What's up with the RDW?

Exploring an Overlooked Lab Value with Statistics & Machine Learning

Final Report for CS 5266: Topics in Big Data

Authors: Davis Zhang, Sachit Bhat, Kenneth Li, & Alvin Jeffery

Background

The red cell (erythrocyte) distribution width (i.e., RDW) is a commonly-measured component of a Complete Blood Count. The Complete Blood Count includes measures of several different blood cells, including but not limited to, erythrocyte counts, hemoglobin concentrations, leukocyte counts/proportions, and platelet counts. The Complete Blood Count and frequently-accompanying Basic Metabolic Profile (which measures electrolyte concentrations) are the most commonly-performed laboratory tests among hospitalized patients, with many patients having these levels measured on a daily basis. For patients in the Intensive Care Unit (ICU), these values might be measured several times per day.

Even though the RDW is always available with the other blood counts, the RDW is typically not reviewed or considered in the typical patient. The RDW is a measure of the coefficient of variation of the mean erythrocyte cell volume in the sample. A normal values is approximately 15%, and this value is increased in situations such as anemia (i.e., blood loss), pregnancy, and recovery from acute hemorrhage. The rationale behind the increased values is that during times of low blood oxygen levels, the body increases production of erythrocytes in an attempt to increase oxygen-carrying capacity of the blood. The increased production and efficiency is associated with smaller erythrocyte sizes, which increases variation in mean cell volume.

Previous Literature

Until recently, reviewing the RDW values has been limited to situations where a health care provider is attempting to differentiate the etiology of a blood disorder (i.e., the underlying cause of a change in erythrocyte counts). To our knowledge, the first study to bring attention to RDW values as more beneficial than aiding in the differential diagnosis of blood disorders was in 2008 when Tonelli et al. found a relationship between higher RDW values and risk of death and cardiovascular events in a prospective cohort of 4,111 patients in a 5-year cholesterol study.

Danese and colleagues (2015) later suggested RDW values might be helpful in the "diagnosis and prognostication of patients with [acute coronary syndrome], ischemic cerebrovascular disease, [pulmonary artery disease], [heart failure], and [atrial fibrillation]." When examining the RDW among critical ill patients,

Bazick et al. (2011) found RDW values (when broken into 5 quantiles) measured at the time of critical care initiation to be associated with 30-day mortality and bloodstream infections in adjusted anlayses.

More recent work in a large dataset of intensive care unit patients has found RDW to be a simple alternative to more complex prognostic predictors (Hunziker at al., 2012; Hu et al., 2016). These studies used data from the MIMIC-II database, and to our knowledge, similar studies have not been conducted in the current version (MIMIC-III). In our own work, we have previously found the RDW value on hospital admission to be the most important clinical variable in predicting in-hospital cardiopumonary arrest (Jeffery et al., in-press).

While previous studies have suggested RDW might reflect inflammation (Bazick et al., 2011), oxidative stress (Bazick et al., 2011), and/or arterial underfilling (Bazick et al., 2011), it remains unclear why the RDW values are associated with a host of negative outcomes.

Objectives

Given the knowledge gap of *why* elevated RDW values are associated with poor outcomes, our goal was to identify associations between RDW values and other patient characteristics (e.g., diagnoses, laboratory values) that could assist in developing hypotheses for future basic science studies.

We specifically sought to:

1. Measure the association between RDW values and other commonly-measured laboratory values upon ICU admission
2. Identify diagnoses associated with elevated RDW values upon ICU admission
3. Measure changes in RDW values from ICU admission to ICU discharge
4. Evaluate the variable importance of RDW in predicting in-hospital mortality

Methods

We used the MIMIC-III database (Johnson et al., 2017; Johnson et al., 2016), a freely accessible critical care database, for all study aims. MIMIC was developed by the MIT Lab for Computational Physiology and comprises demographics, vital signs, laboratory values, medications, and additional information from approximately 40,000 critical care patients. The data is stored in a structure similar to a spreadsheet in which each column contains consistent information and each row contains an instantiation of that information.

Management of "Big Data" for Class

For the class project, we downloaded approximately 50 GB of data and imported the data into a Google Cloud Platform project. After downloading the necessary data that we would be working with, we simply uploaded the files into Google Cloud's Storage Bucket. To manipulate and work with the data, we used Google Datalab, an interactive tool built on Jupyter which allows for analysis of data from the cloud storage. Within Google Cloud, we created four VM compute instances under one instance parent group, which allowed for easy access to the same datalab project but as different users. This was preferable to using

Jupyter Notebooks on local machines because GBQ was closely integrated with Datalab as another GCP service, and we had sensitive data we didn't want exposed. Within the datalab, we used Ungit, a web-based git service that allowed for collaboration and changes to code.

Before finalizing our data management strategies, we also attempted to create an SQL database with Google's SQL Cloud Instance application. We started with a hard-disk drive machine to create a MySQL database, but query execution time was extremely slow. We then switched to a Solid State drive to improve speed, even though this wwas more expensive. Our disk space scaled up as storage needs increased, which was very beneficial. We eventually switched to Google Big Query because it was easier to access the data, queries executed more quickly, and the cost was reduced. We were also motivated to switch to Big Query by its seamless integration with services like Google Colaboratory and a *pandas* module for GBQ. We wanted to make sure our development interface was accessible and practical, and we specifically looked for ways to use Jupyter Notebook to process data from a database. We found that Google Colab, while a novel service, was underperforming compared to traditional notebooks. We finally settled on Google Cloud Datalab as a final alternative, which sets up Compute Engine Instances and a private notebook server.

Data Analysis

For Aim #1, we created bivariate plots with RDW values along the x-axis and other laboratory values along the y-axis for values immediately following ICU admission. We applied an alpha argument to illustrate the density of values. We also applied locally-weighted scatterplot smoothing (LOWESS) curves to identify trends in the associations. In the LOWESS plots, RDW values are on the original scale, but y-axis values have been scaled (variance = 1) and centered (mean = 0).

For Aim #2, we calculated the mean RDW value (upon ICU admission) for each unique ICD-9 diagnosis code and ranked the diagnoses in decreasing order of meand RDW value.

For Aim #3, We analyzed the RDW changes of patients on an individual basis, so we looked at every RDW time series constructed from every ICU stay recorded. We looked the net changes, variance, and autocorrelation of RDW values and visualized the spread of these metrics across the data.

For Aim #4, we created several machine learning algorithms to predict in-hospital mortality (binary outcome) based on laboratory values available on ICU admission. Model predictors included all laboratory values measured at or before the first measured RDW value. We imputed missing values with a last-one carried forward imputation (from hospital admission). For laboratory values with no measures, we conducted a median imputation.

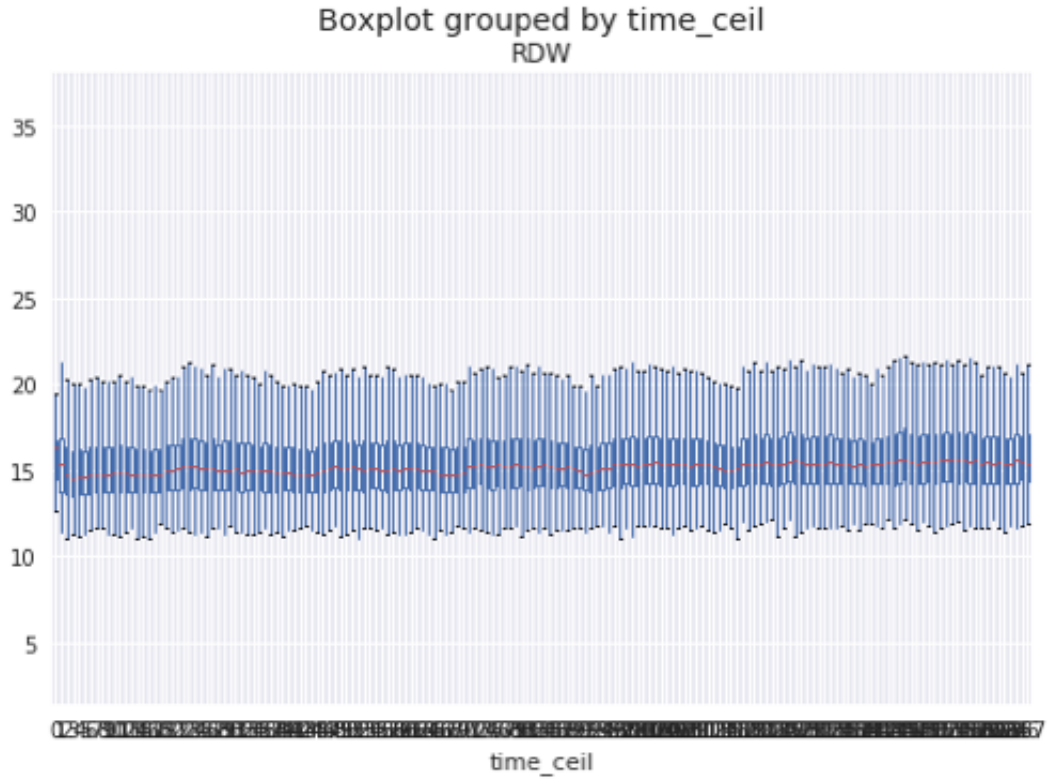
Results

Bivariate Associations with Other Lab Values

```
import pandas as pd
import numpy as np
from scipy import stats
import math
import google.datalab.bigquery as bq
import matplotlib.pyplot as plt
from sklearn import preprocessing

query = bq.Query('SELECT * '
                 'FROM `vu-bigdata.MIMIC.LAB_VIEW`;')
all_labs = query.execute().result().to_dataframe()

# review RDW values
stats.describe(all_labs.RDW, nan_policy='omit')
```

DescribeResult(nobs=685404, minmax=(masked_array(data=3.1, mask=False, fill_value=1e+20), masked_array(data=36.4, mask=False, fill_value=1e+20)), mean=15.971654498660643, variance=6.0680851872526365, skewness=masked_array(data=1.24610802,mask=False, fill_value=1e+20), kurtosis=2.3195188557420963)

```
# create "time since" ICU admission to lab value taken
mini = all_labs#[:100]
#mini.is_copy = False
mini['CHARTTIME'] = pd.to_datetime(mini['CHARTTIME'])
mini['INTIME'] = pd.to_datetime(mini['INTIME'])
mini = mini[mini['CHARTTIME'] >= mini['INTIME']]

mini.boxplot('RDW', by = 'time_ceil')
```




```
# keep first for a patient's ICU Stay
first_vals = mini.loc[mini.groupby("ICUSTAY_ID")["time_std"].idxmin()]
#first_vals = mini.sort_values("time_std").groupby("ICUSTAY_ID", as_index=
False).first()

stats.describe(first_vals['time_std'])
```

DescribeResult(nobs=59732, minmax=(0.0, 157.96972222222223), mean=5.394770865421103, variance=81.4488995040862, skewness=4.577480778948639, kurtosis=30.209671091485184)

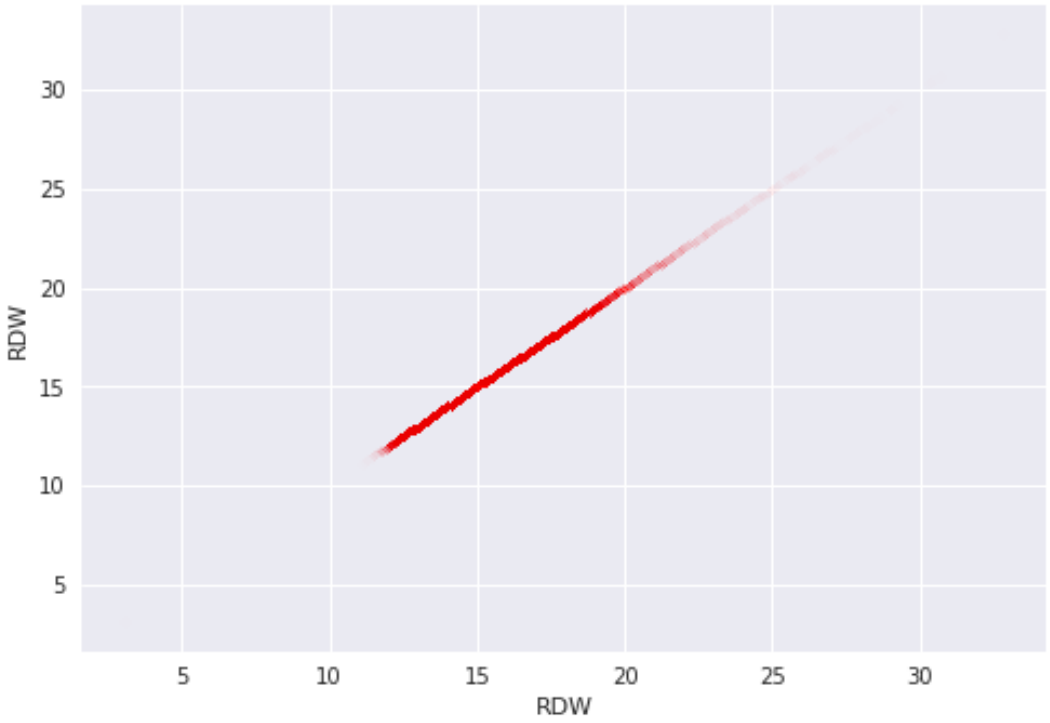
```
description = stats.describe(first_vals['RDW'], nan_policy='omit')
description
```

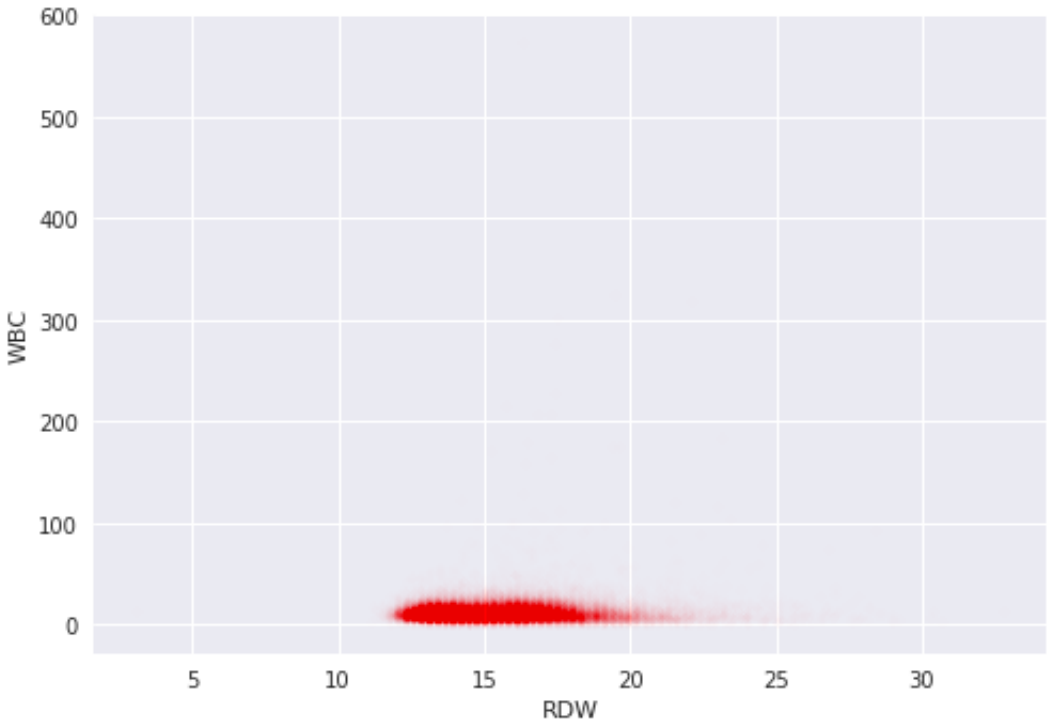
DescribeResult(nobs=37004, minmax=(masked_array(data=3.1, mask=False, fill_value=1e+20), masked_array(data=32.8, mask=False, fill_value=1e+20)), mean=15.282405956112854, variance=4.819572140035329, skewness=masked_array(data=1.25028716, mask=False, fill_value=1e+20), kurtosis=2.5886270475477406)

