Predicting Parkinson's Disease

in Rural America Based on Initial Examination Data

BY

JOERG HEINTZ
B.S. University of Applied Sciences, Kaiserslautern in Germany, 1998

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Public Health Sciences
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:

Edward Mensah, Chair and Advisor
Michael Cailas, Environmental and Occupational Health Sciences
Kevin Croke, Northwestern University

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Edward Mensah, for his guidance. Our research was based on conversation and discussion rather than dictation, which encouraged me to perform research independently. I enjoyed working with you Professor Mensah!

Finally, I want to thank my wife Rosa and our both families for their support. This thesis is dedicated to

Helga† & Holger and Pepin† & Rosa

TABLE OF CONTENTS

1 INTRODUCTION AND AIMS 1

2 LITERATURE REVIEW 2

3 SURROGATE DATA 4

4 DATA ANALYSIS PIPELINE 6

5 VARIABLES 7
- 5.1 Data Imputation 7
- 5.2 Orthostatic Blood Pressure 8
- 5.3 REM Sleep Behavior Disorder 11
- 5.4 Montreal Cognitive Assessment 13
- 5.5 Unified Parkinson's Disease Rating Scale II + III 13
- 5.6 Geriatric Depression Scale 17
- 5.7 Anxiety 17

6 VARIABLE AGGREGATION 21
- 6.1 Aggregation to Motor and Non-Motor Scores 21
- 6.2 Motor and Non-Motor Data Set 22
  - 6.2.1 Non-Motor Scores 22
  - 6.2.2 Motor Scores 24

7 PREDICTING PARKINSON'S 26
- 7.1 Clustering Parkinson's 26
- 7.2 Predicting Parkinson's 28
  - 7.2.1 Introduction 28
  - 7.2.2 Prediction Method Selection Process 29
  - 7.2.3 Creating Training and Test Data Sets 29
- 7.3 Logistic Regression 30
  - 7.3.1 Model Selection 30
  - 7.3.2 Performance Test with all Test - and Initial Visit Test Data 31
- 7.4 Random Forest 31
  - 7.4.1 Parametrization Via 5-fold Cross Validation 31
  - 7.4.2 Performance Test with all Test - and Initial Visit Test Data 32
- 7.5 Support Vector Machine 32
  - 7.5.1 Kernel Parametrization and Selection Via 5-fold Cross Validation 33
  - 7.5.2 Performance Assessment with Test- and Initial Visit Data 33
- 7.6 Results and Discussion 35
- 7.7 Recommendations 36
- 7.8 Limitation and Future Research 37
- 7.9 Conclusion 37

CITED LITERATURE 38

APPENDICES 41
- Appendix A 42
- Appendix B 43

VITA 88

LIST OF TABLES

TABLE		PAGE
I	SURROGATE VARIABLES	4
II	GENDER DISTRIBUTION	5
III	RACE DISTRIBUTION	5
IV	AGGREGATED DATA SET	22
V	NON-MOTOR HYPOTHESIS TEST RESULTS	25
VI	MOTOR HYPOTHESIS TEST RESULTS	25
VII	K-MEANS CLUSTER VARIATION	28
VIII	TEST & TRAINING DATA SET	30
IX	LR MISCLASSIFICATION RATE	30
X	LR TEST RESULTS	31
XI	RF MISCLASSIFICATION RATES	32
XII	RF TEST RESULTS	32
XIII	SVM KERNEL SVM PARAMETRIZATION	33
XIV	SVM MISCLASSIFICATION RATE	33
XV	SVM TEST RESULTS PER KERNEL	35
XVI	PERFORMANCE COMPARISON OF LR, RF, SVM	35
XVII	DATA COLLECTION TIME FOR PCP	37

LIST OF FIGURES

FIGURE PAGE

1 Point in Time Diagnosis 5
2 Age Distribution 5
3 Data Processing Pipeline 6
4 Imputation Approach 7
5 Missing Data Distribution 8
6 Systolic & Diastolic Orthostatic Blood Pressure 9
7 Systolic & Diastolic Orthostatic Blood Pressure (imputed) 10
8 Movement Sleep Behavior Disorder 12
9 Movement Sleep Behavior Disorder (imputed) 12
10 Mild Cognitive Impairment 14
11 Mild Cognitive Impairment (imputed) 14
12 UPDRS Part II 15
13 UPDRS Part II (imputed) 15
14 UPDRS Part III 16
15 UPDRS Part III (imputed) 16
16 Geriatric Depression 19
17 Geriatric Depression (imputed) 19
18 Anxiety 20
19 Anxiety (imputed) 20
20 Scatter Plot Non-Motor vs. Motor 23
21 Non-Motor Distribution 23
22 Motor Distribution 24
23 Association Non-Motor to Motor 26
24 K-Means Clustering 27
25 Model Development Process 29
26 SVM Kernel Overview with Hyperplanes 34

LIST OF ABBREVIATIONS

ANX	Anxiety
APN	Advanced Practical Nurse
BL	Baselining Visit
GDS	Geriatric Depression Scale
HC	Healthy Control
IID	Independent and Identically Distributed
LR	Logistic Regression
MCI	Mild Cognitive Impairment
MDS	Movement Disorder Society
MoCA	Montreal Cognitive Assessment
OH	Orthostatic Hypotension
PCP	Primary Care Physicians
PD	Parkinson's Disease Patient
PPMI	Parkinson's Progression Markers Initiative
PSG	Polysomnography
RBD	REM Sleep Disorder
RBDSQ	REM Sleep Disorder Screening Questionnaire
REM	Rapid Eye Movement
RF	Random Forest
STAI	State Trait Anxiety Inventory
SVM	Support Vector Machine
SYSOH	Systolic Orthostatic Hypotension
UPDRS	Unified Parkinson's Disease Rating Scale
V01 – V15	Visit 01 to Visit 15

SUMMARY

Neurological disorders are frequent causes of morbidity and mortality in the US, whereby Parkinson's is one of the most common. Increasing incident rates and a continuous shortfall of neurologists is prevalent. Additionally, neurophobia has been identified in medical student cohorts, residents and among general practitioners, which gives concern regarding care delivery for neurological patients. Even more dramatic is the situation in rural areas. More patients have to be treated by physicians and primary care providers without formal neurologic training (Sandrone et al., 2018). The American Journal of Managed Care just reported results from a poll that 26% of Parkinson patients having been misdiagnosed (Gavidia, 2020).

In this study I developed a prediction model that addresses partially the need for better Parkinson's disease detection in primary care in rural settings, and the shortage of neurologists. The model developed uses machine learning and routinely accessible data to Primary Care Physicians (PCP) and allows to discriminate between healthy control (HC) and Parkinson's disease patient (PD) with an accuracy of 92 to 96%. This model could assist PCPs with a specificity >96% (Logistic Regression) and sensitivity >97% (Random Forest) in the triaging process and potentially reduce the number of referrals by giving PCPs greater confidence in whether a subject shows sufficient symptom to be diagnosed with Parkinson's. Not investigated was the model's performance with subjects that have a precondition other than being healthy or being diagnosed with Parkinson's disease. In order to have a greater impact on the health care system, the model developed here need to be trained for other neurological conditions, e.g. Alzheimer's, to cover a greater spectrum of diseases. However, this approach has the potential to assist rural hospitals and primary care physicians without formal neurological training in providing better care for patients with neurological conditions such as Parkinson's.

1 INTRODUCTION AND AIMS

Neurological disorders are among the most frequent causes of morbidity and mortality in the US, the most common being Parkinson's and Alzheimer's. Increasing incident rates are being reported for both diseases while a continuous shortfall of neurologists is prevalent. Additionally, neurophobia has been identified in medical student cohorts, residents and among general practitioners, which gives additional cause for concern regarding care delivery for neurological patients. The situation in rural areas is even more problematic: most patients are treated by physicians and primary care providers without formal neurologic training (Sandrone et al., 2018). The insidious and heterogeneous onset of neurodegenerative diseases challenges the abilities of primary care systems to appropriately diagnose and manage these diseases. In early 2020, the American Journal of Managed Care reported that 26% of Parkinson patients are misdiagnosed, with an additional 21% having to see their general provider 3 times before being referred to a specialist (Gavidia, 2020). A predictive model for identifying these neurological disorders based on data from the patients' first visit could facilitate triaging and subsequently improve the quality of life of these patients. I propose to translate recent research results for a practical application in rural hospital settings and to develop a prediction model to identify Parkinson's patients based on initial and routine examination data. **Aim 1:** According to Fereshtehnejad et al., 2015, seven variables were identified as most informative to generate clusters of Parkinson's progression. I propose to reproduce the results from this study with variable surrogates that are available to rural hospitals and primary care physicians. **Aim 2:** Develop an effective statistical learning model to discriminate between healthy patients and patients with Parkinson's using data that is typical for a patient on their initial visit. The ultimate goal is to provide a primary care physician or a rural hospital with an effective tool to support and validate a physician's triaging decision.

2 LITERATURE REVIEW

Predicting Parkinson's is not a new topic. Using Google Scholar and searching for "Predicting Parkinson's" yields more than two hundred hits. The vast majority of the publications identified, though, addressing Parkinson's prediction from the perspectives of computer scientists, clinical, medical, and pharmaceutical researchers; predicting Parkinson's from the perspective and environment of a Primary Care Physician is missing. Computer scientists publish on the effectiveness and precision of machine learning algorithms trained with challenging data sets, and medical – clinical researchers publish on medical breakthroughs enabled through machine learning algorithms. Here, often machine learning is applied to gain deeper insights in observational and experimental data to solve fundamental medical and biological questions on Parkinson's etiology to discover biomarkers and substances for precise diagnosis and safe treatment or to predict disease progression for better care planning. This list is by far not complete and is an attempt to summarize the identified research perspectives on Parkinson's. As mentioned, little publications can be found on Parkinson's from the perspective of Primary Care Physicians, and other non-research groups. This thesis takes the angle of a Primary Care Physician, anticipates the routinely accessible data and suggests how Primary Care Physicians could utilize machine learning with the data at hand for screening, diagnosing, and triaging patients with neurological disorders such as Parkinson's. In the following the literature review overview.

A Google Scholar search performed on 4/23/2020 yield the following results: SEARCH 1 "predicting Parkinson's", 244 results are on predicting Parkinson's based on environmental conditions and factors (Calne & Langston, 1983; Snyder & D'amato, 1985); specific biomarkers, such as dopaminergic transmission ; gene expressions (Maraganore & Lesnick, 2008; Peng et al., 2019); speech- and gait analysis, mental impairments e.g. using data from Montreal Cognitive Assessment (Arora et al., 2015; Byeon, 2020; Cai et al., 2017); based on sensors e.g. smart phone data (Arora et al., 2015). Identifying Parkinson's subtypes using neuroimaging and trajectory clustering via bipartite networks. (Koenen, 2016). SEARCH 2 "predicting Parkinson's" AND "primary care", 30 results on very different topics such as predicting care-

giver and receiver reactions (Loftus, 2008), using single symptoms for Parkinson's screening (Stiasny-Kolster et al., 2007), standard strategies for diagnosis and treatment (Tarsy, 2013), and using wearable technology (Dunn et al., 2018), and other themes. SEARCH 3"Parkinson's" AND "primary care" AND "triaging" AND "decision support", 11 results are on technology evaluations such as telemedicine (AlDossary et al., 2017), or predicting other risks, such as falls, a Parkinson's patient is exposed to (Baus et al., 2017). SEARCH 4 "diagnosing Parkinson's" AND "primary care" AND "decision support"; 6 results are on decisions support systems using data generated via additional technical equipment and not dedicated for Parkinson's primary care settings. Publications: (Cohen et al., 2003; Fallon et al., 2010; Frosch et al., 2012; Gaenslen et al., 2011; Gray, 2000; Zietsma, 2015).

3 SURROGATE DATA

Table I shows the seven variables that were identified as most informative to generate clusters of Parkinson's progression (Fereshtehnejad et al., 2015). The corresponding surrogate variables were selected from the Parkinson's Progression Markers Initiative (PPMI) data. There are two requirements on the surrogates. First, rural hospitals and primary care physicians must be able to produce the data defined by the variables, and second, the data must carry the information to reproduce the progression clusters and train a model to predict people with PD or people that are healthy. The practicality of time and costs to collect the surrogate data is discussed in the variable section.

TABLE I SURROGATE VARIABLES

Variables (Fereshtehnejad et al., 2015)		Surrogate Variables and Data (PPMI Data Set)	
OH	Orthostatic hypotension (OH) is defined as a decrease in systolic blood pressure of 20 mm Hg or a decrease in diastolic blood pressure of 10 mm Hg within 3 minutes of standing when compared with blood pressure from the sitting or supine position.	SYSOH	Systolic Orthostatic Blood Pressure (SYSOH) measured manually supine and after 1 minute of standing. The measurement was annotated with 1, if drop of >10 mm Hg.
RBD	REM Sleep Behavior Disorder (RBD): evaluated by overnight. Polysomnography (PSG), defined with International Classification of Sleep Disorders–II diagnostic criteria and PSG criteria.	RBD	The REM Sleep Behavior Disorder evaluated via subjective self-rating. An item questionnaire was used to self-assess sleep behavior.
MCI	Mild cognitive impairment was defined at baseline according to the 2012 Movement Disorder Society Task Force guidelines using 5 domains.	MoCA	Montreal Cognitive Assessment (MoCA) is conducted by examiner, takes ~ 10 minutes, has a score range of 30 points. It measures e.g. attention, concentration, executive functions, memory, language, etc.
UPDRS Part II	The assessment is conducted via a questionnaire completed by research staff.	UPDRS Part II	The assessment is conducted via a questionnaire completed by the subject and/or caregiver.
UPDRS Part III	The assessment is conducted via a questionnaire completed by research staff.	UPDRS Part III	The assessment is conducted via a questionnaire completed by the subject and/or caregiver.
Depression	The severity of depression was measured via the Beck Depression Self Report Inventory II: a 21-question, multiple-choice assessment	GDS	Depression is measured via Geriatric Depression Scale, a 30-item self-assessment.
Anxiety	The severity of anxiety was measured via the Beck Anxiety Self Report Inventory: a 21-question, multiple-choice assessment.	ANX	Anxiety was measured via The State Trait Anxiety Inventory, a questionnaire given to patients to assess how strong a person's feelings of anxiety are.

The PPMI is an ongoing observational clinical study to identify biomarkers of Parkinson's disease onset and progression (Marek et al., 2011). Throughout this thesis I use PD for Parkinson's Disease Patient, HC for Healthy Control, and BL for Baselining, which is a set of critical observations defined by PPMI for comparison or a control. Baselining data from PPMI are used as surrogates for initial visits of PD and HC. Figure 1 illustrates when and how many PDs were diagnosed before the PPMI study started. The point in time "0 month" marks the study start and when baselining data were collected. The distribution shows that the baselining initial visit data contains data from PDs in different progression statuses. The age of the participating population is between 60 and 75 years, consisting of different races and genders (Figure 2, Table II, III).

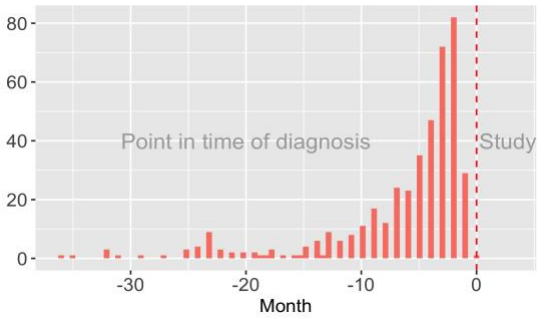


Figure 1 Point in Time Diagnosis

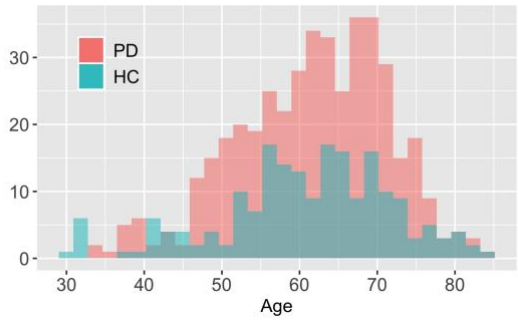


Figure 2 Age Distribution

TABLE II GENDER DISTRIBUTION

	female	male	sum
PD	147	280	427
HC	70	126	196

TABLE III RACE DISTRIBUTION

	Single Race		Double Race	
	PD	HC	PD	HC
HISPLAT	0	0	9	3
RAINDALS	0	0	4	1
RAASIAN	8	1	2	0
RABLACK	6	9	1	0
RAHAWOPI	0	0	0	0
RAWHITE	389	180	20	60
RANOS	2	0	8	2

4 DATA ANALYSIS PIPELINE

My analysis focuses on identifying clusters within the PD, and to develop prediction models to discriminate between HC and PD with data that is routinely accessible to primary care physicians and rural hospitals. The first step was ingesting the PPMI files (0), calculating the diagnostic scores, and merging the data sets. I used different visualization techniques to provide sufficient insights to discover trends, missing values, outliers, determine where hypothesis tests are needed, and to develop prediction models. After normalization, the scores were aggregated to Motor and Non-Motor. I developed and assessed different models to cluster and predict Parkinson's. Figure 3 shows the entire data analysis pipeline.

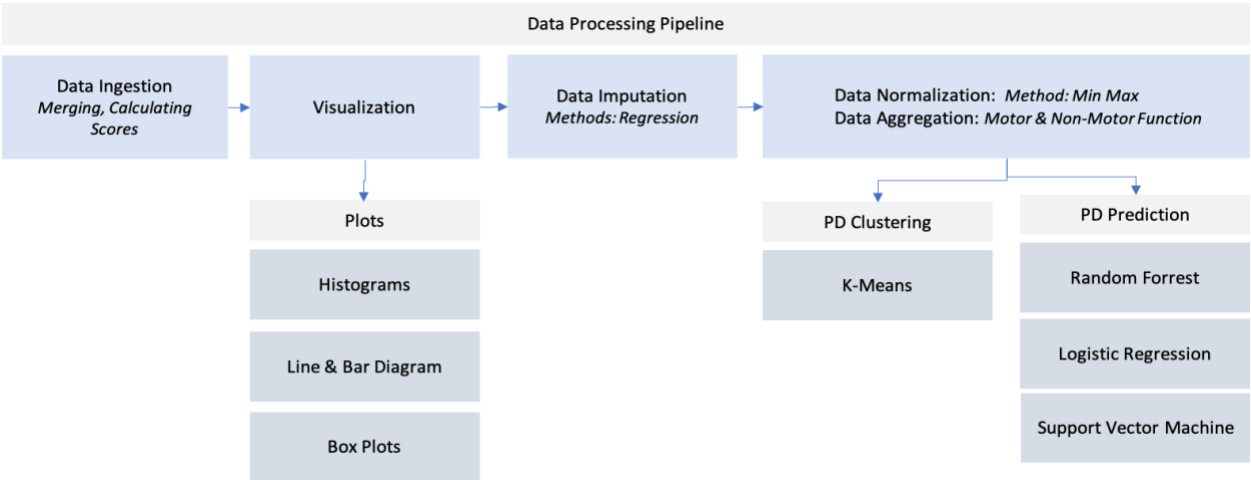


Figure 3 Data Processing Pipeline

5 VARIABLES

5.1 Data Imputation

The imputation method is based on a regression model where missing values are predicted. In order to reduce bias at least 1 visit per visit quartile is required. In Figure 4, subjects 3007, 3006, and 3009 do not fulfill that requirement, and are therefore excluded and not further considered. Only Subject 3000 fulfills that requirement.

	BL	V01	V02	V03	V04	V05	V06	V07	V08	V09	V10	V11	V12	V13	V14	V15
3007	-12															
3006	14	-10	-24	-18	-15											
3009	0	4	-9	24	12	0	20	2	-6	7	30	11				
3000	4	-6	8	1	17	6	-2	6	-11	-8	-6	13	0		-1	

Figure 4 Imputation Approach

The bar chart in Figure 5 shows that out of 619 total, 470 subjects measured orthostatic blood pressure distributed over the 4 quartiles. Twenty-five subjects only measured in 1 quartile, 38 measured in 2 quartiles, and 86 measured in 3 quartiles. In total, 149 subjects were excluded from that experiment.

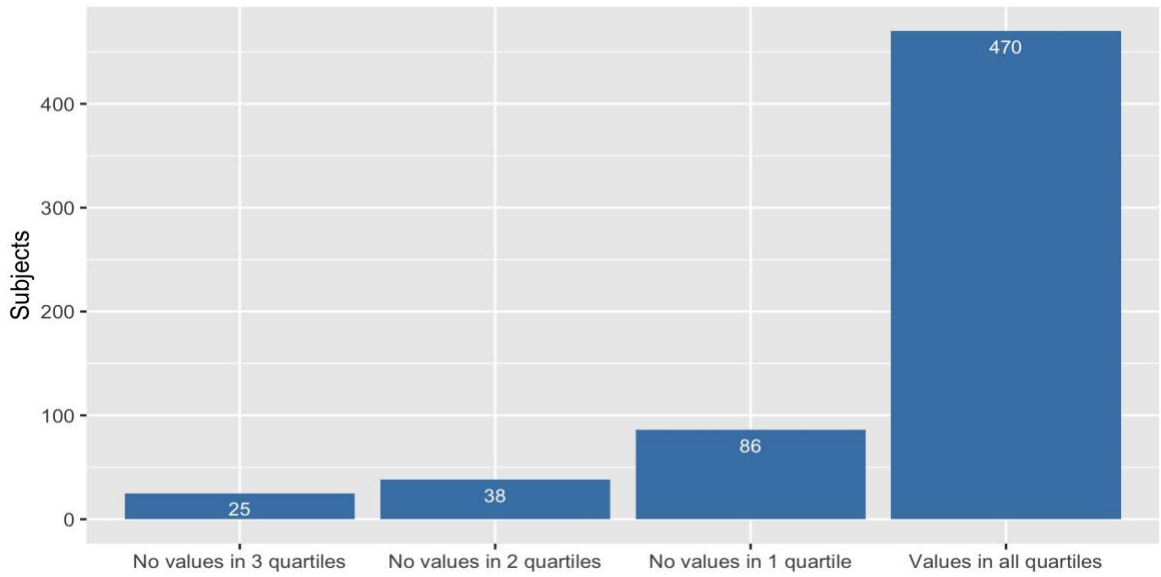


Figure 5 Missing Data Distribution

5.2 Orthostatic Blood Pressure

Orthostatic hypotension is defined as a decrease in systolic blood pressure of 20 mm Hg or a decrease in diastolic blood pressure of 10 mm Hg within three minutes of standing when compared with a supine position. The procedure can be conducted by an APN and takes about 15 minutes. A fall of at least 20 mm Hg of systolic blood pressure was measured in 58.2% of patients. Orthostatic hypotension was asymptomatic in 38.5% and associated with postural events in 19.8% of patients. (Senard et al., 1997). The diagrams (Figure 6) illustrate the orthostatic blood pressure means and variances for PD and HC across all recorded visits. The colored labels contain the sample size per visit and the labels are greyed when the sample size is below 20. The mean difference for the diastolic blood pressure between PD and HC stays constant over time, but seems to grow for systolic blood pressure. The variance is very high when compared to the mean difference. The histogram for systolic blood pressure indicates a shift of the two distributions. The imputed data set (Figure 7) follows the original data set.

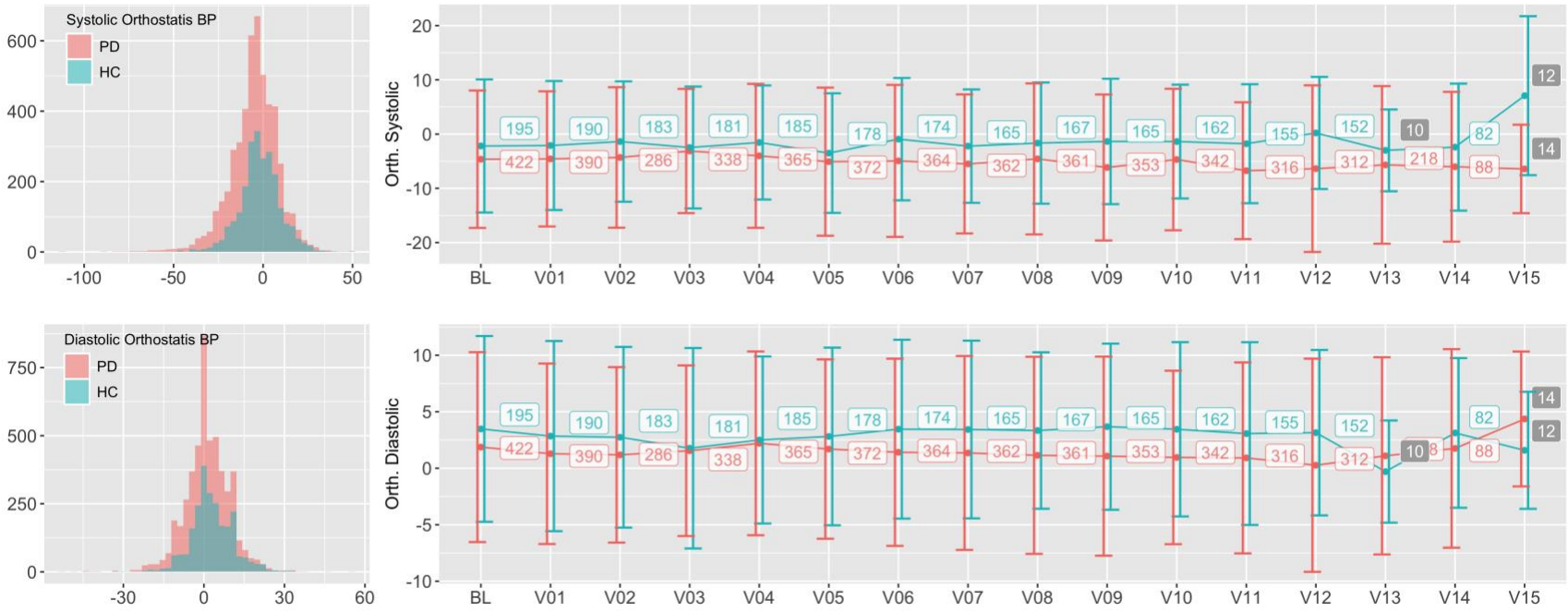


Figure 6 Systolic & Diastolic Orthostatic Blood Pressure

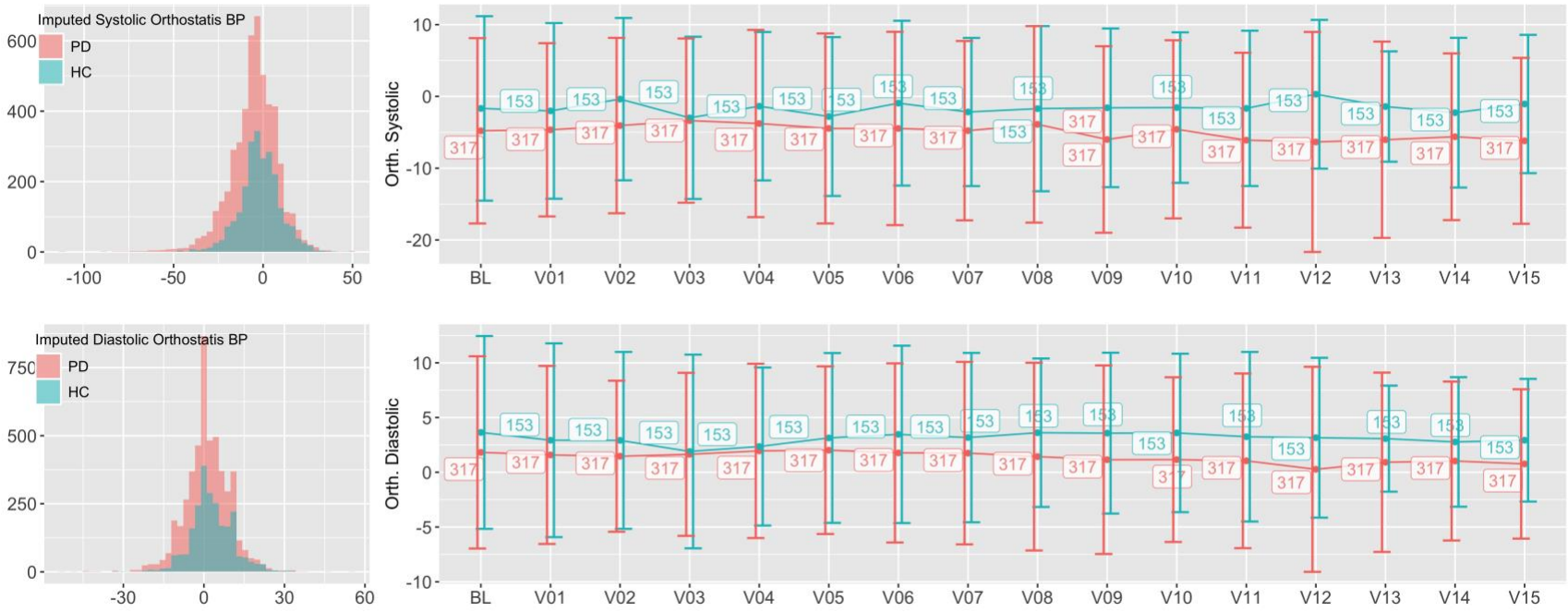


Figure 7 Systolic & Diastolic Orthostatic Blood Pressure (imputed)

5.3 REM Sleep Behavior Disorder

Rapid Eye Movement Sleep Behavior Disorder is thought to be one of the most frequent preceding symptoms of Parkinson's disease. According to (Zhang et al., 2017), the prevalence of RBD in PD was 42.3%. RBD presents as repeated episodes of sleep-related vocalization and/or motor behaviors. These behaviors are documented by costly procedure called polysomnography applied during REM sleep (Högl & Stefani, 2017). Polysomnography records brain waves, blood oxygen level, heart rate, breathing, and eye and leg movements.

