtosis (Figure 2.2j) of the spread across different chain lengths on Au and Ag for both users. There are no observable trends in the data suggesting that, as expected, these fluctuations are random and can therefore be mitigated by comparing large data sets and invoking randomness (Gaussian distribution). This is already being applied and therefore we can infer that user-to-user fluctuations are not the origin of the observed skew in data shown in Figure 2.2a-h.



**Figure 2.3:** Effect of user, time and season on data quality. Thin-area defect ratio with experiments executed by different users (USER1 and USER2) at different time periods (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM) for (a,c) using Ag as substrate (b,d) using Au as substrate (e-f) Relationship between thin-area defect ratio and experiment execution seasons of data collected from USER1 and USER2 with different substrate.

To further understand the origin of the variance in the data, we investigated the role of time-of-day when the experiment was done and seasons on charge tunneling. The time of day was broken into noon (10-2 pm), afternoon (2-6 pm), evening (6-10 pm) and night (10 pm -6 am). Given that humidity is likely to fluctuate during the day, we also correlated the day with changes in ambient humidity obtained from the weather data of the day of the experiment. We assume here that the humidity of the day will affect humidity in the room where the experiment is being run.

Irrespective of the user, molecule, or substrate being used, we observe a minor inverse correlation in the effect of humidity on number of thin-area defects (failed junctions) with humidity of ~65% leading to a lower number of failed junctions (Figure 2.3a-d). This is opposite of what we anticipated since a large amount of adsorbed water should lead to low current and challenging measurements. High humidity could also lead to better packing of the hydrophobic molecules leading to fewer failures and inducing an annealing-type effect. When we compared the failure rate to the seasons, we observe that in both Au and Ag, spring gave the highest number of thin-area defects with Autumn giving the lowest. The number of thin-area defects were also lower on Ag than on Au likely indicating the benefit of high packing density of the SAM on charge tunneling characteristics.



**Figure 2.4:** Population-independent statistical quantification of the role of different parameters.

(a) Estimation and mean difference plots of thin-area defect ratio and the humidity on the time period when experiment execution (b) Estimation and mean difference plots of thin-area defect ratio and experiment execution time (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM) (c) Estimation and mean difference plots of thin-area defect ratio and experiment execution seasons.

To ascertain and quantify these observations, estimation plots were made for humidity, time of day of the experiment and season. For humidity,  $<40\%$  humidity was used as a reference point showing that overall increase in humidity led to  $\sim -20$  decline in the tunneling current. The values were observed to be similar for all experiments with humidity  $>45\%$  (Figure 2.4a). Comparing all day collected at different times of day, however, showed unpaired mean differences around

zero. A slight but noticeable decline occurred during the afternoon time (Figure 2.4b). When we compared data from different seasons with the summer as the reference, we observe a slow but gradual increase. The spring semester led to the highest unpaired mean differences (Figure 2.4c).

**Conclusion:**

By looking into a set of datasets collected over years and by different users, we observe that user-to-user fluctuations are random as are fluctuations due to time of day. We note that humidity has a slight influence in the tunnel junction, but the biggest variable is in seasonal fluctuations. Although a higher number of failed junctions were observed in spring, we also observe a significant increase in charge tunneling in the spring semester (Figure 2.4c). It is therefore important that associated data for all reported values be saved and archived.

## References

1. Chabinyk, M.L., et al., *Organic polymeric thin-film transistors fabricated by selective dewetting*. Applied Physics Letters, 2002. **81**(22): p. 4260-4262.
2. Holmlin, R.E., et al., *Zwitterionic SAMs that Resist Nonspecific Adsorption of Protein from Aqueous Buffer*. Langmuir, 2001. **17**(9): p. 2841-2850.
3. Rampi, M.A. and G.M. Whitesides, *A versatile experimental approach for understanding electron transport through organic materials*. Chemical Physics, 2002. **281**(2-3): p. 373-391.
4. Cui, X.D., et al., *Changes in the Electronic Properties of a Molecule When It Is Wired into a Circuit*. The Journal of Physical Chemistry B, 2002. **106**(34): p. 8609-8614.
5. Wold, D.J. and C.D. Frisbie, *Fabrication and characterization of metal-molecule-metal junctions by conducting probe atomic force microscopy*. J Am Chem Soc, 2001. **123**(23): p. 5549-56.
6. Jing, W. and M. Lundstrom, *Ballistic transport in high electron mobility transistors*. IEEE Transactions on Electron Devices, 2003. **50**(7): p. 1604-1609.
7. Salomon, A., et al., *Comparison of Electronic Transport Measurements on Organic Molecules*. Advanced Materials, 2003. **15**(22): p. 1881-1890.
8. Selzer, Y., A. Salomon, and D. Cahen, *The Importance of Chemical Bonding to the Contact for Tunneling through Alkyl Chains*. The Journal of Physical Chemistry B, 2002. **106**(40): p. 10432-10439.
9. Holmlin, R.E., et al., *Electron transport through thin organic films in metal--insulator--metal junctions based on self-assembled monolayers*. J Am Chem Soc, 2001. **123**(21): p. 5075-85.
10. Haag, J., A. Vermeulen, and A. Borst, *The intrinsic electrophysiological characteristics of fly lobula plate tangential cells: III. Visual response properties*. J Comput Neurosci, 1999. **7**(3): p. 213-34.
11. Cahen, D. and A. Kahn, *Electron Energetics at Surfaces and Interfaces: Concepts and Experiments*. Advanced Materials, 2003. **15**(4): p. 271-277.
12. Rampi, M.A., O.J.A. Schueller, and G.M. Whitesides, *Alkanethiol self-assembled monolayers as the dielectric of capacitors with nanoscale thickness*. Applied Physics Letters, 1998. **72**(14): p. 1781-1783.
13. Wold, D.J., et al., *Distance Dependence of Electron Tunneling through Self-Assembled Monolayers Measured by Conducting Probe Atomic Force Microscopy: Unsaturated*

- versus Saturated Molecular Junctions*. The Journal of Physical Chemistry B, 2002. **106**(11): p. 2813-2816.
14. Reed, M.A., et al., *Conductance of a Molecular Junction*. Science, 1997. **278**(5336): p. 252-254.
  15. Ho Choi, S., B. Kim, and C.D. Frisbie, *Electrical resistance of long conjugated molecular wires*. Science, 2008. **320**(5882): p. 1482-6.
  16. Slowinski, K., H.K.Y. Fong, and M. Majda, *Mercury–Mercury Tunneling Junctions. I. Electron Tunneling Across Symmetric and Asymmetric Alkanethiolate Bilayers*. Journal of the American Chemical Society, 1999. **121**(31): p. 7257-7261.
  17. Sek, S., R. Bilewicz, and K. Slowinski, *Electrochemical wiring of alpha, omega-alkanedithiol molecules into an electrical circuit*. Chem Commun (Camb), 2004(4): p. 404-5.
  18. Chen, J., et al., *Understanding interface (odd-even) effects in charge tunneling using a polished EGaIn electrode*. Phys Chem Chem Phys, 2018. **20**(7): p. 4864-4878.

## CHAPTER 3

### **Metadata-Driven Unveiling of Time-period Effects on Large Tunnel Junctions and the Importance of Metadata**

#### **Introduction**

The creation of molecular electronics ideology in 1971 passionate researchers into the era of studying molecular electronic behaviors, as it pushes the Moore's law into a theoretical achievable highest resolution regime, namely molecular/atomic scale [1]. The experimental and analytical system design for molecular behavior study, however, becomes pivotal considering the challenges of achieving precise control and informative characterizations at molecule scale. Numerous trials and errors efforts have been done for developing the ideal single molecular systems, among which SAM (Self-Assembled Monolayer) coupled with EGaIn (Eutectic Gallium indium alloy) based measuring platform has been adopted as one of the standards considering its in reliability in monolayer structure fabrication as well as repeatable data collection [2]. Other standard single molecular fabrication and measurement systems were also investigated and adopted for different research purposes (e.g. tunable nanomaterial system [3], Structure-property relationships in charge tunneling [4]. Based on the above progresses, parameters affecting single molecule electronic behaviors have been experimentally observed, repeated and verified, which could be divided into two categories, namely internal factors and external factors. Internal factors are defined as systematic factors that impact directly on molecular electronic behaviors, such as molecule length /carbon number (odd-even effects)[5], molecule/electrode interfaces (substrate, electrode, defects) [2], and (local) dipoles (head group type) [6], etc. On the other hand, external factors affect measurements indirectly, either by interacting with internal factors (e.g. temperature, humidity, atmosphere) or affecting measurement systems directly (effect of user). While internal parameters

were generally set as the main controlled variable for experiments thus more thoroughly investigated, the effect of external factors were normally treated as random variables thus frequently ignored or fail to be addressed for single molecule systems.

Apart from system and experiment design, data collection and analysis are also of great importance for SAM systems, albeit challenging considering inherited chaoticities inside molecular-scale systems. With agreements on big data collection to achieve reliable information exploitation (multicycles on multi-spots), the adoption of “proper” data analysis methods is experiencing gradual evolution in the field. Conventionally, current density-voltage (J-V) results are fitted into Gaussian distributions for obtaining mean and standard deviation values, during which process extreme J values are treated as invalid data points (short or open) thus often discarded [7]. In the past decade, more statistical tools have been exploited in analyzing un-preselected raw data (heat map, lag plot, estimation plots etc.) [7]. As un-selected data theoretically reflects every detail that happened during the exact measurement, it contains information of both internal and external effects on SAMs when combined with big data collection. Recently, a work studied environmental effects on alkane SAM systems based on above methodologies, with the conclusion that humidity is a non-negligible external factor when investigating SAM behavior [2].

DAA is a multipurpose and easily synthesized synthon that is capable of generating different head groups in a variety of ways [8]. The polar nature of the amide group and their hydrogen bonding can form the chemisorption monolayers [9]. The amino group of one amino acid condenses with the carboxyl group of another amino acid, generating a peptide bond between them and combining to form a peptide [10]. The charge transport across amide-based SAMs is a multidimensional problem, which means every component in the system influences the behavior of SAMs [11].

Changes in head group degrees of freedom due to different structures of the head group (open vs cyclic) disrupt the bias-dependent stability of junctions [11].