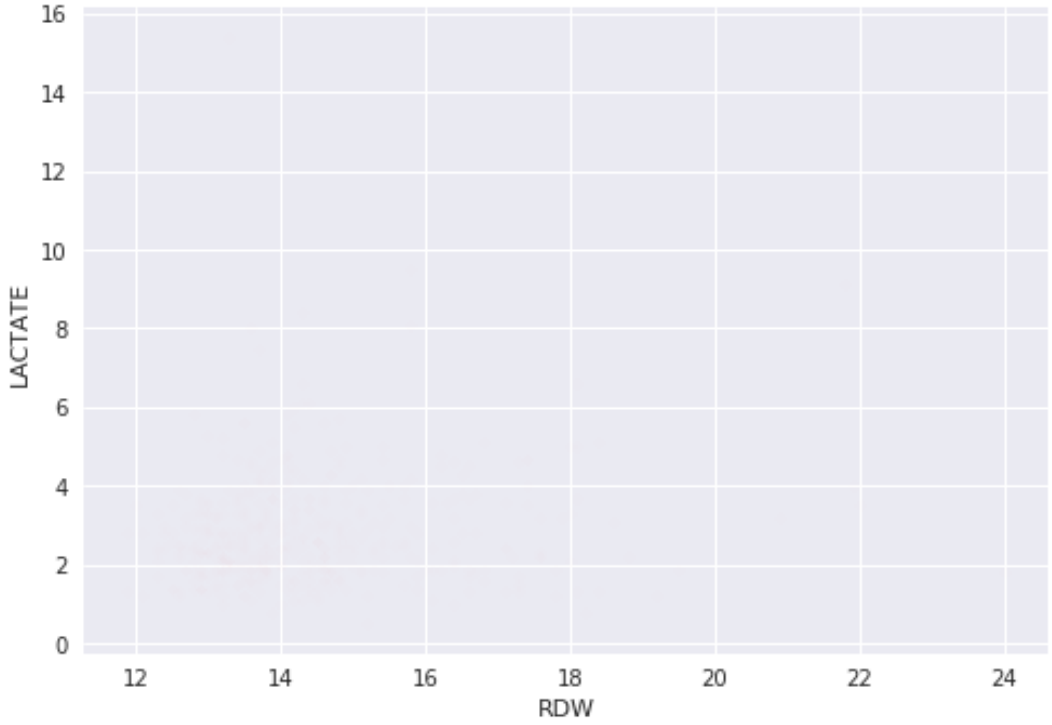
```
def scale_center(var):
  """
  Input: numeric variable
  Output: variable normalized with mean = 0 and sd = 1
  """
  descriptions = stats.describe(var, nan_policy = 'omit')
  mean = descriptions.mean
  variance = descriptions.variance
  return (var - mean)/variance**0.5

# subset to numeric values
cols = ['RDW', 'WBC', 'BANDS', 'HEMOGLOBIN', 'HEMATOCRIT', 'PLATELET', 'PT
', 'PTT',
          'INR', 'SODIUM', 'POTASSIUM', 'ANIONGAP', 'BILIRUBIN', 'BUN', 'C
REATININE',
          'CHLORIDE', 'GLUCOSE', 'LACTATE']
first_vals_std = first_vals[cols]

# normalize
first_vals_std = first_vals_std.apply(scale_center)
#stats.describe(first_vals['RDW'], nan_policy='omit')
first_vals_std.head()
```

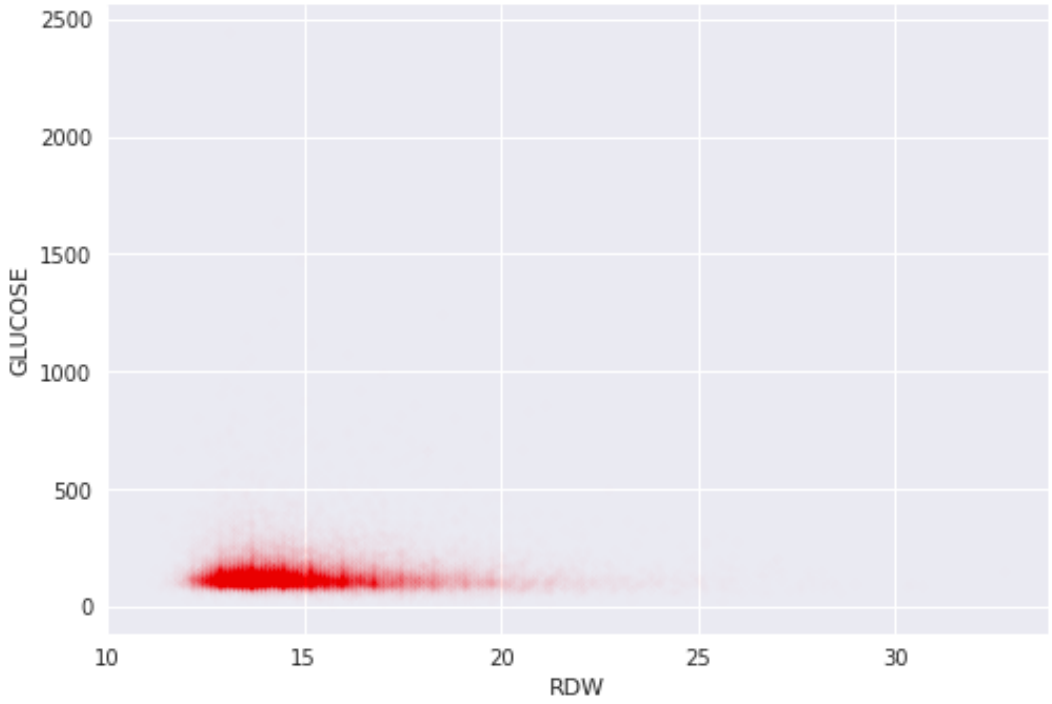
	RDW	WBC	BANDS	HEMOGLOBIN	HEMATOCRIT	PLATELET	PT	PTT	INR	SODIUM	POTASSIUM	ANIONGAP	BILIRUBIN	BUN	CREATININE	CHLORIDE	GLUCOSE	LACTATE
257358	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.538122
1337180	NaN	2.962675	NaN	NaN	0.152032	-1.030938	0.359315	-0.272808	0.247498	0.474877	-1.286072	1.692577	0.0196	-0.296714	-0.284813	0.066720	0.182878	NaN
702482	0.190217	-0.870319	NaN	-0.886298	-1.028104	-0.489583	NaN	NaN	NaN	-0.317901	-0.890686	-0.646060	NaN	-0.564935	-0.404032	-1.045461	-0.961310	NaN
670761	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.119707	-0.495300	0.653183	NaN	-0.743750	-0.404032	-0.251046	1.145035	NaN
427044	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.871471

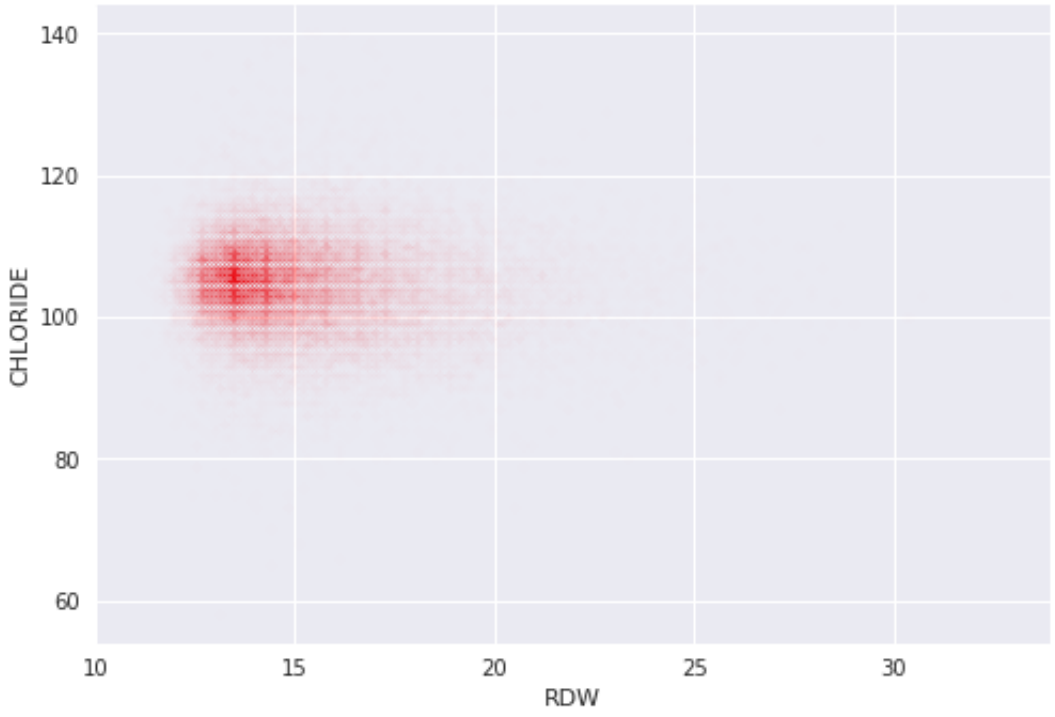
```
for y in ['RDW', 'WBC', 'BANDS', 'HEMOGLOBIN', 'HEMATOCRIT', 'PLATELET', '
PT', 'PTT',
          'INR', 'SODIUM', 'POTASSIUM', 'ANIONGAP', 'BILIRUBIN', 'BUN', 'C
REATININE',
          'CHLORIDE', 'GLUCOSE', 'LACTATE']:
  first_vals.plot(kind='scatter', x='RDW', y=y, alpha=5e-3, color='r')
```