Alternatively, data can be collected via an RBD screening questionnaire. It is important to note that these questionnaires only enable diagnosis of probable RBD. The most frequently applied questionnaire was devised by Karin Stiasny-Kolster and published in 2007. This questionnaire is comprised of 10 items which are answered by 13 "yes" or "no" questions. Subjects with score ≥ 5 are RBD-positive, while subjects with score < 5 are RBD-negative. The data collection procedure can be conducted by any PCP or APN and takes about 10 minutes.

The line diagrams (Figures 8, 9) illustrate the RBD score means and variances for PD and HC across all recorded visits. The mean difference for the RBD score between PD and HC increases over time. The variance is very high when compared to the mean difference. The histogram for RBD indicates a similar distribution with a high overlap.

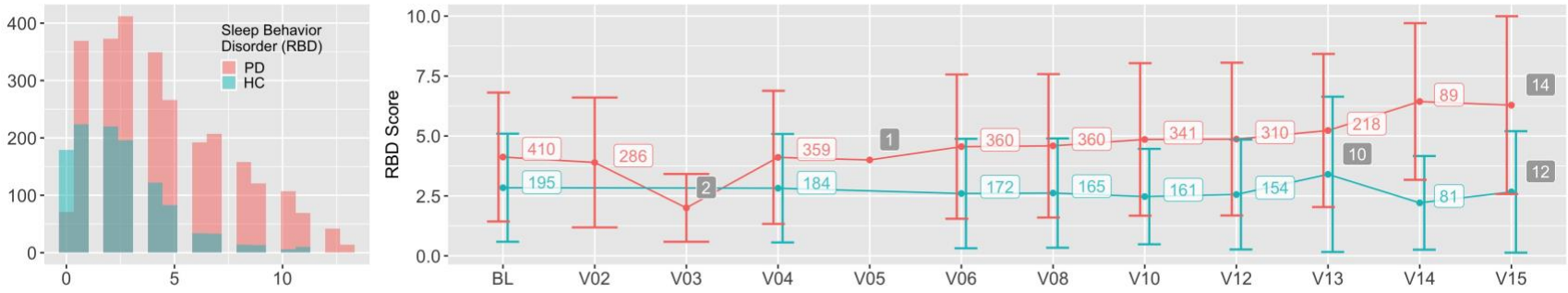


Figure 8 Movement Sleep Behavior Disorder

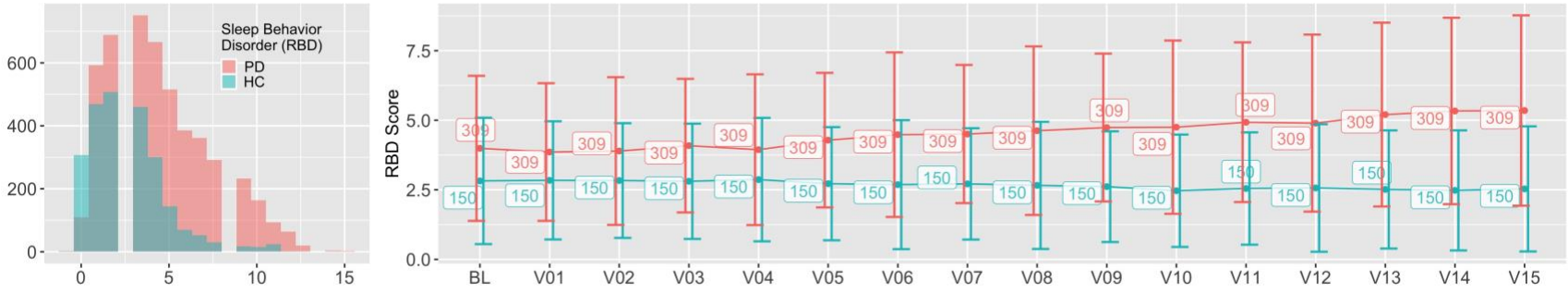


Figure 9 Movement Sleep Behavior Disorder (imputed)

5.4 Montreal Cognitive Assessment

Mild cognitive impairment (MCI), which is an intermediate clinical state between normal cognitive aging and dementia, is a common feature of Parkinson's disease that may be present even in its earliest stages. Studies report prevalence rates of MCI in 25–50% of PD (Weil et al., 2018). The Montreal Cognitive Assessment (MoCA) is often used to assess MCI. The MoCA test is a one-page, 30-point test that can be administered by an APN in around 10 minutes. However, only a health professional with expertise in the cognitive field should interpret the results.

The line diagrams (Figures 10, 11) illustrate the MCI score means and variances for PD and HC across all recorded visits. The mean difference for the MCI scores between PD and HC stays constant over time. The variance is very high when compared to the mean difference. The MCI score variance of PD is higher than the variance of HC. The histogram indicates a large overlap between PD and HC, whereby the PD histogram is slightly skewed to the left (lower scores). The imputed data set follows the original data set.

5.5 Unified Parkinson's Disease Rating Scale II + III

The Movement Disorder Society's (MDS) Unified Parkinson's Disease Rating Scale (UPDRS) is used to quantify the progression of Parkinson's disease. Part II assesses a patient's perception of their ability to carry out activities of daily living including dressing, walking, and eating. Part III covers motor evaluation of disability and includes ratings for tremor, slowness (bradykinesia), stiffness (rigidity), and balance (Goetz et al., 2008). The scores for Part II vary for PD from about 1 to 30, and Part III from about 10 to 40. Part II is self-reported and can be filled out in the waiting room by the patient or caregiver. The review of Part II and filling of Part III can take up to 30 minutes for an APN or PCP. The diagrams (Figures 12, 13, 14, 15) illustrate the MDS-UPDRS score means and variances for PD and HC across all recorded visits.

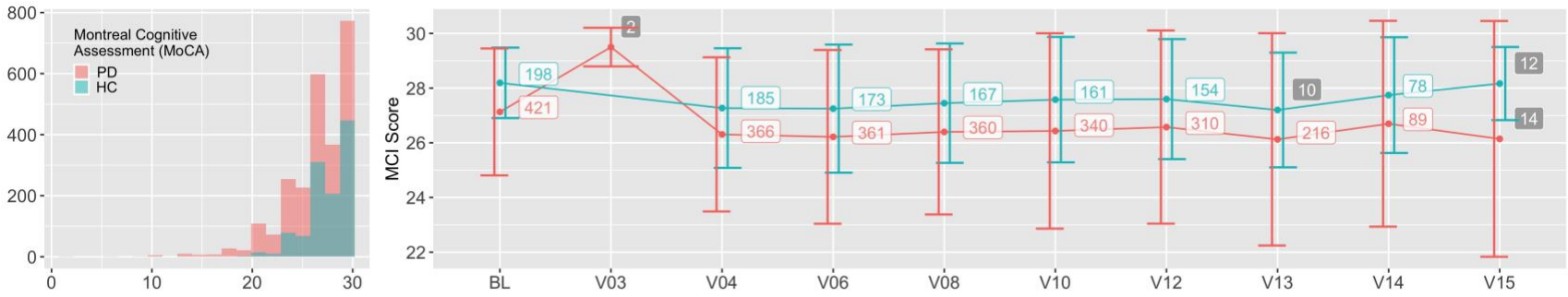


Figure 10 Mild Cognitive Impairment

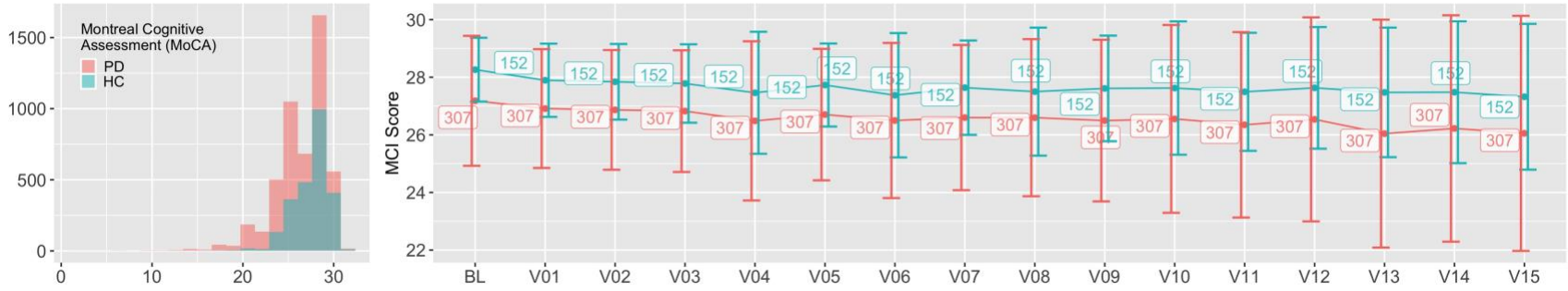


Figure 11 Mild Cognitive Impairment (imputed)

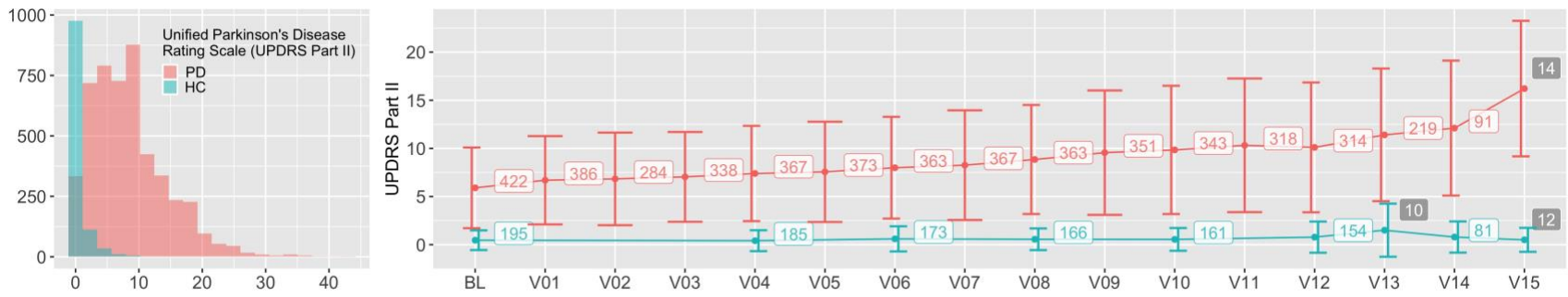


Figure 12 UPDRS Part II

Figure 13 UPDRS Part II (imputed)

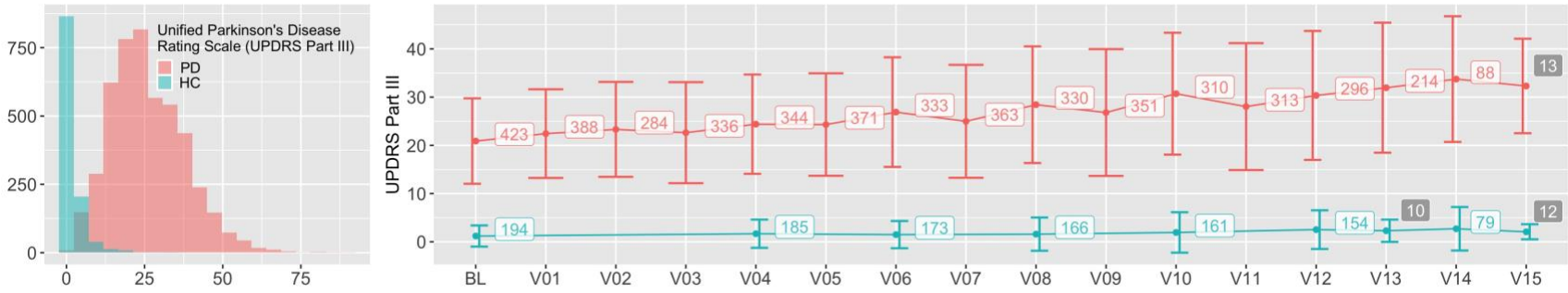


Figure 14 UPDRS Part III

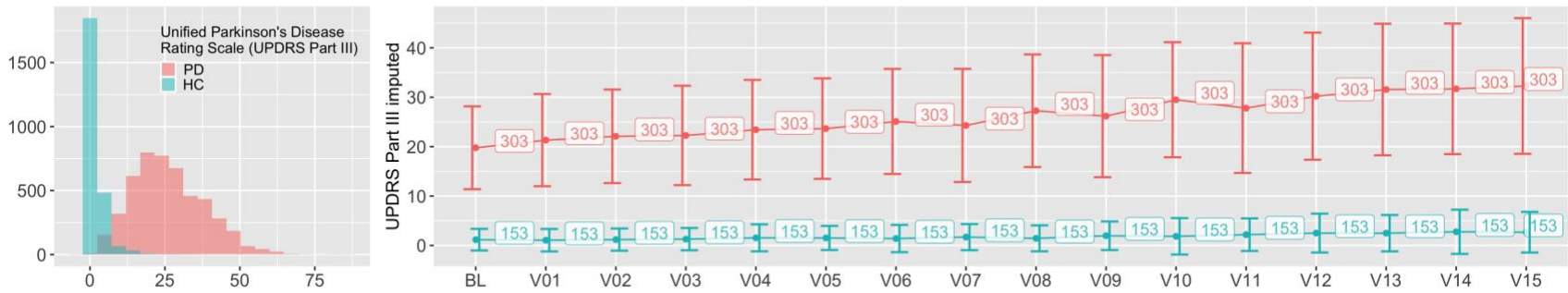


Figure 15 UPDRS Part III (imputed)

The mean difference for the UPDRS scores between PD and HC increases significantly over time and can be discerned easily. The variance is low compared to the mean difference. The score variance of PD is clearly higher than the variance of HC. The histogram indicates a small overlap between PD and HC, whereby the PD histogram is heavily skewed to the right (higher scores). The imputed data set follows the original data set.

5.6 Geriatric Depression Scale

Depressive symptoms are the most common non-motor symptoms of Parkinson's disease, and the Geriatric Depression Scale (GDS) is one of the most commonly used self-rating depression scales in geriatric populations (Kurlowicz & Greenberg, 2007). It is generally accepted that clinically-significant depressive disturbances occur in 40–50 % of patients with PD (Marsh, 2015). Geriatric depression is self-reported and the questionnaire can be filled out in the waiting room by the patient or caregiver.

The diagrams (Figure 16, 17) illustrate the GDS score means and variances for PD and HC across all recorded visits. The mean difference for the GDS score between PD and HC stays close to constant over time. The variance is very high compared to the mean difference. The GDS score variance of PD is higher than the variance of HC. The histogram indicates some overlap between PD and HC, whereby the PD histogram is skewed to the right (higher scores). The imputed data set follows the original data set.

5.7 Anxiety

Anxiety disorders are common in PD and may include panic, agoraphobia, obsessive-compulsive disorder, and social phobia. The prevalence in PD is about 30% with non-episodic anxiety being more prevalent than episodic anxiety (Broen et al., 2016). Anxiety can be identified with The State-Trait Anxiety Inventory (STAI).

The State-Trait Anxiety Inventory is a psychological inventory based on a 4-point Likert scale and consists of 40 questions on a self-reported basis (Renzi, 1985). The questionnaire can be answered in the waiting room by the patient or caregiver and reviewed by a PCP or APN.

These diagrams (Figures 18, 19) illustrate the State/Trait Anxiety score means and variances for PD and HC across all recorded visits. The mean difference for the State/Trait Anxiety score between PD and HC does not increase over time. The variance is high compared to the mean difference. The histogram indicates an overlap between PD and HC, whereby the PD histogram is skewed to the right (higher scores). The imputed data set follows the original data set.

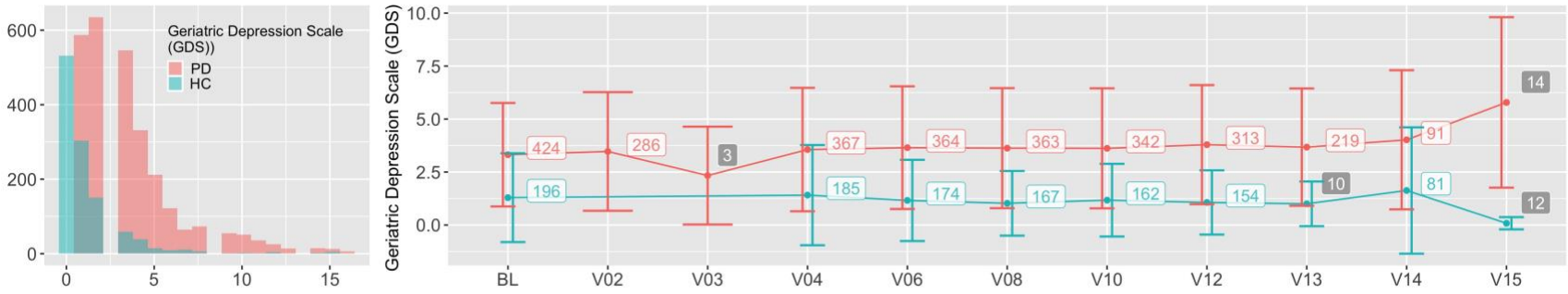


Figure 16 Geriatric Depression

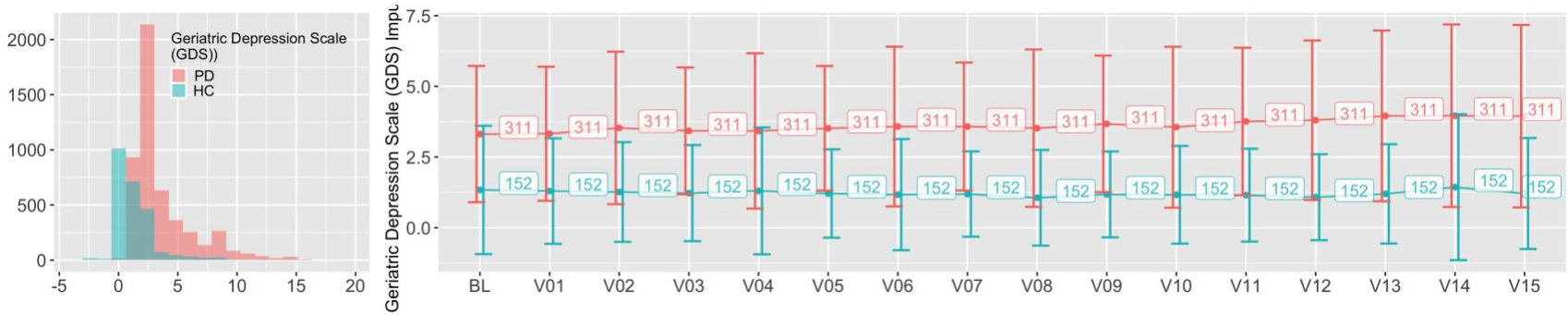


Figure 17 Geriatric Depression (imputed)

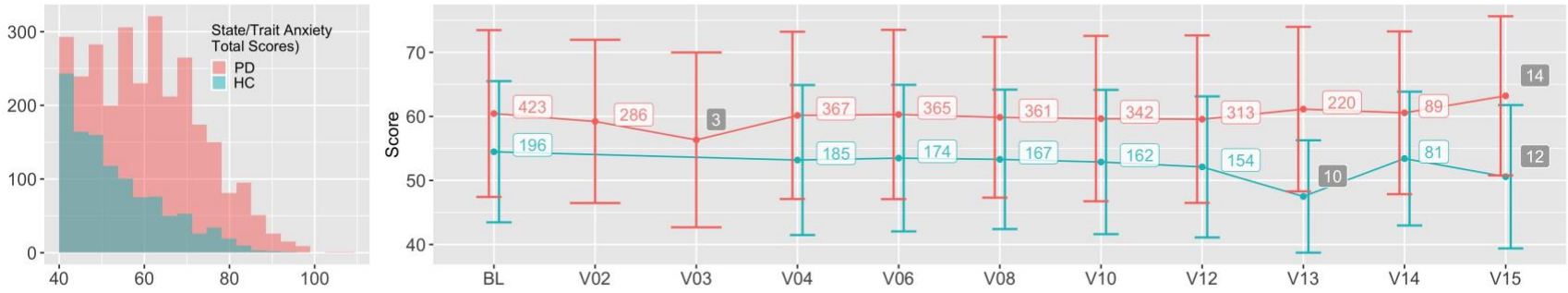


Figure 18 Anxiety

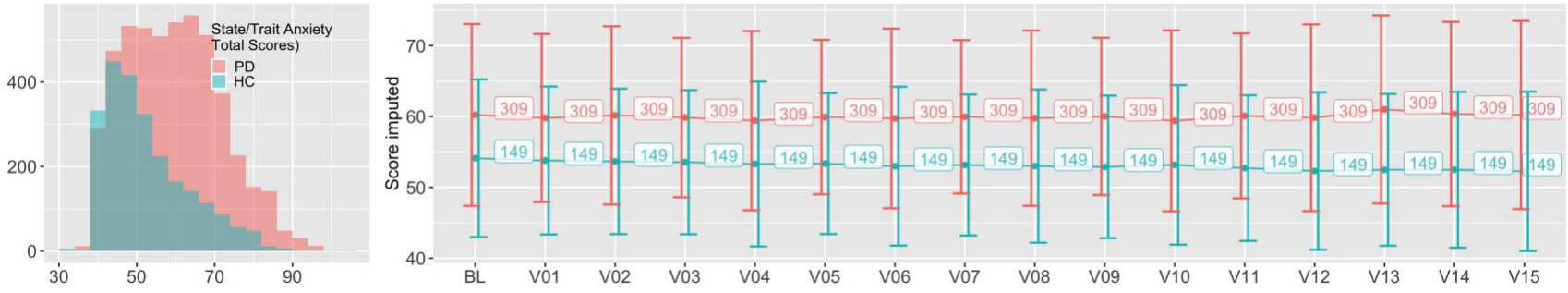


Figure 19 Anxiety (imputed)

6 VARIABLE AGGREGATION

6.1 Aggregation to Motor and Non-Motor Scores

Parkinson's patients show symptoms, which are represented by the surrogate variables, in combinatorial diversity and intensity. To simply the following analytical steps, surrogate variables were accumulated into motor and non-motor symptoms. The data was normalized via min-max normalization. This is method is preferred over z-score normalization when the distribution shape needs to be preserved.

EQUATION 1 DATA NORMALIZATION

$$x_{normalized} = \frac{x_i - \mathrm{x}_{min}}{\mathrm{x}_{max} - \mathrm{x}_{min}}$$

Motor and non-motor related scores were added up to calculate a representative score of a subject's motor and non-motor condition. However, I wanted to keep all measurements that present the mental and cognitive state of a subject in one group therefore I counted the orthostatic blood pressure as a motor symptom.

EQUATION 2 DATA AGGREGATION

$$Motoric = \sum_{i=1}^{n} \sum_{v=1}^{visits} UPDRS_{Part\ II} + UPDRS_{Part\ III} + SYSORHY$$

$$Non - Motoric = \sum_{i=1}^{n} \sum_{v=1}^{visits} ANX + GDS + MoCA + RBD$$

Table IV documents the resulting data set with the following characteristics after imputation, normalization, and accumulation:

TABLE IV AGGREGATED DATA SET

PD	HC	Subjects
289	147	436
Events	Records	Variables
16	6976	2

6.2 Motor and Non-Motor Data Set

The upper scatterplot (Figure 20, BL... V15) displays the normalized motor and non-motor scores from all subjects measured over 16 visits. The middle plot displays the scores of their initial visits for baselining (Figure 20, BL), and the lower scatterplot (Figure 20 V15) illustrates their scores at the 15th visit. All three plots illustrate PD and HC clusters with a visible overlap. The disease progression can be inferred by comparing BL with V15 recognizing the greater spread of motor and non-motor scores.

6.2.1 Non-Motor Scores

This histogram and boxplot (Figure 21) illustrate the distribution of non-motor scores for both HC and PD. The mean non-motor score for PD is higher than for HC. The distributions are independent, unimodal, and skewed to the right, whereby PD's spread to the right is much larger. The boxplot illustrates the median differences and distributions among all 16 visits. Subsequently, PD and HC means are compared for each visit. Therefore, their variances will be tested whether they are significantly different from each other to decide if the two-sample t-test with unequal variances is appropriate (Walker et al., 1989). Table V demonstrates that the means of PD and HC are significantly different for non-motor symptoms for all visits.