The Fc head group contains both pi-orbitals and redox-active [12]. The HOMO level asymmetrically couples to electrodes in head groups due to the redox-active, thus the single molecular orbital appears between the Fermi levels of the electrodes only in one bias direction but not in the other [13]. At sufficient negative bias, when the HOMO level of Fc is vigorously accessible, charges are allowed to transport through the head group of SAMs, and the alkyl chain forms the only tunneling barrier for the SAM [13]. At positive bias, the HOMO level blocks charges, and becomes tunneling barrier with the alkyl chain [14]. Therefore, the width of the tunneling barrier is higher at negative bias than at positive bias, and the current flow at negative bias is higher than at positive bias [13]

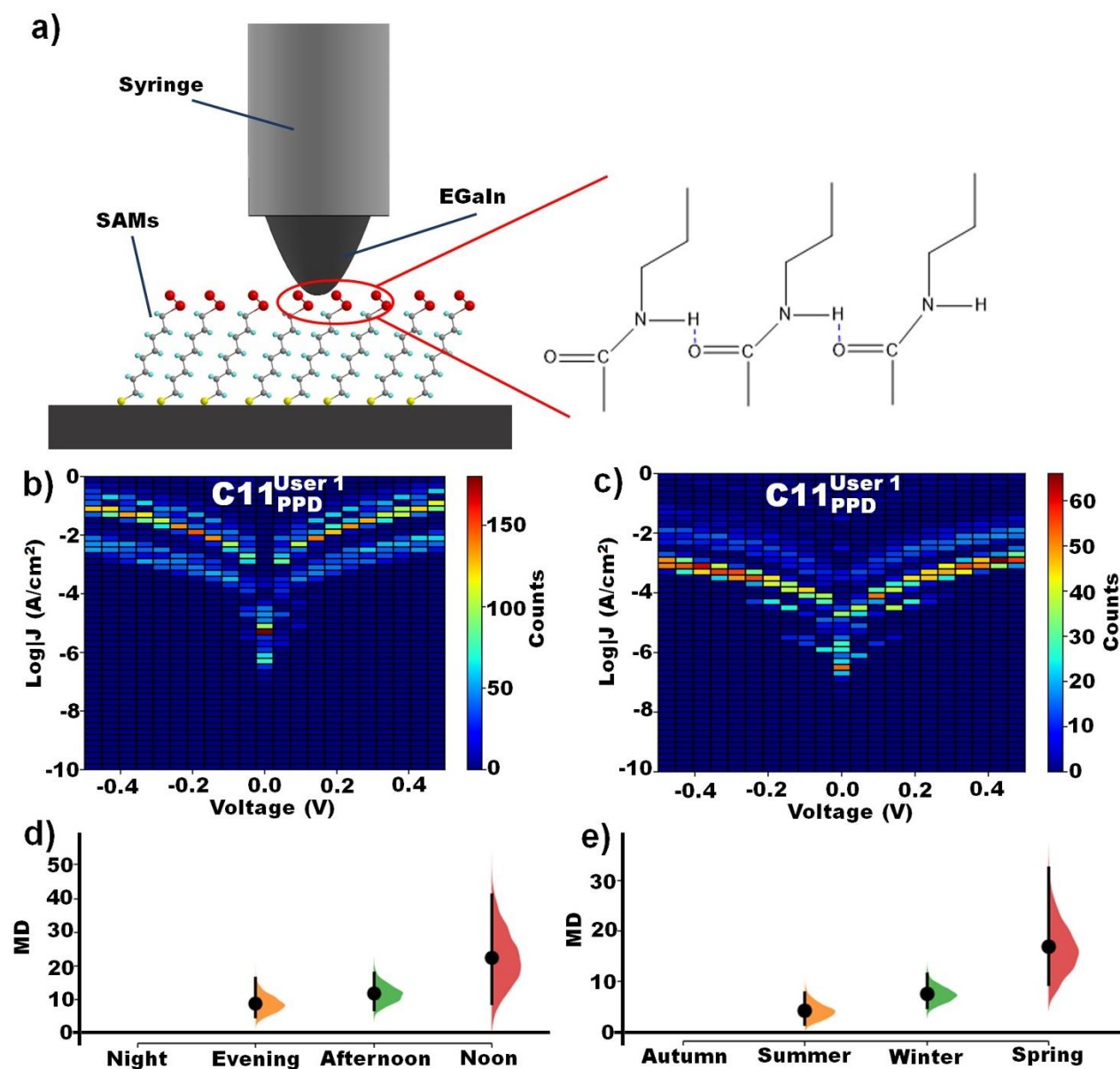
Charge transportation across SAM junctions is typically described by Simmons' equation[4]:

$$J = J_0 e^{-\beta d}$$

Where  $\beta$  is the decay constant,  $J_0$  is the current density that assumes  $d = 0 \text{ \AA}$ . Although the Simon equation directly relates current density with molecule length, it ignores the stereo electronic properties inside molecules.[11] On the contrary, Landauer formula addressed the randomness of possible molecular conformations[15]

$$T(E) = \frac{\Gamma_1 \cdot \Gamma_2}{\left(\frac{\Gamma_1 + \Gamma_2}{2}\right)^2 + (E - E_{HOMO})^2}$$

$T(E)$  is transmission coefficient,  $\Gamma$  is molecule-electrode coupling strength, and  $E_{HOMO}$  is the Highest Occupied Molecular Orbital level of molecule. For molecules containing head groups with non-symmetric electrical charge distributions, it is found that the electron conductance would be affected accordingly [13].



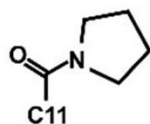
**Figure 3.1.** (a) Scheme of EGaIn tunneling junction system constructed with DAA molecules with the formation of peptide bonds. (b-c) Heat maps constructed on data measured by the same user (user 1), but at data in (b) is taken at 3pm and data in (c) is taken at 6pm. (d-e) Estimation plot constructed from all experimental data of the EGaIn tunneling junction with close-ring DAA as the head group of SAMs.

In this chapter, we adopted above methodologies for data analysis specifying Python methods, for the validation of our hypothesis that humidity would affect SAM behavior internally, especially for dipole SAM systems with which water molecules potentially interact dipolarly (Figure 3.1a). To address this point, pyrrolidine molecule (PRR head group) and piperidine molecule (PPD head group) were fabricated onto Au substrate by and adopted as the standard SAM systems for this study. For data collection, EGaIn was used as the top electrode together with Keithley measurement systems. For heat maps constructed from the data collected on the same SAM system (PPD) by the same user (USER1) but at different times, the data distribution is different with an obvious hard-contact junction formation in one of the measurements (Figure 3.1b, 3.1c). The most important aspect of the Estimation plot is quantification. By reasoning quantitatively about the system under study, it is possible to determine whether the data is noteworthy or relevant to the topic and the correlation between data and hypothesized factors. The measurement data of EGaIn tunneling junctions with DAA as the head group of SAMs will vary depending on the experimental time period and experimental season (Figure 3.1d, 3.1e).

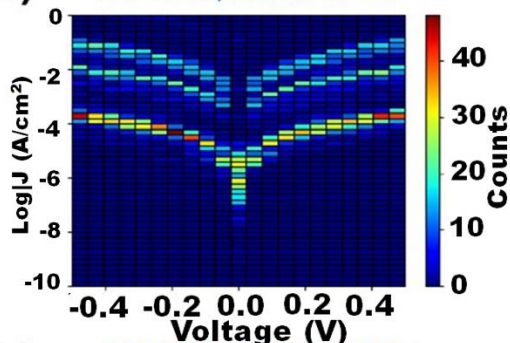
**Result and Discussion:**

**Figure 3.2.** (a-c) Heat maps constructed from carbon number 11 pyrrolidine (PRR) SAM collected by USER1 at different data collection times. (d-e) Heat maps constructed from number 11 piperidine (PPD) SAMs collected by both USER1 and USER2 at different data collection times. (g-h) Three-dimensional (3D) standard deviation plots comparing PRR and PPD SAMs containing different carbons in the molecule.

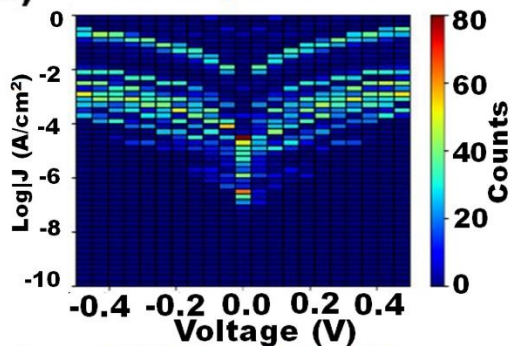
PRR



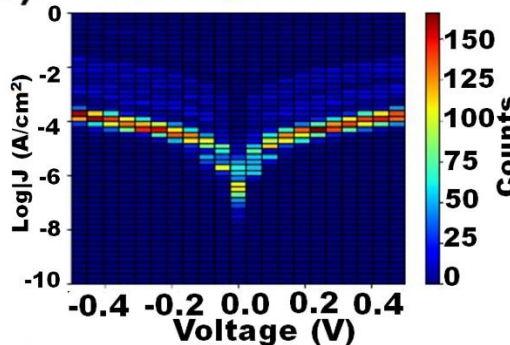
a) USER1, 5PM-6PM



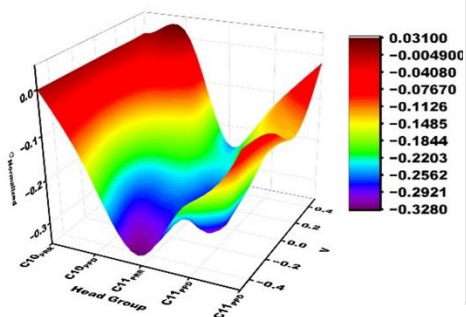
b) USER1, 2PM-4PM



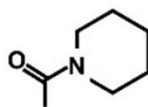
c) USER1, 2PM-4PM



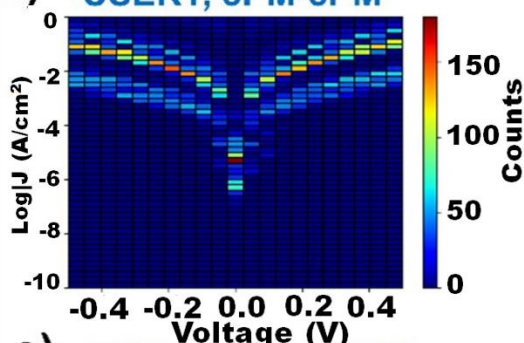
g)



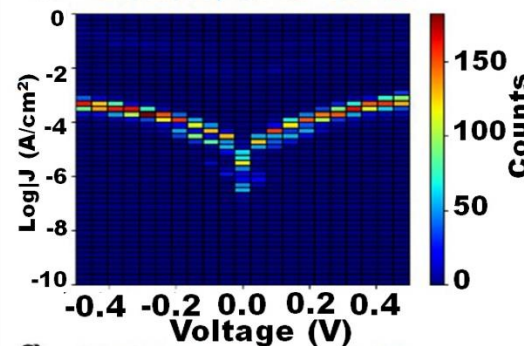
PPD



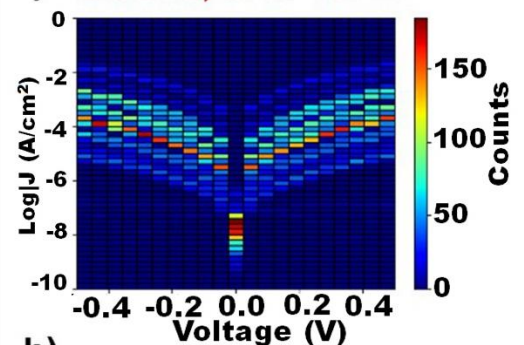
d) USER1, 3PM-5PM



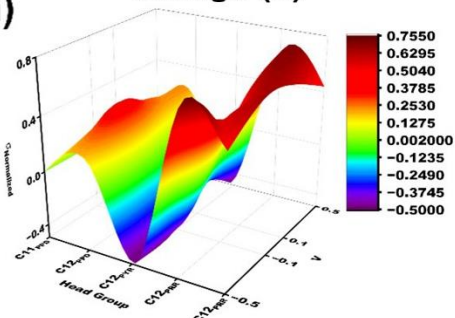
e) USER1, 9PM-10PM



f) USER2, 9PM-10PM

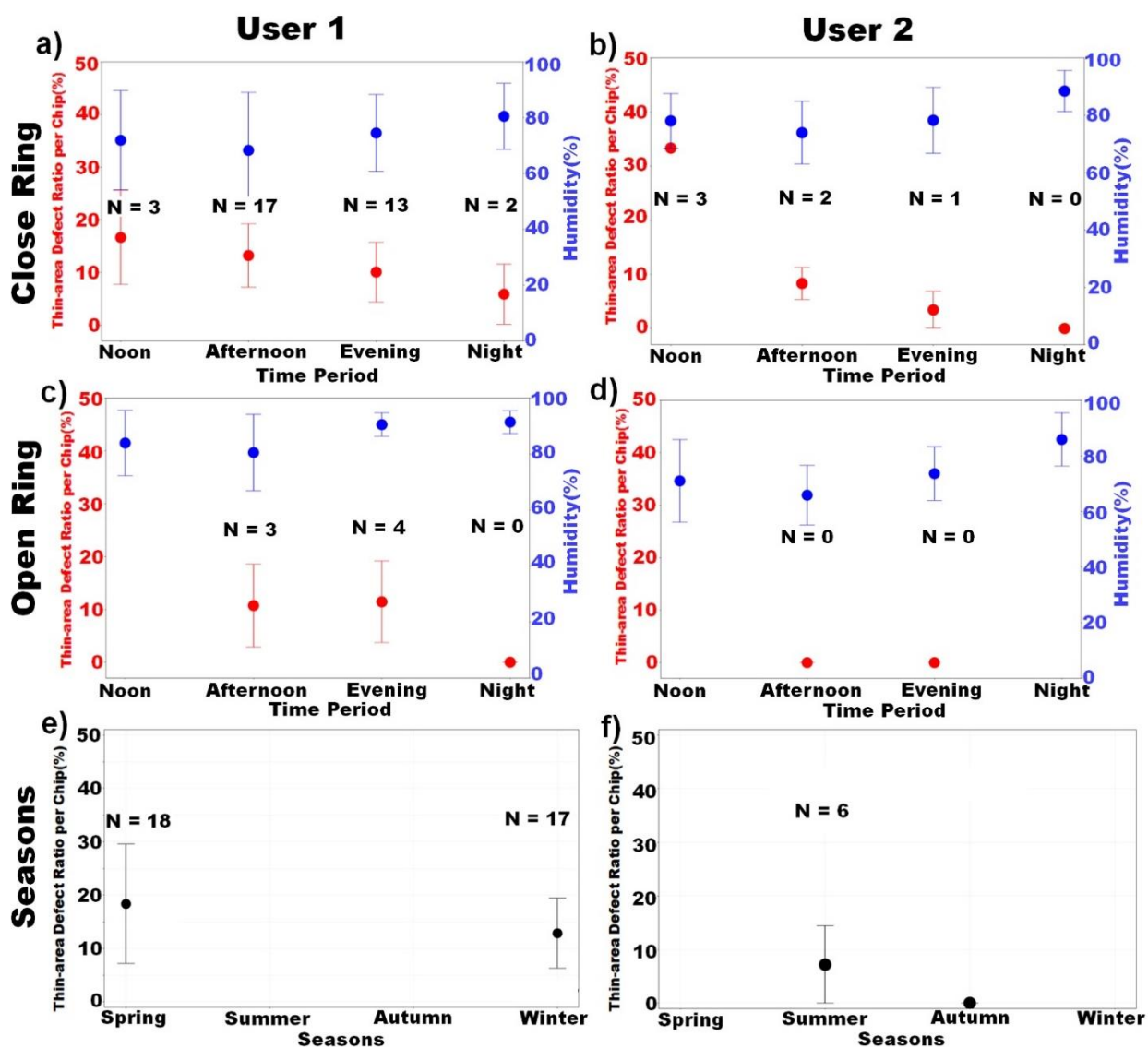


h)



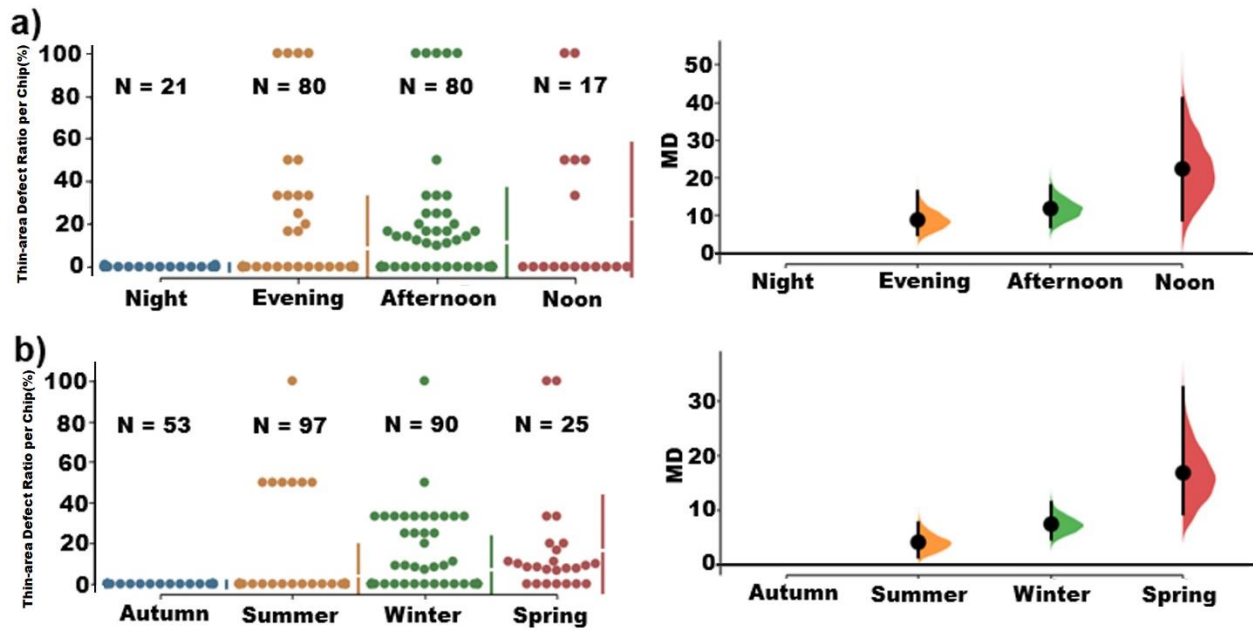
Firstly, user effects (USER1 and USER2) were studied by adopting heat maps constructed from SAM systems with same and same carbon length but different head groups (11 carbon number, PRR and PDD respectively). For the PRR SAM system, two experiments completed by user 1 between 5PM – 6PM and between 2PM – 4PM both exist hard contacts, together with noise presented from -2V to 2V in Figure 3.2b (Figure 3.2a-b). By the same user, there are no bad junction presences in the data measured from another experiment done between 2PM – 4PM (Figure 3.2c). For PPD SAMs, hard contacts existed in the heat map constructed form experiment performed by USER1 at 3PM-5PM, which disappeared when the experiment was conducted by the same user at 9-10PM period. For the same experiment USER2 executed between 4 pm to 8 pm, obtained heat map revealed no obvious hard contacts (Figure 3.2f).

3D standard deviations plots were constructed with normalized data. For USER1, the standard deviation (STD) of C11<sub>PPD</sub> at V=0.5V is -0.134 in the experiment that was done in several different months (December, February, March and April), but in the second experiment, which mainly finished on April, the STD decreased to -0.017 (Figure 3.2g). Using the carbon number 12 pyrrolidine (C12<sub>PRR</sub>) SAM, USER 2 first got data with a STD of -0.462 at 0.5V on a day that has humidity of 86.54%, while for the second experiment on same sample but in a different day that has 70.62% humidity, STD changed to 0.63 (Figure 3.2h). Since there exist clear variation of data constructed from same system with experiments done by different users, as well as for the same user, clear dependence exhibited between data and experiment executing period, we hypothesize here that time period, season, or location would affect data collected, with the emphasis on humidity as the internal factor behind all above external factors that affect the electronic behaviors fundamentally.



**Figure 3.3.** Thin-area defect ratio with experiments executed by different users (USER1 and USER2) at different time periods (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM) for (a-b) close ring DAA SAM systems (PPD and PRR) and (c-d) open ring DAA SAM systems (Diethyl Amide (DEA) and Dipropyl Amide (DPA)). (e-f) Relationship between thin-area defect ratio and experiment execution seasons of close-ring DAA SAM systems.

To further explore user and experiment time effect on electrical measurement, data collected from SAM constructed by open ring DAA molecules (DEA and DPA) were also analyzed and studied. When DAA with close-ring head groups were adopted, thin-area defect ratio per chip for both users were the highest (USER1: 16.67%, USER2: 33.33%) at noon (10AM-2PM), then the thin-area defect ratio of the measurement data from USER1 decreased to 13.23% at afternoon (2PM-6PM) and 10.08% evening (6PM-10PM), which finally reached to the lowest value of 5.88% at night (10PM-6AM). Similarly, USER2's data has a thin-area defect ratio of 8.33% in afternoon, 3.45% in evening, and reach to 0 at night (Figure 3.3a-b). When using open-ring DAA molecules to construct SAM, the thin-area defect ratio per chip that USER1 measured in afternoon and evening are 7.85% and 7.74% respectively (Figure 3.3c). Especially for USER2, there was no defects during the measurement (Figure 3.3d). Towards the close-ring DAA SAM system, thin-area defect ratio for USER1 when measurements were completed in spring and winter, were 19% and 13%, respectively. The thin-area defect ratio for USER2 when experiments were done in summer and fall were 8% and 0 respectively.



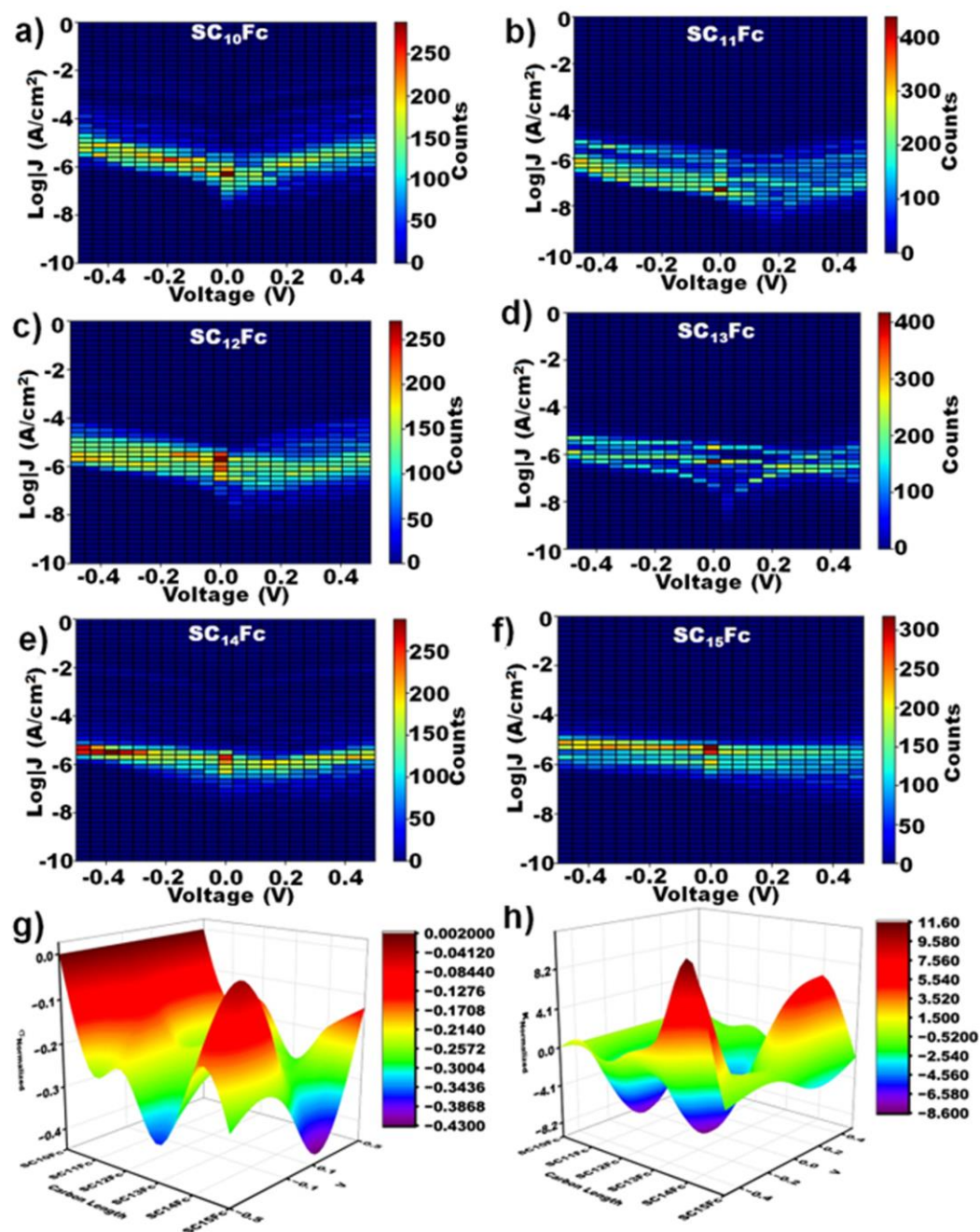
**Figure 3.4.** (a) Estimation and mean difference plots of thin-area defect ratio and experiment execution time (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM) (b) Estimation and mean difference plots of thin-area defect ratio and experiment execution seasons.

With estimation plot, the correlations between thin-area defect ratio per chip, time period and season are defined. By using thin-area defect ratio per chip of experiment in night as the standard, the mean difference of thin-area defect ratios gradually rise in the order of evening (8.8), afternoon (11.8), and noon (22.5) Figure 3.4a). When taking the thin-area defect ratio of autumn as standard, the mean difference of thin-area defect ratios rises gradually in the fall (4.12) and winter (7.39) and rises abruptly in the spring (16.8) (Figure 3.4b).

Specifically, experiments that were done at noon had the highest thin-area defect ratio (16.67% from USER1, 33.33% from USER2) and reached minimum in the night (5.88% from USER1, 0%

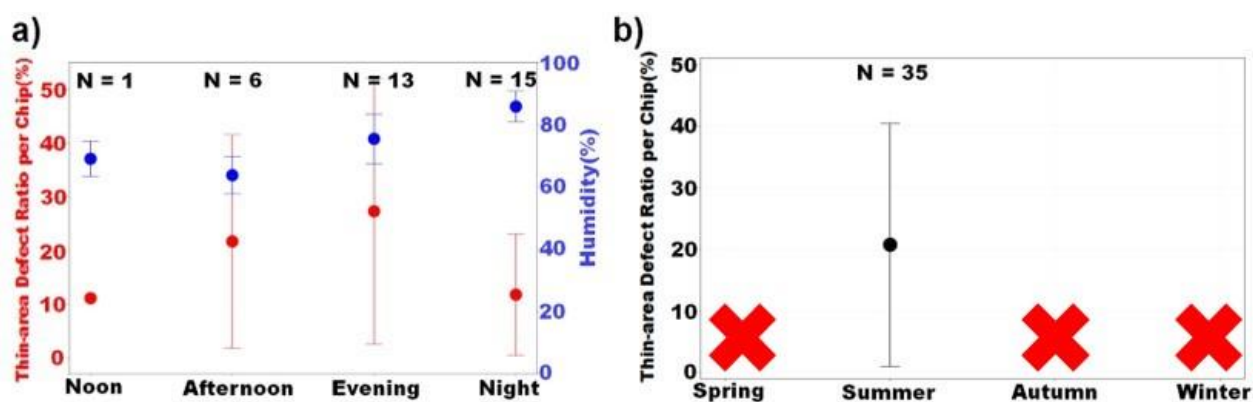
from USER2) for molecule tunneling junctions with close-ring DAA head groups. Since amide hydrogen bonds exist in head groups of DAA, the stability between head groups will be enforced due to the formation of peptide bonds. However, the beneath alkanethiol, which plays a supportive role under the head group, only has a simple chemical structure. This part is fragile and susceptible to external influences compared to the multimolecular head groups. In the noon, there is a high volume of traffic both inside and outside the laboratory. Even if people just go pass through, it is possible to affect the structure of the whole system, and thus have an impact on the measurement results. In the night there is almost no one around, which greatly reduces the probability of being influenced by the outside world, and thus measurements have the lowest thin-area defect ratio per chip.

Towards season effect, spring had the highest thin-area defect ratio per chip that reaches 18.36% and autumn contains the lowest (0%) (Figure 3.3e-f). The fluctuating temperature in Spring causes the air conditioning system to malfunction. High amount of precipitation in the spring raises the humidity level in the room, which increases the probability of water molecules to aggregate between the EGaIn tip and the SAM. As a result, the formation of capillary bridges leads to more hard contacts, and thus causes an increase in thin-area defect ratio.



**Figure 3.5** (a-f) Heat maps of SAM systems made with alkanethiol containing ferrocene headgroups (SC<sub>n</sub>Fc) with varying carbon number n=10 to 15. (g) Three-dimensional (3D) standard deviation (STD) plots showing STD values for varying n and at different voltages. (h) Three-dimensional (3D) kurtosis plots showing values for varying n and at different voltages

To further investigate external factors on head-group containing SAM systems, alkanethiol molecule containing ferrocene group ( $SC_nFc$ ) was chosen as standard for its rectifying electronic behaviors. This specific dataset was from one of our collaborators. We investigated here was acquired by novice users with a poor indoor HVAC system. Since the Fc head group is polar and redox-active, heat maps are expected to show asymmetrical curves and behavior as rectifier. However, due to the bad environmental environment and novice users, all  $SC_nFc$  SAMs with varying carbon numbers  $n$  behaved similarly, directly illustrated by heat maps with data straight line distribution trends (Figure 3.5a-f). Standard deviation show little difference between positive and negative bias. For example,  $SC_{14}Fc$  has an STD of -0.19 at 0.5V and a STD of -0.2 at -0.5V.  $SC_{13}Fc$  has a STD of -0.22 at 0.1V and a STD of -0.4 at -0.1V. When all other samples have negative kurtosis after normalization,  $SC_{14}Fc$  has kurtosis of 6.44 at 0.5V and of 11.6 at -0.5V (Figure 3.5g-h). These fluctuations are intense and unpredictable.

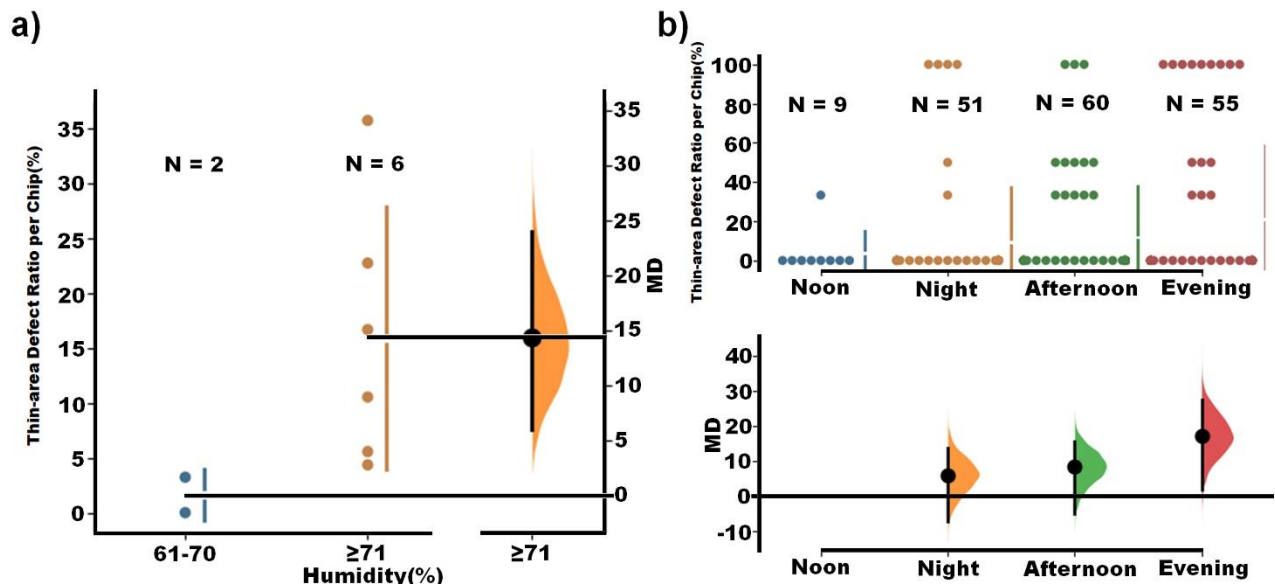


**Figure 3.6** Thin-area defect ratio and humidity plot of SAM constructed by  $SC_nFc$  ( $n = 10-15$ )

with measurement execution time (a) and seasons (b) (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM).

To further understand the quality of the data set, the correlation between the thin-area defect ratio per chip and measurement execution time period is plotted in Figure 3.6a. The data collected in

evening has the highest thin-area defect ratio per chip (27.27%) with a STD of 24.73. The thin-area defect ratio decreases to 21.67% in the afternoon with a STD of 19.84. During noon and night, the thin-area defect ratio is getting close (11.11% in noon, 11.76% in night) (Figure 3.6a). Since all experiments were executed in summer, the thin-area defect ratio per chip reaches 20.71%, and the season-related STD value is 19.76. (Figure 3.6b)



**Figure 3.7** (a) Estimation plots between thin-area defect ratio and humidity for SAM made with  $SC_nFc$  ( $n = 10-15$ ). (b) Estimation and mean difference plots between thin-area defect ratio and measurement execution time periods (Noon: 10AM-2PM, Afternoon: 2PM-6PM, Evening: 6PM-10PM, Night: 10PM-6AM).

A further analysis with estimation plots is revealed in Figure 3.7. A total of eight data points show a deficiency of the data (Figure 3.7a). Figure 3.7b displays the thin-area defect ratio will be lowest at noon, then gradually increase with night and afternoon and reach the highest at evenings. With noon as normalization, the mean difference of thin-area defect ratios rises gradually in the night (5.77) and afternoon (8.24) and rises abruptly in the evening (17.2) (Figure 3.7b).

The problem of this SC<sub>n</sub>Fc data set is the standard deviation for thin-area defect ratio is very large for both time periods and seasons. The poor quality of the data set can be misleading, and thus it is inappropriate to draw any conclusions. This example reflects the importance of metadata. With mining metadata, the reliability of the measurement result can be proved, and thus derive reliable conclusions.

### **Conclusion:**

In conclusion, we adopted Python methods, to generate heat maps, dot plots and estimation plots for the analysis of SAMs data acquired by different users at different time periods. Together with heap map and estimation plots, we obtain the following conclusions: 1) For SAM constructed by alkanethiol containing close-ring head groups (close-ring DAA), the measurement result will be affected by time and season. The thin-area defect ratio per chip will reach the highest in spring, and the lowest in autumn. Since SAMs are easily influenced by the external environment, high traffic nearby will increase the thin-area defect ratio. Thus, experiments at noon have the highest thin-area defect ratio, and at night have the lowest. 2) For SAM constructed by alkanethiol containing open-ring head groups (open-ring DAA), time and seasons will not apparently affect the thin-area defect ratio per chip, but more data is needed to support this conclusion 3) Analysis of metadata is very important, which can prove the reliability of the data set. 4) In summary, humidity serves as an internal factor that fundamentally affect the molecule behavior through dipole-dipole interactions, especially for SAM systems containing dipolar head groups.

## REFERENCES

1. Mann, B. and H. Kuhn, *Tunneling through Fatty Acid Salt Monolayers*. Journal of Applied Physics, 1971. **42**(11): p. 4398-4405.
2. Chen, J., et al., *Understanding interface (odd-even) effects in charge tunneling using a polished EGaIn electrode*. Phys Chem Chem Phys, 2018. **20**(7): p. 4864-4878.
3. Du, C., et al., *Substrate Roughness and Tilt Angle Dependence of Sum-Frequency Generation Odd-Even Effects in Self-Assembled Monolayers*. The Journal of Physical Chemistry C, 2022. **126**(16): p. 7294-7306.
4. Chen, J., et al., *Understanding Keesom Interactions in Monolayer-Based Large-Area Tunneling Junctions*. J Phys Chem Lett, 2018. **9**(17): p. 5078-5085.
5. Chen, J., et al., *Spectroscopic evidence for the origin of odd-even effects in self-assembled monolayers and effects of substrate roughness*. Phys Chem Chem Phys, 2017. **19**(10): p. 6989-6995.
6. Carlotti, M., et al., *Conformation-driven quantum interference effects mediated by through-space conjugation in self-assembled monolayers*. Nat Commun, 2016. **7**: p. 13904.
7. Sporrer, J., et al., *Revealing the Nature of Molecule-Electrode Contact in Tunneling Junctions Using Raw Data Heat Maps*. J Phys Chem Lett, 2015. **6**(24): p. 4952-8.
8. Ilangoan, A., S. Pandaram, and T. Duraisamy, *N,N-Dialkyl Amides as Versatile Synthons for Synthesis of Heterocycles and Acyclic Systems*. 2020.
9. Son, Y.J., et al., *Formation and Thermal Stability of Ordered Self-Assembled Monolayers by the Adsorption of Amide-Containing Alkanethiols on Au(111)*. Int J Mol Sci, 2023. **24**(4).
10. Libretexts, *13.2 Peptides*. 2020.
11. Du, C., et al., *Molecular Conformation in Charge Tunneling across Large-Area Junctions*. J Am Chem Soc, 2021. **143**(34): p. 13878-13886.
12. Reus, W.F., et al., *The SAM, not the electrodes, dominates charge transport in metal-monolayer//Ga<sub>2</sub>O<sub>3</sub>/gallium-indium eutectic junctions*. ACS Nano, 2012. **6**(6): p. 4806-22.
13. Nijhuis, C.A., W.F. Reus, and G.M. Whitesides, *Mechanism of rectification in tunneling junctions based on molecules with asymmetric potential drops*. J Am Chem Soc, 2010. **132**(51): p. 18386-401.
14. Chen, X., et al., *Molecular diodes with rectification ratios exceeding 10<sup>5</sup> driven by electrostatic interactions*. Nat Nanotechnol, 2017. **12**(8): p. 797-803.

15. Chen, J., et al., *(Invited) Towards a Perfect Junction: Effect of Smoothing Electrodes on Tunneling Behaviors across Large Area Molecular Junctions*. ECS Transactions, 2018. **86**(3): p. 79-87.

## Chapter 4

### General Conclusion

This goal of thesis is to mine metadata to improve charge tunneling measurement. In this study, data from multiple tunneling junctions are used for analysis. All tunneling junctions use EGaIn as the top electrode, and Au or Ag as bottom electrode. First the data from the tunneling junction with n-alkanethiol as the SAM is analyzed, based on the analyzed result, heat maps and estimation plots are generated (Chapter 2). The same procedure is repeated on DAA and SC<sub>n</sub>Fc (Chapter 3).

Chapter one is a general introduction to molecular electronics, and why we were interested in mining metadata of charge tunneling measurement.

In chapter two n-alkanethiol is used as the SAM. The heat map and the estimation plot are developed to analysis the data measured. Seasonal changes have been shown to have an impact on measurement results. Experiments in autumn have the lowest thin-area defect ratio per chip, and experiments in spring have the highest. We believe that the indoor air condition system can failure due to fluctuating temperatures in the spring. With more rainfall in the spring, water molecules are more likely to accumulate between the EGaIn tip and functional groups of SAMs. The formation of capillary bridges leads to hard contact and thus creating more thin-area defects.

Similarly, chapter three analysis the data set of DAA and SC<sub>n</sub>Fc by the heat map and the estimation plot. We get the same conclusion on the DAA as we got on the n-alkanethiol in Chapter 2. Fall will have the lowest thin-area defect ratio per chip and spring will have the

highest. Thus, the probabilistic Landauer formula should consider environment effect as a parameter, which should be modified as:

$$T(E) = \frac{\Gamma_1 \cdot \Gamma_2}{\left(\frac{\Gamma_1 + \Gamma_2}{2}\right)^2 + (E - E_{HOMO})^2} + \Phi(E)$$

Where  $\phi(E)$  is the added environmental factor and it should be the sum of possibilities of external factors affecting the measurement, including possibility of time period affects the measurement, the possibility of user affects the measurement, etc. Also, the time period in which the experiment was performed will affect SAMs that contain multiple molecular composed functional groups such as DAA. Because backbone chains are vulnerable to external influences, the thin-area defect ratio will be highest in the noon since there's the most flow of people. Instead, it reaches a minimum in the night. Also, the data measured in extreme environments, such as extremely high humidity, can be very noisy and does not match the actual theory. We believe that this kind of data is not suitable for further analysis or for deriving conclusions.

## **Chapter 5**

### **Future Steps**

In the future, a protocol that specifies universal data processing and reporting should be established for collecting more charge tunneling measurement results. With the uniform format, measurement data derived from SAM-based junction and all other molecular electronics systems can be easily categorized, and a database should be established to centrally store data in the repository. Since the importance of metadata is already shown in the thesis, the database can be used to identify bad datasets, thus eliminating data discrepancies between labs. Once the database reaches a significant size, analysis results will prove the optimal time period, season, and humidity for tunneling junction measurement. We can also take these good datasets and model them through machine learning. As new data is added, the model will be iterated several times until it is complete and reliable. Ultimately, these models can be loaded into an automated remote laboratory operating system such as ChemOS, allowing for error-free measurements without human intervention.

**APPENDIX**

## Appendix A

All plots were created by the Python code developed on Spyder. The method to make estimation plot referred 'dabest' library (<https://github.com/ACCLAB/DABEST-python>), and the weather data was collected from Visual Crossing Weather (<https://www.visualcrossing.com/>). Data was sorted into a hierarchal folder setup. The Python Code compiled could capture applied voltage, current density, absolute current density, current, experiment time and date that provided in the data set, arrange them and finally visualize them into heat maps, dot plots and estimation plots. To load a data set, special formatting was required for code to recognize. Once the code was loaded in Spyder and started running, internal method should automatically show plots. Code and used dataset can be found in [Github](#). 3D-plots were made by using OriginLab to show standard deviation in different voltage and carbon length. Standard deviation was calculated by the equation below:

$$SD = \sqrt{\frac{\sum(X_i - X_m)^2}{n - 1}}$$

Due to space constraints, the python code used to analyze data of n-alkanethiol is placed here. The logic of the code used to analyze the other data is the same, except some little details such as folder names are different. All the code can be found in our group GitHub:

### Capture history weather data from [data base](#):

```
#include sections
#values include days,hours,current,alerts
Include="hours"
df = pd.DataFrame()
#basic query including location
ApiQuery=BaseURL + Location
if (len(StartDate)):
    ApiQuery+="/"+StartDate
    if (len(EndDate)):
        ApiQuery+="/"+EndDate
ApiQuery+="?"
if (len(UnitGroup)):
    ApiQuery+="&unitGroup="+UnitGroup
if (len(ContentType)):
    ApiQuery+="&contentType="+ContentType
if (len(Include)):
    ApiQuery+="&include="+Include
ApiQuery+="&key="+ApiKey
try:
    CSVBytes = urllib.request.urlopen(ApiQuery)
except urllib.error.HTTPError as e:
    ErrorInfo= e.read().decode()
    print('Error code: ', e.code, ErrorInfo)
```

```

    sys.exit()
except urllib.error.URLError as e:
    ErrorInfo= e.read().decode()
    print('Error code: ', e.code,ErrorInfo)
    sys.exit()
CSVText = csv.reader(codecs.iterdecode(CSVBytes, 'utf-8'))
RowIndex = 0
for Row in CSVText:
    if RowIndex == 0:
        FirstRow = Row
    else:
        print('Weather in ', Row[0], ' on ', Row[1])
        ColIndex = 0
        for Col in Row:
            if ColIndex >= 4:
                if(FirstRow[ColIndex] == 'humidity'):
                    df = df.append({'Time':Row[1][11:], 'Humidity':Row[ColIndex]},
ignore_index=True )
                    break
                ColIndex += 1
        RowIndex += 1
df.to_csv('E:\SMMT Group\Weather CSV\Shawn Data in Paper\C10CNC2\\' + StartDate +
.csv')
if RowIndex == 0:
    print('Sorry, but it appears that there was an error connecting to the weather server.')
    print('Please check your network connection and try again..')
if RowIndex == 1:
    print('Sorry, but it appears that there was an error retrieving the weather data.')
    print('Error: ', FirstRow)

```

### **Visualization data by heat map:**

```

RealFull = pd.DataFrame()
# Need to change due to different file path
pathfolder=r'E:\SMMT Group\Origin Data\New data from Martin\MAA-
monoAlkylAmides\20090812(C11NHBu 3h)'
def getExperienceTime(Series):
    time = []
    currentLength = 0
    for file in Series:
        f = open(file)
        count = 0
        for lines in f :
            if(lines[0].isnumeric()) :
                time.append(lines[0:-1])
                count += 1
        if(count != 1):

```

```

    time[currentLength] = time[-1]

    for i in range(count):
        if(len(time) > currentLength + 1):
            del time[- 1]
        currentLength += 1
    return time
def find_related_files(txt_files):
    related_files = []
    times = []
    for txt_file in txt_files:
        directory = os.path.dirname(txt_file) # Get the directory of the TXT file

        base_name = os.path.basename(txt_file) # Get the base name of the TXT file
        base_name_without_data = base_name.replace("_data.txt", "") # Remove the "_data.txt"
suffix
        files_in_directory = os.listdir(directory)
        for file in files_in_directory:
            if file.startswith(base_name_without_data) and not file.endswith(".txt"):
                related_file_path = os.path.join(directory, file)
                related_files.append(txt_file) # Keep the original TXT file path

                # Extract time from the file content using the getExperienceTime function
                time = getExperienceTime([related_file_path])
                if time:
                    times.append(time[0])
                else:
                    times.append(None)

    # Create a DataFrame with related files and their corresponding times
    df = pd.DataFrame({"TXT File": related_files, "Time": times})
    return df
'''
Determine the time period of the measurement
'''
def findTimePeriod(files, time):
    morning = []
    noon = []
    afternoon = []
    evening = []
    night = []
    for index in range(len(time)):
        file = files[index]
        if time[index][-2:] == "AM": # Starting with AM times
            hour = int(time[index][0:time[index].index(':')])

```

```

    #print(hour)
    if 6 <= hour < 10:
        morning.append(file)
    elif hour < 6:
        night.append(file)
    elif hour == 12:
        night.append(file)
    else:
        noon.append(file)
else:
    hour = int(time[index][0:time[index].index(':')])
    if hour < 2:
        noon.append(file)
    elif hour == 12:
        noon.append(file)
    elif 2 <= hour < 6:
        afternoon.append(file)
    elif 6 <= hour < 10:
        evening.append(file)
    else:
        night.append(file)

return morning, noon, afternoon, evening, night

'''
Extract numbersd from file name
'''
def extract_numbers_from_filename(filename):
    numbers = re.findall(r'\d+', filename)
    return [int(num) for num in numbers]

'''
Concatenate data in a series of full pathways together with specific format in these files
(voltage,absJ,J,Current,Time)
'''
def DataConcat(Series):
    i=0
    Full=pd.DataFrame()
    dfs=pd.DataFrame()
    df=[]
    for values in Series:
        df=pd.read_csv(values,sep='\t',on_bad_lines='skip')
        if i==1:
            x=dfs.columns
            b=df.columns
            df=df.rename(columns={b[k]:x[k] for k in range(len(x))})

```

```

    Full=pd.concat([dfs,df],ignore_index=True)
elif i>=2:
    c=df.columns
    df=df.rename(columns={c[j]:x[j] for j in range(len(c))})
    Full=pd.concat([Full,df],ignore_index=True)
    i=i+1
    dfs=df
if i==1:
    Full=df
new_names=['Voltage (V)','Absolute Value of J','Current Density (J)','Current','Time']
columns=Full.columns.tolist()
for i in range(len(new_names)):
    columns[i]=new_names[i]
Full.columns=columns
#if len(Full.columns)>=7:
    #Full=Full.drop(Full.columns[[5,6]],axis=1)
#else:
    #Full=Full.drop(Full.columns[5], axis=1)
Full=Full.iloc[:,5]
return Full

def add_image_slide(presentation, image_path):
    slide_layout = presentation.slide_layouts[6] # 6 corresponds to the blank slide layout
    slide = presentation.slides.add_slide(slide_layout)
    left = Inches(1)
    top = Inches(1.5)
    pic = slide.shapes.add_picture(image_path, left, top, height=Inches(5))

'''
Main part
'''
grouped_files = defaultdict(list)

# Walk through subfolders and read TXT files
for root, dirs, files in os.walk(pathfolder):
    if files:
        # Extract the subfolder name from the root path
        subfolder_name = os.path.basename(root)

        for filename in files:
            if filename.endswith('.txt'):
                filepath = os.path.join(root, filename)

                # Use regular expression to extract the first number from the filename
                first_number_match = re.match(r'D*(\d+)', filename)
                if first_number_match:

```

```

first_number = first_number_match.group(1)
grouped_files[(subfolder_name, int(first_number))].append(filepath)

# Create a dictionary to store the dataframes for each grouping
dataframes_dict = {}
my_dict=grouped_files
counter=0
countertot=[0,0,0,0,0,0,0,0,0,0]
i=0
my_dict['Extra-Ignore'] = 0
#print(my_dict)
check='sc10fc'
devcount=0
devtotal=[0,0,0,0,0,0,0,0,0,0]
exit_flag=False
fresh=[]
stdavg=np.linspace(0,9,10)
scounts=np.linspace(0,9,10)
alls=np.linspace(0,9,10)
stdcount=0
allcounts=0
presentation = Presentation()
exit_flag = False
slide_number = 0 # Slide number counter
slides_per_page = 6 # Number of plots to be shown per slide
image_files = []
left_positions = [Inches(0.5), Inches(4), Inches(7.5)]
top_positions = [Inches(1), Inches(4), Inches(7)]
image_width = Inches(3.5)
image_height = Inches(2.5)
while exit_flag == False:
    for key, nested_values in my_dict.items():
        if key[0] != check:
            # If we encounter a new carbon+date combination, save the previous set of plots in a
            PowerPoint
            if devtotal[3] != 0 and len(image_files) > 0:
                # Save the plots in a PowerPoint presentation
                for i in range(0, len(image_files), slides_per_page):
                    presentation = Presentation()
                    for j, image_file in enumerate(image_files[i:i+slides_per_page]):
                        left = left_positions[j % 3]
                        top = top_positions[j // 3]
                        add_image_slide(presentation, image_file, left, top, image_width, image_height)
                    pptx_filename = f'{check}_slides_{i//slides_per_page}.pptx"

```

```

        presentation.save(pptx_filename)
        print(f"Saved { min(slides_per_page, len(image_files)-i)} plots for {check} to
        {pptx_filename}")
        image_files.clear()

    if exit_flag:
        break

    # Your existing code to generate and plot the data goes here...
    # Example: (You will need to modify this part according to your actual data and plot
    generation)
    series_values = pd.Series(nested_values)
    try:
        Full = DataConcat(series_values)
        RealFull = pd.concat([RealFull,Full],ignore_index = True)

    except ValueError:
        exit_flag = True

for x in range(len(RealFull['Absolute Value of J'])):
    if(isinstance(RealFull['Absolute Value of J'][x],str)):
        print(RealFull['Absolute Value of J'][x])

Abso = np.log10(RealFull['Absolute Value of J'],where=(RealFull['Absolute Value of J']!=0))
Volts = RealFull["Voltage (V)"]
fig2 = plt.figure(dpi=400)
hist = plt.hist2d(Volts, Abso, range=[[-0.5,0.5],[-10,0]], bins =
[21,50],cmap=plt.cm.jet,edgecolor="black")
plt.xlabel('V',fontsize=15)
plt.ylabel('LogJ (A/cm\u00b2)',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
#Currently need to changed manually
#plt.title('C15', fontweight='bold', fontsize=15)
#plt.rc('grid', linestyle="-", color='black')
cbar = plt.colorbar()
cbar.ax.set_ylabel('Counts',fontsize=15)
cbar.ax.tick_params(labels=15)

plt.show()
allcounts += 1

```

**Data analysis on data set(humidity vs thin-area defect ratio, season vs thin-area defect ratio etc.):**

```

sc='sc'
fc='fc'

```

```

c=chr(92)
carbo_folder_path= r'Q:\Shared drives\MSE-Thuo Group\Group Server ISU\Zizhi\Data\Data
Analysis\RE Martin data\Odd-even alkanethiols on Ag'
dogs=np.linspace(7,17,11)
def getDate(Series):
    date = []
    unique_date = []
    dateChangeFormat = []

    """
    Get experiment date and time from input data set
    """
    for file in Series:
        f = open(file)

        for lines in f :
            if(lines[0:3] == 'Mon'
               or lines[0:3] == 'Tue'
               or lines[0:3] == 'Wed'
               or lines[0:3] == 'Thu'
               or lines[0:3] == 'Fri'
               or lines[0:3] == 'Sat'
               or lines[0:3] == 'Sun') :
                if(len(date) == 0):
                    date.append(lines[0:len(lines)-1])

            else:
                if(lines[0:len(lines)-1] != date[0]):
                    if(len(date) < 2):
                        date.append('NaN')
                        date[1] = lines[0:len(lines)-1]

        """
        Change the format of date
        """
        for i in range(len(date)):
            commaPos = date[i].index(',') + 2
            if((date[i][commaPos:commaPos + 3]) == 'Jan'):
                dateChangeFormat.append('01-' + date[i][(commaPos + 8):])
            elif((date[i][commaPos:commaPos + 3]) == 'Feb'):
                dateChangeFormat.append('02-' + date[i][(commaPos + 9):])
        for i in range(len(dateChangeFormat)):
            dateChangeFormat[i].strip(" ")
            for j in range(len(dateChangeFormat[i])):
                if(dateChangeFormat[i][j] == ','):

```

```

        dateChangeFormat[i] = (dateChangeFormat[i][(j + 1):] + '-' +
dateChangeFormat[i][:j])
        if not dateChangeFormat[i][1:] in unique_date:
            unique_date.append(dateChangeFormat[i][1:])
    return unique_date
def getExperienceTime(Series):
    time = []
    currentLength = 0
    for file in Series:
        f = open(file)
        count = 0
        for lines in f :
            if(lines[0].isnumeric()) :
                time.append(lines[0:-1])
                count += 1
        if(count > 1):
            time[currentLength] = time[-1]

        for i in range(count):
            if(len(time) > currentLength + 1):
                del time[- 1]
            currentLength += 1
    return time
def findRelatedData(filename1, filename2): #Finding data file with _data.txt as spot and date
filenames are as follows: spot 1 scan 1-20 vs spot 1 scan 1-20_data.txt

    fileTXT = []

    for file in filename2:
        if(file[-1] == ')'):
            temp_str = file[-4:]
            fileM = file[:-4] + '_data' + temp_str + '.txt'
        else:
            fileM = file + '_data.txt'
        if fileM in filename1:
            fileDnT.append(file)
            fileTXT.append(fileM)

    return fileTXT

def findTimePeriod(file,time): # finding time in file related and binning them into premade bins
(above)
    for index in range(len(time)):
        if(time[index][len(time[index]) - 2 : len(time[index])] == "AM"): # starting with AM
times -

```

```

hour = int(time[index][0:time[index].index(':')])
if hour < 10 and hour >= 6:
    morning.append(file[index])
elif hour == 12:
    night.append(file[index])
elif hour < 6:
    night.append(file[index])
else:
    noon.append(file[index])

else:
hour = int(time[index][0:time[index].index(':')])
if hour < 2:
    noon.append(file[index])
elif hour ==12:
    noon.append(file[index])
elif hour >= 2 and hour < 6:
    afternoon.append(file[index])
elif hour >= 6 and hour <10:
    evening.append(file[index])
else:
    night.append(file[index])

def findSeasons(file,date):
    if(date[0][5:6] == '1'):
        if(date[0][6:7] == '1' or date[0][6:7] == '0'):
            for index in range(len(file)):
                autumn.append(file[index])
        elif(date[0][6:7] == '2'):
            for index in range(len(file)):
                winter.append(file[index])
    elif(date[0][6:7] == '1' or date[0][6:7] == '2'):
        for index in range(len(file)):
            winter.append(file[index])
    elif(date[0][6:7] == '3' or date[0][6:7] == '4' or date[0][6:7] == '5'):
        for index in range(len(file)):
            spring.append(file[index])
    elif(date[0][6:7] == '6' or date[0][6:7] == '7' or date[0][6:7] == '8'):
        for index in range(len(file)):
            summer.append(file[index])
    elif(date[0][6:7] == '9'):
        for index in range(len(file)):
            autumn.append(file[index])

def DataConcat(Series):
    i=0

```

```

Full=pd.DataFrame()
dfs=pd.DataFrame()
df=[]
for values in Series:
    df=pd.read_csv(values,sep='\t',on_bad_lines='skip')
    if i==1:
        x=dfs.columns
        b=df.columns
        df=df.rename(columns={b[k]:x[k] for k in range(len(b))})
        Full=pd.concat([dfs,df],ignore_index=True)
    elif i>=2:
        c=df.columns
        df=df.rename(columns={c[j]:x[j] for j in range(len(c))})
        Full=pd.concat([Full,df],ignore_index=True)
    i=i+1
    dfs=df
if i==1:
    Full=df
new_names=['Voltage (V)','Absolute Value of J','Current Density (J)','Current','Time']
columns=Full.columns.tolist()
for i in range(len(new_names)):
    columns[i]=new_names[i]
Full.columns=columns
if len(Full.columns)>=7:
    Full=Full.drop(Full.columns[[5,6]],axis=1)
#else:
    #Full=Full.drop(Full.columns[5], axis=1)
Full=Full.iloc[:,5]
return Full

def search_files_with_same_number(folder_path, number):
    matching_files = []

    for file in os.listdir(folder_path):
        if file.endswith(".txt"):
            file_path = os.path.abspath(os.path.join(folder_path, file))
            file_name = os.path.splitext(file)[0]
            numbers = re.findall(r"\d+", file_name)
            if numbers and int(numbers[0]) == number:
                matching_files.append(file_path)

    matching_files = sorted(matching_files, key = len)

    return matching_files

def get_text_file_paths(folder_path):

```

```

file_paths = []

for root, dirs, files in os.walk(folder_path):
    for file in files:
        if file.endswith(".txt"):
            file_path = os.path.join(root, file)
            file_paths.append(file_path)

file_paths_series = pd.Series(file_paths)
return file_paths_series

def search_setting_with_same_number(folder_path, number):
    matching_files = []

    for file in os.listdir(folder_path):
        if not file.endswith(".txt"):
            file_path = os.path.abspath(os.path.join(folder_path, file))
            file_name = os.path.splitext(file)[0]
            numbers = re.findall(r"\d+", file_name)
            if numbers and int(numbers[0]) == number:
                matching_files.append(file_path)

    matching_files = sorted(matching_files, key = len)
    return matching_files

def extract_numbers_from_filename(filename):
    numbers = re.findall(r"\d+", filename)
    return [int(num) for num in numbers]

def group_files_by_scfc_and_spot(file_paths):

    #for sc#fc r"sc(\d+)fc\\(\d+)"
    pattern = r"sc(\d+)fc\\(\d+)" # Change if dataset format changes - current (sc15fc/spot 1)

    grouped_files = {}

    for file_path in file_paths:
        match = re.search(pattern, file_path)
        if match:
            scfc = match.group(1)
            spot = match.group(2)
            key = f"sc{scfc}fc" # Change as well if dataset format changes - if just number then can
change whole apparatus
            if key not in grouped_files:
                grouped_files[key] = {}

```

```

    if spot not in grouped_files[key]:
        grouped_files[key][spot] = []
    grouped_files[key][spot].append(file_path)

return grouped_files

large_diff='green'
default_color='black'
Sub='Fc-Cn-SH'
countertot=0
counter=0
i=0
shortcounter=0
devcount=0
devs=np.linspace(0,10,11)
counter2=np.linspace(0,10,11)
stdavg=np.linspace(0,10,11)
scouts=np.linspace(0,10,11)
shorting=np.linspace(0,10,11)
J0 = np.linspace(0,7,8)
counter2tot = [0] * 4
seasonCounter = [0] * 4
J0tot = [0] * 5
for values in dogs:
    carbo_folder_path= r'Q:\Shared drives\MSE-Thuo Group\Group Server ISU\Zizhi\Data\Data
Analysis\RE Martin data\Odd-even alkanethiols on Ag'
    values=str(int(values))
    carbo_folder_path=carbo_folder_path + c + sc + values + fc
    files_paths=get_text_file_paths(carbo_folder_path)
    nums=[]
    for pathss in files_paths:
        a=extract_numbers_from_filename(pathss)
        if(len(a) != 1):
            nums.append(a[1])
    sorted_numbers=natsorted(nums)
    unique_numbers=list(set(sorted_numbers))
    fresh=[]
    stdcount=0
    for n in range(len(unique_numbers)):
        for m in [unique_numbers[n]]:
            Full = pd.DataFrame()
            TotalDnT = pd.DataFrame()
            result=pd.DataFrame()
            number=int(m)
            matching_files = search_files_with_same_number(carbo_folder_path,number)
            matching_setting = search_setting_with_same_number(carbo_folder_path, number)

```

```

fileTXT = findRelatedData(matching_files, matching_setting)

if not len(matching_files) == 0:
    Full = DataConcat(matching_files)
if not len(fileDnT) == 0:
    time = getExperienceTime(fileDnT)

    date = getDate(fileDnT)

    fileDnT = []
if not len(fileTXT) == 0:
    TotalDnT = DataConcat(fileTXT)
    findTimePeriod(fileTXT,time)
    findSeasons(fileTXT,date)

if Full.empty:
    break

Abso = Full['Absolute Value of J']
J0_temp = []
Abso = np.log10(Abso)
if any(value > 0 for value in Abso):
    counter += 1
shortcounter+=1
n11std=[0] * 20
n12std=[0] * 20
k=0
newneg=[-0.50,-0.45,-0.40,-0.35,-0.30,-0.25,-0.20,-0.15,-0.10,-0.05]
newpos=[0.50,0.45,0.40,0.35,0.30,0.25,0.20,0.15,0.10,0.05]
tot_average=[]
if not any(value > 0 for value in Abso):
    for numpos, numneg in zip(newpos,newneg):

        filtered_neg = Full[Full['Voltage (V)'] == numneg]
        filtered_pos = Full[Full['Voltage (V)']== numpos]
        new_df = pd.DataFrame()
        new_df_2=pd.DataFrame()
        new_df_2['Negative']=np.log10(filtered_neg['Absolute Value of J'])
        new_df['Positive']=np.log10(filtered_pos['Absolute Value of J'])
        new_list1=new_df['Positive'].tolist()
        new_list2=new_df_2['Negative'].tolist()
        n11std[k]=np.std(new_list1)

```

```

        nl2std[k]=np.std(new_list2)
        k+=1
    std=0.3
    stdcount+=1
    if any(value > std for value in nl1std) or any(value > std for value in nl2std):
        devcount += 1
    combined_list = nl1std + nl2std
    total_average = sum(combined_list) / len(combined_list)
    fresh.append(total_average)
scounts[i]=stdcount
fresh=sum(fresh)/len(fresh)
stdavg[i]=fresh
devs[i]=devcount
counter2[i]=counter
shorting[i]=shortcounter
counter=0
devcount=0
shortcounter=0
i=i+1
#print(noon)

total_spots_time = [0]*4
total_spots_seasons = [0]*4
std_time = [0] * 4
std_season = [0] * 4

if not(len(noon) == 0):
    noon=group_files_by_scfc_and_spot(noon)
    spots_with_values_greater_than_3_1 = 0
    total_spots=0
    for scfc_line, spot_data in noon.items():
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

    for x in range(len(file_paths)):
        concatenated_data = DataConcat(file_paths)

```

```

    if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
        spots_with_values_greater_than_3_1 += 1
        est_noon.append(1/len(file_paths))
    else:
        est_noon.append(0)

if total_spots > 0:
    spots_with_values_greater_than_31 = spots_with_values_greater_than_3_1 / total_spots *
100
    total_spots_time[0] = total_spots
    counter2tot[0] = spots_with_values_greater_than_31
if total_spots > 10:
    std_noon = []
    for scfc_line, spot_data in noon.items():
        each_spots = len(spot_data)
        for spot_number, file_paths in spot_data.items():
            std_temp = 0
            concatenated_data = DataConcat(file_paths)
            if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
                std_temp += 1

            std_noon.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_31) ** 2 for x in std_noon]
    variance = sum(deviations) / total_spots
    std_time[0] = math.sqrt(variance)
else:
    std_time[0] = -1

else:
    counter2tot[0] = -100

if not(len(afternoon) == 0):
    afternoon=group_files_by_scfc_and_spot(afternoon)
    spots_with_values_greater_than_3_2 = 0
    total_spots=0
    for scfc_line, spot_data in afternoon.items():
        n = 0
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

```

```

concatenated_data = DataConcat(file_paths)

if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
    spots_with_values_greater_than_3_2 += 1

    est_afternoon.append(1/len(file_paths))
else:
    est_afternoon.append(0)
if total_spots > 0:
    spots_with_values_greater_than_32 = spots_with_values_greater_than_3_2 / total_spots *
100
total_spots_time[1] = total_spots
counter2tot[1] = spots_with_values_greater_than_32
if total_spots > 10:
    std_afternoon = []
    for scfc_line, spot_data in afternoon.items():
        each_spots = len(spot_data)
        for spot_number, file_paths in spot_data.items():
            std_temp = 0
            concatenated_data = DataConcat(file_paths)
            if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
                std_temp += 1
            std_afternoon.append(std_temp/each_spots*100)
        deviations = [(x - spots_with_values_greater_than_32) ** 2 for x in std_afternoon]
        variance = sum(deviations) / total_spots
        std_time[1] = math.sqrt(variance)
    else:
        std_time[1] = -1
else:
    counter2tot[1] = -100

if not(len(evening) == 0):
    evening=group_files_by_scfc_and_spot(evening)
    spots_with_values_greater_than_3_3 = 0
    total_spots=0
    for scfc_line, spot_data in evening.items():
        for spot_number, file_paths in spot_data.items():

            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:

```

```

        total_spots+=repeat

concatenated_data = DataConcat(file_paths)

if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
    spots_with_values_greater_than_3_3 += 1
    est_evening.append(1/len(file_paths))
else:
    est_evening.append(0)

if total_spots > 0:
    spots_with_values_greater_than_33 = spots_with_values_greater_than_3_3 / total_spots *
100
    total_spots_time[2] = total_spots
    counter2tot[2] = spots_with_values_greater_than_33
if total_spots > 10:
    std_evening = []
    for scfc_line, spot_data in evening.items():
        each_spots = len(spot_data)
        for spot_number, file_paths in spot_data.items():
            std_temp = 0
            concatenated_data = DataConcat(file_paths)
            if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
                std_temp += 1
            std_evening.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_33) ** 2 for x in std_evening]
    variance = sum(deviations) / total_spots
    std_time[2] = math.sqrt(variance)
else:
    std_time[2] = -1
else:
    counter2tot[2] = -100

if not(len(night) == 0):
    night=group_files_by_scfc_and_spot(night)
    spots_with_values_greater_than_3_4 = 0
    total_spots=0
    for scfc_line, spot_data in night.items():
        for spot_number, file_paths in spot_data.items():

            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1

```

```

else:
    total_spots+=repeat

concatenated_data = DataConcat(file_paths)

if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
    spots_with_values_greater_than_3_4 += 1
    est_night.append(1/len(file_paths))
else:
    est_night.append(0)

if total_spots > 0:
    spots_with_values_greater_than_34 = spots_with_values_greater_than_3_4 /total_spots *
100
    total_spots_time[3] = total_spots
    counter2tot[3] = spots_with_values_greater_than_34
if total_spots > 10:
    std_night = []
    for scfc_line, spot_data in night.items():
        each_spots = len(spot_data)
        for spot_number, file_paths in spot_data.items():
            std_temp = 0
            concatenated_data = DataConcat(file_paths)
            if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
                std_temp += 1
            std_night.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_34) ** 2 for x in std_night]
    variance = sum(deviations) / total_spots
    std_time[3] = math.sqrt(variance)
else:
    std_time[3] = -1
else:
    counter2tot[3] = -100
hums=[61.83482, 54.08589, 58.57513,72.96707589]
label = ['Noon','Afternoon','Evening','Night']
std_hum = [20.09905972,22.70370332,24.58917457,22.17593896]
fig, ax1 = plt.subplots(figsize=(20,12),dpi = 300)
ax1.set_xlabel('Time Period', fontsize=40)
ax1.set_ylabel('Shorts Ratio per Chip(%)', color='red',fontsize=40)
xs = np.linspace(0, 3, 4)
ax1.scatter(label, counter2tot, color='red',s=800)
ax1.yaxis.set_major_formatter(mtick.PercentFormatter())
ax1.tick_params(axis='x',labelsize=40)
ax1.tick_params(axis='y', labelcolor='red',labelsize=40)
ax2 = ax1.twinx()
ax2.scatter(label, hums, color='blue', label='Humidity Data Points',s=800)

```

```

ax2.yaxis.set_major_formatter(mtick.PercentFormatter())
ax2.set_ylabel('Humidity(%)', color='blue',fontsize=40)
ax2.tick_params(axis='y', labelcolor='blue',labelsize=40)
ax1.text(0,-2,'n =' + str(spots_with_values_greater_than_3_1),fontsize=40)
ax1.text(0.75,-2,'n =' + str(spots_with_values_greater_than_3_2),fontsize=40)
ax1.text(1.75,-2,'n =' + str(spots_with_values_greater_than_3_3),fontsize=40)
ax1.text(2.75,-2,'n =' + str(spots_with_values_greater_than_3_4),fontsize=40)
ax1.set_ylim(-3,30)
for x in range(len(timePeriod)):
    if not(std_time[x] == -1):
        ax1.errorbar(label[x], counter2tot[x],yerr=std_time[x],fmt='none',capsize=20,ecolor='red')
        ax2.errorbar(label[x], hums[x],yerr=std_hum[x],fmt='none',capsize=20,ecolor='blue')
ax2.set_ylim(0,100)
plt.tight_layout()
plt.show()
if not(len(spring) == 0):
    spring=group_files_by_scfc_and_spot(spring)
    spots_with_values_greater_than_3_0 = 0
    total_spots=0
    for scfc_line, spot_data in spring.items():
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

        concatenated_data = DataConcat(file_paths)

        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            spots_with_values_greater_than_3_0 += 1
            est_spring.append(1/len(file_paths))
        else:
            est_spring.append(0)

    if total_spots > 0:
        spots_with_values_greater_than_30 = spots_with_values_greater_than_3_0 / total_spots *
100
        total_spots_seasons[0] = total_spots
        seasonCounter[0] = spots_with_values_greater_than_30
    if total_spots > 10:
        std_spring = []

```

```

for scfc_line, spot_data in spring.items():
    each_spots = len(spot_data)
    for spot_number, file_paths in spot_data.items():
        std_temp = 0
        concatenated_data = DataConcat(file_paths)
        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            std_temp += 1
            std_spring.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_30) ** 2 for x in std_spring]
    variance = sum(deviations) / total_spots
    std_season[0] = math.sqrt(variance)
else:
    std_season[0] = -1
else:
    seasonCounter[0] = -100

if not(len(summer) == 0):
    summer=group_files_by_scfc_and_spot(summer)
    spots_with_values_greater_than_3_1 = 0
    total_spots=0
    for scfc_line, spot_data in summer.items():
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

            concatenated_data = DataConcat(file_paths)
            if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
                spots_with_values_greater_than_3_1 += 1
                est_summer.append(1/len(file_paths))
            else:
                est_summer.append(0)

    if total_spots > 0:
        spots_with_values_greater_than_31 = spots_with_values_greater_than_3_1 / total_spots
*100
        total_spots_seasons[1] = total_spots
        seasonCounter[1] = spots_with_values_greater_than_31
    if total_spots > 10:
        std_summer = []
        for scfc_line, spot_data in summer.items():