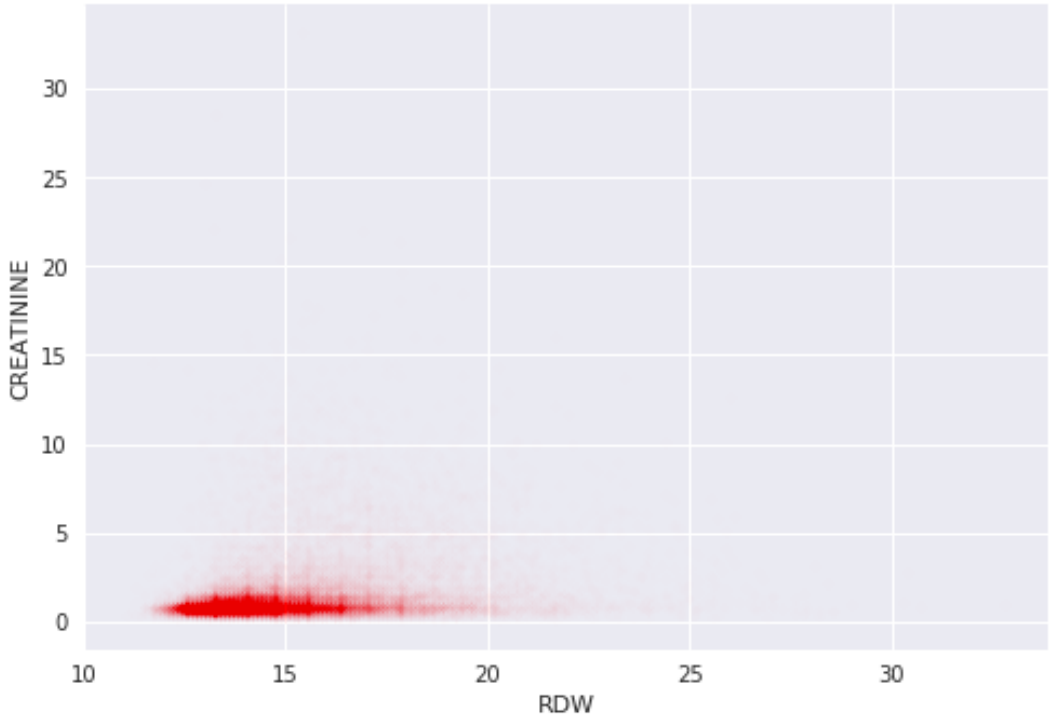

RDW
30
25
20
15
10
5
5
10
15
20
25
30
RDW

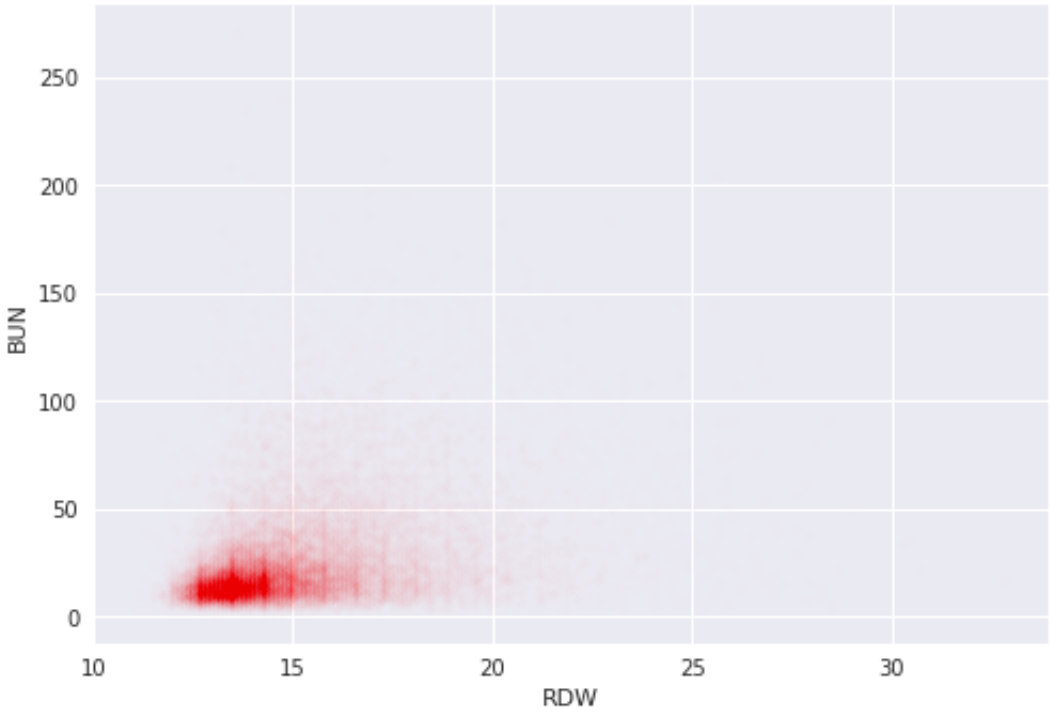

WBC
600
500
400
300
200
100
0
5
10
15
20
25
30
RDW

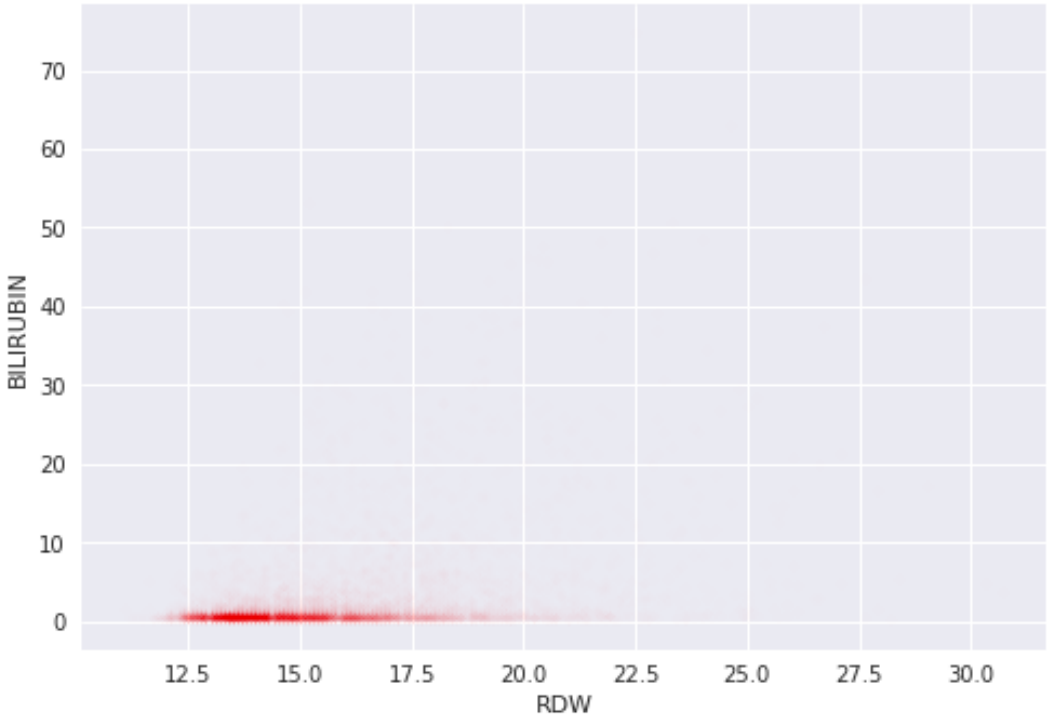

LACTATE
RDW
0
2
4
6
8
10
12
14
16
12
14
16
18
20
22
24

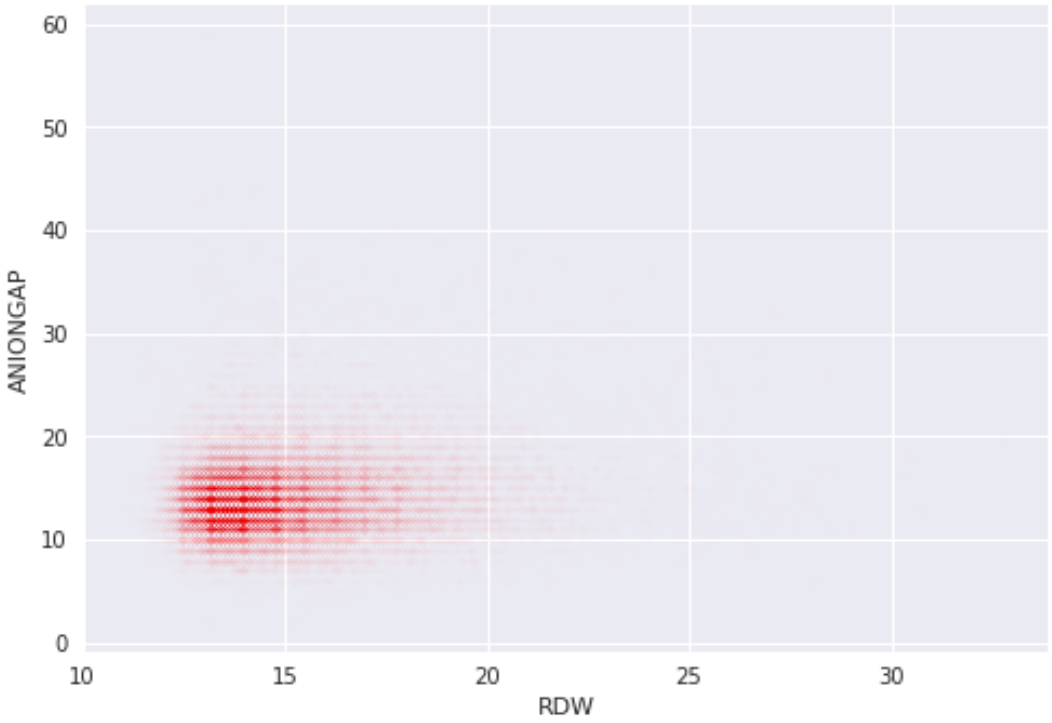

GLUCOSE
RDW
0
500
1000
1500
2000
2500
10
15
20
25
30

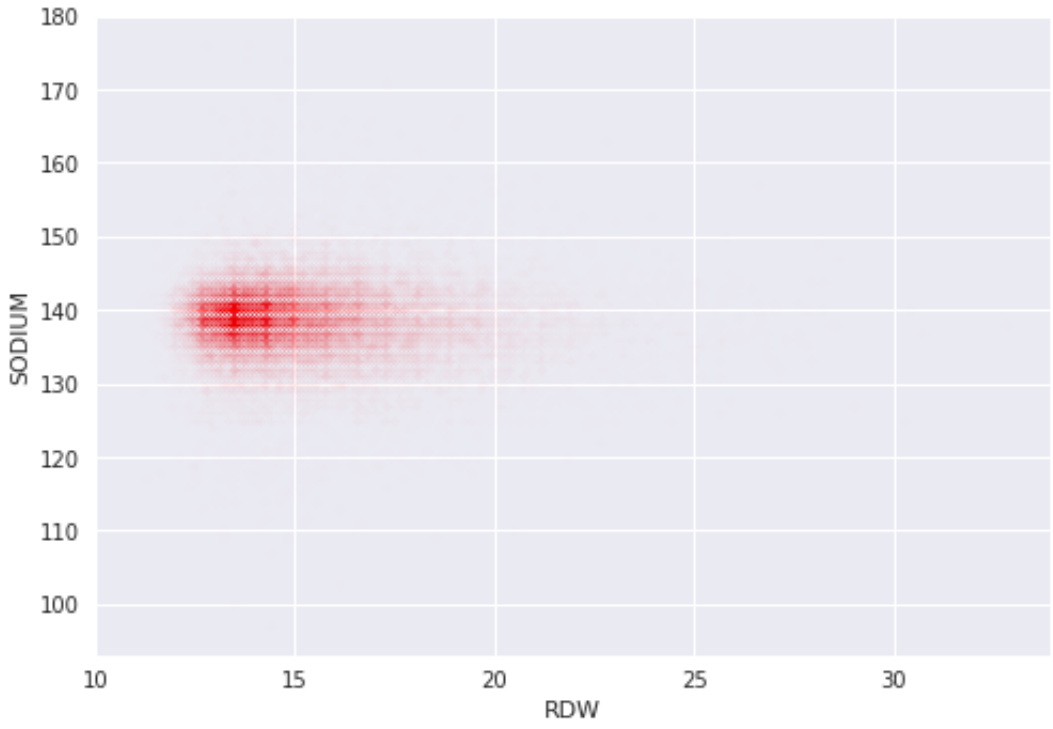

140
120
100
80
60
CHLORIDE
10
15
20
25
30
RDW

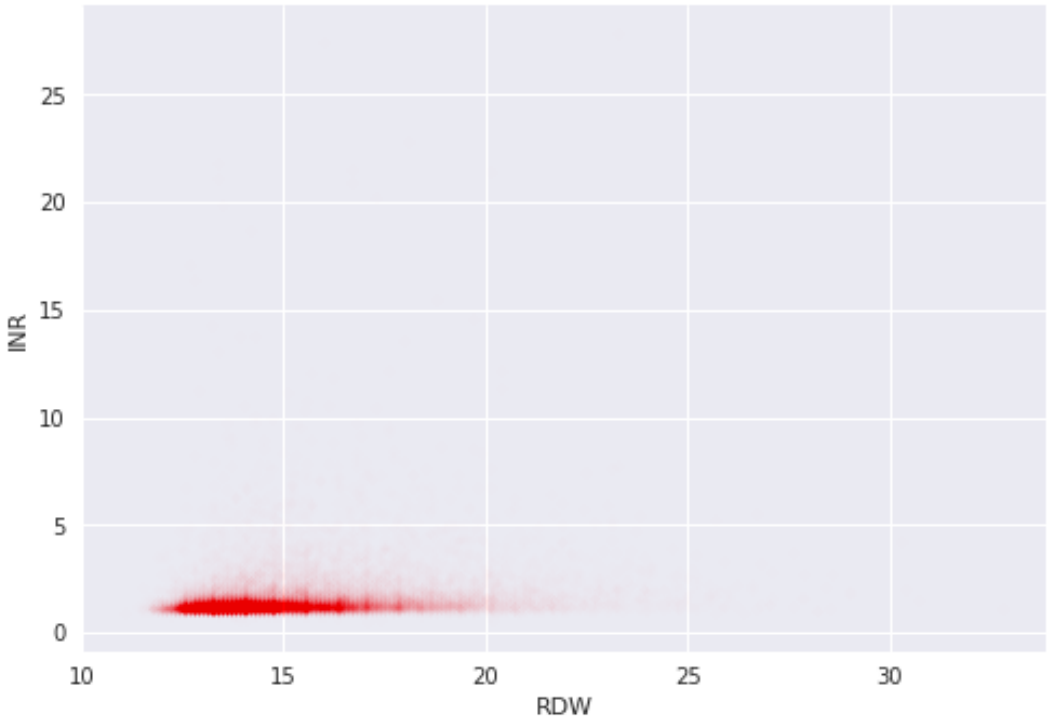

30
25
20
15
10
5
0
CREATININE
10
15
20
25
30
RDW

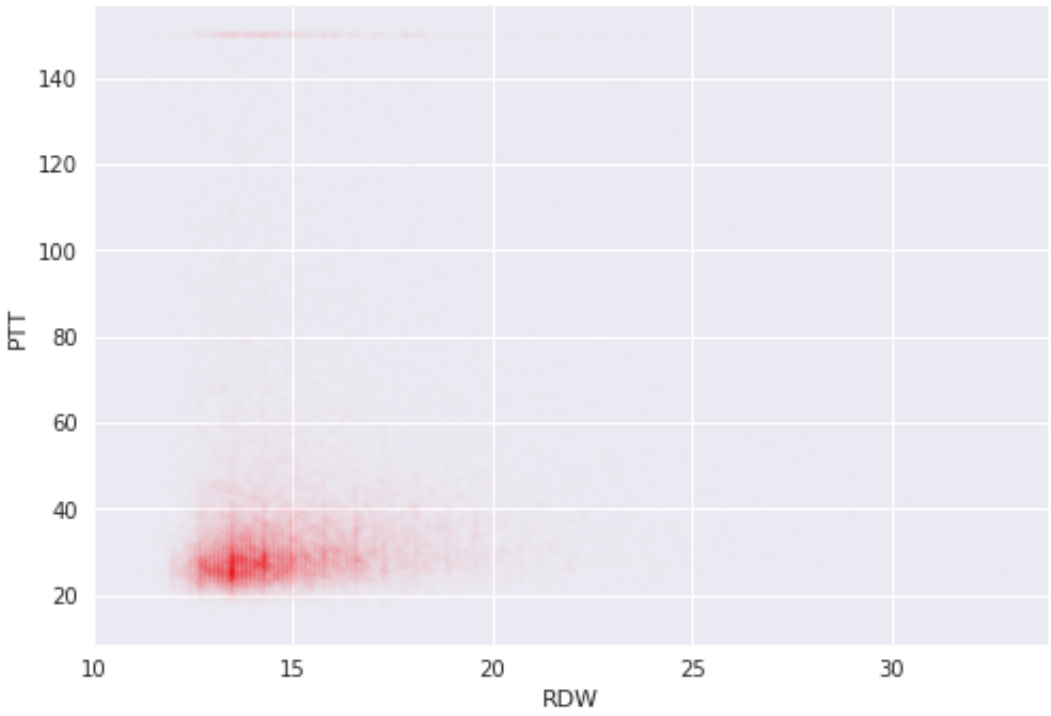

250
200
150
100
50
0
BUN
10
15
20
25
30
RDW

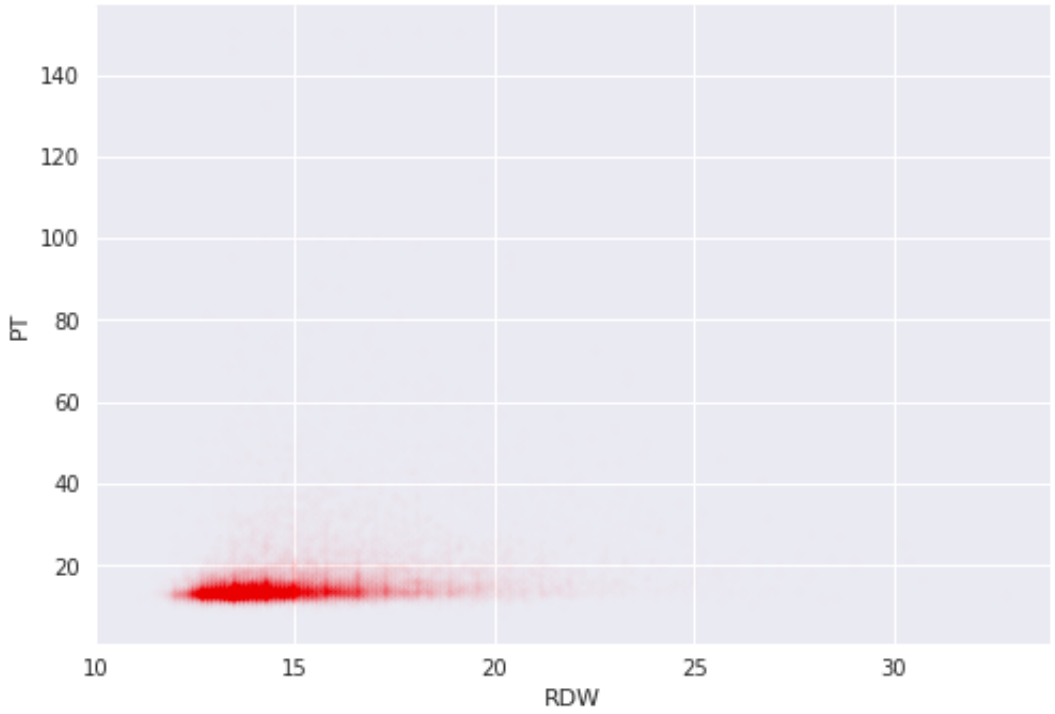

70
60
50
40
30
20
10
0
BILIRUBIN
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
RDW