Figure 20 Scatter Plot Non-Motor vs. Motor

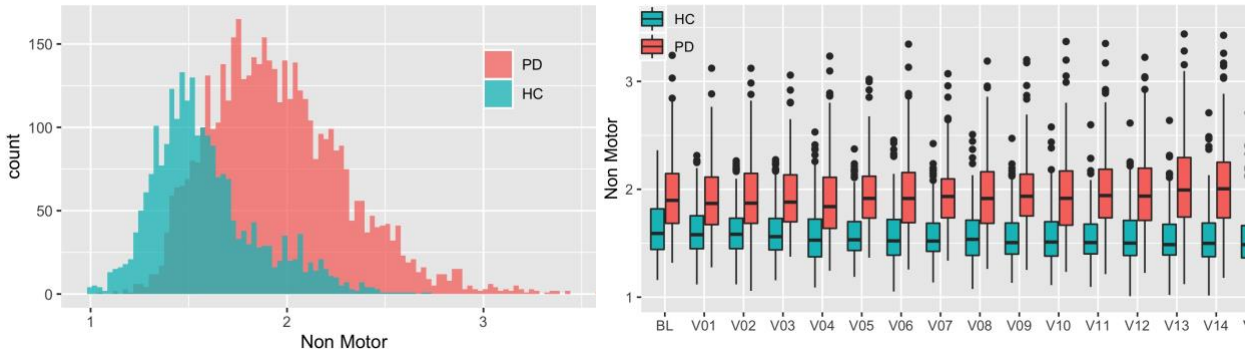


Figure 21 Non-Motor Distribution

EQUATION 3 NON-MOTOR HYPOTHESIS TESTS

Test of Variance Equality	**Two - Sample t-Test, IID, Unequal Variances**	
$H_0 : d_{PD}^2 = d_{HC}^2$ $H_1 : d_{PD}^2 \neq d_{HC}^2$ $If: \quad F = \frac{d_{PD}^2}{d_{HC}^2} > F_{n_{PD}-1;n_{HC}-1,1-a/2} \quad OR$ $F = \frac{d_{PD}^2}{d_{HC}^2} < F_{n_{PD}-1;n_{HC}-1,1-a/2}$ $then\ H_0\ is\ rejected.$	$H_0 : \overline{PD} = \overline{HC}$ $H_1 : \overline{PD} \neq \overline{HC}$ $If: \quad t > (t_{d',1-a/2}) \quad OR$ $t < (-t_{d',1-a/2})$ $then\ H_0\ is\ rejected.$	$Test\ statistic:$ $t = \frac{\overline{PD} - \overline{HC}}{s\sqrt{d_{PD}^2 / n_{PD} + d_{HC}^2 / n_{HC}}}$ $Approximat\ Degrees\ of\ Freedom:$ $d' = \frac{\left(d_{PD}^2 / n_{PD} + d_{HC}^2 / n_{HC}\right)^2}{\frac{\left(d_{PD}^2 / n_{PD}\right)^2}{\left(n_{PD} - 1\right)} + \frac{\left(d_{HC}^2 / n_{HC}\right)^2}{\left(n_{HC} - 1\right)}};$

6.2.2 Motor Scores

The histogram and boxplot (Figure 22) illustrate the distribution of motor scores for both HC and PD. The mean motor score for PD is higher than for HC. Both distributions are independent and unimodal. The HC distribution is symmetrical and narrow, whereby the PD distribution is skewed to the right and spread out. The boxplot illustrates the median differences and distributions among all 16 visits.

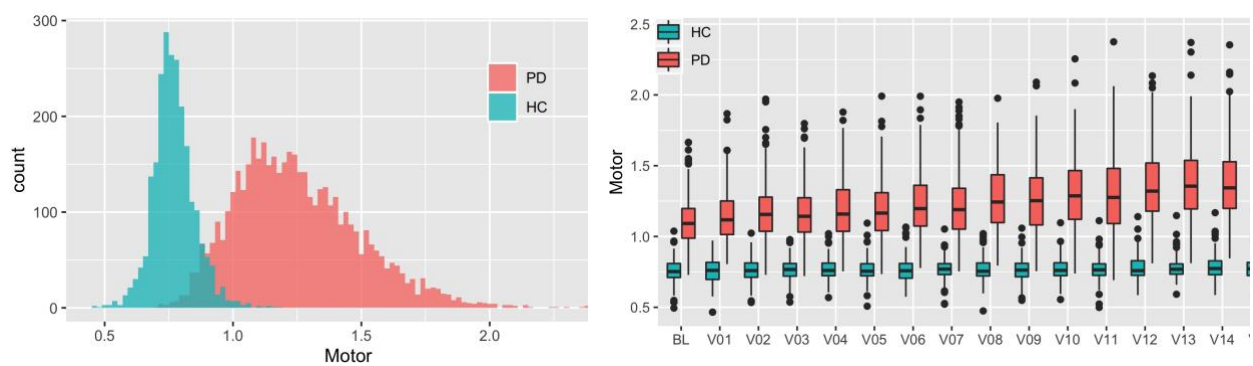


Figure 22 Motor Distribution

Subsequently, PD and HC means are compared for each visit. Table VI demonstrate that the means of PD and HC are significantly different for non-motoric symptoms for all visits.

TABLE V NON-MOTOR HYPOTHESIS TEST RESULTS

Visit	V: Ho	V: p.value	SD: PD	SD: HC	M: Ho	M: p.value	M: PD	M: HC
BL	rejected	0.00658	0.332	0.272	rejected	2.26E-21	1.94	1.63
V01	rejected	0.00232	0.322	0.257	rejected	1.62E-21	1.91	1.62
V02	rejected	2.08E-05	0.343	0.249	rejected	1.13E-23	1.92	1.61
V03	rejected	0.00681	0.309	0.253	rejected	6.7E-26	1.92	1.61
V04	rejected	0.0387	0.351	0.301	rejected	1.73E-18	1.9	1.6
V05	rejected	0.00886	0.303	0.25	rejected	3.59E-30	1.94	1.59
V06	rejected	0.00597	0.351	0.286	rejected	6.41E-26	1.94	1.58
V07	rejected	0.00867	0.306	0.252	rejected	7.26E-33	1.95	1.59
V08	rejected	0.00177	0.352	0.279	rejected	3.73E-28	1.94	1.57
V09	rejected	0.00256	0.319	0.255	rejected	5.63E-36	1.97	1.57
V10	rejected	5.89E-05	0.365	0.27	rejected	1.33E-29	1.95	1.57
V11	rejected	0.00086	0.341	0.266	rejected	1.52E-36	1.98	1.56
V12	rejected	0.000193	0.365	0.276	rejected	5.79E-34	1.98	1.56
V13	rejected	1.23E-06	0.395	0.273	rejected	3.47E-39	2.02	1.54
V14	rejected	0.000252	0.393	0.299	rejected	4.85E-34	2.02	1.56
V15	rejected	4.14E-05	0.4	0.294	rejected	3.9E-35	2.01	1.55

TABLE VI MOTOR HYPOTHESIS TEST RESULTS

Visit	V: Ho	V: p.value	SD: PD	SD: HC	M: Ho	M: p.value	M: PD	M: HC
BL	rejected	0	0.17	0.0856	rejected	9.9E-102	1.11	0.757
V01	rejected	0	0.179	0.088	rejected	1.1E-108	1.14	0.758
V02	rejected	0	0.195	0.0793	rejected	1.3E-110	1.17	0.762
V03	rejected	0	0.187	0.0783	rejected	8E-111	1.16	0.767
V04	rejected	0	0.198	0.0764	rejected	1.6E-110	1.18	0.77
V05	rejected	0	0.206	0.0781	rejected	5.8E-109	1.19	0.765
V06	rejected	0	0.208	0.086	rejected	2.3E-115	1.22	0.769
V07	rejected	0	0.222	0.0757	rejected	9.9E-107	1.21	0.769
V08	rejected	0	0.227	0.0844	rejected	1E-116	1.27	0.77
V09	rejected	0	0.244	0.0823	rejected	4.4E-109	1.27	0.766
V10	rejected	0	0.244	0.0803	rejected	3.8E-119	1.31	0.772
V11	rejected	0	0.261	0.0803	rejected	9.9E-110	1.3	0.769
V12	rejected	0	0.248	0.0829	rejected	1.5E-123	1.34	0.775
V13	rejected	0	0.267	0.0715	rejected	3.9E-120	1.37	0.776
V14	rejected	0	0.261	0.0888	rejected	6.5E-121	1.37	0.785
V15	rejected	0	0.27	0.08	rejected	7.1E-123	1.39	0.779

7 PREDICTING PARKINSON'S

7.1 Clustering Parkinson's

This scatter plot (Figure 23) illustrates the relationship between non-motor and motor functions in Parkinson's patients. The plot indicates a correlation (r = 0.2279, CI 95% [0.196, 0.254]) but no visually identifiable cluster that could indicate intrinsic Parkinson's subtypes. This is also true for using a log or exponential scale of the axis. The K-means algorithm (Hastie, 2017), as one of the most used algorithms for clustering is applied to detect potentially-hidden clusters.

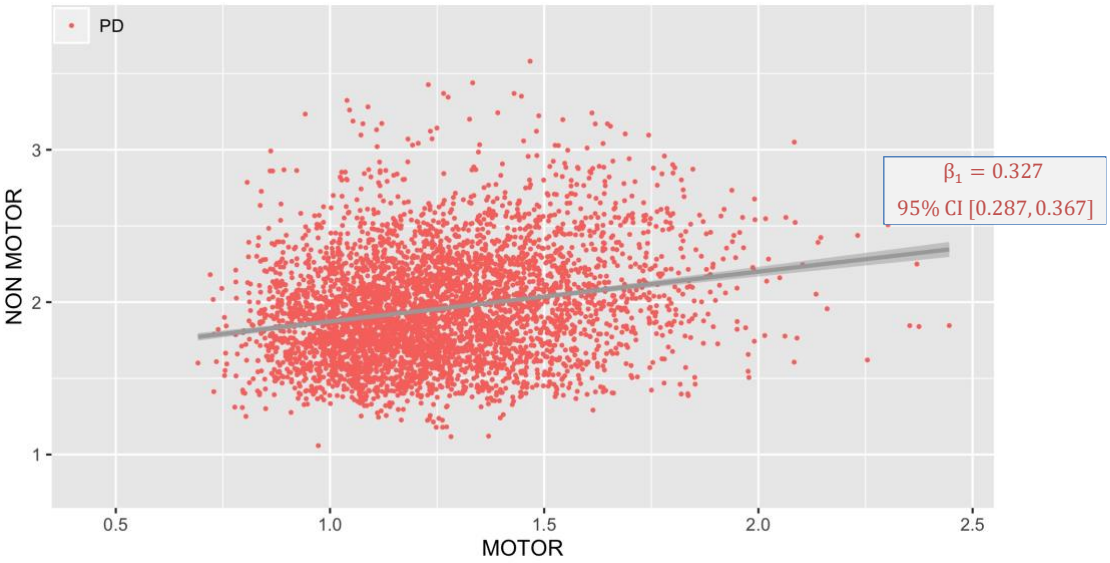


Figure 23 Association Non-Motor to Motor

Since the algorithm expects the number of anticipated clusters from the user, two popular methods to estimated number of clusters are used. The elbow and silhouette method recommend two clusters (Figure 24). The K-means algorithm identifies regions with the highest compactness and maximal distance to each

other. Applying the K-means algorithm yield the following: only ~45.5% of the total variance is explained cluster variance (Table VII). The clusters provide inconclusive information, which can be visually confirmed. The data surrogates and pre-processing approach applied convey insufficient information to reveal potential Parkinson's subtypes.

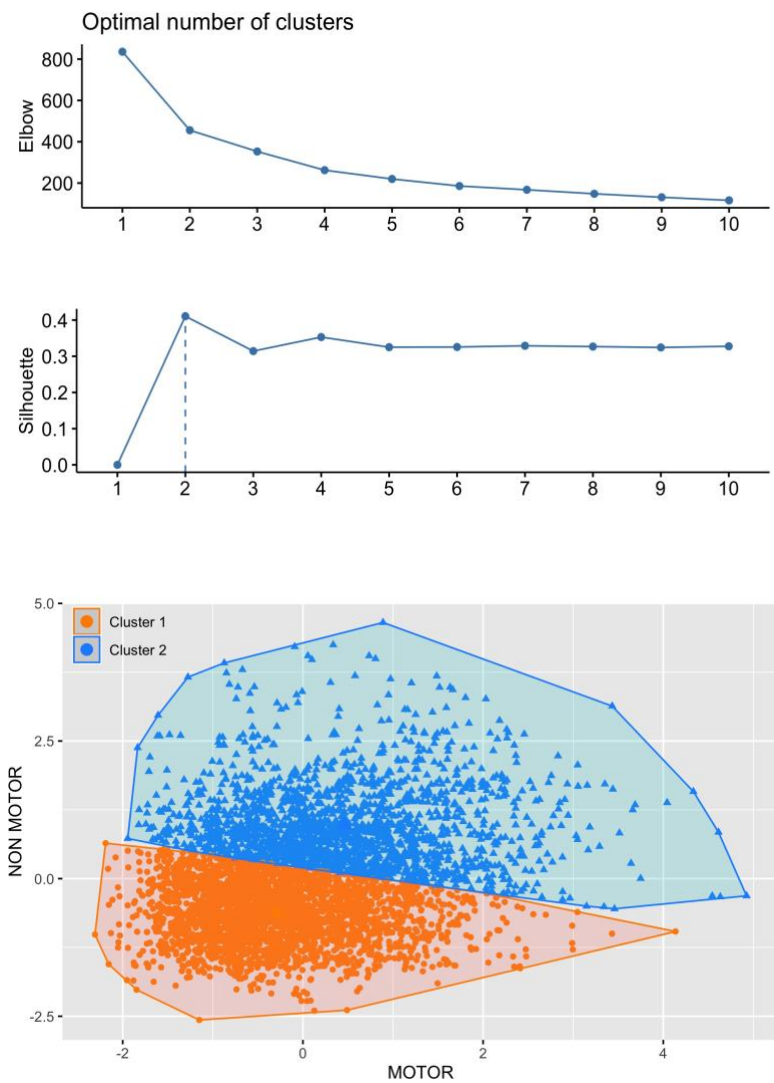


Figure 24 K-Means Clustering

TABLE VII K-MEANS CLUSTER VARIATION

Total Variance	TotSS	835.75
Variance cluster 1	WithinSS	217.67
Variance cluster 1	WithinSS	237.98
Variance explained by distance	BetweenSS	380.1
% of variance explained by clusters	BetweenSS/TotalSS	45.48%

7.2 Predicting Parkinson's

7.2.1 Introduction

In this chapter, I selected a predictive model to discriminate between HC and PD at their first doctor visit. For prediction I used the aggregated data set (Motor, Non-Motor) that contains data over a time span of ~4 years, which entails subjects with no disease, subjects recently diagnosed with Parkinson's, or subjects diagnosed or a longer time ago. The assumption is that in rural areas the subjects who see a doctor for an initial examination have a disease status that follows the distribution showed in Figure 1.

My data set is labeled with 2 classes (PD, HC), is 2-dimensional, imbalanced, consists of continuous data, and as the scatter diagram suggests, linear as well as non-linear classifiers to discriminate between HD, and PD could be successful. Further the statistical power of the data set is high as shown under 6.2.1 and 6.2.2. The sample size is compared to the number of variables large. Based on those parameters I decided to compare the performance of Logistic Regression, linear classifier, and Support Vector Machine as well as Random Forest, both non-linear classifiers. One difference between SVM and Random Forest is that SVM is hardly scalable beyond 10^5 features. Since I am dealing with a very low dimensional space both non-linear classifiers are valid choices for a comparison.

7.2.2 Prediction Method Selection Process

Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM) are the investigated methods to predict PD based on test data and real examination data gathered on the subject's initial visit. Five-fold cross-validation with training data was used to parametrize and select the best model per prediction method. Please see Figure 25 to understand the entire model development process.

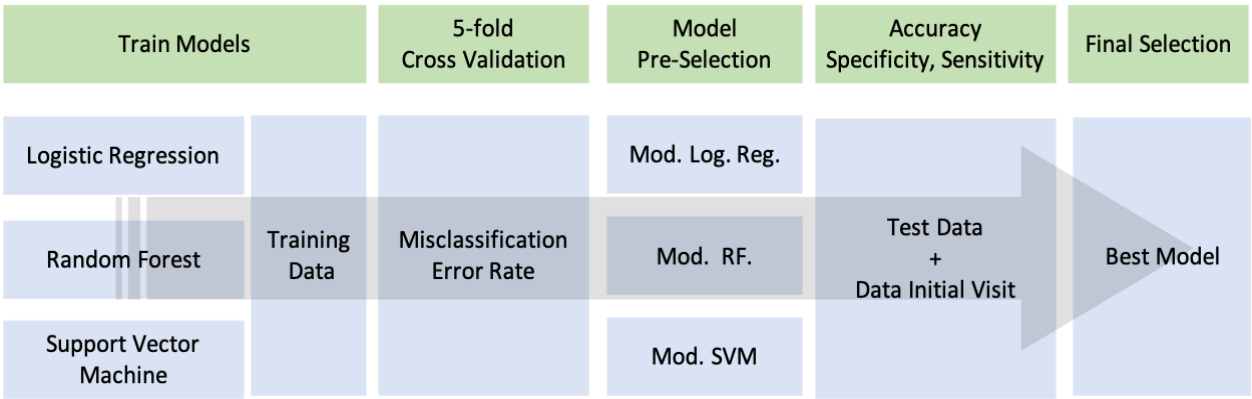


Figure 25 Model Development Process

7.2.3 Creating Training and Test Data Sets

The imputed and imbalanced data set was split (73%, 27%) into an imbalanced Training Data Set and a balanced Testing Data Set. The training data set contains the known labels PD and HC. The model learns with the training data set in order to be generalized to other data sets later on. The test data set is used to test the model's prediction specificity, sensitivity, and accuracy to discriminate between PD and HC. Table VIII provides the overview on how the imputed and imbalanced data set has been split for training and testing.

TABLE VIII TEST & TRAINING DATA SET

Data Set imputed, imbalanced		Training	Testing *All*	Testing *Initial Visit*
HC	2352	1427	925	56
PD	4624	3699	925	56
Sum	6976 (100%)	5126 (73%)	1850 (26%)	

7.3 Logistic Regression

The Logistic Regression (LR) model is a binary classification regression model. The 5 k-fold cross validation technique (Dalpiaz, 2017) with training data was used to select the best model.

7.3.1 Model Selection

Out of three models Model 1 has the lowest misclassification rate as shown in Table IX.

TABLE IX LR MISCLASSIFICATION RATE

Model 1 - 3	5-fold cross validation misclassification rate
$Model\ 1\text{:}\ \log\left[\frac{p(X)}{1-p(X)}\right] = \beta_0 + \beta_1 X_1 + \beta_2 X_2$	0.0269
$Model\ 2\text{:}\ \log\left[\frac{p(X)}{1-p(X)}\right] = \beta_0 + \beta_1 X_1$	0.0327
$Model\ 3\text{:}\ \log\left[\frac{p(X)}{1-p(X)}\right] = \beta_0 + \beta_2 X_2$	0.1473

X_1 = Motor, X_2 = Non-Motor

7.3.2 Performance Test with all Test - and Initial Visit Test Data

Table X lists the performance results of logistic regression Model 1 using the testing data set.

TABLE X LR TEST RESULTS

	LR All Testing	LR Initial Visit
Accuracy	0.9492	0.9286
Sensitivity	0.9373	0.8750
Specificity	0.9611	0.9821

7.4 Random Forest

The Random Forest (RF) algorithm, proposed by L. Breiman in 2001, has been extremely successful as a general-purpose classification and regression method. It combines several randomized decision trees and aggregates their predictions by averaging. RF doesn't require feature scaling, needs very little parameter tuning, and is easily interpretable.

7.4.1 Parametrization Via 5-fold Cross Validation

A 5 k-fold cross validation technique with training data was used to identify the best parameters for RF. The different setups have been compared by their misclassification rate as shown in Table XI. The parameters mtry, which determines the number of variables available for splitting at each tree node, and the number of trees, which is recommended to be set very high (Probst et al., 2019) need to be determined numerically. The table below shows the misclassification rate per parameter combination.

TABLE XI RF MISCLASSIFICATION RATES

mean	ntree	mtry	fold 1	fold 2	fold 3	fold 4	fold 5
0.03843	100	1	0.03711	0.04386	0.04488	0.03512	0.03119
0.03746	1000	1	0.03516	0.03899	0.04390	0.03805	0.03119
0.03707	3000	1	0.03418	0.03996	0.04293	0.03707	0.03119
0.03707	5000	1	0.03418	0.03899	0.04293	0.03805	0.03119
0.03941	100	2	0.03711	0.03899	0.04878	0.04098	0.03119
0.03882	1000	2	0.03613	0.04191	0.04683	0.04098	0.02827
0.03980	3000	2	0.03613	0.04191	0.04878	0.04195	0.03021
0.03921	5000	2	0.03613	0.04191	0.04683	0.04293	0.02827

The model with ntree = 3000, and mtry = 1 has the lowest misclassification rate and performs best. A higher amount of trees (ntree = 5000) leads to overfitting and is therefore not selected.

7.4.2 Performance Test with all Test - and Initial Visit Test Data

Table XII below documents the performance results of random forest using the testing data set.

TABLE XII RF TEST RESULTS

	RF All Testing	RF Initial Visit
Accuracy	0.9481	0.9545
Sensitivity	0.9719	0.9818
Specificity	0.9243	0.9273

7.5 Support Vector Machine

The Support Vector Machine (SVM) is a tool for binary classification. It attempts to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane. The SVM controls the hyperplane to be a linear or non-linear boundary via a kernel. (James et al., 2013).

7.5.1 Kernel Parametrization and Selection Via 5-fold Cross Validation

Linear, polynomial, and radial kernels were compared and Table XIII lists the parametrization space used to identify the best setup. The 5-fold cross validation method was applied to identify the best tuning parameters. Cost is the penalty parameter of the error term for all kernels. Degree is the parameter of the order of the polynomial kernel and allows for splitting the data in a non-linear fashion. Gamma is a parameter of the radial kernel and allows encircling data points class for separation.

TABLE XIII SVM KERNEL SVM PARAMETRIZATION

Kernel	linear	polynomial	radial
Tuning Parameter	cost = [.1,1,10,100 500, 1000]	cost = [0.01, 0.1, 1,10, 40, 50] degree = [2,3,4*]) 4 didn't converge	cost = [.1,1,10, 50,100] γ = [0.01,0.1, 1,10, 100]

Table XIV displays the best tuning parameters for each SVM kernel with misclassification rates. The SVM with linear, polynomial, and radial kernel perform similarly well and are considered for testing.

TABLE XIV SVM MISCLASSIFICATION RATE

Kernel / Fold	linear cost = 10	polynomial cost = 10, degree = 3	radial cost = 10, γ = 0.1
fold 1	0.0419	0.0331	0.0370
fold 2	0.0283	0.0293	0.0254
fold 3	0.0312	0.0332	0.0312
fold 4	0.0351	0.0341	0.0351
fold 5	0.0341	0.0410	0.0371
Mean	0.0341	0.0341	0.0332

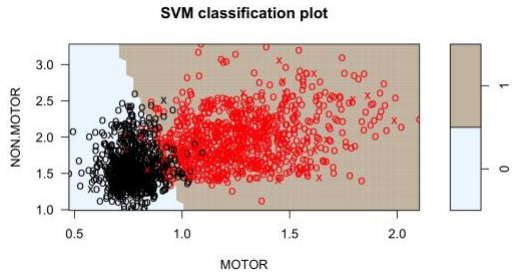
7.5.2 Performance Assessment with Test- and Initial Visit Data

Figure 26 documents the kernel parametrization and visualizes the hyperplane between PD and HC.

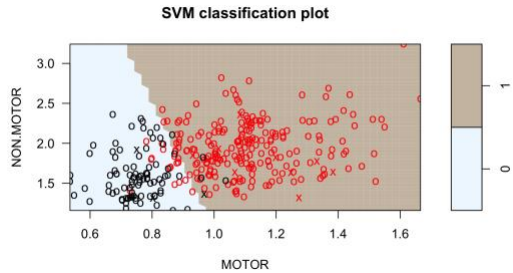
SVM - Linear
cost = 10

$$K(x_i, x_i`) = \sum_{j=1}^{p} (x_{ij}\, x_i`_j)$$

All Testing Data



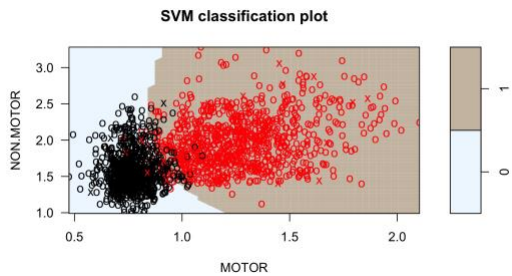
Initial Visit



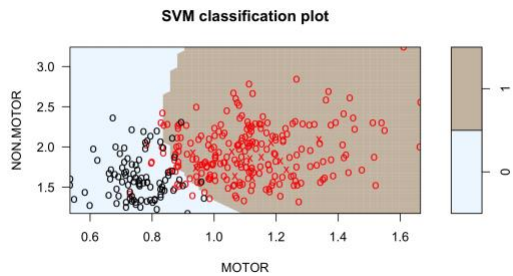
SVM - Polynomial
cost = 10, degree = 3

$$K(x_i, x_i`) = \left(1 + \sum_{j=1}^{p} (x_{ij}\, x_i`_j)\right)^d$$

All Testing Data



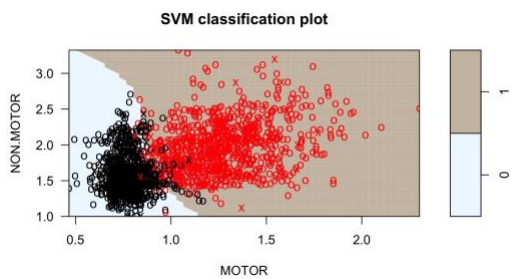
Initial Visit



SVM - Radial
cost = 10, γ = 0.1

$$K(x_i, x_i`) = \exp\left[-\gamma \sum_{j=1}^{p} (x_{ij} - x_i`_j)^2\right]$$

All Testing Data



Initial Visit

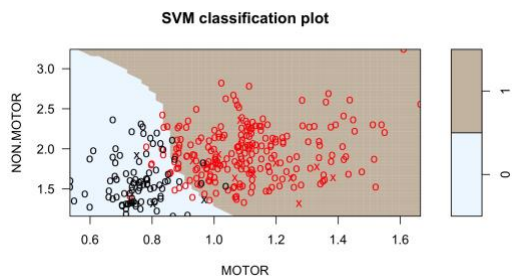


Figure 26 SVM Kernel Overview with Hyperplanes

All 3 SVM Kernels perform similarly well on the test and initial visit data and all of the kernels could be used to discriminate between HC and PD. Table XVI documents the performance results of support vector machine per kernel and test data set.

TABLE XV SVM TEST RESULTS PER KERNEL

	Linear		Polynomial		Radial	
	All Testing Data	Initial Visit	All Testing Data	Initial Visit	All Testing Data	Initial Visit
Accuracy	0.9465	0.9414	0.9508	0.9444	0.9497	0.9444
Sensitivity	0.9632	0.9442	0.9632	0.9485	0.9622	0.9485
Specificity	0.9297	0.9341	0.9384	0.9341	0.9373	0.9341

7.6 Results and Discussion

Table XVII displays an overview of the performance of all classifiers, which perform similarly well in discriminating between PD and HC. Regarding accuracy, Random Forest and Support Vector Machine demonstrate with 95.54%, given initial visit data, and 95.08%, given testing data the best performance. The highest sensitivities can be achieved with Random Forest. Logistic Regression is best in specificity.

TABLE XVI PERFORMANCE COMPARISON OF LR, RF, SVM

	LR All Testing	LR Initial Visit	RF All Testing	RF Initial Visit	SVM Poly All Testing	SVM Poly Initial Visit
Accuracy	0.9492	0.9286	0.9481	0.9554	0.9508	0.9444
Sensitivity	0.9373	0.8750	0.9719	0.9818	0.9632	0.9485
Specificity	0.9611	0.9821	0.9243	0.9273	0.9384	0.9341

The Parkinson's subtypes could not be identified with the variable surrogates and the K-means algorithm. One reason for this could be quality reduction of the surrogate variables. For example, in the research study (Fereshtehnejad et al., 2015) an overnight polysomnography was used to determine REM sleep behavior disorder, and in contrast the surrogate data came from a RBD screening questionnaire (RBDSQ), which is a self-rating questionnaire that distinguishes on a binary level whether RBD is presented or not (Stiasny-Kolster et al., 2007).