```

```

each_spots = len(spot_data)
for spot_number, file_paths in spot_data.items():
    std_temp = 0
    concatenated_data = DataConcat(file_paths)
    if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
        std_temp += 1
        std_summer.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_31) ** 2 for x in std_summer]
    variance = sum(deviations) / total_spots
    std_season[1] = math.sqrt(variance)
else:
    std_season[1] = -1
else:
    seasonCounter[1] = -100

if not(len(autumn) == 0):
    autumn=group_files_by_scfc_and_spot(autumn)
    spots_with_values_greater_than_3_2 = 0
    total_spots=0
    for scfc_line, spot_data in autumn.items():
        #total_spots += len(spot_data)
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

        concatenated_data = DataConcat(file_paths)

        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            spots_with_values_greater_than_3_2 += 1
            est_autumn.append(1/len(file_paths))
        else:
            est_autumn.append(0)

    if total_spots > 0:
        spots_with_values_greater_than_32 = spots_with_values_greater_than_3_2 / total_spots *
100
        total_spots_seasons[2] = total_spots
        seasonCounter[2] = spots_with_values_greater_than_32
    if total_spots > 10:
        std_autumn = []

```

```

for scfc_line, spot_data in autumn.items():
    each_spots = len(spot_data)
    for spot_number, file_paths in spot_data.items():
        std_temp = 0
        concatenated_data = DataConcat(file_paths)
        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            std_temp += 1
            std_autumn.append(std_temp/each_spots*100)
    deviations = [(x - spots_with_values_greater_than_32) ** 2 for x in std_autumn]
    variance = sum(deviations) / total_spots
    std_season[2] = math.sqrt(variance)
else:
    std_season[2] = -1
else:
    seasonCounter[2] = -100

if not(len(winter) == 0):
    winter=group_files_by_scfc_and_spot(winter)
    spots_with_values_greater_than_3_3 = 0
    total_spots=0
    for scfc_line, spot_data in winter.items():
        for spot_number, file_paths in spot_data.items():
            repeat = 0
            for x in range(len(file_paths)):
                if(file_paths[x][-5] == ')') and (int(file_paths[x][-6]) > repeat):
                    repeat = int(file_paths[x][-6])
            if(repeat == 0):
                total_spots+=1
            else:
                total_spots+=repeat

        concatenated_data = DataConcat(file_paths)

        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            spots_with_values_greater_than_3_3 += 1
            est_winter.append(1/len(file_paths))
        else:
            est_winter.append(0)

    if total_spots > 0:
        spots_with_values_greater_than_33 = spots_with_values_greater_than_3_3 / total_spots *
100
    total_spots_seasons[3] = total_spots
    seasonCounter[3] = spots_with_values_greater_than_33
    if total_spots > 10:
        std_winter = []

```

```

for scfc_line, spot_data in winter.items():
    each_spots = len(spot_data)
    for spot_number, file_paths in spot_data.items():
        std_temp = 0
        concatenated_data = DataConcat(file_paths)
        if any(np.log10(concatenated_data['Absolute Value of J']) > 0):
            std_temp += 1
            std_winter.append(std_temp/each_spots*100)
        deviations = [(x - spots_with_values_greater_than_33) ** 2 for x in std_winter]
        variance = sum(deviations) / total_spots
        std_season[3] = math.sqrt(variance)
    else:
        std_season[3] = -1
else:
    seasonCounter[3] = -100
for i in range(len(scounts)):
    scounts[i] = shorting[i] - counter2[i]
scounts[4] = 17
carbonNum = ['C7','C8','C9','C11','C13','C14','C15','C16','C17','C18','C19']
hums=[26.18,23.77, 67.31, 59.82857,
42.57583,79.71375,48.25286,84.9475,95.57,76.43667,61.216]
ratios=counter2/shorting
fig, ax1 = plt.subplots()
ax1.set_xlabel('Carbon Length', fontweight='bold', labelpad=15, fontsize=15)
ax1.set_ylabel('Percentage of # of Shorts/Total', color='red')
degree = 2
for index in range(len(ratios)):
    if(int(carbonNum[index][1:]) % 2) == 0:
        ax1.scatter(carbonNum[index], ratios[index], color='white',s=80, marker="o",
edgcolors="red")
    else:
        ax1.scatter(carbonNum[index], ratios[index], color='red',s=80, marker="o")
ax1.tick_params(axis='x',labelsize = 14)
ax1.tick_params(axis='y', labelcolor='red',labelsize = 15)

ax2 = ax1.twinx()
ax2.scatter(carbonNum, hums, color='blue',s=80)
ax2.set_ylabel('Humidity', color='blue')
ax2.tick_params(axis='y', labelcolor='blue',labelsize = 12)
plt.tight_layout()
plt.show()
densitystd=devs/scounts
fig, ax1 = plt.subplots()
ax1.set_xlabel('Carbon Length', fontweight='bold', labelpad=15, fontsize=15)
ax1.set_ylabel('Percentage of Junctions with STD>{ }'.format(round(std,3)), color='red')
degree = 2

```

```

for index in range(len(densitystd)):
    if(int(carbonNum[index][1:] % 2) == 0:
        ax1.scatter(carbonNum[index], densitystd[index], color='white',s=80, marker="o",
edgcolors="red")
    else:
        ax1.scatter(carbonNum[index], densitystd[index], color='red',s=80, marker="o")
ax1.tick_params(axis='x',labelsize = 14)
ax1.tick_params(axis='y', labelcolor='red',labelsize = 15)
ax2 = ax1.twinx()
ax2.scatter(carbonNum, hums, color='blue',s=80)
ax2.set_ylabel('Humidity', color='blue')
ax2.set_ylim(20,98)
ax2.tick_params(axis='y', labelcolor='blue',labelsize = 12)
plt.tight_layout()
plt.show()
plt.figure(figsize=(20,12))
plt.scatter(timePeriod,seasonCounter,c='black',s=500)
label = ['Spring','Summer','Autumn','Winter']
plt.xticks(np.arange(0,4,1),label,fontsize = 40)
plt.yticks(np.arange(0,51,5),fontsize = 40)
plt.grid(True,alpha=0.15)
plt.xlabel('Seasons',labelpad = 40, fontsize=40)
plt.ylabel('Shorts per Chip(%)',fontsize=40)
plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter())
plt.ylim(-1,51)
plt.text(-0.25,-7,'3/1-5/31',fontsize=40)
plt.text(0.75,-7,'6/1-8/31',fontsize=40)
plt.text(-0.1,13,'n = ' + str(spots_with_values_greater_than_3_0),fontsize=40)
plt.text(0.8,3,'n = ' + str(spots_with_values_greater_than_3_1),fontsize=40)
plt.text(1.75,-7,'9/1-11/30',fontsize=40)
plt.text(2.70,-7,'12/1-2/28(29)',fontsize=40)
for x in range(len(timePeriod)):
    if not(std_season[x] == -1):

plt.errorbar(timePeriod[x],seasonCounter[x],yerr=std_season[x],fmt='none',capsize=20,ecolor='black')

```

### Skewness, Kurtosis, Standard deviation of data set:

Based on the previous code:

```

if( int(values) >= 10 and int(values) <= 15):
    carbon_temp_1d_p1 = [n for carbon_temp_1d_p1 in carbon_temp_p1 for n in
carbon_temp_1d_p1]
    carbon_temp_1d_p5 = [n for carbon_temp_1d_p5 in carbon_temp_p5 for n in
carbon_temp_1d_p5]
    carbon_temp_1d_p1_neg = [n for carbon_temp_1d_p1_neg in carbon_temp_p1_neg for n
in carbon_temp_1d_p1_neg]

```

```

    carbon_temp_1d_p5_neg = [n for carbon_temp_1d_p5_neg in carbon_temp_p5_neg for n
in carbon_temp_1d_p5_neg]
    carbonFull_p1.append(carbon_temp_1d_p1)
    carbonFull_p5.append(carbon_temp_1d_p5)
    carbonFull_p1_neg.append(carbon_temp_1d_p1_neg)
    carbonFull_p5_neg.append(carbon_temp_1d_p5_neg)
#positive voltage 0-0.5
for x in range(len(skew_pos_normalize)):
    skew_pos_normalize[x] = skew_pos[x] - skew_pos[0]
skew_neg_normalize = [0] * 10
for x in range(len(skew_neg_normalize)):
    skew_neg_normalize[x] = skew_neg[x] - skew_neg[0]
for x in range(len(carbonFull_p1)):
    skew_carbon_p1[x] = skew(carbonFull_p1[x], axis=0, bias=True)
    skew_carbon_p5[x] = skew(carbonFull_p5[x], axis=0, bias=True)
    skew_carbon_p1_neg[x] = skew(carbonFull_p1_neg[x], axis=0, bias=True)
    skew_carbon_p5_neg[x] = skew(carbonFull_p5_neg[x], axis=0, bias=True)
for x in range(len(skew_carbon_p1_normalize)):
    skew_carbon_p1_normalize[x] = skew_carbon_p1[x] - skew_carbon_p1[0]
    skew_carbon_p5_normalize[x] = skew_carbon_p5[x] - skew_carbon_p5[0]
    skew_carbon_p1_neg_normalize[x] = skew_carbon_p1_neg[x] - skew_carbon_p1_neg[0]
    skew_carbon_p5_neg_normalize[x] = skew_carbon_p5_neg[x] - skew_carbon_p5_neg[0]
voltage_label = ['0.05','0.10','0.15','0.20','0.25','0.30','0.35','0.40','0.45','0.50']

```

```

fig, ax1 = plt.subplots(figsize=(20,12),dpi = 300)
ax1.set_xlabel('V', fontsize=40)
ax1.set_ylabel(u'$\Delta S_{\text{average}}$', color='red',fontsize=40)
ax1.scatter(voltage_label, skew_pos_normalize, color='red',s=500)
ax1.tick_params(axis='x',labelsize=40)
ax1.tick_params(axis='y', labelcolor='red',labelsize=40)
plt.tight_layout()
plt.show()

```

### Estimation plot of data set:

```

plt.rcParams.update({'font.size': 17})
df = pd.DataFrame(columns=['Humidity<=40%','41%-50%','51%-60%','61%-70%','71%-
80%','81%-90%','>90%'])
for x in range(len(Ag_hums)):
    if(int(Ag_hums[x]) < 40):
        hum_40.append(Ag_short_ratio[x] *100)
    elif(int(Ag_hums[x]) >= 40 and int(Ag_hums[x]) < 50):
        hum_50.append(Ag_short_ratio[x] *100)
    elif(int(Ag_hums[x]) >= 50 and int(Ag_hums[x]) < 60):
        hum_60.append(Ag_short_ratio[x] *100)
    elif(int(Ag_hums[x]) >= 60 and int(Ag_hums[x]) < 70):
        hum_70.append(Ag_short_ratio[x] *100)
    elif(int(Ag_hums[x]) >= 70 and int(Ag_hums[x]) < 80):

```

```
    hum_80.append(Ag_short_ratio[x] *100)
elif(int(Ag_hums[x]) >= 80 and int(Ag_hums[x]) < 90):
    hum_90.append(Ag_short_ratio[x] *100)
elif(int(Ag_hums[x]) >= 90):
    hum_100.append(Ag_short_ratio[x] *100)
df['Humidity<=40%'] = pd.Series(hum_40)
shared = dabest.load(df,idx=('Humidity<=40%','51%-60%','61%-70%','71%-80%'))
shared.mean_diff.plot(dpi=150,swarm_label='Thin-area defect ratio(%)',contrast_label='MD')
```