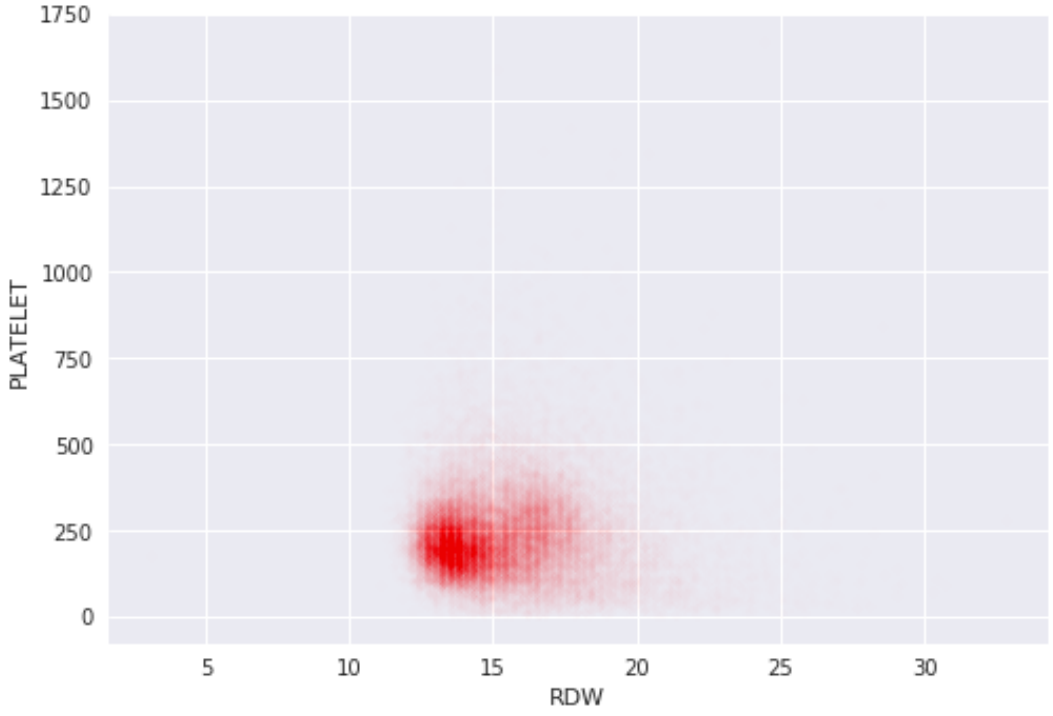

ANIONGAP
RDW
0
10
20
30
40
50
60
10
15
20
25
30

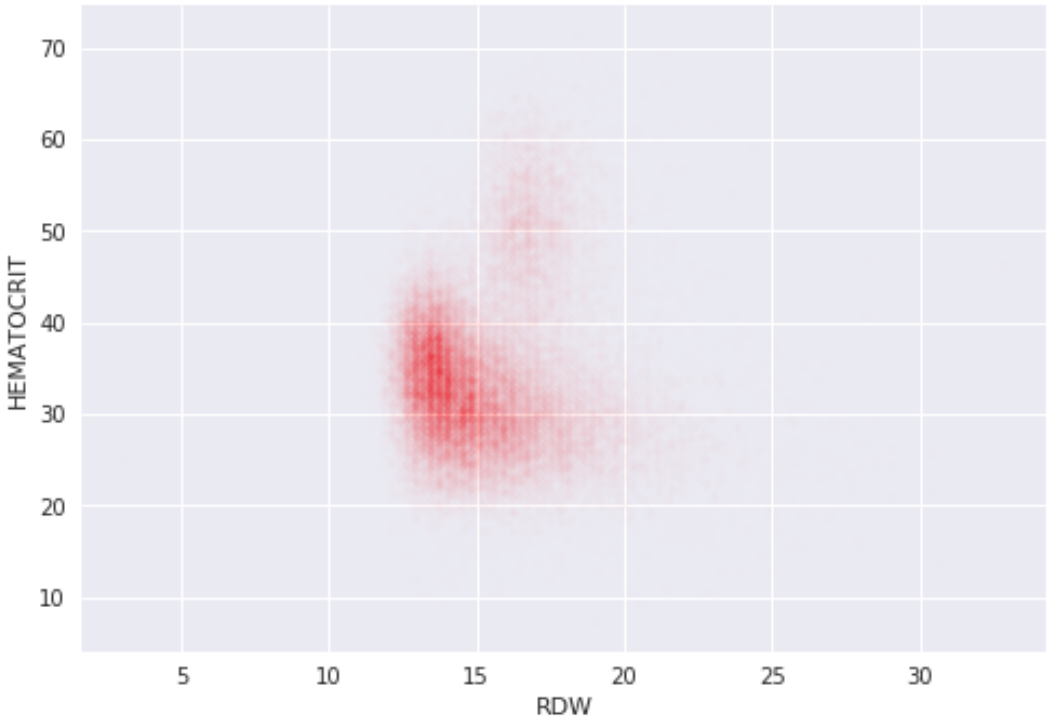

SODIUM
RDW
100
110
120
130
140
150
160
170
180
10
15
20
25
30

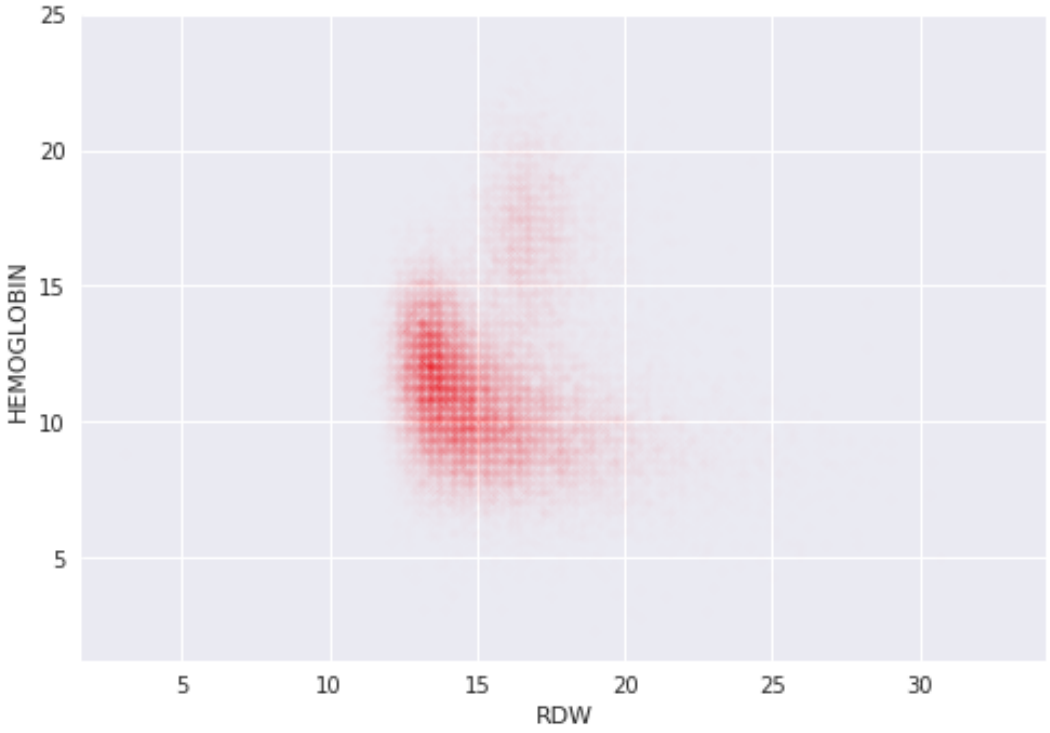

INR
RDW
0
5
10
15
20
25
10
15
20
25
30

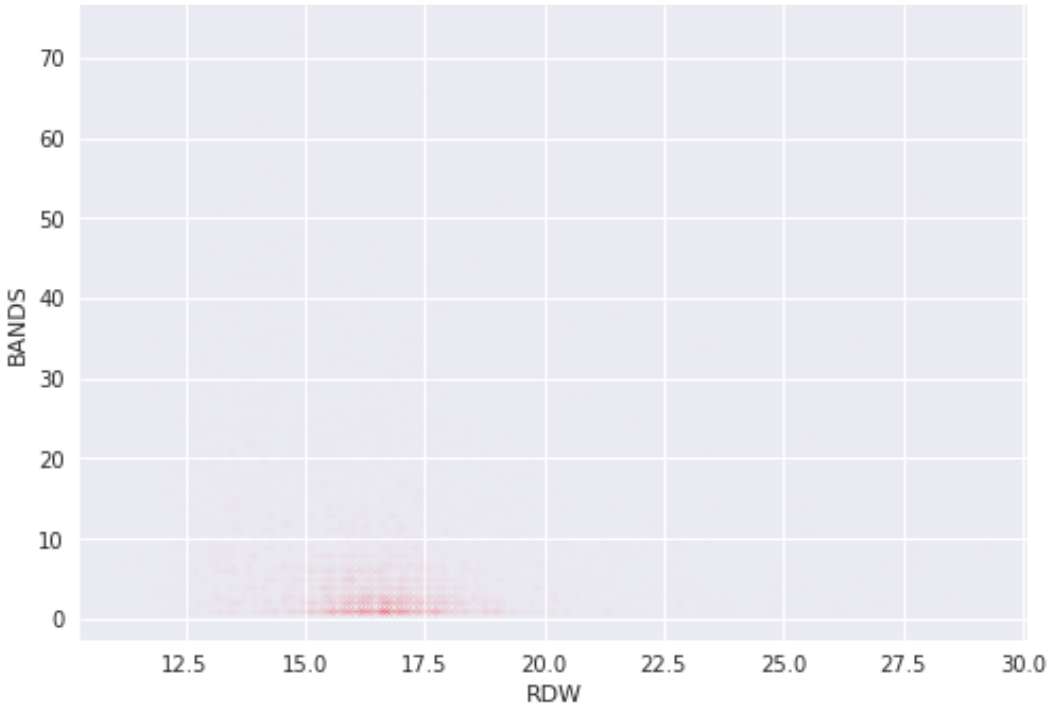

PTT
RDW
20
40
60
80
100
120
140
10
15
20
25
30


PT
RDW
140
120
100
80
60
40
20
10
15
20
25
30


PLATELET
RDW
1750
1500
1250
1000
750
500
250
0
5
10
15
20
25
30

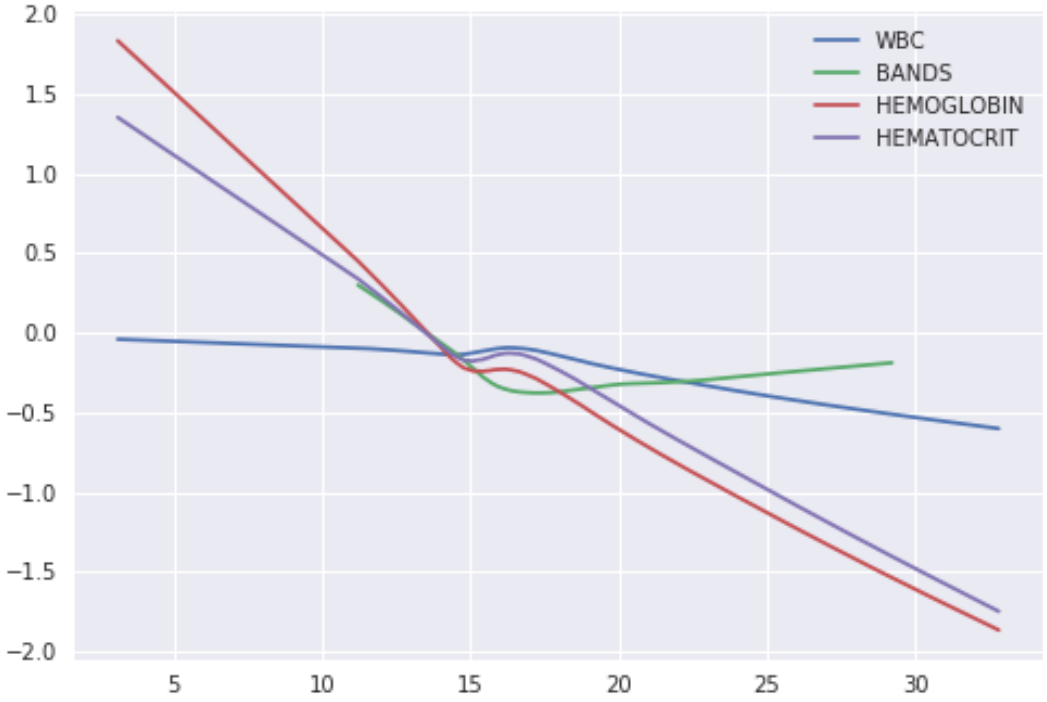

HEMATOCRIT
70
60
50
40
30
20
10
5
10
15
20
25
30
RDW


HEMOGLOBIN
25
20
15
10
5
5
10
15
20
25
30
RDW

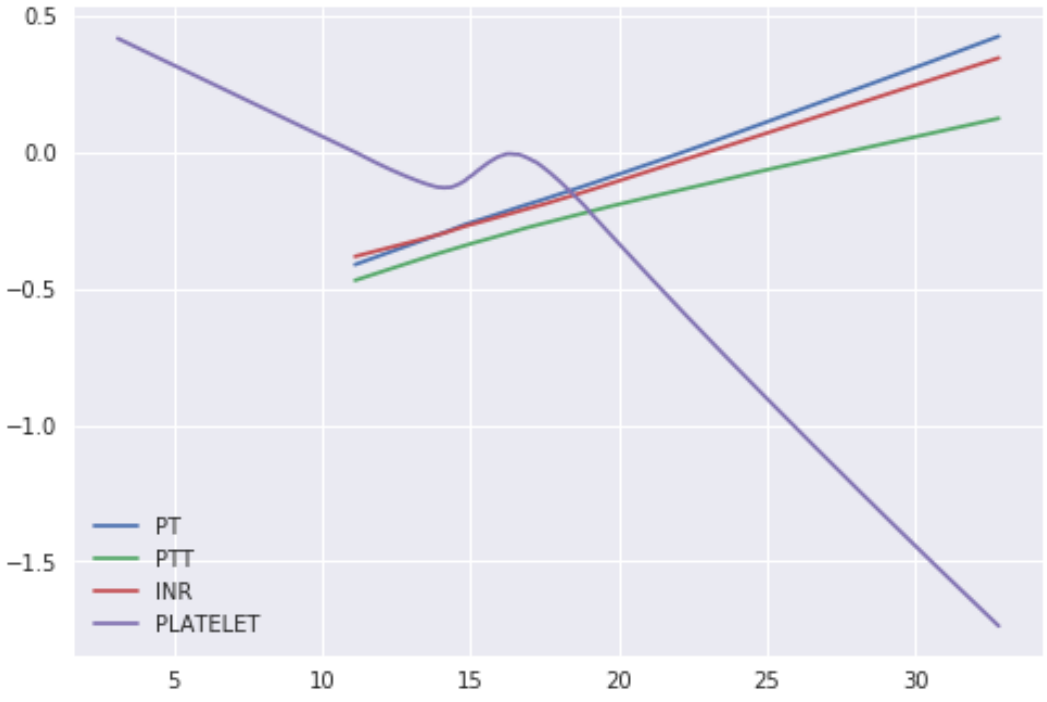


Based on the bivariation association scatterplots, there are very few (if any) associations between changes in RDW values and corresponding changes in another laboratory value.