7.7 Recommendations

In my study I could partially harness the fundamental research results of Fereshtehnejad et al., 2015, and translate them into a prediction model that addresses the need for better PD detection in primary care and rural settings. Using machine learning and routinely accessible data to Primary Care Physicians allows for discrimination between HC and PD with an accuracy between 92 to 95 %. The patient's safety could be increased by using this model to rule out Parkinson's as potential disease and decrease the false negative rates. This model could assist PCPs with a specificity >96% (LR) and sensitivity >97% (RF) in the triaging process and potentially reduce the number of referrals to Neurologists by giving PCPs greater confidence (tackling the problem of neurophobia) in whether a subject shows sufficient symptom to be triaged to a specialist. The effort for a primary care physician, though, is large compared to “just” writing a referral to a Neurologists. I believe incentives are needed to encourage wished behavior.

A rough estimation for how long it would take a Primary Care Physician to collect surrogate data and receive a triaging recommendation in show in Table XVIII. I believe this approach has the potential to assist healthcare workers without formal neurological training in providing better care for patients with neurological conditions such as Parkinson's.

TABLE XVII DATA COLLECTION TIME FOR PCP

Estimated Data Collection Time Assumption: questionnaire filled by patient or care giver	
SYSOH	15 min, measure BP standing & supine
RBD	10 min, review questionnaire
MoCA	10 min, review questionnaire
UPDRS Part II	15 min, interview
UPDRS Part III	15 min, interview
GDS	15 min, review questionnaire
ANX	15 min, review questionnaire
Sum	1h 35min per patient

7.8 Limitation and Future Research

Not investigated was the model's performance with subjects that have a pre-existing condition other than Parkinson's disease or being healthy, and with data gathered in a Primary Care facility under day to day conditions. A validation of my results is strongly needed. Further, focusing solely on PD limits the impact of this support system. Training the model with subjects diagnosed with Alzheimer's preferable with the same surrogate variables could significantly improve the impact and user acceptance.

7.9 Conclusion

Combining Machine learning-based classification methods (Logistic Regression, Random Forest, and Support Vector Machines) with routinely accessible patient data can support Primary Care Physicians in triaging patients to possibly reduce the burden on neurologists caused by overcautious referrals.

CITED LITERATURE

AlDossary, S., Martin-Khan, M. G., Bradford, N. K., & Smith, A. C. (2017). A systematic review of the methodologies used to evaluate telemedicine service initiatives in hospital facilities. *International Journal of Medical Informatics*, *97*, 171–194.

Arora, S., Venkataraman, V., Zhan, A., Donohue, S., Biglan, K. M., Dorsey, E. R., & Little, M. A. (2015). Detecting and monitoring the symptoms of Parkinson's disease using smartphones: a pilot study. *Parkinsonism & Related Disorders*, *21*(6), 650–653.

Baus, A., Coben, J., Zullig, K., Pollard, C., Mullett, C., Taylor, H., Cochran, J., Jarrett, T., & Long, D. (2017). An electronic health record data-driven model for identifying older adults at risk of unintentional falls. *Perspectives in Health Information Management*, *14*(Fall).

Broen, M. P. G., Narayen, N. E., Kuijf, M. L., Dissanayaka, N. N. W., & Leentjens, A. F. G. (2016). Prevalence of anxiety in Parkinson's disease: A systematic review and meta-analysis. *Movement Disorders*, *31*(8), 1125–1133. https://doi.org/10.1002/mds.26643

Byeon, H. (2020). Is the Random Forest Algorithm Suitable for Predicting Parkinson's Disease with Mild Cognitive Impairment out of Parkinson's Disease with Normal Cognition? *International Journal of Environmental Research and Public Health*, *17*(7), 2594.

Cai, Z., Gu, J., & Chen, H.-L. (2017). A new hybrid intelligent framework for predicting Parkinson's disease. *IEEE Access*, *5*, 17188–17200.

Calne, D., & Langston, J. W. (1983). Aetiology of Parkinson's disease. *The Lancet*, *322*(8365–8366), 1457–1459.

Cohen, O., Pullman, S., Jurewicz, E., Watner, D., & Louis, E. D. (2003). Rest tremor in patients with essential tremor: prevalence, clinical correlates, and electrophysiologic characteristics. *Archives of Neurology*, *60*(3), 405–410.

Dalpiaz, D. (2017). *R for Statistical Learning*. Applied Statistics with R - David Dalpiaz

Dunn, J., Runge, R., & Snyder, M. (2018). Wearables and the medical revolution. *Personalized Medicine*, *15*(5), 429–448.

Fallon, J. M., Fallon, J. J., Heil, M., & Weiss, S. J. (2010). *Systems and methods employing remote data gathering and monitoring for diagnosing, staging, and treatment of parkinsons disease, movement and neurological disorders, and chronic pain*. Google Patents.

Fereshtehnejad, S. M., Romenets, S. R., Anang, J. B. M., Latreille, V., Gagnon, J. F., & Postuma, R. B. (2015). New clinical subtypes of Parkinson disease and their longitudinal progression a prospective cohort comparison with other phenotypes. *JAMA Neurology*, *72*(8), 863–873. https://doi.org/10.1001/jamaneurol.2015.0703

Frosch, D. L., May, S. G., Rendle, K. A. S., Tietbohl, C., & Elwyn, G. (2012). Authoritarian physicians and patients' fear of being labeled 'difficult'among key obstacles to shared decision making. *Health Affairs*, *31*(5), 1030–1038.

Gaenslen, A., Swid, I., Liepelt-Scarfone, I., Godau, J., & Berg, D. (2011). The patients' perception of prodromal symptoms before the initial diagnosis of Parkinson's disease. *Movement Disorders*, *26*(4), 653–658.

Goetz, C. G., Tilley, B. C., Shaftman, S. R., Stebbins, G. T., Fahn, S., Martinez-Martin, P., Poewe, W., Sampaio, C., Stern, M. B., Dodel, R., Dubois, B., Holloway, R., Jankovic, J., Kulisevsky, J., Lang, A. E., Lees, A., Leurgans, S., LeWitt, P. A., Nyenhuis, D., … Zweig, R. M. (2008). Movement Disorder Society-Sponsored Revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results. *Movement Disorders*, *23*(15), 2129–2170. https://doi.org/10.1002/mds.22340

Gray, V. K. (2000). *Diagnostic enhancement method and apparatus*. Google Patents.

Hastie, T. T. (2017). The Elements of Statistical Learning Second Edition. *Math. Intell.*, *27*(2), 83–85. https://doi.org/111

Högl, B., & Stefani, A. (2017). REM-Schlaf-Verhaltensstörung (RBD): Was gibt es Neues zur Diagnosestellung und Therapie? *Somnologie*, *21*(April 2016), 1–8. https://doi.org/10.1007/s11818-016-0048-6

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 103). https://doi.org/10.1007/978-1-4614-7138-7

Koenen, E. H. (2016). *Predicting Parkinson's Disease subtypes using neuroimaging*.

Kurlowicz, L., & Greenberg, S. A. (2007). The geriatric depression scale (GDS) validation of a geriatric depression screening scale: A preliminary report. *AJN The American Journal of Nursing*, *107*(10), 67–68. http://www.stanford.edu/~yesavage/GDS.html

Loftus, L. (2008). *Negative Reactions to Receiving Family Care in Older Patients with Parkinson's Disease*. The University of Manchester (United Kingdom).

Maraganore, D. M., & Lesnick, T. G. (2008). *Predicting Parkinson's Disease*. Google Patents.

Marek, K., Jennings, D., Lasch, S., Siderowf, A., Tanner, C., Simuni, T., Coffey, C., Kieburtz, K., Flagg, E., Chowdhury, S., Poewe, W., Mollenhauer, B., Sherer, T., Frasier, M., Meunier, C., Rudolph, A., Casaceli, C., Seibyl, J., Mendick, S., … Taylor, P. (2011). The Parkinson Progression Marker Initiative (PPMI). *Progress in Neurobiology*, *95*(4), 629–635. https://doi.org/10.1016/j.pneurobio.2011.09.005

Marsh, L. M. E. D. (2015). Depression and Parkinson's Disease: Current Knowledge. *The Anatomical Record*, *66*(7), 765–773. https://doi.org/10.1007/s11910-013-0409-5.Depression

Peng, J., Guan, J., & Shang, X. (2019). Predicting Parkinson's disease genes based on node2vec and autoencoder. *Frontiers in Genetics*, *10*.

Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(3), 1–19. https://doi.org/10.1002/widm.1301

Renzi, D. A. (1985). State-Trait Anxiety Inventory. *Measurement and Evaluation in Counseling and Development*, *18*(2), 86–89. https://doi.org/10.1080/07481756.1985.12022795

Senard, J. M., Raï, S., Lapeyre-Mestre, M., Brefel, C., Rascol, O., Rascol, A., & Montastruc, J. L. (1997). Prevalence of orthostatic hypotension in Parkinson's disease. *Journal of Neurology Neurosurgery and Psychiatry*, *63*(5), 584–589. https://doi.org/10.1136/jnnp.63.5.584

Snyder, S. H., & D'amato, R. J. (1985). Predicting Parkinson's disease. *Nature*.

Stiasny-Kolster, K., Mayer, G., Schäfer, S., Möller, J. C., Heinzel-Gutenbrunner, M., & Oertel, W. H. (2007). The REM sleep behavior disorder screening questionnaire - A new diagnostic instrument. *Movement Disorders*, *22*(16), 2386–2393. https://doi.org/10.1002/mds.21740

Stiasny-Kolster, K., Mayer, G., Schäfer, S., Möller, J. C., Heinzel-Gutenbrunner, M., & Oertel, W. H. (2007). The REM sleep behavior disorder screening questionnaire—a new diagnostic instrument. *Movement Disorders*, *22*(16), 2386–2393.

Tarsy, D. (2013). Standard strategies for diagnosis and treatment of patients with newly diagnosed Parkinson disease: UNITED STATES. *Neurology: Clinical Practice*, *3*(6), 482–483.

Walker, E., Kleinbaum, D., Kupper, L., & Muller, K. (1989). Applied Regression Analysis and Other Multivariable Methods. *Technometrics*. https://doi.org/10.2307/1270375

Weil, R. S., Costantini, A. A., & Schrag, A. E. (2018). *Mild Cognitive Impairment in Parkinson ' s Disease — What Is It ?*

Zhang, X., Sun, X., Wang, J., Tang, L., & Xie, A. (2017). Prevalence of rapid eye movement sleep behavior disorder (RBD) in Parkinson's disease: a meta and meta-regression analysis. *Neurological Sciences*, *38*(1), 163–170. https://doi.org/10.1007/s10072-016-2744-1

Zietsma, R. C. (2015). *Apparatus for use in diagnosing and/or treating neurological disorder*. Google Patents.

Gavidia, Matthew. "Poll Finds 1 in 4 People With Parkinson Disease Misdiagnosed." Focus-of-the-week. The American Journal of Managed Care. January 08, 2020.

APPENDICES

Appendix A

A1 PPMI-Files

The PPMI data set consists of 11 files.

Randomization_table.csv	Montreal_Cognitive_Assessment__MoCA_.csv
Screening___Demographics.csv	MDS_UPDRS_Part_II__Patient_Questionnaire.csv
Socio-Economics.csv	MDS_UPDRS_Part_III.csv
PD_Features.csv	State-Trait_Anxiety_Inventory.csv
Vital_Signs.csv	Geriatric_Depression_Scale__Short_.csv
REM_Sleep_Disorder_Questionnaire.csv	

A2 R-Packages

I made use of 34 R packages retrieved from the Comprehensive R Archive Network.

boot	e1071	gridExtra	lubridate	pROC	tibble
brew	ellipse	Hmisc	mclust	purrr	tidyverse
caret	factoextra	kableExtra	NMF	randomForest	VIM
cluster	ggfortify	knitr	openxlsx	rattle	wesanderson
devtools	ggplot2	latex2exp	pander	RColorBrewer	
dplyr	ggrepel	lattice	PerformanceAnalytics	stats	

A3 System

The software was developed and tested on the following platform described below.

- R version 3.6.3 (2020-02-29)
- RStudio Version 1.2.5033 (2019)
- MacBook Pro (13-inch, 2018, Four Thunderbolt 3 Ports)

Appendix B

B1 Data Ingestion

```
# DATA INGESTION
df_enrolled_all = GET_ENROLLED()
path = PASTE0(rpath, "/01_DATA/DATA_LONI/07_MEDICAL_HISTORY/")
file = "VITAL_SIGNS.CSV"
df_orth_blood = READ.CSV(PASTE0(path, file))[,C("PATNO", "EVENT_ID", "INFODT", "SYSSUP", "SYSSTND",
                                "DIASUP","DIASTND")]
# DATE
df_orth_blood$INFODT = PARSE_DATE_TIME(df_orth_blood$INFODT,ORDERS = C("MY"))

# MERGE GET APPRDX TO DISTINGUISH BETWEEN CONTROLS AND PATIENTS.
df_orth_blood = MERGE(df_orth_blood, df_enrolled_all[, C("PATNO", "APPRDX")], BY = "PATNO")
df_orth_blood = df_orth_blood[COMPLETE.CASES(df_orth_blood), ]

# CALCULATE ORTHOSTATIC BLOODPRESSURE STA DING TO SUPINEF
df_orth_blood$SYSORHY = df_orth_blood$SYSSTND - df_orth_blood$SYSSUP
df_orth_blood$DIAORHY = df_orth_blood$DIASTND - df_orth_blood$DIASUP

# ORTHOSTATIC BLOODPRESSURE
df = ARRANGE(df_orth_blood, PATNO, INFODT) %>% GROUP_BY(PATNO) %>%
   SELECT(PATNO, INFODT, EVENT_ID, APPRDX, DIAORHY,SYSORHY)

# SUBSETTING DF INTO PD, CONTROLS, AND PRODROMAL GROUP
df.orth = SUBSET(df, APPRDX == "1" | APPRDX == "2", SELECT = C(PATNO, EVENT_ID, APPRDX, SYSORHY, DIAORHY))
df.orth = SUBSET(df.orth, EVENT_ID != "SC" & EVENT_ID != "ST" & EVENT_ID != "PW" & EVENT_ID != "RS1" &
      EVENT_ID != "U01" & EVENT_ID != "U02")

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df.orth[!DUPLICATED(df.orth$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                        DATA.FRAME(EVENT_ID = all.visits,
                              PATNO = all.patnos.apprdx$PATNO[s],
                              APPRDX =  all.patnos.apprdx$APPRDX[s],
                              X = 1:16)
                        )
}
df = MERGE(X = df.complete.visits, Y = df.orth, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]

## FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$SYSORHY[1:4])) >= 1,
        SUM(!IS.NA(sbjct$SYSORHY[5:8])) >= 1,
        SUM(!IS.NA(sbjct$SYSORHY[9:12])) >= 1,
        SUM(!IS.NA(sbjct$SYSORHY[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}

}

## IMPUTATION DISTRIBUTION
FOR (r IN 1:4){
IF (r == 1){
    na.distr = DATA.FRAME(T(SUBSET(df, relevance == r, SELECT = C(PATNO, EVENT_ID, SYSORHY))[1:16,]))
    NAMES(na.distr) = all.visits}
ELSE {

temp = DATA.FRAME(T(SUBSET(df, relevance == r, SELECT = C(PATNO, EVENT_ID, SYSORHY))[1:16,]))
NAMES(temp) = all.visits
```

```
na.distr = RBIND(na.distr, temp)}
}

na.distr = SUBSET(na.distr, BL!="BL")
rnames = C("3007", "3006", "3009", "3000")
na.distr = na.distr[!GREPL("PA", ROW.NAMES(na.distr)), ]
ROW.NAMES(na.distr) = rnames

AS_IMAGE(
    KABLE(na.distr)   %>%
    KABLE_STYLING(FULL_WIDTH = F), FILE = 'VAR_1_ORTHO_NA_DISTRIBUTION.JPG', WIDTH= 12
    )



## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB('') +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                        "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR1_1.P.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    ysys = AS.NUMERIC(df[df$PATNO == subject, ]$SYSORHY)
    ydia = AS.NUMERIC(df[df$PATNO == subject, ]$DIAORHY)
    x = 1:16

    data = DATA.FRAME(CBIND(ysys, ydia, x))
    IF (SUM(IS.NA(ysys)) > 0)
        {
    imp = REGRESSIONIMP(ysys + ydia ~ x,DATA=data)
    df[df$PATNO == subject, ]$SYSORHY = ROUND(imp$ysys, DIGITS = 0)
    df[df$PATNO == subject, ]$DIAORHY = ROUND(imp$ydia,  DIGITS = 0)
    }

}

df.i.orth = AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "SYSORHY", "DIAORHY")))

# SUMMARIZE BY EVENT ID, SUBJECT GROUP
orth_sys = df.orth %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
   MEAN = MEAN(SYSORHY),
   STD = SD(SYSORHY),
   YMIN = mean - std,
   YMAX = mean + std,
   N = LENGTH(APPRDX))

orth_dia = df.orth %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
   MEAN = MEAN(DIAORHY),
   STD = SD(DIAORHY),
   YMIN = mean - std,
   YMAX = mean + std,
   N = LENGTH(APPRDX))


# IMPUTATED BY EVENT ID, SUBJECT GROUP
orth_i_sys = df.i.orth %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
   MEAN = MEAN(SYSORHY),
   STD = SD(SYSORHY),
   YMIN = mean - std,
```

```
  YMAX = mean + std,
  N = LENGTH(APPRDX))

orth_i_dia = df.i.orth %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
  MEAN = MEAN(DIAORHY),
  STD = SD(DIAORHY),
  YMIN = mean - std,
  YMAX = mean + std,
  N = LENGTH(APPRDX))
```

Histogram and line graph: mean of the orthostatic blood pressure

```
lqnd.xpos = 0.3
lgnd.ypos = 0.83

dodge = 0.2
txtsize = 12

p.line.orth = FUNCTION(
 # DATA, X, Y, .. VALUES
 DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
 COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
 SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

 ## AXIS, LEGEND, LABELS
 XTEXT = "VISITS, DELTA = 3 MNT",
 YTEXT = "TEXT",
 XTITLE = TRUE,
 NM = "XX PER VISIT",
 LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
 LEGEND_POS = C(0.9, 0.9))
{

GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
     GEOM_POINT() + GEOM_LINE() +
     GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
          ALPHA = 0.8, SEGMENT.SIZE = 0) +
     GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
          SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
     GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
          POSITION=POSITION_DODGE(dodge)) +
     THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
         AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


  # LEGEND, AXIX
  LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
     ELSE THEME(LEGEND.POSITION=legend_pos, )) +

  SCALE_COLOR_DISCRETE(LABELS = lbls) +
  XLAB(xtext) +
  YLAB(ytext)
}

p.hist = FUNCTION(
  DATA = df_HVLT_summary, X = "MEAN.TOTAL.RECALL", FILL = "FACTOR(APPRDX)",
  BNS = 50,  LPH = 0.5, XLIMIT = FALSE,
  NM = "MEAN SCORES OF VISITS",
  LBLS = C("PD","HC"),
  XTEXT = "SCORES",
  YTEXT = "COUNTS",
  LGND = TRUE,
  LEGEND_POS = C(0.8, 0.8),
  MN = NULL,
  MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =txtsize-2),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=12)) +
```

```
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(NAME = nm, LABELS = lbls) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos)
        ELSE THEME(LEGEND.POSITION='NONE'))
}

p.orth.sys.line = P.LINE.ORTH(DATA = orth_sys, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX", YTEXT = "ORTH. SYSTOLIC", XTEXT = "",
                 LEGEND_POS = "NONE")

p.orth.dia.line = P.LINE.ORTH(DATA = orth_dia, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX", YTEXT = "ORTH. DIASTOLIC", XTEXT = "",
                 LEGEND_POS = "NONE")

p.sysorhy.hist = P.HIST(DATA = df.orth, X = "SYSORHY", XTEXT = "", YTEXT = "", LGND= TRUE,
                NM = "SYSTOLIC ORTHOSTATIS BP", LEGEND_POS = C(lgnd.xpos, lgnd.ypos))
p.diaorhy.hist = P.HIST(DATA = df.orth, X = "DIAORHY", XTEXT = "", YTEXT = "", LGND= TRUE,
                NM = "DIASTOLIC ORTHOSTATIS BP", LEGEND_POS = C(lgnd.xpos, lgnd.ypos))


# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(3,1,1,1), C(4,2,2,2))
p.grid = GRID.ARRANGE(p.orth.sys.line, p.orth.dia.line,
        p.sysorhy.hist, p.diaorhy.hist,
        LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR1_ORTHO_STATIC.JPG"),
    DPI = 300,
    HEIGHT = 4,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.grid)
```

Plot Imputed Data

```
p.orth.sys.line = P.LINE.ORTH(DATA = orth_i_sys, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
                 YTEXT = "ORTH. SYSTOLIC", XTEXT = "", LEGEND_POS = "NONE")

p.orth.dia.line = P.LINE.ORTH(DATA = orth_i_dia, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
                 YTEXT = "ORTH. DIASTOLIC", XTEXT = "",LEGEND_POS = "NONE")

p.sysorhy.hist = P.HIST(DATA = df.orth, X = "SYSORHY", XTEXT = "", YTEXT = "", LGND= TRUE,
                NM = "IMPUTED SYSTOLIC ORTHOSTATIS BP", LEGEND_POS = C(lgnd.xpos, lgnd.ypos))
p.diaorhy.hist = P.HIST(DATA = df.orth, X = "DIAORHY", XTEXT = "", YTEXT = "", LGND= TRUE,
                NM = "IMPUTED DIASTOLIC ORTHOSTATIS BP", LEGEND_POS = C(lgnd.xpos, lgnd.ypos))


# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(3,1,1,1), C(4,2,2,2))
p.i.grid = GRID.ARRANGE(p.orth.sys.line, p.orth.dia.line,
        p.sysorhy.hist, p.diaorhy.hist,
        LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images, "VAR1_ORTHO_I_STATIC.JPG"),
    DPI = 300,
    HEIGHT = 4,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.i.grid)
```

Sleep Behavior Disorder - Rapid Eye Movement (REM)

Calculating Score:

1. Add 1 point for each response of YES for DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV, SLPINJUR, DRMVERBL, #DRMFIGHT, DRMUMV, DRMOBJFL, MVAWAKEN, DRMREMEM, SLPDSTRB
2. Add 1 point if any of the following variables has a response of "Yes" (1): STROKE, HETRA, PARKISM, RLS, NARCLPSY, DEPRS, EPILEPSY, BRNINFM, CNSOTH
3. If any of the previous variables are missing, then RBD score is missing.

Subjects with score ≥ 5 are RBD Positive
Subjects with score < 5 are RBD Negative

```
# GET ENROLLED SUBJECTS (PD, CONTROLS, PRODROMAL)
df_enrolled_all = GET_ENROLLED(C(1,2))

# GET REM DATA
path = PASTE0(rpath, "01_DATA/DATA_LONI/09_NON-MOTOR_ASSESSMENTS/")
file = "REM_SLEEP_DISORDER_QUESTIONNAIRE.CSV"
df_REM = READ.CSV(PASTE0(path, file))
df_REM = SUBSET(df_REM, SELECT = C(PATNO, EVENT_ID, DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV,
                SLPINJUR, DRMVERBL,DRMFIGHT, DRMUMV, DRMOBJFL,MVAWAKEN, DRMREMEM,
                SLPDSTRB, STROKE, HETRA, PARKISM, RLS, NARCLPSY, DEPRS, EPILEPSY,
                BRNINFM, CNSOTH))

# EXTRACT COMPLETE CASES
df_REM = df_REM[COMPLETE.CASES(df_REM), ]

# MERGE ENROLLED SUJECTS WITH REM DATA
df_REM = MERGE(df_enrolled_all, df_REM, BY = 'PATNO')
df_REM = df_REM[ORDER(df_REM$EVENT_ID, df_REM$PATNO), ]

# CALCULATE SCORES ACCORDING TO 1, AND 2
  # 1. ADD 1 POINT FOR EACH RESPONSE OF YES  FOR DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV, SLPINJUR, DRMVERBL,
  # DRMFIGHT, DRMUMV, DRMOBJFL, MVAWAKEN, DRMREMEM, SLPDSTRB
sum1 = ROWSUMS(SUBSET(df_REM, SELECT = C(DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV, SLPINJUR, DRMVERBL,DRMFIGHT,
                DRMUMV, DRMOBJFL,MVAWAKEN, DRMREMEM,SLPDSTRB)))
  # 2. ADD 1 POINT IF ANY OF THE FOLLOWING VARIABLES HAS A RESPONSE OF "YES" (1): STROKE, HETRA, PARKISM,
  # RLS, NARCLPSY, DEPRS, EPILEPSY, BRNINFM, CNSOTH
sum2 = IFELSE(ROWSUMS(SUBSET(df_REM, SELECT = C(STROKE, HETRA, PARKISM, RLS, NARCLPSY, DEPRS, EPILEPSY,
                BRNINFM, CNSOTH  ))) > 0, 1, 0)
df_REM$SCORE = sum1 + sum2
df_REM = SUBSET(df_REM, SELECT = C(PATNO, APPRDX, EVENT_ID, SCORE))

df_REM = df_REM[ORDER(df_REM$PATNO, df_REM$EVENT_ID), ]


# SUMMARIZING DATA
df_REM_summary = df_REM %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(SCORE, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(SCORE, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)

df_REM_summary = SUBSET(df_REM_summary, EVENT_ID != "SC" & EVENT_ID != "ST" & EVENT_ID != "PW" &
                EVENT_ID != "RS1" & EVENT_ID != "U01" & EVENT_ID != "U02", n > 1)
```

Imputing REM

```
#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df_REM = SUBSET(df_REM, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df_REM[!DUPLICATED(df_REM$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                DATA.FRAME(EVENT_ID = all.visits,
                        PATNO = all.patnos.apprdx$PATNO[s],
                        APPRDX =  all.patnos.apprdx$APPRDX[s],
                        X = 1:16)
                )
}
df = MERGE(X = df.complete.visits, Y = df_REM, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$SCORE[1:4])) >= 1,
```

```
        SUM(!IS.NA(sbjct$SCORE[5:8])) >= 1,
        SUM(!IS.NA(sbjct$SCORE[9:12])) >= 1,
        SUM(!IS.NA(sbjct$SCORE[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}

}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB('') +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                     "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR2_REM.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$SCORE)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$SCORE = ROUND(imp$score, DIGITS = 0)
}

df.i.rem = AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "SCORE")))

# SUMMARIZING DATA
df_REM_i_summary = df.i.rem %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(SCORE, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(SCORE, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)

# FUNCTION
p.line.rem = FUNCTION(
     # DATA, X, Y, .. VALUES
     DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
     COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
     SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

     ## AXIS, LEGEND, LABELS
     XTEXT = "VISITS, DELTA = 3 MNT",
     YTEXT = "TEXT",
     XTITLE = TRUE,
     NM = "XX PER VISIT",
     LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
     LEGEND_POS = C(0.8, 0.8))
    {

    GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
    GEOM_POINT() + GEOM_LINE() +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
       ALPHA = 0.8, SEGMENT.SIZE = 0) +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
       SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
    GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
       POSITION=POSITION_DODGE(dodge)) +
```

```
    THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
     AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


    # LEGEND, AXIX
    LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
      ELSE THEME(LEGEND.POSITION=legend_pos, )) +

    SCALE_COLOR_DISCRETE(LABELS = lbls) +
    XLAB(xtext) +
    YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df_HVLT_summary, X = "MEAN.TOTAL.RECALL", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lgnd.xpos = 0.7; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.rem.line  = P.LINE.REM(DATA = df_REM_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "RBD SCORE",
    XTEXT = "", LEGEND_POS = "NONE")

p.rem.hist = P.HIST(DATA = df_REM, X = "SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("SLEEP BEHAVIOR DISORDER (RBD)",18),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.rem.grid = GRID.ARRANGE(p.rem.hist, p.rem.line,
    LAYOUT_MATRIX = lay)



GGSAVE(PASTE0(p_images, "VAR2_REM_SCORE.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.rem.grid)

# FUNCTION
p.line.rem = FUNCTION(
      # DATA, X, Y, .. VALUES
      DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
      COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
```

```
    SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

    ## AXIS, LEGEND, LABELS
    XTEXT = "VISITS, DELTA = 3 MNT",
    YTEXT = "TEXT",
    XTITLE = TRUE,
    NM = "XX PER VISIT",
    LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
    LEGEND_POS = C(0.8, 0.8))
   {

   GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
   GEOM_POINT() + GEOM_LINE() +
   GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
     ALPHA = 0.8, SEGMENT.SIZE = 0) +
   GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
     SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
   GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
     POSITION=POSITION_DODGE(dodge)) +
   THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
    AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


   # LEGEND, AXIX
   LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
     ELSE THEME(LEGEND.POSITION=legend_pos, )) +

   SCALE_COLOR_DISCRETE(LABELS = lbls) +
   XLAB(xtext) +
   YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df_HVLT_summary, X = "MEAN.TOTAL.RECALL", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lqnd.xpos = 0.7; lqnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.rem.line  = P.LINE.REM(DATA = df_REM_i_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "RBD SCORE",
    XTEXT = "", LEGEND_POS = "NONE")

p.rem.hist = P.HIST(DATA = df.i.rem, X = "SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("SLEEP BEHAVIOR DISORDER (RBD)",18),
    LEGEND_POS = C(lqnd.xpos, lqnd.ypos))
```

```
# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.rem.grid = GRID.ARRANGE(p.rem.hist, p.rem.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images, "VAR2_REM_I_SCORE.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.rem.grid)
```

Output Data

- df_REM
- df_i_REM
- df_REM_i_summary
- df_REM_summary

Cognitive Impairment Assessment(MoCA)

Data from MOCA, SOCIOECO files.