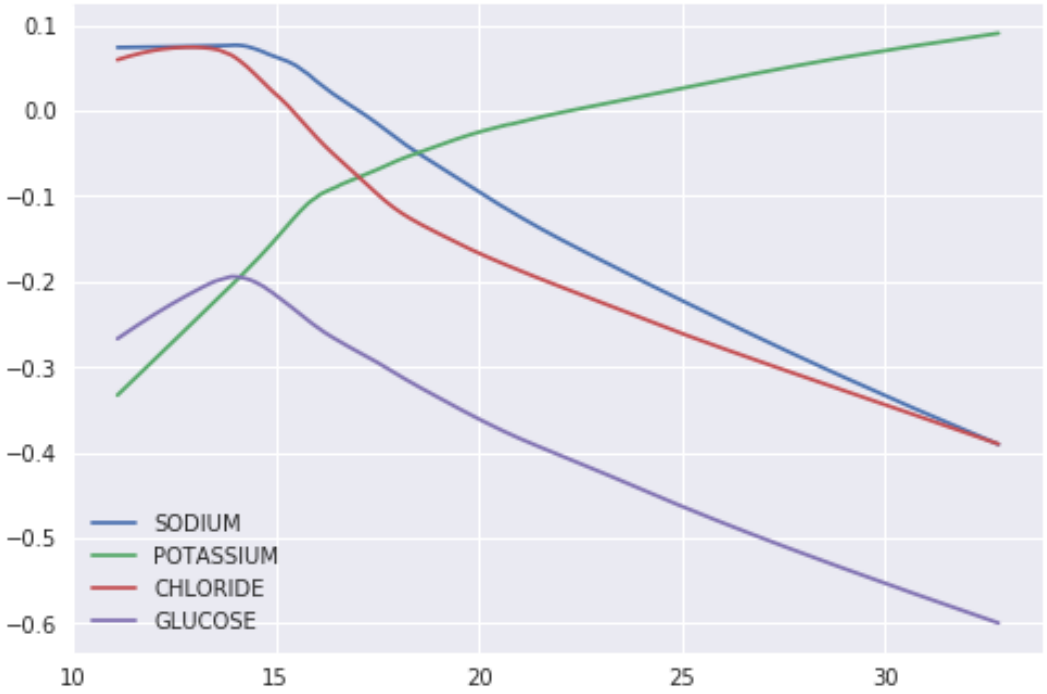
```
import statsmodels.api as sm
x = first_vals['RDW']

def build_lowess(y_var, x_var = x, df = first_vals_std):
  return sm.nonparametric.lowess(df[y_var], x_var, frac=2/3)

y_vars = ['WBC', 'BANDS', 'HEMOGLOBIN', 'HEMATOCRIT']
for y in y_vars:
  line = build_lowess(y)
  plt.plot(line[:, 0], line[:, 1], label = y);
plt.legend();
plt.show();
```



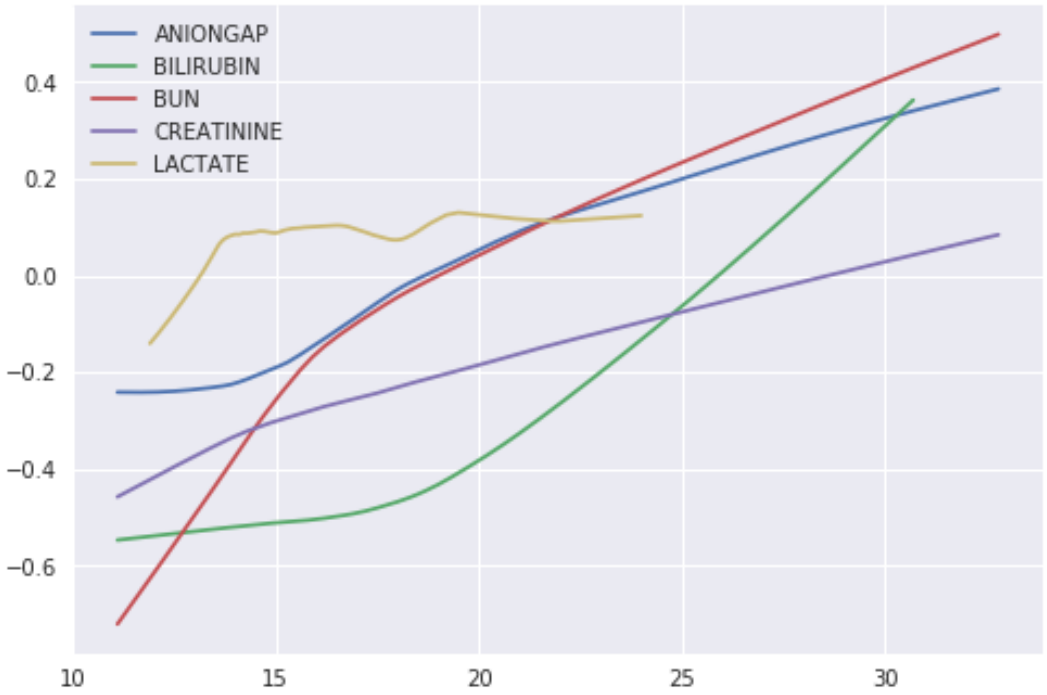

```
y_vars = ['PT', 'PTT', 'INR', 'PLATELET']
for y in y_vars:
  line = build_lowess(y)
  plt.plot(line[:, 0], line[:, 1], label = y);
plt.legend();
plt.show();
```



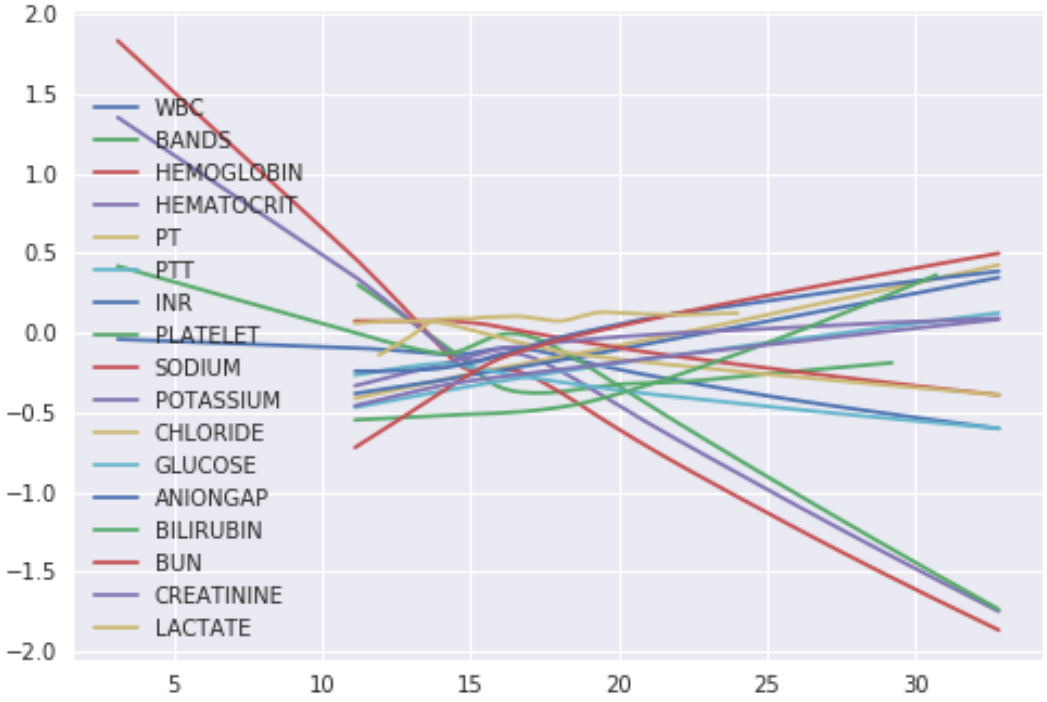
```
y_vars = ['SODIUM', 'POTASSIUM', 'CHLORIDE', 'GLUCOSE']
for y in y_vars:
  line = build_lowess(y)
  plt.plot(line[:, 0], line[:, 1], label = y);
plt.legend();
plt.show();
```




```
y_vars = ['ANIONGAP', 'BILIRUBIN', 'BUN', 'CREATININE', 'LACTATE']
for y in y_vars:
  line = build_lowess(y)
  plt.plot(line[:, 0], line[:, 1], label = y);
plt.legend();
plt.show();
```




```
y_vars = ['WBC', 'BANDS', 'HEMOGLOBIN', 'HEMATOCRIT',
          'PT', 'PTT', 'INR', 'PLATELET',
          'SODIUM', 'POTASSIUM', 'CHLORIDE', 'GLUCOSE',
          'ANIONGAP', 'BILIRUBIN', 'BUN', 'CREATININE', 'LACTATE']
for y in y_vars:
  line = build_lowess(y)
  plt.plot(line[:, 0], line[:, 1], label = y);
plt.legend();
plt.show();
```



LOWESS plots illustrate some associations not seen in the scatterplots. For example, as RDW values increase, the hemoglobin and hematocrit values decrease; a similar pattern is noted with platelet values. All other values tend to remain close to the mean. For those values remaining close to the mean, there is a general decrease in white blood cell (WBC) count, sodium, chloride, glucose as RDW values increase. Additionally, there is a general increase in prothrombin time (PT), partial thromboplastin time (PTT), international normalizing ratio (INR), potassium, blood urea nitrogen (BUN), creatinine, bilirubin, and anion gap as RDW values increase.

Identify Diagnoses Associated with High RDW Values

```
%bq query
WITH myQuery AS
( SELECT Subject_ID, ICD.ICD9_CODE, COUNT(*) AS DIAGNOSES FROM
  `vu-bigdata.MIMIC.DIAGNOSES_ICD` ICD
  INNER JOIN `vu-bigdata.MIMIC.D_ICD_DIAGNOSES` D_ICD
  ON ICD.ICD9_CODE = D_ICD.ICD9_CODE
  GROUP BY Subject_ID, ICD.ICD9_CODE
  ORDER BY DIAGNOSES DESC )
SELECT * FROM myQuery
```

```
%bq query
WITH myQuery AS
( SELECT Subject_ID, HADM_ID, COUNT(*) AS DIAGNOSES FROM
  `vu-bigdata.MIMIC.DIAGNOSES_ICD` ICD
  INNER JOIN `vu-bigdata.MIMIC.D_ICD_DIAGNOSES` D_ICD
  ON ICD.ICD9_CODE = D_ICD.ICD9_CODE
  GROUP BY Subject_ID, HADM_ID
  ORDER BY DIAGNOSES DESC )
SELECT * FROM myQuery
```

	Subject_ID	HADM_ID	DIAGNOSES
1	94,762	160,707	39
2	96,958	103,076	39
3	67,348	179,548	39
4	61,667	146,220	39
5	18,353	163,579	39
6	21,202	194,067	39
7	82,806	137,134	39
8	49,555	111,955	39
9	87,692	196,705	39
10	23,568	198,677	39
11	878	102,365	39
12	63,733	145,445	39
13	22,933	124,281	39
14	12,540	195,743	39

15	54,247	169,804	39
16	2,092	105,566	39
17	48,479	123,178	39
18	74,554	120,011	39
19	95,201	192,304	39
20	96,958	102,063	39
21	23,568	133,076	39
22	69,000	111,892	39
23	65,837	184,409	39
24	55,204	198,883	39
25	99,469	179,324	39

◄ ►

	Subject_ID	HADM_ID	DIAGNOSES
1	94,762	160,707	39
2	96,958	103,076	39
3	67,348	179,548	39
4	61,667	146,220	39
5	18,353	163,579	39
6	21,202	194,067	39
7	82,806	137,134	39

8	49,555	111,955	39
9	87,692	196,705	39
10	23,568	198,677	39
11	878	102,365	39
12	63,733	145,445	39
13	22,933	124,281	39
14	12,540	195,743	39
15	54,247	169,804	39
16	2,092	105,566	39
17	48,479	123,178	39
18	74,554	120,011	39
19	95,201	192,304	39
20	96,958	102,063	39
21	23,568	133,076	39
22	69,000	111,892	39
23	65,837	184,409	39
24	55,204	198,883	39
25	99,469	179,324	39

```
SELECT
Codes.CODE,
Codes.TITLE,
AVG(RDW) AS Avg_RDW,
AVG(RDW_ZScore) AS Avg_RDW_ZScore
--Median_RDW = PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY RDW) OVER(PARTI
TION BY CODE)
FROM
( SELECT HID, SID, RDW, AGE,
  CASE
    WHEN AGE <= 40 AND P.GENDER = 'F' THEN (RDW - 12.3) / 0.7
    WHEN AGE <= 55 AND AGE > 40 AND P.GENDER = 'F' THEN (RDW - 12.1) / 0.5
    WHEN AGE <= 70 AND AGE > 55 AND P.GENDER = 'F' THEN (RDW - 12.3) / 0.6
    WHEN AGE <= 85 AND AGE > 70 AND P.GENDER = 'F' THEN (RDW - 12.6) / 0.8
    WHEN AGE > 85 AND P.GENDER = 'F' THEN (RDW - 12.7) / 0.7
    WHEN AGE <= 40 AND P.GENDER = 'M' THEN (RDW - 12.1) / 0.5
    WHEN AGE <= 55 AND AGE > 40 AND P.GENDER = 'M' THEN (RDW - 12.2) / 0.6
    WHEN AGE <= 70 AND AGE > 55 AND P.GENDER = 'M' THEN (RDW - 12.3) / 0.6
    WHEN AGE <= 85 AND AGE > 70 AND P.GENDER = 'M' THEN (RDW - 12.6) / 0.8
    WHEN AGE > 85 AND P.GENDER = 'M' THEN (RDW - 12.9) / 0.8
    ELSE NULL END AS RDW_ZScore
  FROM
  ( SELECT RDW,
      L.HADM_ID AS HID,
      L.SUBJECT_ID AS SID,
    P.GENDER,
      CAST(LEFT(CAST(CHARTTIME AS STRING), 4) AS INTEGER) - CAST(LEFT(CAST
(DOB AS STRING), 4) AS INTEGER) AGE
      FROM [vu-bigdata:MIMIC.LAB_VIEW] L
      INNER JOIN [vu-bigdata:MIMIC.PATIENTS] P
      ON L.SUBJECT_ID = P.SUBJECT_ID
      LIMIT 5000 ) TEMP
) LV
INNER JOIN
( SELECT
    HADM_ID AS HID,
    SUBJECT_ID AS SID,
    D_ICD.ICD9_CODE AS CODE,
    D_ICD.SHORT_TITLE AS TITLE
    FROM [vu-bigdata:MIMIC.DIAGNOSES_ICD] ICD
    INNER JOIN [vu-bigdata:MIMIC.D_ICD_DIAGNOSES] D_ICD
    ON ICD.ICD9_CODE = D_ICD.ICD9_CODE
```

```
) Codes
ON LV.HID = Codes.HID
AND LV.SID = Codes.SID
WHERE AGE > 18
GROUP BY Codes.Code, Codes.Title
HAVING COUNT(*) > 50
ORDER BY Avg_RDW_ZScore DESC
```

Codes_CODE	Codes_TITLE	Avg_RDW	Avg_RDW_ZScore
42	Human immuno virus dis	19.3826531	12.6690476
24900	Sec DM wo cmp nt st uncn	18.8514286	12.432517
1123	Cutaneous candidiasis	19.1015625	12.3762835
1122	Candidias urogenital NEC	19.8535714	12.329156
785	Cytomegaloviral disease	18.8387755	12.2770408
4568	Varices of other sites	19.5365079	12.1996032
53789	Gastroduodenal dis NEC	19.2493333	11.9295
99685	Compl marrow transplant	19.0241379	11.7938424
5723	Portal hypertension	19.1356201	11.7477541
45621	Esoph varice oth dis NOS	19.0672131	11.6435467
45620	Bleed esoph var oth dis	19.0908257	11.5440585
5724	Hepatorenal syndrome	19.0188679	11.4456713
E9320	Adv eff corticosteroids	18.9746032	11.4437547
5715	Cirrhosis of liver NOS	18.9772436	11.3985043
7854	Gangrene	18.931746	11.3421391
56723	Spontan bact peritonitis	18.9949495	11.3193122
7044	Chrnc hpt C w hepat Coma	18.9533333	11.2834392
5711	Ac alcoholic hepatitis	18.7880734	11.275841
5722	Hepatic encephalopathy	18.8215827	11.1700097
452	Portal vein thrombosis	19.0775758	11.1167893
7847	Epistaxis	18.6072917	11.0723586
20500	Ac myl leuk wo achv rmsn	18.7568627	10.9804855
5728	Oth sequela, chr liv dis	18.6532787	10.8909055
28959	Spleen disease NEC	18.6962963	10.822112
78959	Ascites NEC	18.627013	10.5314811
45821	Hemododialysis hypotensn	18.5535211	10.3909624
99682	Compl liver transplant	18.3879518	10.3799197
7070	Hpt C w/o hepat coma NOS	17.9788618	10.3352981
5712	Alcohol cirrhosis liver	18.2921687	10.2887622
2867	Acq coagul factor defic	18.3898601	10.2745921
56969	Colstmy/enteros comp NEC	18.5808824	10.2673669
V4983	Await organ transplnt st	18.1573171	10.2574332
3843	Pseudomonas septicemia	18.6833333	10.1467014
7824	Jaundice NOS	18.56	10.137619
25541	Glucocorticoid deficient	17.6836066	10.0669204
7100	Syst lupus erythematosus	17.8529412	10.0548786
1125	Disseminated candidiasis	18.2738255	10.0501119
48282	Pneumonia e coli	18.3612903	10.022331
1173	Aspergillosis	18.1229508	9.99478923
27542	Hypercalcemia	18.3949495	9.93811929
70715	Ulcer other part of foot	18.4308824	9.92039566
5589	Noninf gastroenterit NEC	18.2145455	9.74980519
5780	Hematemesis	17.7535354	9.72493987
1550	Mal neo liver, primary	18.2382716	9.72067901
5762	Obstruction of bile duct	18.5127907	9.66647287
45981	Venous insufficiency NOS	18.3567164	9.63320896
5718	Chronic liver dis NEC	18.1705882	9.57326681

4168	Chr pulmon heart dis NEC	18.2596429	9.55192177
56721	Peritonitis (acute) gen	18.045098	9.49925303
7994	Cachexia	17.9153846	9.4599359
2866	Defibrination syndrome	17.9995485	9.44394281
7054	Chrnc hpt C wo hpat coma	17.8589372	9.31478031
4104	Enterococcus group d	17.8640845	9.31301979
4210	Ac/subac bact endocard	17.9301887	9.28921833
V0980	Inf mcr rst ot drg nt ml	17.5778846	9.19655449
34982	Toxic encephalopathy	17.8632911	9.17357595
2639	Protein-cal malnutr NOS	17.8872671	9.13919698
99931	Oth/uns inf-cen ven cath	17.6247934	9.12100551
5768	Dis of biliary tract NEC	18.316129	9.06104071
70709	Pressure ulcer, site NEC	18.0897059	9.01910014
6826	Cellulitis of leg	17.9634731	9.00522526
V08	Asymp hiv infectn status	17.4979592	8.97534014
99591	Sepsis	17.9481481	8.97066248
78550	Shock NOS	17.4807692	8.95807387
V433	Heart valve replac NEC	18.678	8.95471429
2512	Hypoglycemia NOS	17.3236364	8.95179654
27801	Morbid obesity	17.4202381	8.92749433
1985	Secondary malig neo bone	18.1767857	8.9118835
E8791	Abn react-renal dialysis	17.6768293	8.90866725
28984	Heparin-indu thrombocyto	18.558	8.88028571
E9331	Adv eff antineoplastic	17.8252525	8.87997835
4185	Oth gram negatv bacteria	17.912963	8.86752646
56722	Peritoneal abscess	17.6042254	8.85930584
99659	Malfunc oth device/graft	17.954	8.85716667
413	Klebsiella pneumoniae	17.7376471	8.83480392
5856	End stage renal disease	17.9115824	8.83412763
7823	Edema	17.5574074	8.80846561
5781	Blood in stool	17.8450292	8.79507101
E8781	Abn react-artif implant	17.7065574	8.78881733
E8780	Abn react-org transplant	17.3604278	8.78636364
99661	React-cardiac dev/graft	18.0775862	8.77227011
570	Acute necrosis of liver	17.5574338	8.7659308
2869	Coagulat defect NEC/NOS	17.67543	8.75898853
E8798	Abn react-procedure NEC	17.7516556	8.75700095
2761	Hyposmolality	17.538817	8.73428175
5789	Gastrointest hemorr NOS	17.9075209	8.72494031
5761	Cholangitis	17.8458647	8.72028285
5680	Peritoneal adhesions	17.3641975	8.70315991
43820	Late ef-hemplga side NOS	17.534	8.69166667
5750	Acute cholecystitis	18.0626506	8.66123064
99674	Comp-oth vasc dev/graft	17.4555556	8.64767416
2536	Neurohypophysis dis NEC	17.661194	8.64278607
1977	Second malig neo liver	17.8367347	8.61026482
2851	Ac posthemorrhag anemia	17.6972857	8.56945408
1970	Secondary malig neo lung	17.7666667	8.53222222
389	Septicemia NOS	17.6760976	8.47952033

The 10 diagnoses associated with the highest mean RDW values upon admission include:

1. Human immununodeficiency virus
2. Secondary Diabetes Mellitus
3. Candidiasis of the skin
4. Candidiasis of urogenital sites
5. Cytomegaloviral disease
6. Varices of other sites
7. Other specified disorders of stomach and duodenum
8. Stem cell transplant
9. Portal hypertension
10. Esophageal varices in diseases classified elsewhere

Changes in RDW values from ICU Admission to discharge

```
print(all_labs.shape)
all_labs.head()
```

	SUBJECT_ID	ICUSTAY_ID	INTIME	OUTTIME	CHARTTIME	RDW
0	44018	241676	2155-11-05 15:42:18	2155-11-06 21:23:21	2155-11-03 06:04:00	16.2
1	40412	269822	2103-06-22 18:28:48	2103-06-25 22:06:34	2103-06-22 15:45:00	15.6
2	22130	246999	2112-05-27 00:45:59	2112-06-06 18:10:06	2112-05-26 23:38:00	12.5
3	73755	241261	2138-12-06 12:14:34	2138-12-14 12:31:33	2138-12-06 12:38:00	18.0
4	53615	240377	2167-07-01 06:29:30	2167-07-04 01:53:48	2167-07-03 15:30:00	15.4

Clean data and separate by patient

```
# Clean data - get standardized times of measurements within stays
# Time format is YYYY-MM-DD
targets = ['ICUSTAY_ID','INTIME','OUTTIME','CHARTTIME','RDW']
rdw = all_labs[targets]
rdw = rdw.dropna(how='any')

rdw['CHARTTIME'] = pd.to_datetime(rdw['CHARTTIME'])
rdw['INTIME'] = pd.to_datetime(rdw['INTIME'])
rdw = rdw[rdw['CHARTTIME'] >= rdw['INTIME']] #Only take charttimes taken w
ithin ICU stay

rdw['time_std'] = (rdw['CHARTTIME']-rdw['INTIME'])/np.timedelta64(1, 'h')
rdw['time_ceil'] = rdw['time_std'].apply(math.ceil)

print(rdw.shape)
```

(526285, 9)

```
#subset complete lab_view by unique ICUSTAY ID's
icustays = rdw['ICUSTAY_ID'].unique()
eachstay = {}
for stay in icustays:
  eachstay[stay] = rdw[rdw['ICUSTAY_ID'] == int(stay)]


  #subset complete lab_view by unique ICUSTAY ID's
icustays = rdw['ICUSTAY_ID'].unique()
eachstay = {}
for stay in icustays:
  eachstay[stay] = rdw[rdw['ICUSTAY_ID'] == int(stay)]
```

Visualizing time series

```
def getSeries(df):
  '''Get sorted time series of a patient RDW given a single patient datafr
ame'''
  series = pd.Series(df['RDW'].values, index=df['time_std'])
  series = series.sort_index()
  return series

  #Sample time series
s = getSeries(eachstay[269822])
s
```

time_std	RDW Value
10.336667	15.8
31.236667	15.9
61.603333	16.2
83.520000	16.3

Analyzing individual time series

```
#initialize df with delta column meaning difference between first and last
rdw value
cols = ['delta','std','num_obs']

autocorrs = ["autocorr_"+str(lag) for lag in range(1,7)]
cols = cols+autocorrs

features = pd.DataFrame(index=[list(eachstay.keys())],columns=cols)

for p, df in eachstay.items():
  if (df.shape[0]>10): #only take stays with more than n measurements
    s = getSeries(df)
    features['delta'].loc[p] = round(s.iloc[-1] - s.iloc[0], 1) # cut off
floating point errors
    features['std'].loc[p] = s.std()
    features['num_obs'].loc[p] = len(s)
    for lag, label in enumerate(autocorrs):
      features[label].loc[p] = s.autocorr(lag+1)
      #Since about 2 measurements are taken daily, it could be interesting
to look at autocorrelation values in multiples of 2

features = features.dropna(axis=0, how='any')
print(features.shape)
features.head()
```

	delta	std	num_obs	autocorr_1	autocorr_2	autocorr_3	autocorr_4	autocorr_5	autocorr_6
262154	2.2	0.781757	18	0.949756	0.906671	0.866668	0.824122	0.780179	0.761075
262164	1.3	1.24324	23	0.814928	0.636755	0.374669	0.086883	-0.0820771	-0.269624
262169	2	0.710939	24	0.941761	0.91058	0.82794	0.766686	0.648299	0.557474
262172	0.2	0.210263	20	0.525238	0.157157	-0.0630348	-0.323945	-0.578445	-0.560018
262197	1.2	0.638508	13	0.915668	0.781159	0.61639	0.372031	0.0695496	-0.68592

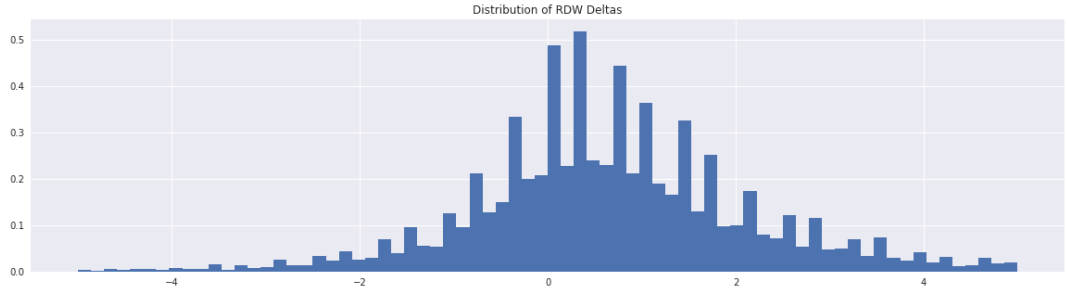
Because some patients only have one observation, standard deviation could not be properly calculated.

Min/Max

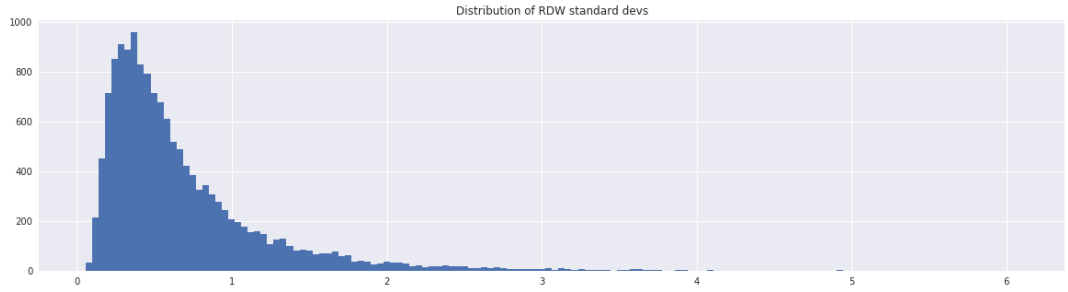
```
print(min(features['delta'].tolist()))
print(max(features['delta'].tolist()))
```

-16.699999999999996 15.400000000000002

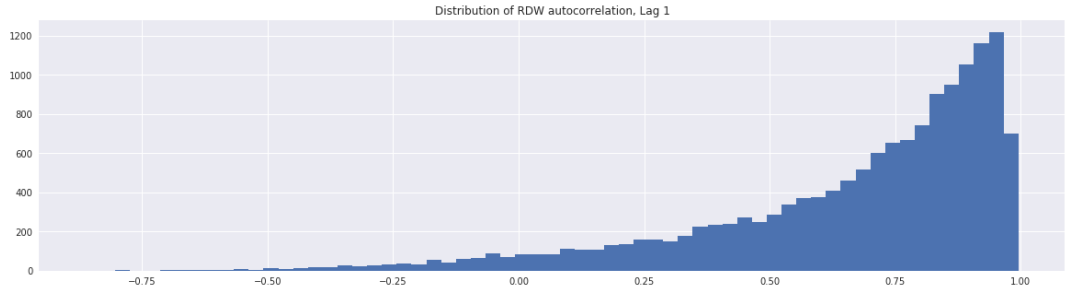
```
plt.figure(figsize=(20,5))
plt.hist(features['delta'].tolist(),bins='auto',range=(-5,5),density =True
);
plt.title("Distribution of RDW Deltas")
# Only values within the range are shown
```



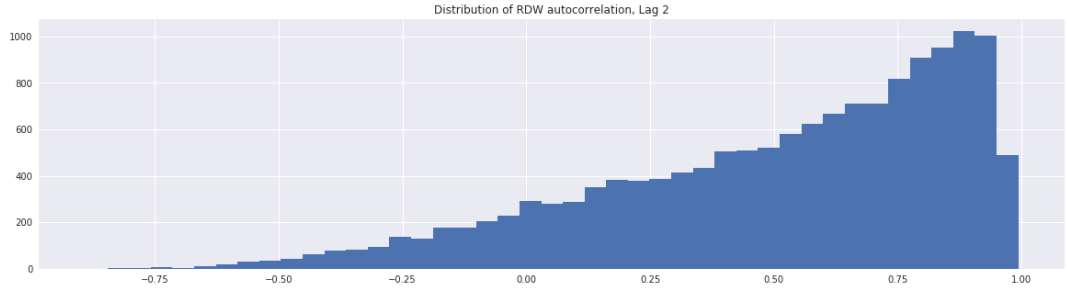

```
plt.figure(figsize=(20,5))
plt.hist(features['std'].dropna().tolist(),bins='auto');
plt.title("Distribution of RDW standard devs")
```




```
plt.figure(figsize=(20,5))
plt.hist(features['autocorr_1'].dropna().tolist(),bins='auto');
plt.title("Distribution of RDW autocorrelation, Lag 1")
```




```
plt.figure(figsize=(20,5))
plt.hist(features['autocorr_2'].dropna().tolist(),bins='auto');
plt.title("Distribution of RDW autocorrelation, Lag 2")
```