Unadjusted Score = sum of MCAALTTM, MCACUBE, MCACLCKC, MCACLCKN, MCACLCKH, MCALION, MCARHINO, MCACAMEL, MCAFDS, MCABDS, MCAVIGIL, MCASER7, MCASNTNC, MCAVF, MCAABSTR, MCAREC1, MCAREC2, MCAREC3, MCAREC4, MCAREC5, MCADATE, MCAMONTH, MCAYR,MCADAY, MCAPLACE, MCACITY

If EDUCYRS ≤ 12 and Unadjusted Score < 30, add 1 more point to score. If EDUCYRS > 12, do not add any more points to score.

```
# DATA INGESTION
## MOCA
path = PASTE0(rpath, "01_DATA/DATA_LONI/09_NON-MOTOR_ASSESSMENTS/")
file = "MONTREAL_COGNITIVE_ASSESSMENT__MOCA_.CSV"
df_moca= READ.CSV(PASTE0(path, file))
## SOCIO ECONOMIC "EDUCYRS"
path = PASTE0(rpath, "01_DATA/DATA_LONI/02_SUBJECT_CHARACTERISTICS/")
file = "SOCIO-ECONOMICS.CSV"
df_soc= READ.CSV(PASTE0(path, file))
## ENROLLMENT DATA
df_enrolled_all =  GET_ENROLLED(C(1,2))

# MERGING MOCA WITH ENROLLED
df_moca = MERGE(df_moca, df_enrolled_all, BY = "PATNO")

# UNADJUSTED SCORE
slct = C('MCAALTTM', "MCACUBE", "MCACLCKC", "MCACLCKN", "MCACLCKH", "MCALION", "MCARHINO", "MCACAMEL", "MCAFDS", "MC
ABDS", "MCAVIGIL",
"MCASER7", "MCASNTNC", "MCAVF", "MCAABSTR", "MCAREC1", "MCAREC2", "MCAREC3", "MCAREC4", "MCAREC5", "MCADATE", "MCAM
ONTH",
"MCAYR","MCADAY", "MCAPLACE", "MCACITY")

df_sum = SUBSET(df_moca, SELECT = slct)
df_sum$sum = ROWSUMS(df_sum)
df_moca$SUM = df_sum$sum
df_moca = SUBSET(df_moca, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "SUM"))

# MERGING SOCIOECONOMIC WITH DF_MOCA UNADJUSTED SCORE
df_soc = SUBSET(df_soc, SELECT = C("PATNO", "EDUCYRS"))
df_moc_soc = MERGE(df_moca, df_soc, BY = "PATNO", ALL.X = TRUE)

# ADJUST SCORE
## IF EDUCYRS ≤ 12 AND UNADJUSTED SCORE < 30, ADD 1 MORE POINT TO SCORE.
## IF EDUCYRS > 12, DO NOT ADD ANY MORE POINTS TO SCORE.
df_moc_soc$adj = IFELSE((df_moc_soc$EDUCYRS <= 12 & df_moc_soc$SUM < 30), 1, 0)
df_moc_soc$adjSUM = df_moc_soc$SUM + df_moc_soc$adj
```

```
# SUBSETTING BL, V01 ... V02, DELETE UNNECESSARY VARIABLES
df_moc = SUBSET(df_moc_soc, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df_moc[df_moc$EVENT_ID == "SC", ]$EVENT_ID = "BL"
df_moc = df_moc[COMPLETE.CASES(df_moc), ]

# # SUMMARIZING DATA
df_moc_summary = df_moc %>% GROUP_BY(APPRDX, EVENT_ID) %>%
  SUMMARISE(
        N = LENGTH(APPRDX),
        MEAN = MEAN(adjSUM),
        STD = SD(adjSUM),
        YMIN = mean - std,
        YMAX = mean + std)
df_moc_summary = SUBSET(df_moc_summary, n>1)

#DF_MOC_SUMMARY[, C(4,5)] = DF_MOC_SUMMARY[, C(4,5)] *-1
#DF_MOC_SUMMARY[, C(4,5)] = MIN.MAX.SCALING(DF_MOC_SUMMARY[, C(4,5)])

# FUNCTION
p.line.moca = FUNCTION(
      # DATA, X, Y, .. VALUES
      DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
      COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
      SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

      ## AXIS, LEGEND, LABELS
      XTEXT = "VISITS, DELTA = 3 MNT",
      YTEXT = "TEXT",
      XTITLE = TRUE,
      NM = "XX PER VISIT",
      LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
      LEGEND_POS = C(0.9, 0.9))
    {

    GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
    GEOM_POINT() + GEOM_LINE() +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
       ALPHA = 0.8, SEGMENT.SIZE = 0) +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
       SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
    GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
       POSITION=POSITION_DODGE(dodge)) +
    THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
     AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


    # LEGEND, AXIX
    LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
      ELSE THEME(LEGEND.POSITION=legend_pos, )) +

    SCALE_COLOR_DISCRETE(LABELS = lbls) +
    XLAB(xtext) +
    YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df_HVLT_summary, X = "MEAN.TOTAL.RECALL", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
```

```
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lgnd.xpos = 0.3; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.moca.line  = P.LINE.MOCA(DATA = df_moc_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "MCI SCORE",
    XTEXT = "", LEGEND_POS = "NONE")

p.moca.hist = P.HIST(DATA = df_moc, X = "ADJSUM", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("MONTREAL COGNITIVE ASSESSMENT (MOCA)",22),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.moca.grid = GRID.ARRANGE(p.moca.hist, p.moca.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR3_MOCA.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.moca.grid)

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df_moc = SUBSET(df_moc, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df_moc[!DUPLICATED(df_moc$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                        DATA.FRAME(EVENT_ID = all.visits,
                                PATNO = all.patnos.apprdx$PATNO[s],
                                APPRDX =  all.patnos.apprdx$APPRDX[s],
                                X = 1:16)
                        )
}
df = MERGE(X = df.complete.visits, Y = df_moc, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$adjSUM[1:4])) >= 1,
        SUM(!IS.NA(sbjct$adjSUM[5:8])) >= 1,
        SUM(!IS.NA(sbjct$adjSUM[9:12])) >= 1,
        SUM(!IS.NA(sbjct$adjSUM[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}
}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
```

```
p.na.i.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB('') +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                     "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR3_MOCA.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.i.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$adjSUM)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$adjSUM = ROUND(imp$score, DIGITS = 0)
}

df_i_moc = AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "ADJSUM")))

# SUMMARIZING DATA
df_i_moc_summary = df_i_moc %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(adjSUM, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(adjSUM, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)
```

Variables

```
# PLOTS
lqnd.xpos = 0.3; lqnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.moca.line  = P.LINE.MOCA(DATA = df_i_moc_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "MCI SCORE",
    XTEXT = "", LEGEND_POS = "NONE")

p.moca.hist = P.HIST(DATA = df_i_moc, X = "ADJSUM", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("MONTREAL COGNITIVE ASSESSMENT (MOCA)",22),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.moca.grid = GRID.ARRANGE(p.moca.hist, p.moca.line,
    LAYOUT_MATRIX = lay)



GGSAVE(PASTE0(p_images,"VAR3_I_MOCA.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.moca.grid)
```

UPDRS Part II

```
# DATA INGESTION
path = PASTE0(rpath, "01_DATA/DATA_LONI/08_MOTOR_MDS_UPDRS/")
var = C("NP2SPCH", "NP2SALV", "NP2SWAL", "NP2EAT", "NP2DRES", "NP2HYGN", "NP2HWRT", "NP2HOBB", "NP2TURN", "NP2TRMR", "NP2R
ISE", "NP2WALK", "NP2FREZ")
id = C("PATNO", "EVENT_ID")

## UPDRS PART II
```

```
# PART I SCORE = SUM OF THESE 13 VARIABLES
file = "MDS_UPDRS_PART_II__PATIENT_QUESTIONNAIRE.CSV"
df1 = READ.CSV(PASTE0(path, file))
df1 = df1[, NAMES(df1) %IN% C(id, var)]

# SUM UP VARIABLES
df_sum = SUBSET(df1, SELECT = var)
df1$UPDRS.2 = ROWSUMS(df_sum)

# MERGE WITH ENROLLED
df_enrolled_all = GET_ENROLLED(C(1,2))
df1 = MERGE(df1, df_enrolled_all, BY = "PATNO")
df1 = df1[COMPLETE.CASES(df1), ]

# UPDRS 2
df.updrs.2 = SUBSET(df1, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01" & EVENT_ID != "SC")
df.updrs.2 = df.updrs.2[, C(id, "APPRDX",  "UPDRS.2")]

# # SUMMARIZING DATA
df_updrs.2.summary = df.updrs.2 %>% GROUP_BY(APPRDX, EVENT_ID) %>%
   SUMMARISE(
         N = LENGTH(APPRDX),
         MEAN = MEAN(UPDRS.2),
         SDT = SD(UPDRS.2),
         YMIN = mean - sdt,
         YMAX = mean + sdt)
df_updrs.2.summary = SUBSET(df_updrs.2.summary, n>2)

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df.updrs.2 = SUBSET(df.updrs.2, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df.updrs.2[!DUPLICATED(df.updrs.2$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
     df.complete.visits = RBIND(df.complete.visits,
                          DATA.FRAME(EVENT_ID = all.visits,
                                PATNO = all.patnos.apprdx$PATNO[s],
                                APPRDX =  all.patnos.apprdx$APPRDX[s],
                                X = 1:16)
                          )
}
df = MERGE(X = df.complete.visits, Y = df.updrs.2, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
     sbjct = SUBSET(df, PATNO == subject)
     missing.na.per.quantile  =  SUM(
         SUM(!IS.NA(sbjct$UPDRS.2[1:4])) >= 1,
         SUM(!IS.NA(sbjct$UPDRS.2[5:8])) >= 1,
         SUM(!IS.NA(sbjct$UPDRS.2[9:12])) >= 1,
         SUM(!IS.NA(sbjct$UPDRS.2[13:16])) >= 1)
     IF (missing.na.per.quantile >= 1){
         df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
         ELSE{
         df[df$PATNO == subject,]$relevance = 0}
}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.i.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
     GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
     GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
     XLAB('') +
     YLAB("SUBJECTS") +
     SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                          "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR4_UPDRS2.NA.DISTRIBUTION.JPG"),
     DPI = 300,
```

```
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.i.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$UPDRS.2)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$UPDRS.2 = ROUND(imp$score, DIGITS = 0)
}

## NO MISSINGS IN SCORE.
## NO MISSINGS IN SCORE.
## NO MISSINGS IN SCORE.

df_i_updrs2= AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "UPDRS.2")))

# SUMMARIZING DATA
df_i_updrs2_summary = df_i_updrs2 %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(UPDRS.2, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(UPDRS.2, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)
```

Variables

```
# FUNCTION
p.line.updrs.part.ii = FUNCTION(
      # DATA, X, Y, .. VALUES
      DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
      COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
      SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

      ## AXIS, LEGEND, LABELS
      XTEXT = "VISITS, DELTA = 3 MNT",
      YTEXT = "TEXT",
      XTITLE = TRUE,
      NM = "XX PER VISIT",
      LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
      LEGEND_POS = C(0.9, 0.9))
    {

    GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
    GEOM_POINT() + GEOM_LINE() +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
       ALPHA = 0.8, SEGMENT.SIZE = 0) +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
       SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
    GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
       POSITION=POSITION_DODGE(dodge)) +
    THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
      AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


    # LEGEND, AXIX
    LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
      ELSE THEME(LEGEND.POSITION=legend_pos, )) +

    SCALE_COLOR_DISCRETE(LABELS = lbls) +
    XLAB(xtext) +
    YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df.updrs.2, X = "UPDRS2", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
```

```
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.updrs2.line  = P.LINE.UPDRS.PART.II(DATA = df_updrs.2.summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "UPDRS PART II",
    XTEXT = "", LEGEND_POS = "NONE")

p.updrs2.hist = P.HIST(DATA = df.updrs.2, X = "UPDRS.2", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("UNIFIED PARKINSON'S DISEASE RATING SCALE (UPDRS PART II)",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.updrs2.grid = GRID.ARRANGE(p.updrs2.hist, p.updrs2.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR4_UPDRS2.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.updrs2.grid)

# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.updrs2.line  = P.LINE.UPDRS.PART.II(DATA = df_i_updrs2_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "UPDRS PART II IMPUTED",
    XTEXT = "", LEGEND_POS = "NONE")

p.updrs2.hist = P.HIST(DATA = df_i_updrs2, X = "UPDRS.2", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("UNIFIED PARKINSON'S DISEASE RATING SCALE (UPDRS PART II)",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.updrs2.grid = GRID.ARRANGE(p.updrs2.hist, p.updrs2.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR4_I_UPDRS2.JPG"),
    DPI = 300,
```

```
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.updrs2.grid)
```

UPDRS Part III

sum of:
NP3SPCH, NP3FACXP, NP3RIGN, NP3RIGRU, NP3RIGLU, PN3RIGRL, NP3RIGLL, NP3FTAPR, NP3FTAPL, NP3HMOVR, NP3HMOVL, NP3PRSPR, NP3PRSPL, NP3TTAPR, NP3TTAPL, NP3LGAGR, NP3LGAGL, NP3RISNG, NP3GAIT, NP3FRZGT, NP3PSTBL, NP3POSTR, NP3BRADY, NP3PTRMR, NP3PTRML, NP3KTRMR, NP3KTRML, NP3RTARU, NP3RTALU, NP3RTARL, NP3RTALL, NP3RTALJ, NP3RTCON

NUPDRS3A is variables “in clinic post-dose levadopa” - delted from this data set

```
# DATA INGESTION
path = PASTE0(rpath, "01_DATA/DATA_LONI/08_MOTOR_MDS_UPDRS/")
var = C("NP3SPCH", "NP3FACXP", "NP3RIGN", "NP3RIGRU", "NP3RIGLU", "PN3RIGRL", "NP3RIGLL", "NP3FTAPR", "NP3FTAPL", "NP3HMOVR", "NP3HMOVL", "NP3PRSPR", "NP3PRSPL", "NP3TTAPR", "NP3TTAPL", "NP3LGAGR", "NP3LGAGL", "NP3RISNG", "NP3GAIT", "NP3FRZGT", "NP3PSTBL", "NP3POSTR", "NP3BRADY", "NP3PTRMR", "NP3PTRML", "NP3KTRMR", "NP3KTRML", "NP3RTARU", "NP3RTALU", "NP3RTARL", "NP3RTALL", "NP3RTALJ", "NP3RTCON" )
id = C("PATNO", "EVENT_ID")

## UPDRS PART III
# PART I SCORE = SUM OF THESE 13 VARIABLES
file = "MDS_UPDRS_PART_III.CSV"
df1 = READ.CSV(PASTE0(path, file))
df1 = df1[df1$PAG_NAME != "NUPDRS3A", ]
df1 = df1[, NAMES(df1) %IN% C(id, var)]



# SUM UP VARIABLES
df_sum = SUBSET(df1, SELECT = var)
df1$UPDRS.3 = ROWSUMS(df_sum)

# MERGE WITH ENROLLED
df_enrolled_all = GET_ENROLLED(C(1,2))
df1 = MERGE(df1, df_enrolled_all, BY = "PATNO")
df1 = df1[COMPLETE.CASES(df1), ]

# UPDRS 2
df.updrs.3 = SUBSET(df1, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01" & EVENT_ID != "SC")
df.updrs.3 = df.updrs.3[, C(id, "APPRDX",  "UPDRS.3")]

# # SUMMARIZING DATA
df_updrs.3.summary = df.updrs.3 %>% GROUP_BY(APPRDX, EVENT_ID) %>%
  SUMMARISE(
      N = LENGTH(APPRDX),
      MEAN = MEAN(UPDRS.3),
      SDT = SD(UPDRS.3),
      YMIN = mean - sdt,
      YMAX = mean + sdt)
df_updrs.3.summary = SUBSET(df_updrs.3.summary, n>2)

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df.updrs.3 = SUBSET(df.updrs.3, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df.updrs.3[!DUPLICATED(df.updrs.3$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                        DATA.FRAME(EVENT_ID = all.visits,
                              PATNO = all.patnos.apprdx$PATNO[s],
                              APPRDX =  all.patnos.apprdx$APPRDX[s],
                              X = 1:16)
                        )
}
df = MERGE(X = df.complete.visits, Y = df.updrs.3, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
```

```
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$UPDRS.3[1:4])) >= 1,
        SUM(!IS.NA(sbjct$UPDRS.3[5:8])) >= 1,
        SUM(!IS.NA(sbjct$UPDRS.3[9:12])) >= 1,
        SUM(!IS.NA(sbjct$UPDRS.3[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}
}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.i.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB('') +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                     "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR5_UPDRS3.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.i.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$UPDRS.3)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$UPDRS.3 = ROUND(imp$score, DIGITS = 0)
}

## NO MISSINGS IN SCORE.
## NO MISSINGS IN SCORE.

df_i_updrs3= AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "UPDRS.3")))

# SUMMARIZING DATA
df_i_updrs3_summary = df_i_updrs3 %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(UPDRS.3, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(UPDRS.3, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)

# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.updrs3.line  = P.LINE.UPDRS.PART.II(DATA = df_updrs.3.summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "UPDRS PART III",
    XTEXT = "", LEGEND_POS = "NONE")

p.updrs3.hist = P.HIST(DATA = df.updrs.3, X = "UPDRS.3", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("UNIFIED PARKINSON'S DISEASE RATING SCALE (UPDRS PART III)",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
```

```
p.updrs3.grid = GRID.ARRANGE(p.updrs3.hist, p.updrs3.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR5_UPDRS3.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.updrs3.grid)

# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.updrs3.line  = P.LINE.UPDRS.PART.II(DATA = df_i_updrs3_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "UPDRS PART III IMPUTED",
    XTEXT = "", LEGEND_POS = "NONE")

p.updrs3.hist = P.HIST(DATA = df_i_updrs3, X = "UPDRS.3", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("UNIFIED PARKINSON'S DISEASE RATING SCALE (UPDRS PART III)",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.updrs3.grid = GRID.ARRANGE(p.updrs3.hist, p.updrs3.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR5_I_UPDRS3.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.updrs3.grid)
```

Depression

Depression: Geriatric Depression Scale (GDS)

Add 1 point for each response of “No” (0) to any of the following variables: GDSSATIS, GDSGSPIR, GDSHAPPY,GDSALIVE, GDSENRGY

Add 1 point for each response of “Yes” (1) to any of the following variables: GDSDROPD, GDSEMPTY, GDSBORED,GDSAFRAD, GDSHLPLS, GDSHOME,GDSMEMRY, GDSWRTLS, GDSHOPLS, GDSBETER Subjects with GDS ≥ 5 are “Depressed” Subject with GDS < 5 are “Not Depressed”

Histogram and line graph

```
# FUNCTION
p.line.depresson = FUNCTION(
     # DATA, X, Y, .. VALUES
     DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
     COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
     SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

     ## AXIS, LEGEND, LABELS
     XTEXT = "VISITS, DELTA = 3 MNT",
     YTEXT = "TEXT",
     XTITLE = TRUE,
     NM = "XX PER VISIT",
     LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
     LEGEND_POS = C(0.9, 0.9))
    {

    GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
    GEOM_POINT() + GEOM_LINE() +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
       ALPHA = 0.8, SEGMENT.SIZE = 0) +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
       SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
    GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
```

```
        POSITION=POSITION_DODGE(dodge)) +
    THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
     AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


    # LEGEND, AXIX
    LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
      ELSE THEME(LEGEND.POSITION=legend_pos, )) +

    SCALE_COLOR_DISCRETE(LABELS = lbls) +
    XLAB(xtext) +
    YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df.updrs.2, X = "UPDRS2", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.depression.line  = P.LINE.DEPRESSON(DATA = df_dep_qds_summaried, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "GERIATRIC DEPRESSION SCALE (GDS) ",
    XTEXT = "", LEGEND_POS = "NONE")

p.depression.hist = P.HIST(DATA = df_dep_gds, X = "GDS_RAW_SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("GERIATRIC DEPRESSION SCALE (GDS))",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.depression.grid = GRID.ARRANGE(p.depression.hist, p.depression.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR6_DEPRESSION.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.depression.grid)

# PLOTS
lgnd.xpos = 0.65; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12
```

```
p.depression.line  = P.LINE.DEPRESSON(DATA = df_i_GDS_RAW_SCORE_summary, Y = "MEAN", YMIN = "YMIN",
                                      YMAX = "YMAX", YTEXT = "GERIATRIC DEPRESSION SCALE (GDS) IMPUTED ",
    XTEXT = "", LEGEND_POS = "NONE")

p.depression.hist = P.HIST(DATA = df_i_GDS_RAW_SCORE, X = "GDS_RAW_SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("GERIATRIC DEPRESSION SCALE (GDS))",30),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.depression.grid = GRID.ARRANGE(p.depression.hist, p.depression.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR6_I_DEPRESSION.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.depression.grid)
```

Anxiety

STAIAD1 - STAIAD40 Add values for the following questions: 3, 4, 6, 7, 9, 12, 13, 14, 17, 18, 22, 24, 25, 28, 29, 31, 32, 35, 37, 38, 40 Use reverse scoring for the remaining questions and add to the first score (e.g., if value = 1, add 4 points to score; if value = 2, add 3 points to score, etc).

```
# GET ENROLLED SUBJECTS (PD, CONTROLS, PRODROMAL)
df_enrolled_all = GET_ENROLLED(C(1,2))

# GET ANXIETY DATA
path = PASTE0(rpath, "01_DATA/DATA_LONI/09_NON-MOTOR_ASSESSMENTS/")
file = "STATE-TRAIT_ANXIETY_INVENTORY.CSV"
df_ANX = READ.CSV(PASTE0(path, file))

# MERGE ENROLLED SUJECTS WITH ANXIETY DATA
df_ANX = MERGE(df_enrolled_all, df_ANX, BY = 'PATNO')
df_ANX = df_ANX[ORDER(df_ANX$EVENT_ID, df_ANX$PATNO), ]

# PREPARING THE VARIABLES TO CALCULATE THE SCORE
var = 1:40
var_add = C(3, 4, 6, 7, 9, 12, 13, 14, 17, 18, 22, 24, 25, 28, 29, 31, 32, 35, 37, 38, 40)
var_rev = var[-var_add]

var_add = SAPPLY(var_add, FUNCTION(x) PASTE0("STAIAD",x))
var_rev = SAPPLY(var_rev, FUNCTION(x) PASTE0("STAIAD",x))

# SUM 1
sum1 = ROWSUMS(SUBSET(df_ANX, SELECT = var_add))

# SUM 2 REVERSRE
a = C(1,2,3,4)
v_rev = SUBSET(df_ANX, SELECT = var_rev)
v = v_rev
FOR (i IN a){
   IF (i == 1) v_rev[v_rev == i] = 4
   IF (i == 2) v_rev[v_rev == i] = 3
   IF (i == 3) v_rev[v_rev == i] = 2
   IF (i == 4) v_rev[v_rev == i] = 1
}
sum2 = ROWSUMS(v_rev)
# SCORE = SUM1 + SUM2
df_ANX$SCORE = sum1 + sum2
df_ANX = SUBSET(df_ANX, SELECT = C(PATNO, EVENT_ID, APPRDX, SCORE))

# SUMMARIZING DATA
df_ANX_summary = df_ANX %>% GROUP_BY(APPRDX, EVENT_ID) %>%
   SUMMARISE(MEAN = MEAN(SCORE,NA.RM = TRUE),
          N = LENGTH(APPRDX),
          SDT = SD(SCORE, NA.RM = TRUE),
          YMIN = mean - sdt,
```

```
            YMAX = mean + sdt)
df_ANX_summary = SUBSET(df_ANX_summary, EVENT_ID != "SC" & EVENT_ID != "ST" & EVENT_ID !=
                        "PW" & EVENT_ID != "RS1" & EVENT_ID != "U01" & EVENT_ID != "U02" & n>1)

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df_ANX = SUBSET(df_ANX, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df_ANX[!DUPLICATED(df_ANX$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                        DATA.FRAME(EVENT_ID = all.visits,
                                PATNO = all.patnos.apprdx$PATNO[s],
                                APPRDX =  all.patnos.apprdx$APPRDX[s],
                                X = 1:16)
                        )
}
df = MERGE(X = df.complete.visits, Y = df_ANX, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$SCORE[1:4])) >= 1,
        SUM(!IS.NA(sbjct$SCORE[5:8])) >= 1,
        SUM(!IS.NA(sbjct$SCORE[9:12])) >= 1,
        SUM(!IS.NA(sbjct$SCORE[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}
}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.i.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB(") +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                        "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR7_ANX.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.i.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$SCORE)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$SCORE = ROUND(imp$score, DIGITS = 0)
}

df_i_ANX= AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "SCORE")))

# SUMMARIZING DATA
df_i_ANX_summary = df_i_ANX %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(SCORE, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(SCORE, NA.RM = TRUE),
```

```
    YMAX = mean + sdt,
    YMIN = mean - sdt)

# FUNCTION
p.line.anxiety = FUNCTION(
      # DATA, X, Y, .. VALUES
      DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL", YMIN = "YMIN", YMAX = "YMAX",
      COLOUR = "FACTOR(APPRDX)", GROUP = "APPRDX",
      SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

      ## AXIS, LEGEND, LABELS
      XTEXT = "VISITS, DELTA = 3 MNT",
      YTEXT = "TEXT",
      XTITLE = TRUE,
      NM = "XX PER VISIT",
      LBLS = C("PARKINSON'S","CONTROLS / HEALTHY", "PRODROMAL"),
      LEGEND_POS = C(0.9, 0.9))
    {

    GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
    GEOM_POINT() + GEOM_LINE() +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 4,
       ALPHA = 0.8, SEGMENT.SIZE = 0) +
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 4,
       SEGMENT.SIZE = 0, COLOR = "WHITE", FILL ="DARKGREY") +
    GEOM_ERRORBAR(AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=colour), WIDTH=.5, SIZE = .7,
       POSITION=POSITION_DODGE(dodge)) +
    THEME(AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
      AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize))  +


    # LEGEND, AXIX
    LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
      ELSE THEME(LEGEND.POSITION=legend_pos, )) +

    SCALE_COLOR_DISCRETE(LABELS = lbls) +
    XLAB(xtext) +
    YLAB(ytext)
}

p.hist = FUNCTION(
    DATA = df.updrs.2, X = "UPDRS2", FILL = "FACTOR(APPRDX)",
    BNS = 20,  LPH = 0.5, XLIMIT = FALSE,
    NM = "HH",
    LBLS = C("PD", "HC"),
    XTEXT = "SCORES",
    YTEXT = "COUNTS",
    LGND = TRUE,
    LEGEND_POS = C(0.2, 0.2),
    MN = NULL,
    MX = NULL)
{
    GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
    GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
    THEME(
        LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = txtsize-2),
        LEGEND.TITLE = ELEMENT_TEXT(SIZE =  txtsize-2),
        LEGEND.KEY.SIZE = UNIT(.7, "LINE"),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        AXIS.TEXT=ELEMENT_TEXT(SIZE=txtsize),
        AXIS.TITLE=ELEMENT_TEXT(SIZE=txtsize)) +
    GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = .2))) +
    SCALE_FILL_DISCRETE(LABELS = lbls, NAME = nm) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
    XLAB(xtext) + YLAB(ytext) +
    LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos, LEGEND.TEXT.ALIGN = 1)
        ELSE THEME(LEGEND.POSITION='NONE'))
}


# PLOTS
lgnd.xpos = 0.7; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12
```

```
p.anxiety.line  = P.LINE.ANXIETY(DATA = df_ANX_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "SCORE",
    XTEXT = "", LEGEND_POS = "NONE")

p.anxiety.hist = P.HIST(DATA = df_ANX, X = "SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("STATE/TRAIT ANXIETY TOTAL SCORES)",20),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.anxiety.grid = GRID.ARRANGE(p.anxiety.hist, p.anxiety.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR7_ANXIETY.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.anxiety.grid)

# PLOTS
lgnd.xpos = 0.7; lgnd.ypos = 0.8
dodge = 0.2; txtsize = 12

p.anxiety.line  = P.LINE.ANXIETY(DATA = df_i_ANX_summary, Y = "MEAN", YMIN = "YMIN", YMAX = "YMAX",
    YTEXT = "SCORE IMPUTED",
    XTEXT = "", LEGEND_POS = "NONE")

p.anxiety.hist = P.HIST(DATA = df_i_ANX, X = "SCORE", XTEXT = "", YTEXT = "", LGND= T,
    NM = STR_WRAP("STATE/TRAIT ANXIETY TOTAL SCORES)",20),
    LEGEND_POS = C(lgnd.xpos, lgnd.ypos))

# MEAN VAOUE OF COMBINING PLOT AND TABLES INTO ONE PLOT
lay = RBIND(C(1,2,2,2))
p.anxiety.grid = GRID.ARRANGE(p.anxiety.hist, p.anxiety.line,
    LAYOUT_MATRIX = lay)
```

```
GGSAVE(PASTE0(p_images,"VAR7_I_ANXIETY.JPG"),
    DPI = 300,
    HEIGHT = 2,
    WIDTH = 10,
    SCALE  = 1.5,
    PLOT= p.anxiety.grid)
```

Merging and Storing Imputed Data

- df.i.orth

```
# CURATED DATA PER MEASUREMENT, BLODPRESSURE, REM,  MOC, UPDARS, DEPRESSION , ANXIETY
#DF.I.ORTH
COLNAMES(df.i.rem)[4] ="REM"
COLNAMES(df_i_moc)[4] ="MOC"
#DF_I_UPDRS2
#DF_I_UPDRS3
COLNAMES(df_i_GDS_RAW_SCORE)[4] = "GDS"
COLNAMES(df_i_ANX)[4] = "ANX"

df.i = data.frame
df.i = MERGE(df.i.orth, df.i.rem, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)
df.i = MERGE(df.i, df_i_moc, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)
df.i = MERGE(df.i, df_i_updrs2, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)
df.i = MERGE(df.i, df_i_updrs3, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)
df.i = MERGE(df.i, df_i_GDS_RAW_SCORE, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)
df.i = MERGE(df.i, df_i_ANX, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE)

df.i = df.i[COMPLETE.CASES(df.i), ]
WRITE.CSV(df.i, PASTE0(p, "PPMI_IMPUTED.CSV"), ROW.NAMES=FALSE)
```

B2 Data Aggregation

- Data need: df.ppmi_imputed.csv generated in file "01_DataClean_2.Rmd"

```
mydf = READ.CSV(PASTE0(p, "DF.PPMI_IMPUTED.CSV"))
mydf[,-C(1,2,3)] = MIN.MAX.SCALING(mydf[,-C(1,2,3)])

PD = LENGTH(UNIQUE(mydf[mydf$APPRDX == 1,]$PATNO))
HC = LENGTH(UNIQUE(mydf[mydf$APPRDX == 2,]$PATNO))
Records = LENGTH(mydf$PATNO)

aggreg.subjects = KABLE(DATA.FRAME(PD = PD, HC = HC),  TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%  KABLE_STYLING(FULL_WI
DTH = F)  %>%
    SAVE_KABLE("02_IMAGES/AGGREG.SUBJECTS.PNG")
```

Adjusting mean per event of SYSORH and MOCA

The average scores for HC where higher in SYSORH and MOCA higher than for PD. In all other tests e.g. ANX, Depression, … the PD Parkions score where higher. In order to cummulate the all scores in Nonmotor and motor representative scores the scores mean has been shifted without change the difference between PD and HC.

Aggregating to Motor and Nonmotor Scores

```
p.hist.motor = GGPLOT(DATA = df.nonmotor.motor, AES(X = MOTOR, FILL = FACTOR(APPRDX)))+
    GEOM_HISTOGRAM(BINS =100, POSITION = "IDENTITY", ALPHA=0.75) +
    THEME(LEGEND.POSITION = C(0.8, 0.8),
        LEGEND.BACKGROUND = ELEMENT_BLANK()) +
    SCALE_FILL_DISCRETE(NAME = "", LABELS = C("PD", "HC")) +
    XLAB("MOTOR")

p.hist.motor
```

```
IMAGE.SAVE(MYPLOT = p.hist.motor, PIC.NAME = "AGGREG.HIST.MOTOR", SCL = 3)

p.hist.nonmotor = GGPLOT(DATA = df.nonmotor.motor, AES(X = NON.MOTOR, FILL = FACTOR(APPRDX)))+
    GEOM_HISTOGRAM(BINS =100, POSITION = "IDENTITY", ALPHA=0.75) +
    THEME(LEGEND.POSITION = C(0.8, 0.8),
        LEGEND.BACKGROUND = ELEMENT_BLANK()) +
    SCALE_FILL_DISCRETE(NAME = "", LABELS = C("PD", "HC")) +
    XLAB("NON MOTOR")

p.hist.nonmotor
```

```
IMAGE.SAVE(MYPLOT = p.hist.nonmotor, PIC.NAME = "AGGREG.HIST.NONMOTOR", SCL = 3)

dt = df.nonmotor.motor
dt$APPRDX = IFELSE(dt$APPRDX == 1, "PD", "HC")


p.boxplot.nonmotor = GGPLOT(DATA = dt, AES(X = FACTOR(EVENT_ID),
                            Y = NON.MOTOR,  FILL = FACTOR(APPRDX))) +
    GEOM_BOXPLOT() +
            THEME(LEGEND.POSITION=C(.04, .95),
                LEGEND.BACKGROUND=ELEMENT_BLANK()) +
            SCALE_X_DISCRETE(NAME =NULL) +
            SCALE_FILL_MANUAL(VALUES = C("#00BFC4", "#F8766D")) +
            GUIDES(FILL=GUIDE_LEGEND(TITLE="")) +
            YLAB("NON MOTOR")

p.boxplot.nonmotor
```

```
IMAGE.SAVE(MYPLOT = p.boxplot.nonmotor, PIC.NAME = "AGGREG.BOXPLOT.NONMOTOR", SCL = 3, HGT = 1, WDTH = 2)

dt = df.nonmotor.motor
dt$APPRDX = IFELSE(dt$APPRDX == 1, "PD", "HC")


p.boxplot.motor = GGPLOT(DATA = dt, AES(X = FACTOR(EVENT_ID),
                                Y = MOTOR,  FILL = FACTOR(APPRDX))) +
    GEOM_BOXPLOT() +
            THEME(LEGEND.POSITION=C(.04, .95),
                LEGEND.BACKGROUND=ELEMENT_BLANK()) +
            SCALE_X_DISCRETE( NAME =NULL) +
            SCALE_FILL_MANUAL(VALUES = C("#00BFC4", "#F8766D")) +
            GUIDES(FILL=GUIDE_LEGEND(TITLE="")) +
            YLAB("MOTOR")

p.boxplot.motor
```

```
IMAGE.SAVE(MYPLOT = p.boxplot.motor, PIC.NAME = "AGGREG.BOXPLOT.MOTOR", SCL = 3, HGT = 1, WDTH = 2)
```

t - test

MOTOR

```
df.tst = DATA.FRAME()
FOR (event IN all.visits){

mc.pd = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 1 & EVENT_ID == event, SELECT = C("MOTOR"))))
mc.hc = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 2 & EVENT_ID == event, SELECT = C("MOTOR"))))

# F-TEST TO CHECK VARIANCE
ftst = VAR.TEST(X = mc.pd, Y = mc.hc)
pftst = SIGNIF(ftst$p.value,4)

# TEST TO CHECK DIFF HC, AND PD
IF (pftst < 0.05) vql = FALSE
ttst = T.TEST(X = mc.pd, Y = mc.hc, VAR.EQUAL = vql)
pttst = SIGNIF(ttst$p.value, 4)

temp = DATA.FRAME(EVENT_ID = event, F.TEST_VARIANCE.HD.PD = pftst, STATISTIC = ttst$statistic,
            T.TEST_MEAN.HD.PD = pttst, MEAN.HC = MEAN(mc.hc), SD.HC = SD(mc.hc),
            MEAN.PD = MEAN(mc.pd), SD.PD = SD(mc.pd))
     df.tst = RBIND(df.tst, temp)
}

NAMES(df.tst)

## [1] "EVENT_ID"            "F.TEST_VARIANCE.HD.PD" "STATISTIC"
## [4] "T.TEST_MEAN.HD.PD"    "MEAN.HC"               "SD.HC"
## [7] "MEAN.PD"              "SD.PD"

KABLE(df.tst, DIGITS = 200, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

df.tst$HoVar = IFELSE(df.tst$f.test_variance.HD.PD < 0.05,"REJECTED", "FAIL TO REJECT")
df.tst$HoMean = IFELSE(df.tst$t.test_mean.HD.PD < 0.05,"REJECTED", "FAIL TO REJECT")
df.tst[,-C(1,9,10)] = SIGNIF(df.tst[,-C(1, 9,10)], 3)


df.tst = df.tst[,C(1,9,2,8,6,10,4,7,5)]
cnames = C("VISIT", "V: HO", "V: P.VALUE",   "SD: PD", "SD: HC", "M: HO", "M: P.VALUE", "M: PD", "M: HC" )
COLNAMES(df.tst) =  cnames
ROWNAMES(df.tst) = NULL


WRITE.XLSX(df.tst, PASTE0(p_images, "AGGREG.MOTOR.TESTS.XLSX"))

KABLE(df.tst, DIGITS = 40, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

```
# ENROLLED
rpath = "/USERS/JOERGHEINTZ/BOX/02_DATA_ANALYTICS_PROJECTS/001_MASTERTHESIS/"
# LOAD RANDOM
path = PASTE0(rpath, "01_DATA/DATA_LONI/05_STUDY_ENROLLMENT/")
df_ran  = READ.CSV(PASTE0(path, "RANDOMIZATION_TABLE.CSV"))
# LOAD SCREEN
path = PASTE0(rpath, "01_DATA/DATA_LONI/02_SUBJECT_CHARACTERISTICS/")
df_scr  = READ.CSV(PASTE0(path, "SCREENING___DEMOGRAPHICS.CSV"))
# MERGE
df_temp = MERGE(df_ran, df_scr, BY = 'PATNO')
df_enrolled_all = GET_ENROLLED(C(1,2))

# DEPRESSION
path = PASTE0(rpath, "01_DATA/DATA_LONI/09_NON-MOTOR_ASSESSMENTS/")
file = "GERIATRIC_DEPRESSION_SCALE__SHORT_.CSV"
df_depression = READ.CSV(PASTE0(path, file))

# MERGE
df_depression = MERGE(df_enrolled_all, df_depression, BY = 'PATNO')
df_depression = df_depression[ORDER(df_depression$EVENT_ID, df_depression$PATNO), ]

#ADD 1 POINT FOR EACH RESPONSE OF "NO" (0) TO ANY OF THE FOLLOWING VARIABLES:
df_dep_sum1 = SUBSET(df_depression, SELECT = C(PATNO,APPRDX, GDSSATIS, GDSGSPIR, GDSHAPPY,GDSALIVE, GDSENRGY))
df_dep_sum1$sum = ROWSUMS(df_dep_sum1[,-C(1)]==0, NA.RM = TRUE)

#ADD 1 POINT FOR EACH RESPONSE OF "YES" (1) TO ANY OF THE FOLLOWING VARIABLES:
df_dep_sum2 = SUBSET(df_depression, SELECT = C(PATNO, APPRDX, GDSDROPD, GDSEMPTY, GDSBORED,GDSAFRAD, GDSHLPLS, GDSHOM
E,GDSMEMRY, GDSWRTLS, GDSHOPLS,GDSBETER))
df_dep_sum2$sum = ROWSUMS(df_dep_sum2[,-C(1)]==1, NA.RM = TRUE)

df_dep_gds = DATA.FRAME(PATNO = df_depression$PATNO,
                        APPRDX = df_depression$APPRDX,
                        EVENT_ID = df_depression$EVENT_ID,
                        GDS_SUM1 = df_dep_sum1$sum,
                        GDS_SUM2 = df_dep_sum2$sum)
df_dep_gds$GDS_RAW_SCORE = ROWSUMS(df_dep_gds[,C("GDS_SUM1", "GDS_SUM2")])
df_dep_gds = SUBSET(df_dep_gds, EVENT_ID != "SC" & EVENT_ID != "ST" & EVENT_ID != "PW" & EVENT_ID != "RS1" & EVENT_ID != "U01"
& EVENT_ID != "U02")

# SUMMARIZE BY EVENT ID, SUBJECT GROUP
df_dep_gds_summaried = df_dep_gds %>% GROUP_BY(APPRDX, EVENT_ID) %>%
   SUMMARISE(MEAN = MEAN(GDS_RAW_SCORE, NA.RM = TRUE),
          N = LENGTH(APPRDX),
          SDT = SD(GDS_RAW_SCORE),
          YMIN = mean - sdt,
          YMAX = mean + sdt)

# TAKE OUT VISITS < VIST
df_dep_gds_summaried = df_dep_gds_summaried[df_dep_gds_summaried$n > 1,]

#  IMPUTATION
## GENERATE TEMPLATE WITH PATNO, EVENT_ID AS MASTER TO DETERMINE MISSING VALUES
df_dep_gds = SUBSET(df_dep_gds, EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
df.complete.visits = DATA.FRAME()
all.patnos.apprdx = df_dep_gds[!DUPLICATED(df_dep_gds$PATNO), C("PATNO", "APPRDX")]
FOR (s IN 1:LENGTH(all.patnos.apprdx$PATNO)){
    df.complete.visits = RBIND(df.complete.visits,
                        DATA.FRAME(EVENT_ID = all.visits,
                                PATNO = all.patnos.apprdx$PATNO[s],
                                APPRDX =  all.patnos.apprdx$APPRDX[s],
                                X = 1:16)
                        )
}
df = MERGE(X = df.complete.visits, Y = df_dep_gds, BY = C("PATNO", "EVENT_ID", "APPRDX"), ALL = TRUE, SORT = FALSE)
df = df[ORDER(df$PATNO, df$x), ]


# FIND SUBJECTS THAT HAVE ATTENDED AT LAST ONCE IN EACH QUARTILE OF THE 15 VISITS PLUS BASELINE.
df$relevance = NA

FOR (subject IN all.patnos.apprdx$PATNO){
    sbjct = SUBSET(df, PATNO == subject)
    missing.na.per.quantile  =  SUM(
        SUM(!IS.NA(sbjct$GDS_RAW_SCORE[1:4])) >= 1,
```

```
        SUM(!IS.NA(sbjct$GDS_RAW_SCORE[5:8])) >= 1,
        SUM(!IS.NA(sbjct$GDS_RAW_SCORE[9:12])) >= 1,
        SUM(!IS.NA(sbjct$GDS_RAW_SCORE[13:16])) >= 1)
    IF (missing.na.per.quantile >= 1){
        df[df$PATNO == subject,]$relevance = missing.na.per.quantile}
        ELSE{
        df[df$PATNO == subject,]$relevance = 0}
}


## HISTOGRAM THAT SHOWS THE HOW MANY SUBJECTS HAVE VISITED ALL, 3,  2 AND ONE  QUARTILE
d = DATA.FRAME(TABLE(df$relevance)/16)
p.na.i.distribution = GGPLOT(DATA = d, AES(X = Var1, Y = Freq))  +
    GEOM_BAR(STAT = "IDENTITY",FILL = "STEELBLUE") +
    GEOM_TEXT(AES(LABEL=Freq), VJUST=1.7, SIZE=3, COLOR = 'WHITE')+
    XLAB('') +
    YLAB("SUBJECTS") +
    SCALE_X_DISCRETE(LABELS=C("1" = "NO VALUES IN 3 QUARTILES", "2" = "NO VALUES IN 2 QUARTILES",
                     "3" = "NO VALUES IN 1 QUARTILE","4" = "VALUES IN ALL QUARTILES"))

GGSAVE(PASTE0(p_images, "VAR6_GDS_RAW_SCORE.NA.DISTRIBUTION.JPG"),
    DPI = 300,
    HEIGHT = 1,
    WIDTH = 1.75,
    SCALE  = 4,
    PLOT= p.na.i.distribution)


## IMPUTATION PER PATNO
df = df[df$relevance > 3, ]
FOR (subject IN UNIQUE(df$PATNO)){
    score = AS.NUMERIC(df[df$PATNO == subject, ]$GDS_RAW_SCORE)
    x = 1:16

    data = DATA.FRAME(CBIND(score, x))
    imp = REGRESSIONIMP(score ~ x,DATA=data)
    df[df$PATNO == subject, ]$GDS_RAW_SCORE = ROUND(imp$score, DIGITS = 0)
}

df_i_GDS_RAW_SCORE= AS.DATA.FRAME(SUBSET(df, SELECT = C("PATNO", "EVENT_ID", "APPRDX", "GDS_RAW_SCORE")))

# SUMMARIZING DATA
df_i_GDS_RAW_SCORE_summary = df_i_GDS_RAW_SCORE %>% GROUP_BY(APPRDX, EVENT_ID) %>% SUMMARISE(
    MEAN = MEAN(GDS_RAW_SCORE, NA.RM = TRUE),
    N = LENGTH(APPRDX),
    SDT = SD(GDS_RAW_SCORE, NA.RM = TRUE),
    YMAX = mean + sdt,
    YMIN = mean - sdt)
```

NON.MOTOR

```
df.tst = DATA.FRAME()
FOR (event IN all.visits){

mc.pd = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 1 & EVENT_ID == event, SELECT = C("NON.MOTOR"))))
mc.hc = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 2 & EVENT_ID == event, SELECT = C("NON.MOTOR"))))

# F-TEST TO CHECK VARIANCE
ftst = VAR.TEST(X = mc.pd, Y = mc.hc)
pftst = SIGNIF(ftst$p.value, 4)

# TEST TO CHECK DIFF HC, AND PD
IF (pftst < 0.05) vql = FALSE
ttst = T.TEST(X = mc.pd, Y = mc.hc, VAR.EQUAL = vql)
pttst = SIGNIF(ttst$p.value, 4)

temp = DATA.FRAME(EVENT_ID = event, F.TEST_VARIANCE.HD.PD = pftst, STATISTIC = ttst$statistic,
            T.TEST_MEAN.HD.PD = pttst, MEAN.HC = MEAN(mc.hc), SD.HC = SD(mc.hc),
            MEAN.PD = MEAN(mc.pd), SD.PD = SD(mc.pd))
df.tst = RBIND(df.tst, temp)
}

NAMES(df.tst)
```

```
## [1] "EVENT_ID"            "F.TEST_VARIANCE.HD.PD" "STATISTIC"
## [4] "T.TEST_MEAN.HD.PD"     "MEAN.HC"             "SD.HC"
## [7] "MEAN.PD"             "SD.PD"

df.tst$HoVar = IFELSE(df.tst$f.test_variance.HD.PD < 0.05,"REJECTED", "FAIL TO REJECT")
df.tst$HoMean = IFELSE(df.tst$t.test_mean.HD.PD < 0.05,"REJECTED", "FAIL TO REJECT")
df.tst[,-C(1,9,10)] = SIGNIF(df.tst[,-C(1, 9,10)], 3)


df.tst = df.tst[,C(1,9,2,8,6,10,4,7,5)]
cnames = C("VISIT", "V: HO", "V: P.VALUE",   "SD: PD", "SD: HC", "M: HO", "M: P.VALUE", "M: PD", "M: HC" )
COLNAMES(df.tst) =  cnames
ROWNAMES(df.tst) = NULL


WRITE.XLSX(df.tst, PASTE0(p_images, "AGGREG.NONMOTOR.TESTS.XLSX"))

KABLE(df.tst, DIGITS = 40, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

ml.pd = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 2 & EVENT_ID == "BL", SELECT = C("NON.MOTOR"))))
ml.hc = AS.VECTOR(UNLIST(SUBSET(df.nonmotor.motor, APPRDX == 1 & EVENT_ID == "BL", SELECT = C("NON.MOTOR"))))

VAR.TEST(X = ml.pd, Y = ml.hc)

##
##  F TEST TO COMPARE TWO VARIANCES
##
## DATA:  ML.PD AND ML.HC
## F = 0.66838, NUM DF = 146, DENOM DF = 288, P-VALUE = 0.006584
## ALTERNATIVE HYPOTHESIS: TRUE RATIO OF VARIANCES IS NOT EQUAL TO 1
## 95 PERCENT CONFIDENCE INTERVAL:
##  0.5072358 0.8924487
## SAMPLE ESTIMATES:
## RATIO OF VARIANCES
##         0.6683845

T.TEST(X = ml.pd, Y = ml.hc, VAR.EQUAL = FALSE)

##
##  WELCH TWO SAMPLE T-TEST
##
## DATA:  ML.PD AND ML.HC
## T = -10.145, DF = 350.01, P-VALUE < 2.2E-16
## ALTERNATIVE HYPOTHESIS: TRUE DIFFERENCE IN MEANS IS NOT EQUAL TO 0
## 95 PERCENT CONFIDENCE INTERVAL:
##  -0.3600392 -0.2431099
## SAMPLE ESTIMATES:
## MEAN OF X MEAN OF Y
##  1.634587  1.936162
```

B3 Clustering

Ingestion

```
df.nonmotor.motor = READ.CSV(PASTE0(p,'DF.NONMOTOR.MOTOR.CSV'))
#DF.NONMOTOR.MOTOR$APPRDX = AS.FACTOR(IFELSE(DF.NONMOTOR.MOTOR$APPRDX == 1, "1", "0"))
df.ppmi_basis = READ.CSV(PASTE0(p,'DF.PPMI_BASIS_DATA_CLEAN.CSV'))

# SCALING Z-SALING
df.pd.hc.cluster = df.nonmotor.motor
#DF.PD.HC.CLUSTER[,-C(1,2,3)] = SCALE(DF.NONMOTOR.MOTOR[,-C(1,2,3)])
df.pd.cluster = SUBSET(df.nonmotor.motor, APPRDX == 1)



## LINEAR RELATIONSHIP BETWEEN MOTOR AND NON.MOTOR FUNCTION
    fit = LM(NON.MOTOR ~ MOTOR, DATA = SUBSET(df.pd.cluster, APPRDX == 1, SELECT = C("MOTOR","NON.MOTOR")))
    SUMMARY(fit)$coefficients[2,]
```

```
##     ESTIMATE   STD. ERROR      T VALUE     PR(>|T|)
## 3.269448E-01 2.063355E-02 1.584530E+01 4.179845E-55

    beta1 = SUMMARY(fit)$coefficients[2,1]
    SE_beta1 = SUMMARY(fit)$coefficients[2,2]
    alpha = 0.05
    n= LENGTH(SUBSET(df.pd.cluster, APPRDX == 1, SELECT = C("MOTOR","NON.MOTOR"))$NON.MOTOR)
    crit = QT((0.05)/2, n-2) # TWO SIDED TEST

    c = LIST(ROUND(beta1 + C(1,-1) * SE_beta1 * crit, 4))
    PASTE("CONFIDENCE INTERVALL SLOPE BETA 1 = ", c)

## [1] "CONFIDENCE INTERVALL SLOPE BETA 1 =  C(0.2865, 0.3674)"

### CORRELATION BETWEEN MOTOR AND NON MOTOR
    d = SUBSET(df.pd.cluster, APPRDX == 1, SELECT = C("MOTOR","NON.MOTOR"))
    COR(X = d$MOTOR, Y = d$NON.MOTOR)

## [1] 0.2269859

    ctest = COR.TEST(X = d$MOTOR, Y = d$NON.MOTOR, METHOD = "PEARSON")
    PASTE("CI  95% ",ctest$estimate,  LIST(ctest$conf.int))

## [1] "CI  95%  0.226985897727729 C(0.199466585216809, 0.254147448391625)"

x1 = 0.45; x2 =2.45; y1=0.8; y2 = 3.8

p.scatter.pd = GGPLOT(DATA = SUBSET(df.pd.cluster, APPRDX == 1),
            AES(X = MOTOR, Y = NON.MOTOR, COLOR = FACTOR(APPRDX))) +
    GEOM_POINT(SIZE = .5, ALPHA =1) +
    XLIM(x1,x2) +
    YLIM(y1,y2) +
    XLAB("MOTOR") +
    YLAB("NON MOTOR") +
    THEME(LEGEND.POSITION = C(0.04, 0.97),
        LEGEND.BACKGROUND = ELEMENT_BLANK(),
        LEGEND.TITLE = ELEMENT_BLANK()) +
    SCALE_COLOR_DISCRETE(NAME = "", LABELS = C("PD")) +
    GEOM_SMOOTH(METHOD = "LM", COLOUR = 'DARKGREY')

p.scatter.pd

IMAGE.SAVE(
    p.scatter.pd,
    PIC.NAME = "CLUST.SCATTER.PD",
    )


# IMAGE.SAVE(
#     GRID.ARRANGE(P.SCATTER.PD.HC, P.SCATTER.PD, NCOL = 1, HEIGHTS = C(1,1)),
#     PIC.NAME = "CLUST.SCATTER.PD.HC.PD",
#     )
```

Transformation

Neither applying log, exp, or sqrt on the axis revealed a cluster.

```
df.kmeans = df.pd.cluster[,C("MOTOR", "NON.MOTOR")]
MOTOR =  df.kmeans$MOTOR
NON.MOTOR = df.kmeans$NON.MOTOR

PLOT(LOG(df.kmeans$MOTOR), (df.kmeans$NON.MOTOR), CEX = .1, PCH = 10)
```

Determine the centers

```
### ELBOW
    alg = C("HARTIGAN-WONG", "LLOYD", "FORGY",
                "MACQUEEN")
    ### OPTIMAL NUMBRER OF LCUSGTERS
    SET.SEED(123)

    # FUNCTION TO COMPUTE TOTAL WITHIN-CLUSTER SUM OF SQUARE
    wss <- FUNCTION(k) {
```

```
    KMEANS(df.kmeans, k,
          ALGORITHM = alg[1],
          ITER.MAX = 100,
          NSTART = 100 )$tot.withinss
    }

    # COMPUTE AND PLOT WSS FOR K = 1 TO K = 15
    # K.VALUES <- 1:5
    #
    # # EXTRACT WSS FOR 2-5 CLUSTERS
    # WSS_VALUES <- MAP_DBL(K.VALUES, WSS)
    # PLOT(K.VALUES, WSS_VALUES,
    #       TYPE="B", PCH = 19, FRAME = FALSE,
    #       XLAB="NUMBER OF CLUSTERS K",
    #       YLAB="TOTAL WITHIN-CLUSTERS SUM OF SQUARES")


# SILHOUETTE
### FUNCTION TO COMPUTE AVERAGE SILHOUETTE FOR K CLUSTERS
#      AVG_SIL <- FUNCTION(K) {
#        KM.RES <- KMEANS(DF.KMEANS, CENTERS = K, NSTART = 100, ITER.MAX = 100)
#        SS <- SILHOUETTE(KM.RES$CLUSTER, DIST(DF.KMEANS))
#        MEAN(SS[, 3])
#      }
#
# ### COMPUTE AND PLOT WSS FOR K = 2 TO K = 15
#      K.VALUES <- 2:5

## EXTRACT AVG SILHOUETTE FOR 2-15 CLUSTERS
    # AVG_SIL_VALUES <- MAP_DBL(K.VALUES, AVG_SIL)
    #
    # PLOT(K.VALUES, AVG_SIL_VALUES,
    #       TYPE = "B", PCH = 19, FRAME = FALSE,
    #       XLAB = "NUMBER OF CLUSTERS K",
    #       YLAB = "AVERAGE SILHOUETTES")

# GAP STAT
    gap_stat <- CLUSGAP(df.kmeans, FUN = kmeans, NSTART = 100,
                        K.MAX = 4, B = 50)


# PLOT
IMAGE.SAVE(
    GRID.ARRANGE(
        FVIZ_NBCLUST(df.kmeans, kmeans, METHOD = "WSS") +
            YLAB("ELBOW") + XLAB(""),
        FVIZ_NBCLUST(df.kmeans, kmeans, METHOD = "SILHOUETTE") +
            YLAB("SILHOUETTE") +
            XLAB("") + LABS(TITLE = "")
    #      FVIZ_GAP_STAT(GAP_STAT) + YLAB("GAB STATISTIC")+
    #          XLAB("") +
    #          LABS(TITLE = "")
    ),
    PIC.NAME = "CLUST.CENTER.DETERM.",
    HGT = 1,
    WDTH = 1.5,
    SCL = 4.5
    )
```

Cluster Parkinsons into 2 clsuter

```
SET.SEED(1968)

km.out = KMEANS((df.kmeans),
            ITER.MAX = 100,
            CENTERS  = 2,
            NSTART = 1000,
            ALGORITHM = alg[4])

p.cluster =  FVIZ_CLUSTER(km.out,
              DATA = df.kmeans,
              GEOM = "POINT",
              YLAB = "NON MOTOR",
              XLAB = "MOTOR",
              MAIN = "",
```

```
            SHOW.CLUST.CENT = TRUE,
            SHOW.CLUST.CENTER = F) +
        THEME(LEGEND.TITLE = ELEMENT_BLANK(),
            LEGEND.POSITION = C(0.08, 1.05),
            LEGEND.BACKGROUND = ELEMENT_BLANK()) +
        SCALE_COLOR_MANUAL(
            VALUES=C("DARKORANGE", "DODGERBLUE"),
            LABELS = C("CLUSTER 1", "CLUSTER 2"))


p.cluster

IMAGE.SAVE(p.cluster,
    PIC.NAME = "CLUST.PARKINSONS",
        HGT = 1,
        WDTH = 1.5,
        SCL = 4.5
)

myres = DATA.FRAME(T(DATA.FRAME(TOTAL.VARIANCE = km.out$totss, VARIANCE.WITHIN.CLUSTER.1 = km.out$withinss[1],
         VARIANCE.WITHIN.CLUSTER.1 = km.out$withinss[2],
         VARIANCE.BETWEEN.CLUSTERS = km.out$betweenss,
        VARIANCE.EXPLAINED.BY.CLUSTERS = km.out$betweenss/km.out$totss
)))

rownames = C("TOTAL VARIANCE", "VARIANCE WITHIN CLUSTER 1", "VARIANCE WITHIN CLUSTER 2",
         "VARIANCE EXPLAINED BY CLUSTER DISTANCE", "% OF VARIANCE EXPLAINED BY CLUSTERS")

COLNAMES(myres) = "VALUES"

KABLE(myres, DIGITS = 200, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

B4 Predicting

```
df.nonmotor.motor = READ.CSV(PASTE0(p,'DF.NONMOTOR.MOTOR.CSV'))
df.nonmotor.motor$APPRDX = AS.FACTOR(IFELSE(df.nonmotor.motor$APPRDX == 1, "1", "0"))
df.ppmi_basis = READ.CSV(PASTE0(p,'DF.PPMI_BASIS_DATA_CLEAN.CSV'))
## SET CLASSES
df.nonmotor.motor$PATNO = AS.FACTOR(df.nonmotor.motor$PATNO)
df.nonmotor.motor$APPRDX = AS.FACTOR(df.nonmotor.motor$APPRDX)
df.nonmotor.motor$EVENT_ID = AS.FACTOR(df.nonmotor.motor$EVENT_ID)
df.nonmotor.motor$NON.MOTOR = AS.NUMERIC(df.nonmotor.motor$NON.MOTOR)
df.nonmotor.motor$MOTOR = AS.NUMERIC(df.nonmotor.motor$MOTOR)


# SAMPLE SIZE
spsz = DATA.FRAME(C(AS.CHARACTER(AS.VECTOR(TABLE(df.nonmotor.motor$APPRDX))), LENGTH(df.nonmotor.motor$APPRDX)))

pertg = AS.NUMERIC(AS.CHARACTER(UNLIST(spsz))) / AS.NUMERIC(AS.CHARACTER(UNLIST(spsz)[[3]]))
pertg = ROUND(pertg, 3)
spsz$Percent = 100 * pertg

ROWNAMES(spsz) = C("HC", "PD", "SUM")
COLNAMES(spsz) = C("SIZE", "PERCENTAGE")
KABLE(spsz, DIGITS = 200, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

# MEAN, SD, SAMPLE SIZE, MEDIAN

dt.stat = DATA.FRAME(APPLY(df.nonmotor.motor[,-C(1,2,3)], 2, FUNCTION(x){
    CBIND(MEAN(x),
        SD(x))
}))

ROWNAMES(dt.stat) = C("M", "SD")
COLNAMES(dt.stat) = C("MOTOR", "NON MOTOR")
KABLE(dt.stat, DIGITS = 200, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

Create Training and Testing Data

the imbalanced data set (HC = 2352, PD = 4624) to sampling unbalanced data set, nbalanced data set

```
# HC, PD = 1378 TEST DATA EACH
SET.SEED(1968)
x = SAMPLE(2352, ROUND(4624*.2, 0), REPLACE = F)
# TESTING
pd.data.testing = SUBSET(df.nonmotor.motor, APPRDX == 1)[x,]
hc.data.testing = SUBSET(df.nonmotor.motor, APPRDX == 0)[x,]
testing = RBIND(pd.data.testing, hc.data.testing)


# TRAINING
pd.data.training = SUBSET(df.nonmotor.motor, APPRDX == 1)[-x,]
hc.data.training = SUBSET(df.nonmotor.motor, APPRDX == 0)[-x,]
training = RBIND(pd.data.training, hc.data.training)
LIST(DIM(training), DIM(testing))

## [[1]]
## [1] 5126    5
##
## [[2]]
## [1] 1850    5

# GENERATE TABLE TO SHOW TEST AND TRAINING DATA SET
tst.trn.dta = CBIND(DATA.FRAME(TABLE(training$APPRDX)),
            DATA.FRAME(TABLE(testing$APPRDX))[,2],
            DATA.FRAME(TABLE(testing[testing$EVENT_ID == "BL", ]$APPRDX))[,2],
            SUM = DATA.FRAME(TABLE(training$APPRDX))[,2] +
                DATA.FRAME(TABLE(testing$APPRDX))[2])
tst.trn.dta = RBINDLIST(LIST(tst.trn.dta, DATA.FRAME("SUM", T(AS.VECTOR(COLSUMS(tst.trn.dta[,-1])[1:4])))))
COLNAMES(tst.trn.dta) = C("", "TRAINING", "TESTING [ALL]", "TESTING [INITIAL VISIT]", "SUM")
tst.trn.dta[1:2,1] = C("HC", "PD")


KABLE(tst.trn.dta, DIGITS = 200, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

WRITE.XLSX(tst.trn.dta, PASTE0(p_images, "PRED.BALANCED.IMBALANCED.DATASET.XLSX"), ROW.NAMES=FALSE)
```

Feature Plot Aggregated model

```
FEATUREPLOT(X = training[, 4:5],
        Y = AS.FACTOR(training$APPRDX),
        PLOT = "DENSITY",
        SCALES = LIST(X = LIST(RELATION = "FREE"),
        Y = LIST(RELATION = "FREE")),
        ADJUST = 1.5,
        PCH = "|",
        LAYOUT = C(2, 1),
        AUTO.KEY = LIST(COLUMNS = 2))
```

Logistic Regression Model

Cross validation logist regression

```
model_full = GLM(FACTOR(APPRDX) ~ NON.MOTOR + MOTOR, DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")], FAMILY = "BINO
MIAL")
model_motor = GLM(FACTOR(APPRDX) ~ MOTOR, DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")], FAMILY = "BINOMIAL")
model_nonmotor = GLM(FACTOR(APPRDX) ~ NON.MOTOR, DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")], FAMILY = "BINOMIAL
")

# MISCLASSIFICATION RATE
df.mis.class = DATA.FRAME(
    MIS.CLASS.FULL = CV.GLM(training, model_full,  K = 10)$delta[1],
    MIS.CLASS.MOTOR = CV.GLM(training, model_motor,  K = 10)$delta[1],
    MIS.CLASS.NONMOTOR = CV.GLM(training, model_nonmotor,  K = 10)$delta[1]
)

ROWNAMES(df.mis.class) = "MISCLASSIFICATION RATE"
COLNAMES(df.mis.class) = C("FULL MODEL", "MOTOR MODEL", "NON-MOTOR MODEL")

KABLE(df.mis.class, DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

Model Motor and Mental Training Data

```
model_full = GLM(FACTOR(APPRDX) ~.,
            DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")],
            FAMILY = "BINOMIAL", MAXIT = 100)

SUMMARY(model_full)

##
## CALL:
## GLM(FORMULA = FACTOR(APPRDX) ~ ., FAMILY = "BINOMIAL", DATA = TRAINING[,
##     C("APPRDX", "MOTOR", "NON.MOTOR")], MAXIT = 100)
##
## DEVIANCE RESIDUALS:
##     MIN       1Q   MEDIAN       3Q      MAX
## -3.4287  -0.0532   0.0017   0.0391   3.4073
##
## COEFFICIENTS:
##             ESTIMATE STD. ERROR Z VALUE PR(>|Z|)
## (INTERCEPT) -30.7025     1.2719  -24.14   <2E-16 ***
## MOTOR        27.0592     1.1281   23.99   <2E-16 ***
## NON.MOTOR     3.8614     0.2972   12.99   <2E-16 ***
## ---
## SIGNIF. CODES:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (DISPERSION PARAMETER FOR BINOMIAL FAMILY TAKEN TO BE 1)
##
##     NULL DEVIANCE: 6063.3  ON 5125  DEGREES OF FREEDOM
## RESIDUAL DEVIANCE: 1022.4  ON 5123  DEGREES OF FREEDOM
## AIC: 1028.4
##
## NUMBER OF FISHER SCORING ITERATIONS: 9

model_full = GLM(FACTOR(APPRDX) ~.,
            DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")],
            FAMILY = "BINOMIAL", MAXIT = 100)
# PREDICTION TYP 'LINK' GIVE LOGODDS
    model_full_pred = IFELSE(PREDICT(model_full, training, TYPE = "LINK") > 1, "1", "0")

    calc_class_err = FUNCTION(actual, predicted) {
        MEAN(actual != predicted)
    }

    # CLASSIFICATION ERROR
    CALC_CLASS_ERR(ACTUAL = training$APPRDX, PREDICTED = model_full_pred)

## [1] 0.04564963

    # CONFUSTION MATRIX
    train_tab = TABLE(PREDICTED = model_full_pred, ACTUAL = training$APPRDX)
    train_tab

##         ACTUAL
## PREDICTED    0    1
##         0 1382  189
##         1   45 3510

    train_con_mat = CONFUSIONMATRIX(train_tab, POSITIVE = "1")
    df.pred.training = DATA.FRAME(C(train_con_mat$overall["ACCURACY"],
    train_con_mat$byClass["SENSITIVITY"],
    train_con_mat$byClass["SPECIFICITY"]))

    COLNAMES(df.pred.training) = "LR TRAINING"
```

Model Motor and Non Motor Testing Data

```
# PREDICTION WITH TESTING DATA
    pred_test = IFELSE(PREDICT(model_full, testing[,-C(1,2,3)], TYPE = "LINK") > 1, "1", "0")
    # CLASSIFICATION ERROR
    CALC_CLASS_ERR(ACTUAL = testing$APPRDX, PREDICTED = pred_test)

## [1] 0.04648649
```

```
    # CONFUSTION MATRIX
    test_tab = TABLE(PREDICTED = pred_test, ACTUAL = testing$APPRDX)
    test_tab

##        ACTUAL
## PREDICTED   0   1
##        0 895  56
##        1  30 869

    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    df.pred.testing = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))

    COLNAMES(df.pred.testing) = "LR TESTING"

# PREDICT TESTING EVENT BL
    dwdata = testing[testing$EVENT_ID == "BL",  -C(1,2,3)]
    actl = testing[testing$EVENT_ID == "BL",  ]$APPRDX
    pred.testing.event = IFELSE(PREDICT(model_full, dwdata, TYPE = "LINK") > 1, "1", "0")

    CALC_CLASS_ERR(ACTUAL = actl, PREDICTED = pred.testing.event)

## [1] 0.05454545

    # CONFUSTION MATRIX
    test_tab_event = TABLE(PREDICTED = pred.testing.event, ACTUAL = actl)
    test_tab_event

##        ACTUAL
## PREDICTED  0  1
##        0 53  4
##        1  2 51

    test_con_mat = CONFUSIONMATRIX(test_tab_event, POSITIVE = "1")
    df.pred.event = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))

    COLNAMES(df.pred.event) = "LR EVENT BL"

KABLE(CBIND(df.pred.testing, df.pred.event), DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

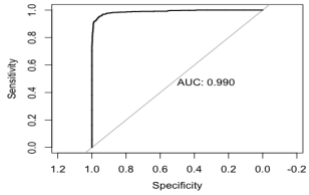
Random Forest

```
model.rf = RANDOMFOREST(APPRDX ~ ., MTRY=2, NTREE  = 1000, DATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")])

# PREDICT TRAINING
    pred.training = PREDICT(model.rf, NEWDATA = training[,-C(1,2,3)])

    train_tab = TABLE(PREDICTED = pred.training, ACTUAL = training$APPRDX)
    train_tab

##        ACTUAL
## PREDICTED    0    1
##        0 1427    0
##        1    0 3699

    train_con_mat = CONFUSIONMATRIX(train_tab, POSITIVE = "1")
    df.pred.training = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    train_con_mat$byClass["SENSITIVITY"],
    train_con_mat$byClass["SPECIFICITY"]))

    COLNAMES(df.pred.training) = "RF TRAINING"


# PREDICT TESTING
    pred.testing = PREDICT(model.rf, NEWDATA = testing[,-C(1,2,3)])
    CALC_CLASS_ERR(ACTUAL = testing$APPRDX, PREDICTED = pred.testing)

## [1] 0.05621622
```

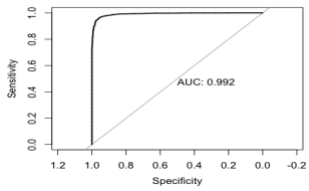
```
    # CONFUSTION MATRIX
    test_tab = TABLE(PREDICTED = pred.testing, ACTUAL = testing$APPRDX)
    test_tab

##        ACTUAL
## PREDICTED   0   1
##        0 850  29
##        1  75 896

    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    df.pred.testing = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))

    COLNAMES(df.pred.testing) = "RF TESTING"


# PREDICT EVENT BL
    dwdata = testing[testing$EVENT_ID == "BL",  -C(1,2,3)]
    actl = testing[testing$EVENT_ID == "BL",  ]$APPRDX
    pred.testing.event = PREDICT(model.rf, NEWDATA = dwdata)
    CALC_CLASS_ERR(ACTUAL = actl, PREDICTED = pred.testing.event)

## [1] 0.03636364

    ## CONFUSION MATRIX
    test_tab_event = TABLE(PREDICTED = pred.testing.event, ACTUAL = actl)
    test_tab_event

##        ACTUAL
## PREDICTED  0  1
##        0 52  1
##        1  3 54

    test_con_mat = CONFUSIONMATRIX(test_tab_event, POSITIVE = "1")
    df.pred.event = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.event) = "RF EVENT BL"

## SUMMARY
KABLE(CBIND(df.pred.testing, df.pred.event), DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)
```

Support Vector Machine
ROC Curve

```
## TESTING
test_prob = PREDICT(model_full, NEWDATA = testing[,C("APPRDX", "MOTOR", "NON.MOTOR")], TYPE = "RESPONSE")
test_roc = ROC(testing$APPRDX ~ test_prob, PLOT = TRUE, PRINT.AUC = TRUE)

## SETTING LEVELS: CONTROL = 0, CASE = 1

## SETTING DIRECTION: CONTROLS < CASES
```




```
## TRAINING
train_prob = PREDICT(model_full, NEWDATA = training[,C("APPRDX", "MOTOR", "NON.MOTOR")], TYPE = "RESPONSE")
train_roc = ROC(training$APPRDX ~ train_prob, PLOT = TRUE, PRINT.AUC = TRUE)

## SETTING LEVELS: CONTROL = 0, CASE = 1
## SETTING DIRECTION: CONTROLS < CASES
```



Support Vector Machine

Tuning Linear Cost function

```
# 5 K-FOLD ACCURACY
SET.SEED(1968)
folds = CREATEFOLDS(training$APPRDX, K=5)
missclass = DATA.FRAME()
temp = VECTOR()
list.svm.lin = LIST()
f = 1
k = 1
kernels = C("LINEAR", "POLY")
cost.f = C(.1, 1, 10, 100, 500, 1000)
#DEGREE =  C(1,2)
i = 1
FOR (k IN 1:1) {
#FOR (D IN 1:LENGTH(DEGREE)){
FOR (cst IN 1:LENGTH(cost.f)){
FOR (f IN 1:5){
    x = folds[[f]]
    training_fold = training[-x,]
    test_fold = training[x,]
    krnl <<- kernels[[k]]
    classifier = SVM(APPRDX ~ MOTOR + NON.MOTOR,
                DATA = training_fold[,-C(1,3)],
                KERNEL = krnl,
                COST = cost.f[cst])
    y_pred = PREDICT(classifier, NEWDATA = test_fold[, -C(1,2,3)])
    cm = TABLE(test_fold$APPRDX, y_pred)
    #PRINT(CM)
    acc.temp = (cm[2,1] + cm[1,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
    missclass[f,k] = acc.temp
    COLNAMES(missclass)[k] = kernels[k]

}
    missclass$cost = cost.f[cst]
   # MISSCLASS$GAMMA = DEGREE[D]
missclassification = RBIND(missclass,APPLY(missclass, 2, mean))
ROWNAMES(missclassification) = C("FOLD 1", "FOLD 2", "FOLD 3", "FOLD 4", "FOLD 5", "MEAN")
list.svm.lin[[i]]  = missclassification
i = i+1
PRINT(i)
}


}#}

KABLE(list.svm.lin[[3]], DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

WRITE.XLSX(list.svm.lin[[3]], PASTE0(p_images, "PRED.SVM.CV.LIN.XLSX"))
```

Tuning Polynomial

```
# 5 K-FOLD ACCURACY
SET.SEED(1968)
folds = CREATEFOLDS(training$APPRDX, K=5)
missclass = DATA.FRAME()
temp = VECTOR()
list.svm.poly = LIST()
f = 1
k = 1
kernels = C("POLY")
cost.f = C(1, 10, 40)
degree =  C(2,3)
```

```
i = 1
FOR (k IN 1:1) {
FOR (d IN 1:LENGTH(degree)){
FOR (cst IN 1:LENGTH(cost.f)){
FOR (f IN 1:5){
    x = folds[[f]]
    training_fold = training[-x,]
    test_fold = training[x,]
    krnl <<- kernels[[k]]
    classifier = SVM(APPRDX ~ MOTOR + NON.MOTOR,
                DATA = training_fold[,-C(1,3)],
                KERNEL = krnl,
                COST = cost.f[cst],
                DEGREE = degree[d])
    y_pred = PREDICT(classifier, NEWDATA = test_fold[, -C(1,2,3)])
    cm = TABLE(test_fold$APPRDX, y_pred)
    #PRINT(CM)
    acc.temp = (cm[2,1] + cm[1,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
    missclass[f,k] = acc.temp
    COLNAMES(missclass)[k] = kernels[k]

}
    missclass$cost = cost.f[cst]
    missclass$degree = degree[d]
missclassification = RBIND(missclass,APPLY(missclass, 2, mean))
ROWNAMES(missclassification) = C("FOLD 1", "FOLD 2", "FOLD 3", "FOLD 4", "FOLD 5", "MEAN")
list.svm.poly[[i]]  = missclassification
i = i+1
PRINT(i)
}


}}

KABLE(list.svm.poly[[5]], DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%


WRITE.XLSX(list.svm.poly[[5]], PASTE0(p_images, "PRED.SVM.CV.POLY.XLSX"))
```

Tuning Radial

```
# 5 K-FOLD ACCURACY
SET.SEED(1968)
folds = CREATEFOLDS(training$APPRDX, K=5)
missclass = DATA.FRAME()
temp = VECTOR()
list.svm.rad = LIST()
f = 1
k = 1
kernels = C("RADIAL", "POLY")#C("RADIAL", "LINEAR", "POLY")
cost.f = C(.1, 1, 10, 100)
gma =  C(.01, .1, 1)
i = 1
FOR (k IN 1:1) {
FOR (g IN 1:LENGTH(gma)){
FOR (cst IN 1:LENGTH(cost.f)){
FOR (f IN 1:5){
    x = folds[[f]]
    training_fold = training[-x,]
    test_fold = training[x,]
    krnl <<- kernels[[k]]
    classifier = SVM(APPRDX ~ MOTOR + NON.MOTOR,
                DATA = training_fold[,-C(1,3)],
                KERNEL = krnl,
                COST = cost.f[cst],
                GAMMA = gma[g])
    y_pred = PREDICT(classifier, NEWDATA = test_fold[, -C(1,2,3)])
    cm = TABLE(test_fold$APPRDX, y_pred)
    #PRINT(CM)
    acc.temp = (cm[2,1] + cm[1,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
    missclass[f,k] = acc.temp
    COLNAMES(missclass)[k] = kernels[k]
```

```
}
    missclass$cost = cost.f[cst]
    missclass$gamma = gma[g]
missclassification = RBIND(missclass,APPLY(missclass, 2, mean))
ROWNAMES(missclassification) = C("FOLD 1", "FOLD 2", "FOLD 3", "FOLD 4", "FOLD 5", "MEAN")
list.svm.rad[[i]]  = missclassification
i = i+1
PRINT(i)
}


}}

KABLE(list.svm.rad[[7]], DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

# # FIND  LOWEST MISSCLASSIFICATION
# DF = DATA.FRAME()
# FOR (I IN 1:12){
#      DF = RBIND(DF, DATA.FRAME(LIST.SVM.RAD[I]))
# }
# DF[GREPL("MEAN", ROWNAMES(DF)) == TRUE, ]

WRITE.XLSX(list.svm.rad[[7]], PASTE0(p_images, "PRED.SVM.CV.RAD.XLSX"))

svmfit.lin =SVM(APPRDX ~  NON.MOTOR + MOTOR, DATA=training[,-C(1,3)] ,
         KERNEL ="LINEAR", COST =10, SCALE =FALSE)
svmfit.pol =SVM(APPRDX ~  NON.MOTOR + MOTOR, DATA=training[,-C(1,3)] ,
         KERNEL ="POLY", COST =10, DEGREE = 3, SCALE =FALSE)
svmfit.rad =SVM(APPRDX ~  NON.MOTOR + MOTOR, DATA=training[,-C(1,3)] ,
         KERNEL ="RADIAL", COST =1, GAMMA = 1, SCALE =FALSE)


## PLOTS
mycols <- COLORS()[C(2,6)]


# LINEAR
IMAGE.SAVE(MYPLOT = PLOT(svmfit.lin, testing[,-C(1,3)], COL = mycols), PIC.NAME = "PRED.SVM.LIN", SCL = 4)

# POLYNOMIAL
IMAGE.SAVE(MYPLOT = PLOT(svmfit.pol, testing[,-C(1,3)], COL = mycols), PIC.NAME = "PRED.SVM.POL", SCL = 4)

# RADIAL
IMAGE.SAVE(MYPLOT = PLOT(svmfit.rad , testing[,-C(1,3)], COL = mycols),PIC.NAME = "PRED.SVM.RAD", SCL = 4)

# PREDICT TESTING LINEAR
pred.testing.lin = PREDICT(svmfit.lin, NEWDATA  = testing[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    test_tab = TABLE(PREDICTED = pred.testing.lin, ACTUAL = testing$APPRDX)
    test_tab

##         ACTUAL
## PREDICTED   0   1
##         0 865  30
##         1  60 895

    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    df.pred.testing.lin = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.testing.lin) = "SVM TESTING KERNEL - LINEAR"


# PREDICT TESTING POLY
pred.testing.pol = PREDICT(svmfit.pol, NEWDATA  = testing[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    test_tab = TABLE(PREDICTED = pred.testing.pol, ACTUAL = testing$APPRDX)
    test_tab

##         ACTUAL
## PREDICTED   0   1
##         0 866  31
##         1  59 894
```

```
    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    df.pred.testing.pol = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.testing.pol) = "SVM TESTING KERNEL - POLYNOMIAL"



## PREDICET TESTING RADIAL
 pred.testing.rad = PREDICT(svmfit.rad, NEWDATA  = testing[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    test_tab = TABLE(PREDICTED = pred.testing.rad, ACTUAL = testing$APPRDX)
    test_tab

##         ACTUAL
## PREDICTED   0   1
##         0 865  31
##         1  60 894

    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    df.pred.testing.rad = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.testing.rad) = "SVM TESTING KERNEL - RADIAL"

## SUMMARY
    df.test.lin.rad = CBIND(df.pred.testing.lin, df.pred.testing.pol,  df.pred.testing.rad)
    KABLE(df.test.lin.rad, DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

event.data = SUBSET(training, EVENT_ID == "BL")

# PREDICT INITIAL  VISIT LINDEAR
pred.training.lin = PREDICT(svmfit.lin, NEWDATA  = event.data[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    train_tab = TABLE(PREDICTED = pred.training.lin, ACTUAL = event.data$APPRDX)
    train_tab

##         ACTUAL
## PREDICTED   0   1
##         0  87  16
##         1   5 218

    train_con_mat = CONFUSIONMATRIX(train_tab, POSITIVE = "1")
    df.pred.event.lin = DATA.FRAME(C(train_con_mat$overall["ACCURACY"],
    train_con_mat$byClass["SENSITIVITY"],
    train_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.event.lin) = "SVM EVENT BL KERNEL - LINEAR"

# PREDICT INITIAL VISIT POLY
pred.training.pol = PREDICT(svmfit.pol, NEWDATA  = event.data[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    test_tab = TABLE(PREDICTED = pred.training.pol, ACTUAL = event.data$APPRDX)
    test_tab

##         ACTUAL
## PREDICTED   0   1
##         0  88  14
##         1   4 220

    test_con_mat = CONFUSIONMATRIX(test_tab, POSITIVE = "1")
    pred.training.pol = DATA.FRAME(C(test_con_mat$overall["ACCURACY"],
    test_con_mat$byClass["SENSITIVITY"],
    test_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(pred.training.pol) = "SVM EVENT BL KERNEL - POLYNOMIAL"


## RADIAL INITIAL VISIT
 pred.training.rad = PREDICT(svmfit.rad, NEWDATA = event.data[,C("APPRDX", "MOTOR", "NON.MOTOR")])

    train_tab = TABLE(PREDICTED = pred.training.rad, ACTUAL = event.data$APPRDX)
    train_tab
```

```
##          ACTUAL
## PREDICTED   0   1
##         0  88  15
##         1   4 219

    train_con_mat = CONFUSIONMATRIX(train_tab, POSITIVE = "1")
    df.pred.event.rad = DATA.FRAME(C(train_con_mat$overall["ACCURACY"],
    train_con_mat$byClass["SENSITIVITY"],
    train_con_mat$byClass["SPECIFICITY"]))
    COLNAMES(df.pred.event.rad) = "SVM EVENT BL KERNEL - RADIAL"

# SUMMARY
    df.event.lin.rad = CBIND(df.pred.event.lin,  pred.training.pol, df.pred.event.rad)
    KABLE(df.event.lin.rad, DIGITS = 8, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

# LINEAR
IMAGE.SAVE(MYPLOT = PLOT(svmfit.lin, event.data[,-C(1,3)], COL = mycols), PIC.NAME = "PRED.SVM.BL.LIN", SCL = 4)

# POLYNOMIAL
IMAGE.SAVE(MYPLOT = PLOT(svmfit.pol, event.data[,-C(1,3)], COL = mycols), PIC.NAME = "PRED.SVM.BL.POL", SCL = 4)

# RADIAL
IMAGE.SAVE(MYPLOT = PLOT(svmfit.rad , event.data[,-C(1,3)], COL = mycols),PIC.NAME = "PRED.SVM.BL.RAD", SCL = 4)

df.svm = CBIND(df.test.lin.rad, df.event.lin.rad)
KABLE(df.svm, DIGITS = 4, TABLE.ATTR = "STYLE = \"COLOR: BLACK;\"")  %>%
    KABLE_STYLING(FULL_WIDTH = F)

WRITE.XLSX(df.svm, PASTE0(p_images, "PRED.SVM.XLSX"))
```

B6 Defined R-Functions

```
LIBRARY(lubridate)
LIBRARY(knitr)
LIBRARY(kableExtra)
LIBRARY(devtools)
LIBRARY(ggplot2)
LIBRARY(ggrepel)
LIBRARY(gridExtra)
LIBRARY(tibble)
LIBRARY(pander)
LIBRARY(lubridate)
LIBRARY(dplyr)
LIBRARY(caret)
LIBRARY(randomForest)
LIBRARY(e1071)
LIBRARY(stats)
LIBRARY(cluster)
LIBRARY(factoextra)
LIBRARY(tidyverse)
LIBRARY(purrr)
LIBRARY(NMF)
LIBRARY(ggfortify)
LIBRARY(latex2exp)
LIBRARY(rattle)
LIBRARY(pROC)
LIBRARY(lattice)
LIBRARY(Hmisc)
LIBRARY(PerformanceAnalytics)
LIBRARY(mclust)
LIBRARY(VIM)
LIBRARY(brew)
LIBRARY(RColorBrewer)
LIBRARY(wesanderson)
LIBRARY(ellipse)
LIBRARY(boot)
LIBRARY(openxlsx)


rpath = "/USERS/JOERGHEINTZ/BOX/02_DATA_ANALYTICS_PROJECTS/001_MASTERTHESIS/"
```

```
 p_images = "/USERS/JOERGHEINTZ/BOX/02_DATA_ANALYTICS_PROJECTS/001_MASTERTHESIS/01_DATA/01_THESIS/02_IMAGES/"
all.visits = C("BL", "V01", "V02", "V03", "V04", "V05", "V06", "V07", "V08", "V09", "V10", "V11",
               "V12", "V13", "V14", "V15")


# TAKES VECTOR WITH SUBJECTS GROUPS SEE DOC "VARIABLE DEFINITIONS AND SCORE CALCULATIONS"
get_enrolled = FUNCTION(SBTS = C(1,2,4)){

   path_randomization_table = PASTE0(rpath, "01_DATA/DATA_LONI/05_STUDY_ENROLLMENT/")
   path_demographics = PASTE0(rpath, "01_DATA/DATA_LONI/02_SUBJECT_CHARACTERISTICS/")

   # ENROLLED SUBJECTS
   df_ran  = READ.CSV(PASTE0(path_randomization_table, "RANDOMIZATION_TABLE.CSV"))

   # LOAD SCREEN
   df_scr  = READ.CSV(PASTE0(path_demographics, "SCREENING___DEMOGRAPHICS.CSV"))

   # MERGE
   df_temp = MERGE(df_ran, df_scr, BY = 'PATNO')[, C("PATNO", "APPRDX", "ENROLLDT")]
   df_enrolled = df_temp[df_temp$APPRDX %IN% sbts, ]

   df_enrolled$ENROLLDT = PARSE_DATE_TIME(df_enrolled$ENROLLDT, ORDERS = C("MY"))
   df_enrolled


}

cbind.fill <- FUNCTION(...){
       nm <- LIST(...)
       nm <- LAPPLY(nm, as.matrix)
       n <- MAX(SAPPLY(nm, nrow))
       DO.CALL(cbind, LAPPLY(nm, FUNCTION (x)
             RBIND(x, MATRIX(n-NROW(x), NCOL(x)))))
}

####### PLOTTING FUNCTION #########

# LINE DIAGRAM
p.line = FUNCTION(
  # DATA, X, Y, .. VALUES
  DATA = df_HVLT_summary, X = "FACTOR(EVENT_ID)", Y = "MEAN.TOTAL.RECALL",
  COLOUR = "FACTOR(APPRDX)", GROUP = "FACTOR(APPRDX)",
  SMPLS.VST = 15,  # SAMPLE SIZSE PER VISIT, MARKS GREY IF # OF VISITS IS BELOW S

  ## AXIS, LEGEND, LABELS
  XTEXT = "VISITS, DELTA = 3 MNT",
  YTEXT = "TEXT",
  XTITLE = TRUE,
  NM = "XX PER VISIT",
  LBLS = C("PARKINSON'S","CONTROLS / HEALTHY"),
  LEGEND_POS = C(0.9, 0.9))
{

  GGPLOT(DATA = data, AES_STRING(X = x, Y = y, COLOUR = colour, GROUP = group))+
   GEOM_POINT() + GEOM_LINE() +
   GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")), HJUST= -.5,VJUST=.2, SIZE = 2,
                ALPHA = 0.8, SEGMENT.SIZE = 0) +
   GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n<smpls.vst, n,"")), HJUST= -.5,VJUST=-.4, SIZE = 2,
                SEGMENT.SIZE = 0, COLOR = "WHITE",
                FILL ="DARKGREY") +

   # LEGEND, AXIX
   LIST(IF(xtitle == FALSE) THEME(LEGEND.POSITION=legend_pos, AXIS.TITLE.X = ELEMENT_BLANK())
       ELSE THEME(LEGEND.POSITION=legend_pos)) +
   SCALE_COLOR_DISCRETE(NAME = nm, LABELS = lbls) +
   XLAB(xtext) +
   YLAB(ytext)
}

# HISTOGRAM
p.hist = FUNCTION(
   DATA = df_HVLT_summary, X = "MEAN.TOTAL.RECALL", FILL = "FACTOR(APPRDX)",
   BNS = 50,  LPH = 0.5, XLIMIT = FALSE,
   NM = "MEAN SCORES OF VISITS",
```

```
  LBLS = C("PARKINSON'S","CONTROLS / HEALTHY"),
  XTEXT = "SCORES",
  YTEXT = "COUNTS",
  LGND = TRUE,
  LEGEND_POS = C(0.8, 0.8),
  MN = NULL,
  MX = NULL)
{

  GGPLOT(DATA = data, AES_STRING(X = x, FILL = fill)) +
     GEOM_HISTOGRAM(ALPHA = lph, BINS = bns,  POSITION = "IDENTITY") +
  THEME(
    LEGEND.TEXT = ELEMENT_TEXT(COLOR = "BLACK", SIZE = 12),
    LEGEND.TITLE = ELEMENT_TEXT(SIZE =12)) +
  GUIDES(SHAPE = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = 1))) +
  GUIDES(COLOR = GUIDE_LEGEND(OVERRIDE.AES = LIST(SIZE = 1))) +
     SCALE_FILL_DISCRETE(NAME = nm, LABELS = lbls) + {IF(xlimit) COORD_CARTESIAN(XLIM = C(mn,mx))} +
     XLAB(xtext) + YLAB(ytext) +
     LIST(IF(lgnd == TRUE) THEME(LEGEND.POSITION=legend_pos)
        ELSE THEME(LEGEND.POSITION='NONE'))
}

# ERRORBARS
p.errorbar.2 = FUNCTION(
 DATA = df_HVLT_summary,
 X = "EVENT_ID",
 Y = "MEAN.DVT.RETENTION",
 YMIN = "MEAN.DVT.RETENTION - SD.DVT.RETENTION",
 YMAX = "MEAN.DVT.RETENTION + SD.DVT.RETENTION",
 COLOUR = "FACTOR(APPRDX)",
 GROUP = "FACTOR(APPRDX)",
 COLOR = "FACTOR(APPRDX)",
 XTEXT = "ASDSD",
 YTEXT = "ADADS",
 TITLE = "ASDFASDF",
 LEGEND.LABELS = C("1", "2"),
 LEGEND.TITLE = "ERRORBAR",
 LEGEND.POSITION = C(0, 0.9),
 LEGEND.TRANS = 1,
 DODGE = 0,
 XTITLE = FALSE,
 SMPLS.VST = 20,
 HJ = 0,
 VJ = 0,
 AGL = 0,
 TXTSIZE = 3
)
{
 GGPLOT(DATA = data, AES_STRING(X = x, Y = y)) +
  # POINTS AND LINE
  GEOM_POINT(POSITION=POSITION_DODGE(dodge)) + AES_STRING(COLOUR = colour) +
  GEOM_LINE(SIZE = 0.5, AES_STRING(GROUP = group), POSITION=POSITION_DODGE(dodge)) +
  # LABELS
  GEOM_TEXT(AES(LABEL = IFELSE(n>smpls.vst, PASTE0("N=",n, ", SD=", ROUND(data$sdt,1), COLOUR = "BLACK"),""),
            HJUST= hj,VJUST=vj), SIZE = txtsize, ALPHA = 1, ANGLE = agl) +

  # ERRORBAR
  GEOM_ERRORBAR(DATA = data, AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=color), WIDTH=.2, SIZE = .3,
            POSITION=POSITION_DODGE(dodge)) +

  # LEGEND & THEME & AXIS
  SCALE_COLOR_DISCRETE(NAME = legend.title, LABELS = legend.labels) +

  LIST(
   IF(xtitle == FALSE){
     THEME(LEGEND.POSITION=legend.position, AXIS.TITLE.X=ELEMENT_BLANK(),
         LEGEND.BACKGROUND = ELEMENT_RECT(FILL=ALPHA('LIGHTGREY', legend.trans)),
         PLOT.TITLE = ELEMENT_TEXT(HJUST=0,VJUST= 0))
   } ELSE
     THEME(LEGEND.POSITION=legend.position, PLOT.TITLE = ELEMENT_TEXT(HJUST=0,VJUST= 0),
         LEGEND.BACKGROUND = ELEMENT_RECT(FILL=ALPHA('LIGHTGREY', legend.trans)))) +
  XLAB(xtext) +
  YLAB(ytext) +
  GGTITLE(title)
```

```
}



p.errorbar = FUNCTION(
  DATA = df_HVLT_summary,
  X = "EVENT_ID",
  Y = "MEAN.DVT.RETENTION",
  YMIN = "MEAN.DVT.RETENTION - SD.DVT.RETENTION",
  YMAX = "MEAN.DVT.RETENTION + SD.DVT.RETENTION",
  COLOUR = "FACTOR(APPRDX)",
  GROUP = "FACTOR(APPRDX)",
  COLOR = "FACTOR(APPRDX)",
  XTEXT = "ASDSD",
  YTEXT = "ADADS",
  TITLE = "ASDFASDF",
  LEGEND.LABELS = C("1", "2"),
  LEGEND.TITLE = "ERRORBAR",
  LEGEND.POSITION = C(0, 0.9),
  LEGEND.TRANS = 1,
  DODGE = 0,
  XTITLE = FALSE,
  SMPLS.VST = 20,
  HJ = 0,
  VJ = 0,
  AGL = 0
)
{
  GGPLOT(DATA = data, AES_STRING(X = x, Y = y)) +
    # POINTS AND LINE
    GEOM_POINT(POSITION=POSITION_DODGE(dodge)) + AES_STRING(COLOUR = colour) +
    GEOM_LINE(SIZE = 0.5, AES_STRING(GROUP = group), POSITION=POSITION_DODGE(dodge)) +
    # LABELS
    GEOM_LABEL_REPEL(AES(LABEL = IFELSE(n>smpls.vst, n,"")),
                     HJUST= -.5,VJUST=.2), SIZE = 2,
            ALPHA = 0.8, ANGLE = agl) +

    # ERRORBAR
    GEOM_ERRORBAR(DATA = data, AES_STRING(YMIN=ymin, YMAX=ymax, COLOR=color), WIDTH=1.5, SIZE = 1,
              POSITION=POSITION_DODGE(dodge)) +

    # LEGEND & THEME & AXIS
    SCALE_COLOR_DISCRETE(NAME = legend.title, LABELS = legend.labels) +
    LIST(
      IF(xtitle == FALSE){
        THEME(LEGEND.POSITION=legend.position, AXIS.TITLE.X=ELEMENT_BLANK(),
            LEGEND.BACKGROUND = ELEMENT_RECT(FILL=ALPHA('LIGHTGREY', legend.trans)),
            PLOT.TITLE = ELEMENT_TEXT(HJUST=0,VJUST= 0))
      } ELSE
        THEME(LEGEND.POSITION=legend.position, PLOT.TITLE = ELEMENT_TEXT(HJUST=0,VJUST= 0),
            LEGEND.BACKGROUND = ELEMENT_RECT(FILL=ALPHA('LIGHTGREY', legend.trans)))) +
    XLAB(xtext) +
    YLAB(ytext) +
    GGTITLE(title)
}


pd.controls.per.visit <- FUNCTION(data){
    ba = df %>% GROUP_BY(EVENT_ID, APPRDX) %>% SUMMARISE(N = LENGTH(APPRDX))
    GGPLOT(DATA = ba, AES(X = EVENT_ID, Y = n, FILL = AS.FACTOR(APPRDX))) +
        GEOM_BAR(STAT = "IDENTITY", POSITION="DODGE") +
        XLAB("VISIT") +
        YLAB("NUMBER OF SUBJECTS") +
        SCALE_FILL_DISCRETE(NAME = "", LABELS = lbls)
}

## IMPUTE DATA RANDOM
### USES DATA AVAILABLE AND RANDOMLY IMPUTES WITH THOSE DATA NA
random.imp <- FUNCTION (a){
    missing <- IS.NA(a)
    n.missing <- SUM(missing)
    a.obs <- a[!missing]
    imputed <- a
    imputed[missing] <- SAMPLE(a.obs, n.missing, REPLACE=TRUE)
```

```
  RETURN (imputed)
}

## STRTABLE
#' CREATES A \CODE{DATA.FRAME} VERSION OF THE STR FUNCTION FOR DATA.FRAMES.
#'
#' NOTE THAT THIS FUNCTION ONLY WORKS WITH \CODE{DATA.FRAMES}. THE FUNCTION
#' WILL THROW AN ERROR FOR ANY OTHER OBJECT TYPES.
#'
#' @PARAM N THE FIRST N ELEMENT TO SHOW
#' @PARAM WIDTH MAXIMUM WIDTH IN CHARACTERS FOR THE EXAMPLES TO SHOW
#' @PARAM N.LEVELS THE FIRST N LEVELS OF A FACTOR TO SHOW.
#' @PARAM WIDTH.LEVELS MAXIMUM WIDTH IN CHARACTERS FOR THE NUMBER OF LEVELS TO SHOW.
#' @PARAM FACTOR.VALUES FUNCTION DEFINING HOW FACTOR EXAMPLES SHOULD BE PRINTED.
#'       POSSIBLE VALUES ARE \CODE{AS.CHARACTER} OR \CODE{AS.INTEGER}.
#' @EXPORT
#' @EXAMPLES
#' DATA(IRIS)
#' STR(IRIS)
#' STRTABLE(IRIS)
#' STRTABLE(IRIS, FACTOR.VALUES=AS.INTEGER)
strtable <- FUNCTION(df, N=4, WIDTH=60,
                N.LEVELS=n, WIDTH.LEVELS=width,
                FACTOR.VALUES=as.character) {
  STOPIFNOT(IS.DATA.FRAME(df))
  tab <- DATA.FRAME(VARIABLE=NAMES(df),
                CLASS=REP(AS.CHARACTER(NA), NCOL(df)),
                LEVELS=REP(AS.CHARACTER(NA), NCOL(df)),
                EXAMPLES=REP(AS.CHARACTER(NA), NCOL(df)),
                STRINGSASFACTORS=FALSE)
  collapse.values <- FUNCTION(col, n, width) {
    result <- NA
    FOR(j IN 1:MIN(n, LENGTH(col))) {
      el <- IFELSE(IS.NUMERIC(col),
                PASTE0(col[1:j], COLLAPSE=', '),
                PASTE0("", col[1:i], "", COLLAPSE=', '))
      IF(NCHAR(el) <= width) {
        result <- el
      } ELSE {
        BREAK
      }
    }
    IF(LENGTH(col) > n) {
      RETURN(PASTE0(result, ', ...'))
    } ELSE {
      RETURN(result)
    }
  }

  FOR(i IN SEQ_ALONG(df)) {
    IF(IS.FACTOR(df[,i])) {
      tab[i,]$class <- PASTE0('FACTOR W/ ', NLEVELS(df[,i]), ' LEVELS')
      tab[i,]$levels <- COLLAPSE.VALUES(LEVELS(df[,i]), N=n.levels, WIDTH=width.levels)
      tab[i,]$examples <- COLLAPSE.VALUES(FACTOR.VALUES(df[,i]), N=n, WIDTH=width)
    } ELSE {
      tab[i,]$class <- CLASS(df[,i])[1]
      tab[i,]$examples <- COLLAPSE.VALUES(df[,i], N=n, WIDTH=width)
    }

  }

  CLASS(tab) <- C('STRTABLE', 'DATA.FRAME')
  RETURN(tab)
}

#' PRINTS THE RESULTS OF \CODE{\LINK{STRTABLE}}.
#' @PARAM X RESULT OF CODE \CODE{\LINK{STRTABLE}}.
#' @PARAM ... OTHER PARAMETERS PASSED TO \CODE{\LINK{PRINT.DATA.FRAME}}.
#' @EXPORT
print.strtable <- FUNCTION(x, ...) {
  NEXTMETHOD(x, ROW.NAMES=FALSE, ...)
}
## SCALING MIN MAX
min.max.scaling = FUNCTION(mydata){
```

```
  mydata = AS.DATA.FRAME(APPLY(mydata, 2, FUNCTION(x) {
    (x - MIN(x)) / (MAX(x)-MIN(x))

  }))
  mydata
}
normalize <- FUNCTION(x)
{
      RETURN((x- MIN(x)) /(MAX(x)-MIN(x)))
}
get_mydf = FUNCTION(){
  mydf = READ.CSV(PASTE0(rpath, "I.PARKINSONS.CSV"))
  mydf = SUBSET(mydf,EVENT_ID != "SC" & EVENT_ID != "PW" & EVENT_ID != "ST" & EVENT_ID != "U01")
  motor = C("UPDRS.1", "UPDRS.2", "UPDRS.3")
  mydf$mds_updrs_total = ROWSUMS(mydf[, motor])
  filter = C("INFODT", "SYSSUP", "SYSSTND", "DIASUP", "DIASTND", "DVS_SFTANIM", "JLO_TOTRAW",
             "UPDRS.1", "UPDRS.2", "UPDRS.3", "UPDRS.4")
  mydf = SUBSET(mydf, SELECT = !NAMES(mydf) %IN% filter)
  mydf = mydf[COMPLETE.CASES(mydf), ]
  mydf}

## GET BALANCED DATA SET (PD AND HC)
get_mydf.balanced = FUNCTION(MYSETSEED = 1, NORM = "M"){

  # DATA INGESTION
  df = GET_MYDF()
  IF (norm == "M") df[,-C(1,2,3)] = MIN.MAX.SCALING(df[,-C(1,2,3)])
  IF (norm == "Z") df[,-C(1,2,3)] = SCALE(df[,-C(1,2,3)])

  # SUMMARIZE DATA BY PATIENT
  dfsum = df[,-C(2, 3)] %>%
    GROUP_BY(PATNO) %>%
    SUMMARISE_ALL(LIST(mean, sd)
    )
  # INCLUDE LABELS PD, HC IN SUMMARIZED DATA SET
  APPRDX = df$APPRDX[!DUPLICATED(df$PATNO)]
  dfsum$APPRDX = APPRDX

  # COLUMNAMING
  mynames = NAMES(dfsum)
  mynames = SUB("FN1", "MEAN", mynames)
  mynames = SUB("FN2", "SD", mynames)
  NAMES(dfsum) = mynames
  dfsum = dfsum[, ORDER(NAMES(dfsum))]
  dfsum = dfsum[COMPLETE.CASES(dfsum), ]

  # ORDER COLUMN NAMES
  nameorder = NAMES(dfsum)
  nameorder = C("PATNO", "APPRDX", nameorder[!(nameorder %IN% C("PATNO", "APPRDX"))])
  dfsum =  dfsum[, nameorder]

  # BALANCED DATA SET
  pd = dfsum[dfsum$APPRDX == 2, ]
  hc = dfsum[dfsum$APPRDX == 1, ]

  # DRAW SAMPLE FROM THE
  SET.SEED(mysetseed)
  index = SAMPLE(LENGTH(hc$APPRDX), LENGTH(pd$APPRDX))
  df.sum.balanced = RBIND(hc[index, ], pd)
  df.sum.balanced[ORDER(df.sum.balanced$PATNO), ]
}

image.save = FUNCTION(myplot, PTH = p_images, PIC.NAME = "DT", SCL = 4, HGT = 1, WDTH =1.75){
      IF (pic.name == "DT") pic.name = SYS.TIME()
      GGSAVE(PASTE0(p_images, pic.name, ".JPG"),
            DPI = 300,
            HEIGHT = hgt,
            WIDTH = wdth,
            SCALE  = scl,
            PLOT= myplot)
}
```

VITA

EDUCATION

Dipl. Ing. Electrical Engineering	University of Applied Sciences at Kaiserslautern, GER

CERTIFICATIONS	PMP Project Management Professional CISA Certified Information System Auditor ITIL Foundation Certificate in IT Services

PROFESSIONAL EXPERIENCES

I have a proven track record in developing and delivering data science, and IT transformation projects aimed to align both the decision processes and the IT infrastructure services with business needs. I have profound knowledge and experience in designing enterprise architectures and implementing IT service and delivery processes. My past clients include Fortune 50 companies, banks, healthcare providers, reinsurers, and automotive manufacturers. I served as executive manager and management consultant for IBM, Swisscom, and 1&1 Internet AG. In the academic environment I was instrumental as Chief Security Architect at the National Center for Supercomputing Applications (NCSA), and Associate Director at the NIH BD2K Center for Large Genomic Data before I took on the Senior Program Management Position at the College of Engineering to develop and deliver health data analytics projects to advance clinical and medical research.

PROFESSIONAL SKILLSET

- Program Management: Managing the development and delivery of health data analytics projects and serving as liaison between the College of Engineering, healthcare providers, and health care industries.
- Strategy Delivery: Project development, requirement engineering, project portfolio management, infrastructure and service & delivery process Implementation
- Data Science: Machine Learning, Biostatistics, Advanced Statistical Modeling, R
- IT Governance: Policy Development, Risk Based Control Systems

EMPLOYMENT

2017 – date Program Manager Health Data Analytics Initiative, College of Engineering at UIUC
Associated Centers:
- Health Care Engineering Systems Center (HCESC)
- Center for Computational Biotechnology and Genomic Medicine (CCBGM)
- Center for AI Driven Health Data Systems and Analytics
- Cancer Center at Illinois
- OSF Innovation Infrastructure Committee, Hospital System in Peoria, IL

2016 – 2017 Head of Integrated Risk Control System, Allianz Insurance, GER
2014 – 2016 Associate Director NIH BD2K Center at UIUC, USA
2013 - 2014 NCSA Chief Security Architect at UIUC, USA
2011 - 2013 Executive Risk Manager at Swisscom IT Services, CH
2006 - 2010 Senior Management Consultant, IBM Global Technology, GER

PUBLICATIONS

Zhang, Y., Manjunath, M., Kim, Y., Heintz, J., & Song, J. S. (2018). SequencEnG: an Interactive Knowledge Base of Sequencing Techniques. BioRxiv, 319079. http://doi.org/10.1101/319079

Shoaib, T., Heintz, J., Lopez-Berganza, J. A., Muro-Barrios, R., Egner, S. A., & Espinosa-Marzal, R. M. (2018). Stick-Slip Friction Reveals Hydrogel Lubrication Mechanisms. Langmuir. http://doi.org/10.1021/acs.langmuir.7b02834