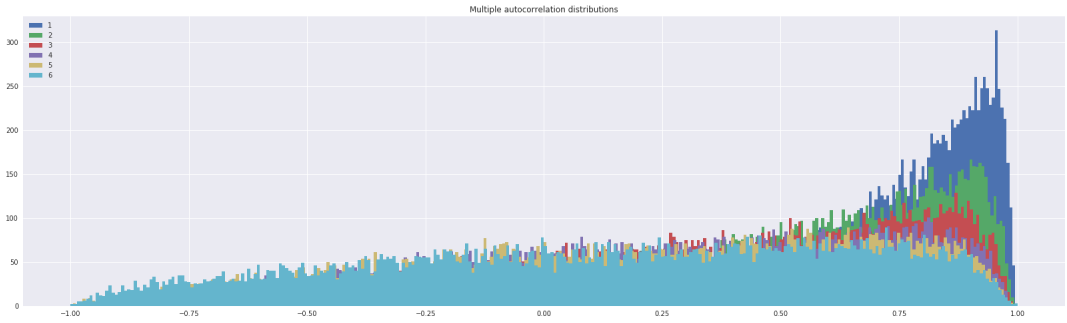
Time Series Plots with High/Low Autocorrelation

```
features.sort_values(by='autocorr_2').head()
```

	delta	std	num_obs	autocorr_1	autocorr_2	autocorr_3	autocorr_4	autocorr_5	autocorr_6
284103	0.7	0.358786	11	-0.0157876	-0.889408	0.206474	0.790639	-0.568939	-0.774496
243947	0.1	0.24606	11	0.151407	-0.845488	-0.328652	0.549216	0.38358	-0.42135
278936	-1.1	0.321926	12	0.332662	-0.835241	0.151697	0.536141	-0.389039	-0.224575
262720	0.3	0.163485	11	0.131635	-0.816497	-0.135788	0.404977	-0.491354	0.173702
279159	0.2	0.238175	11	0.229972	-0.770727	-0.237666	0.612251	0.0362818	-0.776062

```
highSeries = features.loc[features[features['autocorr_2']>0.85].sample(10)
.index] # Look at 10 random series with autocorr_2 at highest or lowest pe
rcentiles
lowSeries = features.loc[features[features['autocorr_2']<-0.75].sample(10)
.index]
plt.figure(figsize=(10,6))
plt.subplot(2,1,1)
for t in highSeries.index:
  s = getSeries(eachstay[t[0]])
  plt.plot(s)
plt.title("High autocorr_2 series samples")
plt.xlim(0,1000)

plt.subplot(2,1,2)
for t in lowSeries.index:
  s = getSeries(eachstay[t[0]])
  plt.plot(s)
plt.title("Low autocorr_2 series samples")
plt.ylabel("RDW")
plt.xlabel("Time (hours)")
plt.xlim(0,1000)

plt.tight_layout()
```


```
plt.figure(figsize=(28,8))
plt.title("Multiple autocorrelation distributions")
for lag, label in enumerate(autocorrs):
  plt.hist(features[label].dropna().tolist(),bins=300,label=str(lag+1));
plt.legend(loc='upper left')
```



Variable Importance of RDW in Predicting In-hosptial Mortality

```
import pandas as pd
import numpy as np
import google.datalab.bigquery as bq
from sklearn.model_selection import train_test_split

# Get measurements for each patient when CHARTTIME
# occurs after ICU admission
query = bq.Query('''SELECT *
                    FROM `vu-bigdata.MIMIC.LAB_DEATH_VIEW`
                    WHERE LV_CHARTTIME >= LV_INTIME''')

data = query.execute().result().to_dataframe()
data.head()
```

	LV_SUBJECT_ID	LV_HADM_ID	LV_ICUSTAY_ID	LV_INTIME	LV_OUTTIME	LV_LOS	LV_CHARTTIME	LV_RDW	LV_ANIONGAP	LV_BANDS
0	94762	160707	215473	2165-06-17 15:22:41	2165-06-24 11:33:22	6.8408	2165-06-23 04:30:00	16.5	33.0	NaN
1	50004	131407	208406	2141-04-14 00:39:48	2141-04-14 18:00:14	0.7225	2141-04-18 19:27:00	20.6	40.0	NaN
2	32624	118491	226705	2106-06-17 16:56:35	2106-06-20 19:01:14	3.0866	2106-06-19 02:57:00	25.8	25.0	NaN
3	6702	157559	220172	2129-06-09 16:18:26	2129-09-01 17:17:17	84.0409	2129-10-07 00:45:00	16.2	10.0	NaN
4	58702	114246	287187	2114-11-09 13:53:46	2114-12-06 12:33:47	26.9445	2114-12-01 03:47:00	20.0	11.0	2.0

LV_POTASSIUM	LV_PTT	LV_INR	LV_PT	LV_SODIUM	LV_BUN	LV_WBC	A_DEATHTIME	DEATH_FLAG	A_HOSPITAL_EXPIRE_FLAG
4.3	38.4	2.3	24.7	135.0	90.0	0.1	2165-06-24 03:23:00	1	1
3.5	77.1	6.4	54.9	144.0	NaN	21.2	2141-04-18 21:35:00	1	1
4.7	150.0	2.4	23.8	125.0	97.0	11.0	2106-06-20 16:00:00	1	1
4.3	39.0	3.6	33.4	145.0	59.0	21.5	2129-10-27 01:00:00	1	1
4.6	45.8	0.9	11.0	134.0	31.0	12.4	2114-12-06 10:34:00	1	1

Clean Data

```
df = data.rename(columns = lambda x: x[3:]) #get rid of bad names
#df = df.drop(df.columns[[0,1]], axis=1) #get rid of redundant cols

# carry observed values forward within each ICUSTAY
df.sort_values(by=['ICUSTAY_ID', 'CHARTTIME'], inplace=True)
df_locf = df.groupby(['ICUSTAY_ID']).fillna(method='ffill')
df_locf.head()
```

	ANIONGAP	BANDS	BILIRUBIN	BUN	CHARTTIME	CHLORIDE	CREATININE	EATHTIME	GLUCOSE	HADM_ID
259995	NaN	NaN	NaN	NaN	2181-11-25 19:27:00	NaN	NaN	NaT	NaN	152234
654187	14.0	NaN	0.2	83.0	2181-11-26 05:36:00	101.0	2.7	NaT	87.0	152234
634144	17.0	NaN	0.2	96.0	2181-11-27 05:54:00	97.0	3.5	NaT	108.0	152234
309835	17.0	NaN	0.2	96.0	2181-11-27 12:24:00	97.0	3.5	NaT	108.0	152234
782559	13.0	NaN	0.2	58.0	2181-11-28 02:56:00	98.0	2.9	NaT	99.0	152234

OUTTIME	PLATELET	POTASSIUM	PT	PTT	RDW	SODIUM	SUBJECT_ID	TH_FLAG	WBC
2181-11-28 20:59:25	NaN	NaN	NaN	NaN	NaN	NaN	55973	0	NaN
2181-11-28 20:59:25	128.0	4.3	26.1	40.6	17.7	139.0	55973	0	2.8
2181-11-28 20:59:25	155.0	5.0	39.1	40.6	17.5	131.0	55973	0	3.5
2181-11-28 20:59:25	155.0	5.0	39.1	40.6	17.5	131.0	55973	0	3.5
2181-11-28 20:59:25	168.0	4.2	39.1	40.6	17.2	135.0	55973	0	3.4

```
# find non-missing RDW values
df_first_rdw = df_locf[np.isfinite(df_locf['RDW'])]
# keep the first
df_first_rdw = df_first_rdw.groupby(['SUBJECT_ID']).first()
# used median imputation for all other variables (for now - not for paper)
#df_imputed = df_first_rdw.apply(lambda x: x.fillna(np.nanmedian(x)), axis
=0)
df_imputed = df_first_rdw

df_imputed.head()
```

	ANIONGAP	BANDS	BILIRUBIN	BUN	CHARTTIME	CHLORIDE	CREATININE	EATHTIME	GLUCOSE	HADM_ID
SUBJECT_ID										
3	23.0	5.0	0.8	41.0	2101-10-20 19:26:00	111.0	2.4	NaT	162.0	145834
4	15.0	NaN	1.9	10.0	2191-03-16 05:42:00	108.0	0.5	NaT	183.0	185777
5	NaN	NaN	NaN	NaN	2103-02-02 06:10:00	NaN	NaN	NaT	NaN	178980
6	20.0	NaN	0.2	65.0	2175-05-31 01:48:00	101.0	10.0	NaT	181.0	107064
7	NaN	NaN	4.7	NaN	2121-05-25 02:30:00	NaN	NaN	NaT	NaN	118037

	ANIONGAP	BANDS	BILIRUBIN	BUN	CHARTTIME	CHLORIDE	CREATININE	EATHTIME	GLUCOSE	HADM_ID
SUBJECT_ID										
3	23.0	5.0	0.8	41.0	2101-10-20 19:26:00	111.0	2.4	NaT	162.0	145834
4	15.0	NaN	1.9	10.0	2191-03-16 05:42:00	108.0	0.5	NaT	183.0	185777
5	NaN	NaN	NaN	NaN	2103-02-02 06:10:00	NaN	NaN	NaT	NaN	178980
6	20.0	NaN	0.2	65.0	2175-05-31 01:48:00	101.0	10.0	NaT	181.0	107064
7	NaN	NaN	4.7	NaN	2121-05-25 02:30:00	NaN	NaN	NaT	NaN	118037


```
# NA values as percent of total (some columns are really empty)
df_imputed.isnull().sum()/df_imputed.shape[0]
```

```
ANIONGAP      0.067703
BANDS         0.777310
BILIRUBIN     0.426432
BUN           0.095536
CHARTTIME     0.000000
CHLORIDE      0.067115
CREATININE    0.095724
EATHTIME      0.875620
GLUCOSE       0.105057
HADM_ID       0.000000
HEMATOCRIT    0.000000
HEMOGLOBIN    0.000000
INR           0.185054
```

```
INTIME                  0.000000
LACTATE                 0.493735
LOS                     0.000212
OSPITAL_EXPIRE_FLAG     0.000000
OUTTIME                 0.000212
PLATELET                0.000705
POTASSIUM               0.067138
PT                      0.185054
PTT                     0.188345
RDW                     0.000000
SODIUM                  0.067115
TH_FLAG                 0.000000
WBC                     0.000000
dtype: float64
```

Splitting Data

```
#You can choose to add factors like time into the targets
#targets = ['LV_RDW', 'LV_ANIONGAP', 'LV_BANDS', 'LV_BILIRUBIN', 'LV_CREAT
ININE', 'LV_CHLORIDE', 'LV_GLUCOSE', 'LV_HEMATOCRIT', 'LV_HEMOGLOBIN', 'LV
_LACTATE', 'LV_PLATELET', 'LV_POTASSIUM', 'LV_PTT', 'LV_INR', 'LV_PT', 'LV
_SODIUM', 'LV_BUN', 'LV_WBC']
targets = ['ANIONGAP', 'BANDS', 'BILIRUBIN', 'BUN', 'CHLORIDE', 'CREATININ
E', 'GLUCOSE', 'HEMATOCRIT', 'HEMOGLOBIN', 'INR', 'LACTATE', 'PLATELET', '
POTASSIUM', 'PT', 'PTT', 'RDW', 'SODIUM', 'WBC']
X = df_first_rdw[targets]
y = df_first_rdw['TH_FLAG']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)

# impute missing values with medians
X_train = X_train.apply(lambda x: x.fillna(np.nanmedian(x)), axis=0)
X_train.head()
```

	ANIONGAP	BANDS	BILIRUBIN	BUN	CHLORIDE	CREATININE	GLUCOSE	HEMATOCRIT	HEMOGLOBIN	INR	LACTATE	PLATELET	POTASSIUM	PT	PTT	RDW	SODIUM	WBC
SUBJECT_ID																		
92379	12.0	3.0	0.8	14.0	108.0	1.2	133.0	34.4	12.1	1.3	0.6	176.0	4.1	14.0	30.6	14.8	143.0	7.9
16619	11.0	3.0	1.5	19.0	112.0	0.9	134.0	34.9	11.5	1.1	1.7	219.0	4.4	13.1	32.8	21.2	144.0	9.7
23275	10.0	3.0	0.6	10.0	101.0	0.5	119.0	35.6	12.1	1.1	1.8	275.0	3.4	13.0	25.0	12.6	134.0	16.7
23911	13.0	3.0	5.5	18.0	106.0	0.9	37.0	53.3	17.7	1.3	1.8	158.0	4.1	14.0	30.6	19.7	139.0	11.9
6244	13.0	3.0	0.8	15.0	106.0	0.9	143.0	39.2	13.4	1.1	1.8	242.0	4.0	13.1	111.6	13.9	140.0	10.7

Make Models

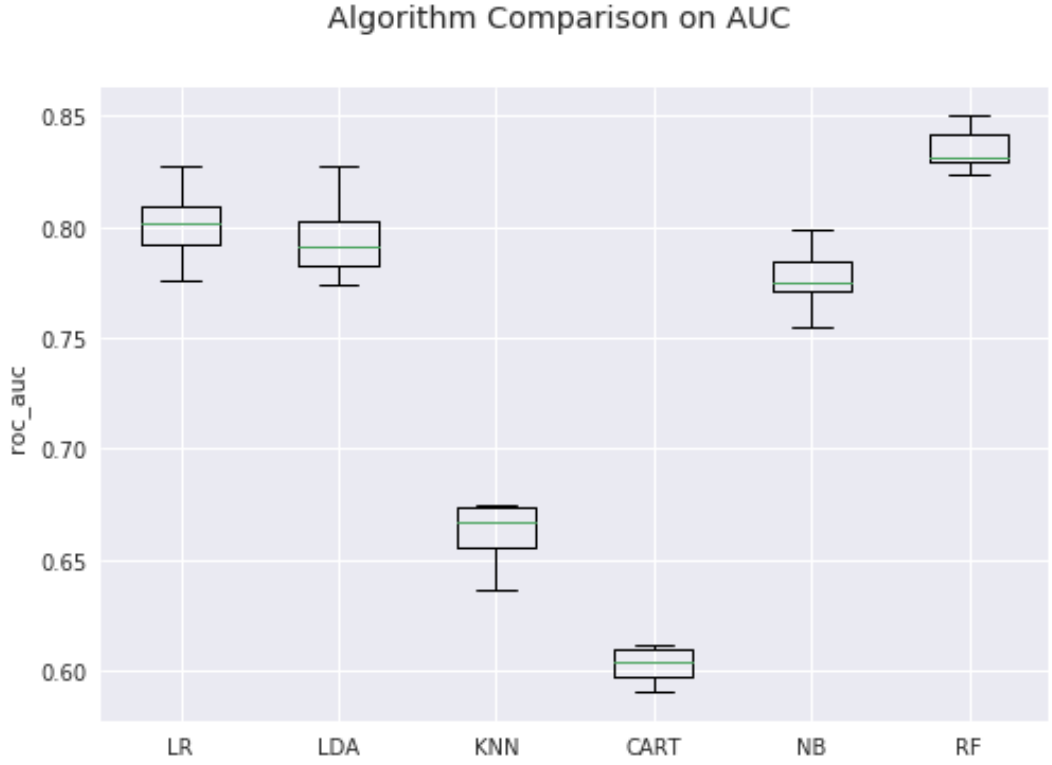
```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

# prepare configuration for cross validation test harness
# (from https://machinelearningmastery.com/compare-machine-learning-algori
thms-python-scikit-learn/)
seed = 123
# prepare models
models = []
models.append(('LR', LogisticRegression(class_weight='balanced')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
#models.append(('SVM', SVC())) # takes a long time
models.append(('RF', RandomForestClassifier(n_estimators=100, class_weight
='balanced')))
```

AUC Comparisons

```
# evaluate each model in turn
results = []
names = []
scoring = 'roc_auc' # others include: 'accuracy', 'f1', 'roc_auc',
                    # or found here: http://scikit-learn.org/stable/module
s/model_evaluation.html
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, y_train,
cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.801285 (0.013491)
LDA: 0.794056 (0.015722)
KNN: 0.663351 (0.011795)
CART: 0.607158 (0.015576)
NB: 0.776854 (0.013618)
RF: 0.832981 (0.011916)
```

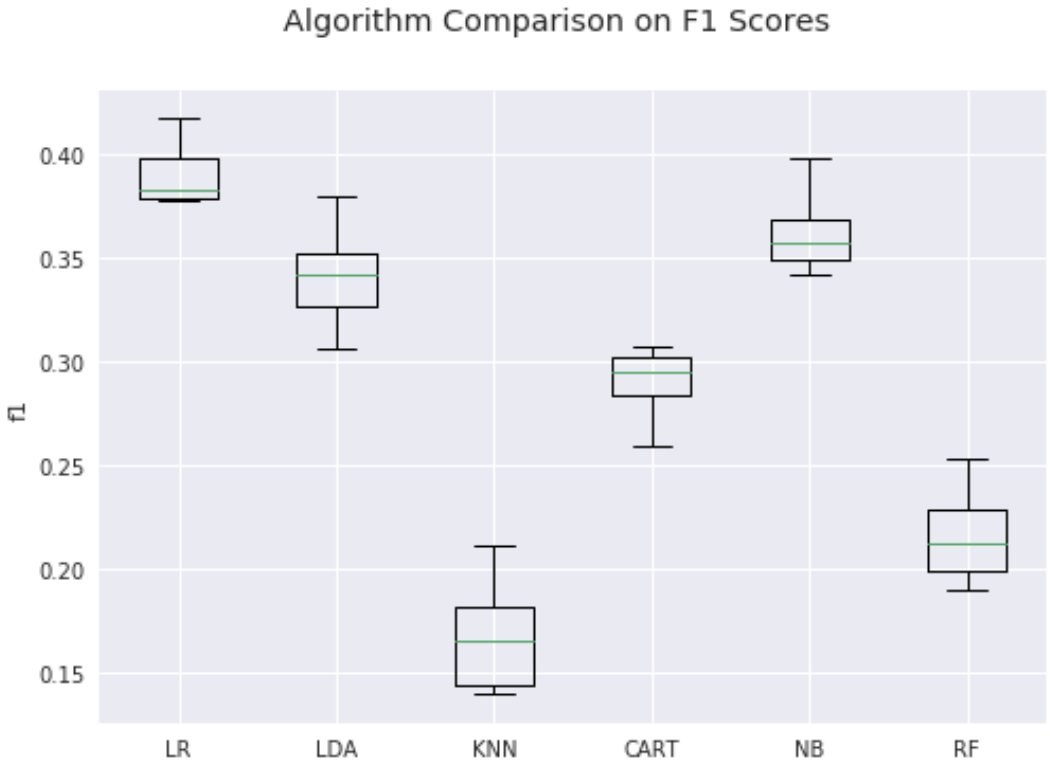
```
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison on AUC')
ax = fig.add_subplot(111)
plt.boxplot(results)
plt.ylabel(scoring)
ax.set_xticklabels(names)
plt.show()
```




```
results = []
names = []
scoring = 'f1'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, y_train,
cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.390029 (0.013872)
LDA: 0.341680 (0.023317)
KNN: 0.166617 (0.023179)
CART: 0.293222 (0.020713)
NB: 0.364083 (0.022115)
RF: 0.212223 (0.028137)
```

```
fig = plt.figure()
fig.suptitle('Algorithm Comparison on F1 Scores')
ax = fig.add_subplot(111)
plt.boxplot(results)
plt.ylabel(scoring)
ax.set_xticklabels(names)
plt.show()
```



Based on the model, performance evaluation with respect both area under the curve (AUC) and F1 scores, logistic regression and random forest approaches performed the best. Therefore, we used logistic regression and random forests to evaluate variable importance.

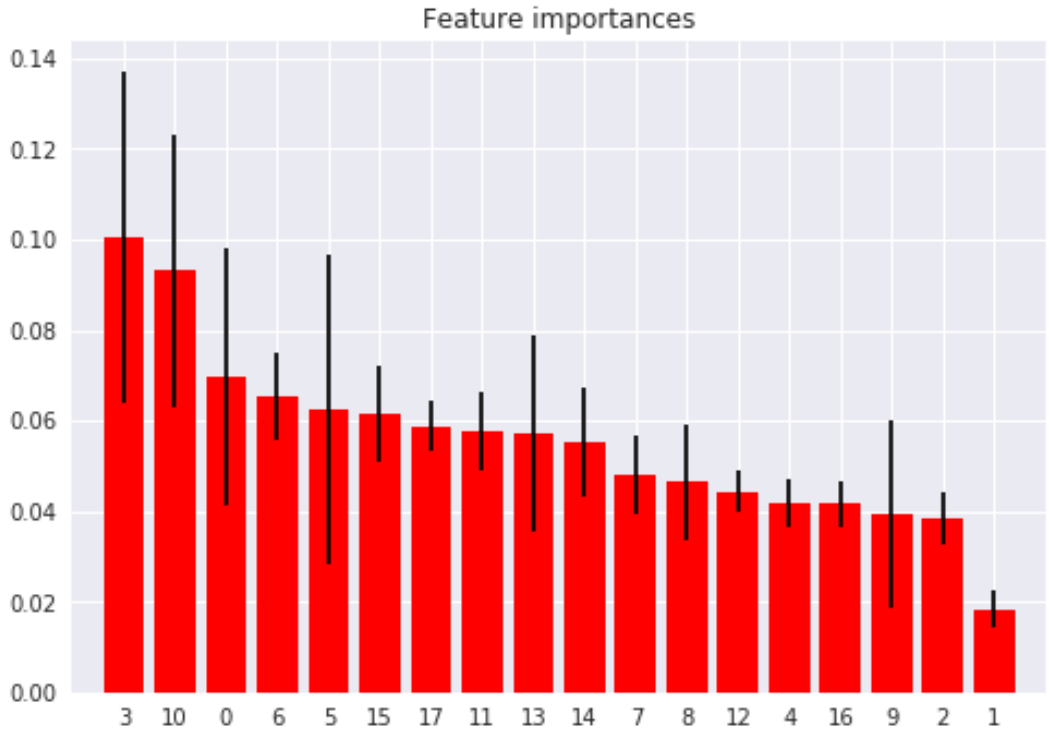
```
from sklearn.feature_selection import RFE
names = X_train.columns
model = LogisticRegression(class_weight='balanced')
rfe = RFE(model, 3) # top 3 attributes
rfe = rfe.fit(X_train, y_train)

print("Features sorted by their rank:")
print(sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), names)))
```

Features sorted by their rank: [(1, 'HEMOGLOBIN'), (1, 'INR'), (1, 'LACTATE'), (2, 'HEMATOCRIT'), (3, 'POTASSIUM'), (4, 'ANIONGAP'), (5, 'RDW'), (6, 'BANDS'), (7, 'CREATININE'), (8, 'BUN'), (9, 'WBC'), (10, 'CHLORIDE'), (11, 'SODIUM'), (12, 'PT'), (13, 'PTT'), (14, 'GLUCOSE'), (15, 'BILIRUBIN'), (16, 'PLATELET')]

```
# Random Forest
forest = RandomForestClassifier(n_estimators=100, class_weight='balanced',
random_state=seed)
forest.fit(X_train, y_train)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X_train.shape[1]):
    #print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices
[f]]))
    print("%d. %s (%f)" % (f + 1, X_train.columns[indices[f]], importances
[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X_train.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X_train.shape[1]), indices)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```

Feature Ranking:

1. BUN (0.100535)
2. LACTATE (0.093031)
3. ANIONGAP (0.069771)
4. GLUCOSE (0.065349)
5. CREATININE (0.062566)
6. RDW (0.061578)
7. WBC (0.058777)
8. PLATELET (0.057647)
9. PT (0.057211)
10. PTT (0.055198)
11. HEMATOCRIT (0.048193)
12. HEMOGLOBIN (0.046353)
13. POTASSIUM (0.044411)
14. CHLORIDE (0.041633)
15. SODIUM (0.041619)
16. INR (0.039318)
17. BILIRUBIN (0.038442)
18. BANDS (0.018370)



In both logistic regression and random forest approaches, the lactate value was ranked as 1 of the top 3 most important features. In our original hypothesis that RDW values would be important, we found the RDW value ranked as #5 in the logistic regression and #6 in the random forest.

Discussion

Until recently, RDW values have not been considered as helpful measures in understanding the diagnosis and pronostication of patients with certain diseases. As a fairly common component for patients in the ICU, we sought to use this data to pull insights from and analyze the relationship between RDW values and other critical hospital measures.

Our first question dealt with the measuring the association between RDW values and other commonly-measured laboratory values. Because RDW is measured so frequently, there is potential of hidden patterns that could have significant correlations. In our analysis, there was very little correlation between RDW and any of the other major lab values. As shown in the graphs, the increase in RDW did not seem to cause any significant change in the associated lab value. We then created a LOESS curve to better visualize the trends between different values.

The next question we analyzed was whether or not certain diseases were associated with the elevated RDW values upon ICU admission. We first created tables that contained data for the subject ID, hospital admission ID, and count for the number of corresponding diagnoses. While we initially classified the diagnoses based on whether they corresponded with an RDW value that was greater or less than the standard, we then revised our query to generate the average RDW values and ZScores for each of the diagnoses. This not only gave us an idea of the which diagnoses corresponded with higher RDW values, but also how far off the mean the observations were. In our analysis, we found that diseases such as HIV (RDW of 19.38) and Yeast Infection (RDW of 19.10) were associated with a much higher average RDW value, while other diseases such as Analgesic Poisoning (RDW of 14.91) and Aortic valve disorder (RDW of 16.24) were associated with much lower values overall.

Our third question sought to measure the changes from ICU admission to ICD discharge. After cleaning the data and separating our values by patient, we were then able to analyze the RDW time series for each patient. After creating our own dataframe to store the features of the time series, we calculated and created a visual representation of the RDW Deltas throughout the stay. This was a fairly standard distribution with a few outliers, though it did indicate that the RDW values increase on average. An average delta of 0 indicates that RDW values do not change from admission to departure. We also plotted the distribution of RDW standard deviations, which peaked at a value of around 0.4, further supporting the relatively low distribution of RDW deltas. As for the analysis of the time series, we measured the distribution of autocorrelation both at a lag of 1 time period and lag of 2. For the 1 period lag, the autocorrelation was much higher, as most values were very similar to their predecessors. For a 2 period lag, we can see a greater spread of autocorrelation values as there is a greater difference in time. After finding the autocorrelation of the different time series, we plotted the RDW values associated with some of the highest and lowest autocorrelation values. Our analysis showed some interesting behavior, in which the low values for the autocrrelation for the lag of 2 were quite irregular, with many jumps between RDW values for the time measured. The high autocorrelation values were much more stable and consistent throughout time. The histograms show that there are more "flat" time series with a more stability, meaning that RDW would usually have a more steady change over time. Finally, we plotted the autocorrelation values with different lags, from a time period of one step back up to six. Our results suggest that RDW time series seem to be non-periodic within lag intervals of 1-6 since the flattening distribution is expected from a rather random time series, since autocorrelation generally becomes random

when the change in series is random. However, there is much more positive autocorrelation than negative autocorrelation, meaning that RDW values usually trend in one direction and remain that way. If the time series were more highly periodic, we would expect to see peaks in the histogram at different levels. In order to create our visualizations, we did have to manipulate the data to fit with our models. For one, we had to cut off about 2/3 of the total data to operate on time series with more than 10 observations. We believed this to be a reasonable step since statistical analysis on time series with less than 10 observations would yield too much variance and too little value. Secondly, the combined autocorrelation histogram had a different y-axis range than the individual plots as it was necessary to use more bins. It is also noted that the time series for the RDW observations were not recorded at regular time intervals and the data was only present for times of measurement. To make our procedure more thorough, we could have used interpolation to fill in even time intervals, which would have required more resources and approximation.

The RDW value was not as important of a variable in the high-performing models as we had hypothesized. However, our modeling approaches were very simple to date, and our imputation approach is sub-standard at the moment. Interestingly, lactate levels were ranked as very important in both models, but this could be a result of the median imputation approach. In the future, we will attempt to impute a missing value of "0" because in the clinical arena, missing lactate values are more likely to indicate a value of "0" (i.e., clinicians are not requesting lactate levels because the patient is not sick enough to warrant this value being measured).

Conclusion

In our analysis, we tackled some of the problems that we believed to have significant implications for hospital ICU stays. RDW values are common measurements for intensive care patients, and have seen to be associated with mortality and bloodstream infections. While our study was to identify associations between RDW and other patient characteristics, there are still a number of analyses that would further aid our research. For one, we plan to create a random forest that would use regression on any data available pre-admission (chronic diseases, age, gender, etc.) and use it as a predictor for an RDW outcome. Secondly, we hope to collapse our ICD codes as a form of PheCodes, to facilitate phenome-wide association studies. In our analysis, we did not gain much further insight on the associations between RDW values and other relevant lab values, but found some very interesting information regarding the associations between the RDW values and ICD codes. There are likely certain diagnoses that would not be associated with a higher RDW value, with this analysis it is clear that there could be some hidden factors that could cause such a correlation. While our project only focused on a few key questions, there are many further insights that we hope to uncover.

References

Bazick, H. S., Chang, D., Mahadevappa, K., Gibbons, F. K., & Christopher, K. B. (2011). Red cell distribution width and all cause mortality in critically ill patients. *Critical Care Medicine, 39*(8), 1913.

Danese, E., Lippi, G., & Montagnana, M. (2015). Red blood cell distribution width and cardiovascular diseases. *Journal of Thoracic Disease, 7*(10), E402.

Hu, Z. D., Wei, T. T., Tang, Q. Q., Fu, H. T., Yang, M., Ma, N., ... & Zhong, R. Q. (2016). Prognostic value of red blood cell distribution width in acute pancreatitis patients admitted to intensive care units: An analysis of a publicly accessible clinical database MIMIC II. *Clinical Chemistry and Laboratory Medicine (CCLM), 54*(7), e195-e197.

Hunziker, S., Celi, L. A., Lee, J., & Howell, M. D. (2012). Red cell distribution width improves the simplified acute physiology score for risk prediction in unselected critically ill patients. *Critical Care, 16*(3), R89.

Jeffery, A.D., Dietrich, M.S., Fabbri, D., Kennedy, B., Novak, L.L., Coco, J., & Mion, L.C. (In-Press). Advancing In-Hospital Clinical Deterioration Prediction Models. *American Journal of Critical Care.*

Johnson, A. E., Stone, D. J., Celi, L. A., & Pollard, T. J. (2017). The MIMIC Code Repository: Enabling reproducibility in critical care research. *Journal of the American Medical Informatics Association, 25*(1), 32-39.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., ... & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data, 3,* 160035.

Tonelli, M., Sacks, F., Arnold, M., Moye, L., Davis, B., & Pfeffer, M. (2008). Relation between red blood cell distribution width and cardiovascular event rate in people with coronary disease. *Circulation, 117*(2), 163-168.

In [ ]:

In [ ]: