PHASE 4:
PROJECT TITLE: CREDIT CARD FRAUD DETECTION
INTRODUCTION:

In proposed system, I present a behavior and Location Analysis (BLA).Which does not require fraud signatures and yet is able to detect frauds by considering a cardholder's spending habit. Card transaction processing sequence by the stochastic process of an BLA. The details of items purchased in Individual transactions are usually not known to any Fraud Detection System (FDS) running at the bank that issues credit card to the card holders. Hence I feel that BLA is an ideal choice for addressing this problem. Another important advantage of the BLA -based approach is a drastic reduction in the number of False Positives transactions identified as malicious by an FD Sal though they are actually genuine. An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification.FDS receive the card details and the value of purchase to verify, whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find any anomaly in the transaction based on the spending profile of the cardholder, shipping address, and billing address, etc. If the FDS confirms the transaction to be of fraud, it raises an alarm, and the issuing bank declines the transaction.

The credit card fraud detection features uses user behavior and location scanning to check for unusual patterns. These patterns include user characteristics such as user spending patterns as well as usual user geographic locations to verify his identity. If any unusual pattern is detected, the system requires re-verification.

The system analyses user credit card data for various characteristics. These characteristics include user country, usual spending procedures. Based upon previous data of that user the system recognizes unusual patterns in the payment procedure. So now the system may require the user to login again or even block the user for more than 3 invalid attempts.

FEATUREENGINEERING:

Project Overview:

Credit fraud detection is a crucial application of data science that aims to identify and prevent fraudulent transactions in the financial industry. Effective feature engineering plays a significant role in building accurate and efficient fraud detection models. This report outlines the feature engineering process for a credit fraud detection project.

1. Data Preprocessing:

Before diving into feature engineering, it is essential to preprocess the data. This typically involves handling missing values, encoding categorical variables, and scaling numerical features.

2. Feature Extraction:

Feature extraction involves creating new features from the existing data that may provide additional information to the model. Here are some common feature extraction techniques for credit fraud detection:

- **Transaction Amount Statistics:** Calculate statistics (mean, median, standard deviation) for transaction amounts within specific time windows. This can help identify unusual transaction amounts.
- **Transaction Frequency:** Calculate the frequency of transactions for a specific card or account within a given time frame. Unusual spikes in transaction frequency may indicate fraud.
- **Time-based Features**: Extract information from the transaction timestamp, such as the day of the week, hour of the day, or time since the last transaction. This can help capture temporal patterns in fraud.
- **Geographical Features:** If available, incorporate the location of the transaction (e.g., latitude and longitude) and assess whether it matches the cardholder's usual locations.
- **Merchant Category:** Create features that encode the type of merchant where the transaction occurred. Certain merchant categories may have a higher likelihood of fraud.
- **Transaction History:** Utilize historical transaction information for each account or card to capture patterns and anomalies in a cardholder's spending behavior.

3. Dimensionality Reduction:

Feature selection and dimensionality reduction techniques can be employed to reduce the complexity of the dataset and remove irrelevant or redundant features. Common methods include:

Correlation Analysis: Identify and remove highly correlated features to reduce multi co linearity.

Feature Importance: Use techniques such as tree-based algorithms to evaluate the importance of each feature and select the most relevant ones.

Principal Component Analysis (PCA): If the dataset is high-dimensional, PCA can be used to reduce dimensionality while retaining most of the variance in the data.

4. Feature Scaling:

Ensure that all numerical features are appropriately scaled to prevent some features from dominating the model. Common scaling techniques include standardization and min-max scaling.

5. Anomaly Detection:

In addition to creating new features, consider in corporating anomaly detection algorithms (e.g., Isolation Forest, One-Class SVM) to identify outliers in the data, which may be indicative of fraud.

6. Model-Specific Features:

Different machine learning algorithms may benefit from specific types of features. For example, if using deep learning models, embedding layers can be employed for categorical variables. Gradient boosting algorithms can take advantage of interaction features.

7. Cross-Validation:

Ensure that feature engineering is performed within the cross-validation loop to prevent data leakage and to evaluate the impact of feature engineering on model performance effectively.

8. Monitoring and Updating:

Feature engineering is an iterative process. After model deployment, continuously monitor the performance of the fraud detection system and update features as needed to adapt to evolving fraud patterns.

MODELTRAINING:

1) **Data Collection:** Gather a comprehensive dataset containing transaction details, including both legitimate and fraudulent cases.
2) **Feature Selection:** Choose relevant features that have a significant impact on fraud detection, such as transaction amount, location, and customer history.
3) **Model Selection:** Opt for machine learning algorithms like logistic regression, random forests, or deep learning techniques like neural networks that are suitable for classification tasks.
4) **Training and Validation:** Split the dataset into training and validation sets to train the model and evaluate its performance using metrics like accuracy, precision, recall, and F1-score.
5) **Hyper parameter Tuning:** Fine-tune the model parameters to optimize its performance.
6) **Deployment:** Implement the model into a real-time or batch processing system for continuous fraud detection.
7) **Monitoring and Updating:** Continuously monitor the model's performance and update it as new data becomes available to adapt to evolving fraud patterns.

EVALUTION:

Credit fraud detection in data science is a critical application. Effective evaluation involves metrics like accuracy, precision, recall, and F1-score. Additionally, the area under the ROC curve(AUC)and the confusion matrix are essential tools for assessing model performance. Cross-validation and stratified sampling are used to ensure robustness. Anomaly detection techniques, machine learning algorithms, and deep learning models are commonly employed. Timeliness and scalability are key concerns. Real-world testing with historical and fresh data is crucial. Continuous monitoring and adaptation are necessary to combat evolving fraud patterns. Balancing false positives and false negatives while maintaining high accuracy is the ultimate goal.

PROGRAM:

```
Import numpy as np
Import pandas as pd
Import matplotlib.pyplot as plt
Import seaborn as sns
From ipy widgets import interact,FloatSlider

from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.model_selection importtrain_test_split,GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,roc_curve,precision_recall_curve

from imblearn.under_sampling import TomekLinks
from imblearn.over_sampling import SMOTE


pd.options.display.max_columns=100
pd.options.display.max_rows=100
pd.options.display.width=100
plt.style.use('ggplot')

df=pd.read_csv('creditcard.csv') df.head()
```

OUTPUT1:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196

df.shape

OUTPUT2:

(284807,31)

df.isnull().sum().sum()

OUTPUT3:

0

df.describe()

OUTPUT4:

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848(
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927(
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.1943
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.3216
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086.
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.0007

df.dtypes

OUTPUT5:

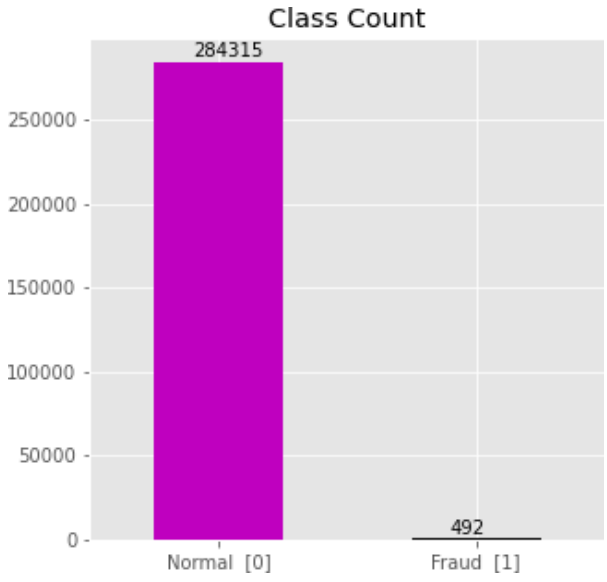
Time float64
V1 float64
V2 float64
V3 float64
V4 float64
V5 float64
V6 float64
V7 float64
V8 float64
V9 float64
V10 float64
V11 float64
V12 float64

```
V13       float64
V14       float64
V15       float64
V16       float64
V17       float64
V18       float64
V19       float64
V20       float64
V21       float64
V22       float64
V23       float64
V24       float64
V25       float64
V26       float64
V27       float64
V28       float64
Amount    float64
Class       int64
dtype: object
```

INPUT:

```
diff_class = df['Class'].value_counts()
diff_class.plot(kind='bar', color=['m', 'k'], figsize=(5, 5))
plt.xticks(range(2),['Normal[0]','Fraud[1]'],rotation=0)

 for i, v in enumerate(diff_class):
    plt.text(i-0.1,v+3000,str(v))
plt.title('Class Count')
plt.show()
```

OUTPUT6:

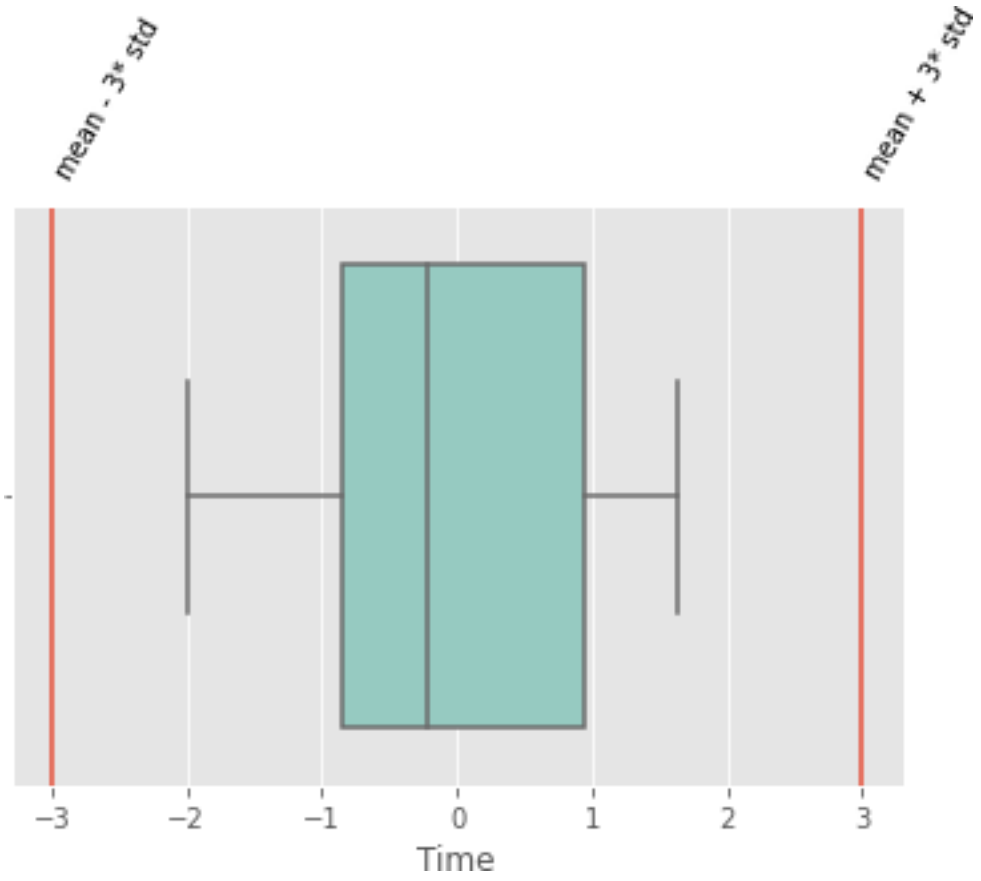


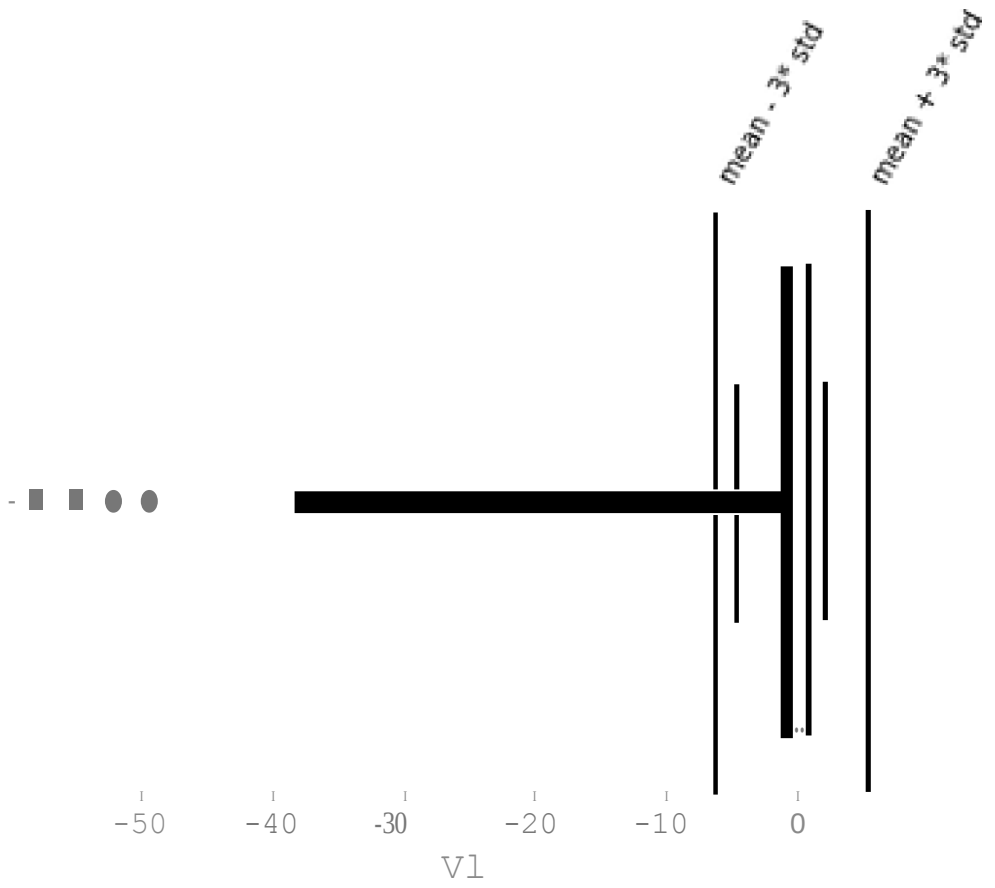
```
ss=StandardScaler()
df['Amount']=ss.fit_transform(df[['Amount']])
df['Time'] = ss.fit_transform(df[['Time']])
```

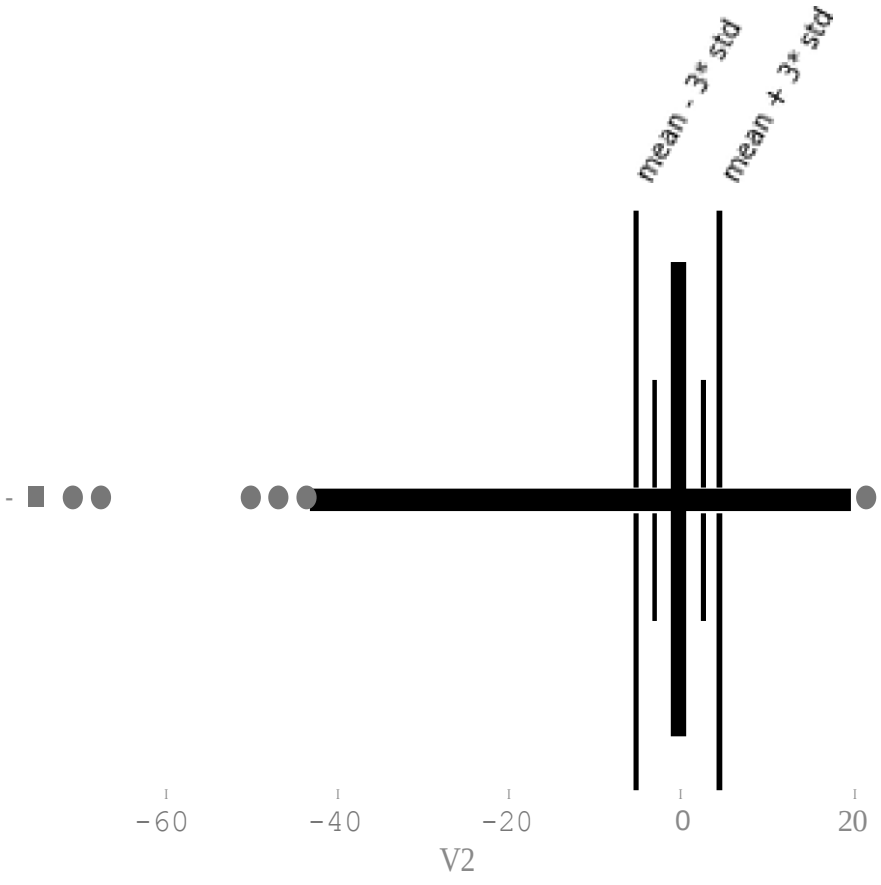
#Distributionofdifferentcolumns. for

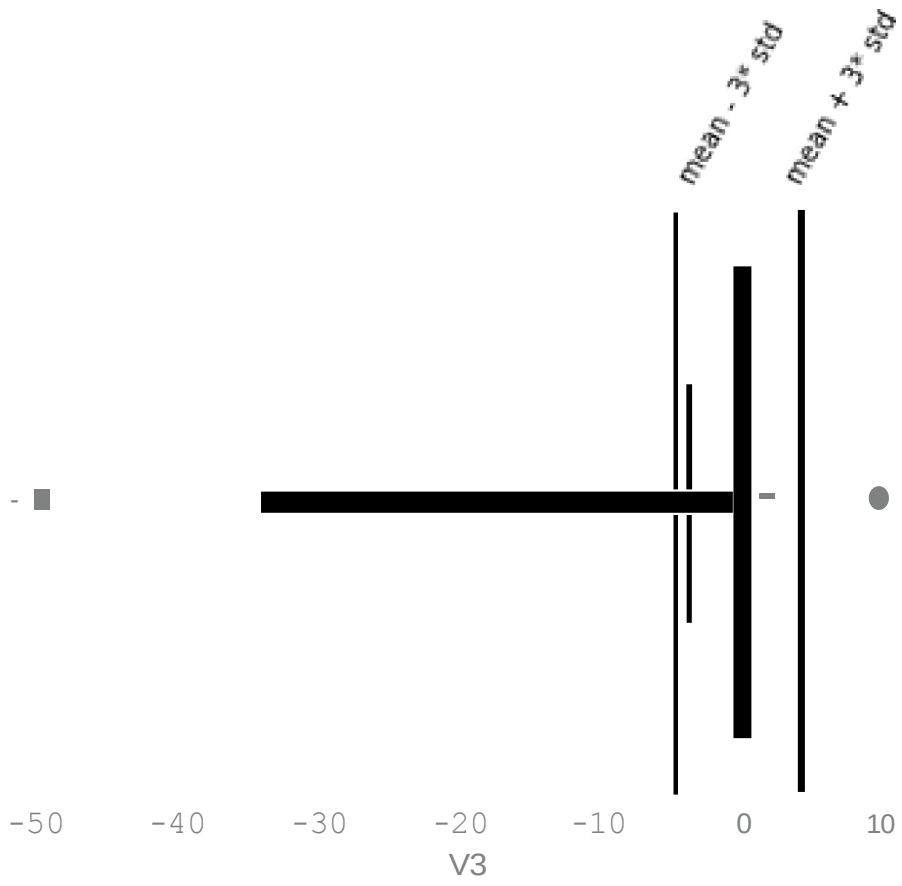
```
var in df.columns[:-1]:

    sns.boxplot(df[var],hue=df['Class'],palette='Set3')
    mean = df[var].mean()
    std = df[var].std()
    plt.axvline(mean-3*std,0,1)
    plt.text(mean-3*std,-0.55,'mean-3*std',rotation=60)
    plt.axvline(mean + 3 * std, 0, 1)
    plt.text(mean+3*std,-0.55,'mean+3*std',rotation=60) plt.show()
```

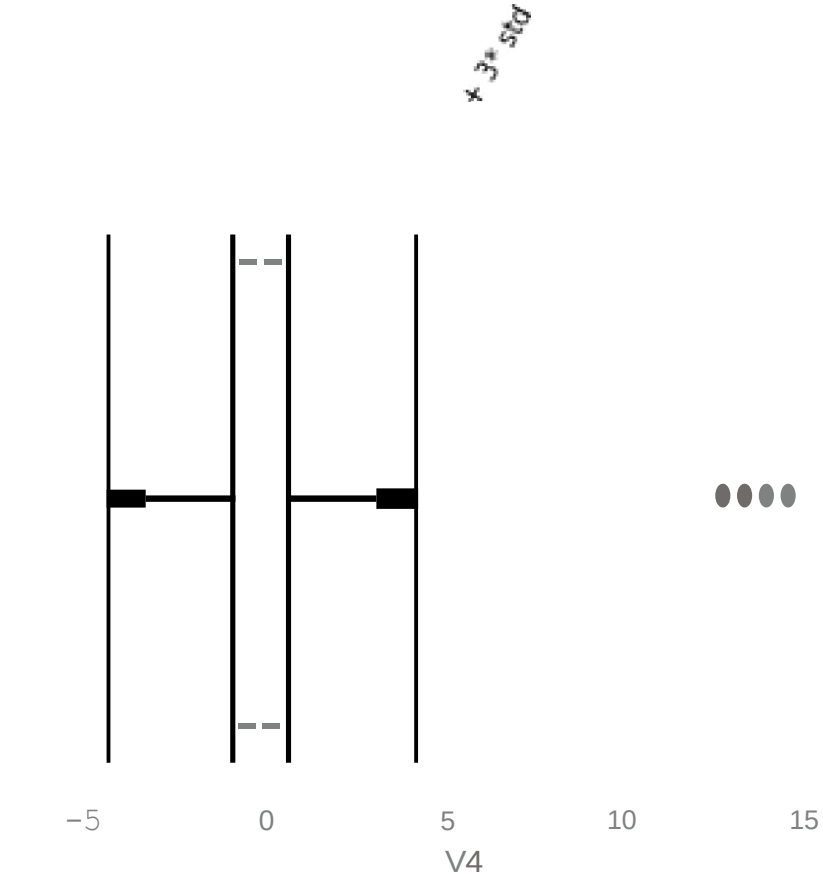
OUTPUTFORDATAINTRAININGANDDATASET:

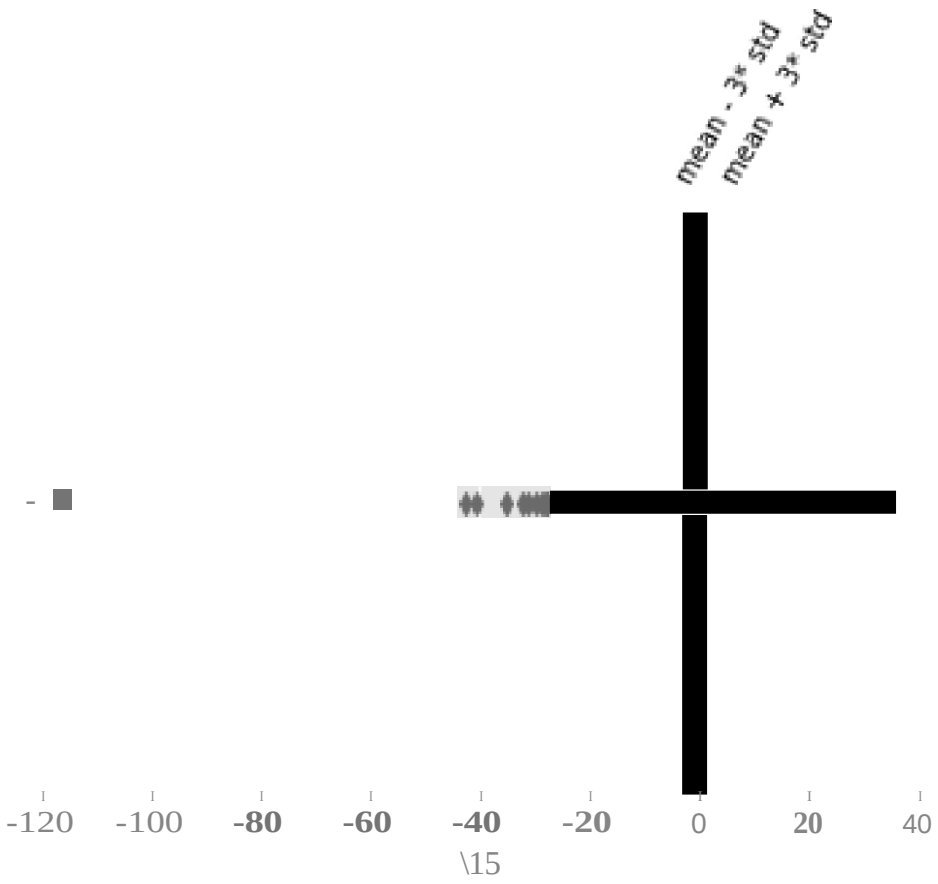


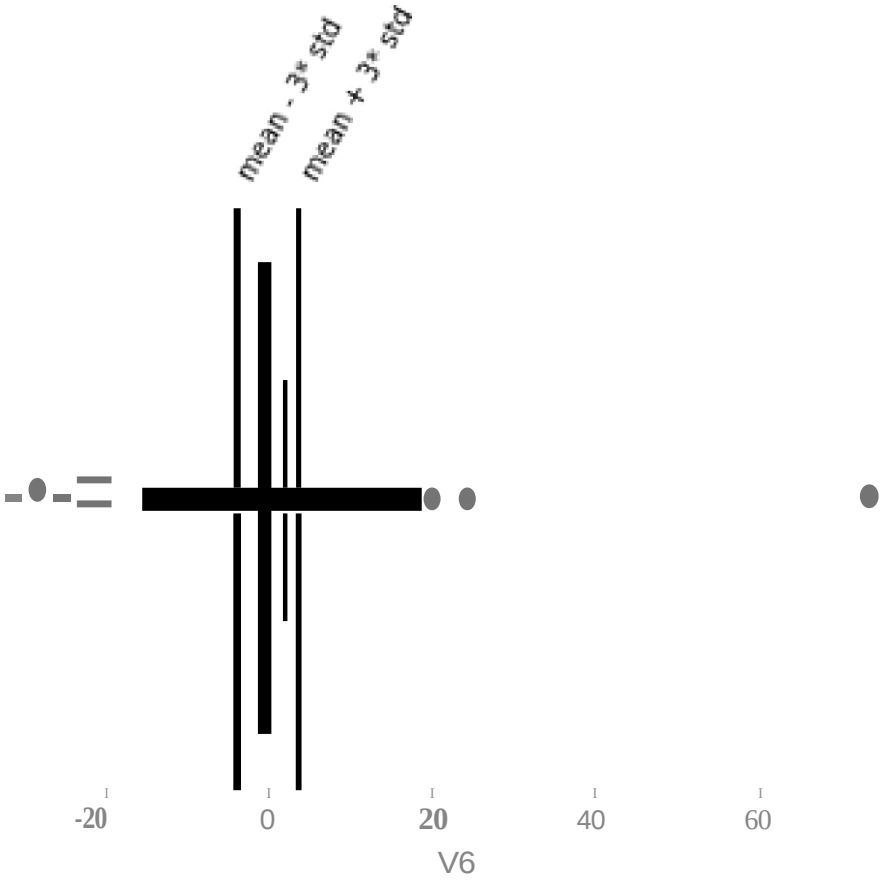

mean - 3* std
mean + 3* std
-50
-40
-30
-20
-10
0
V1

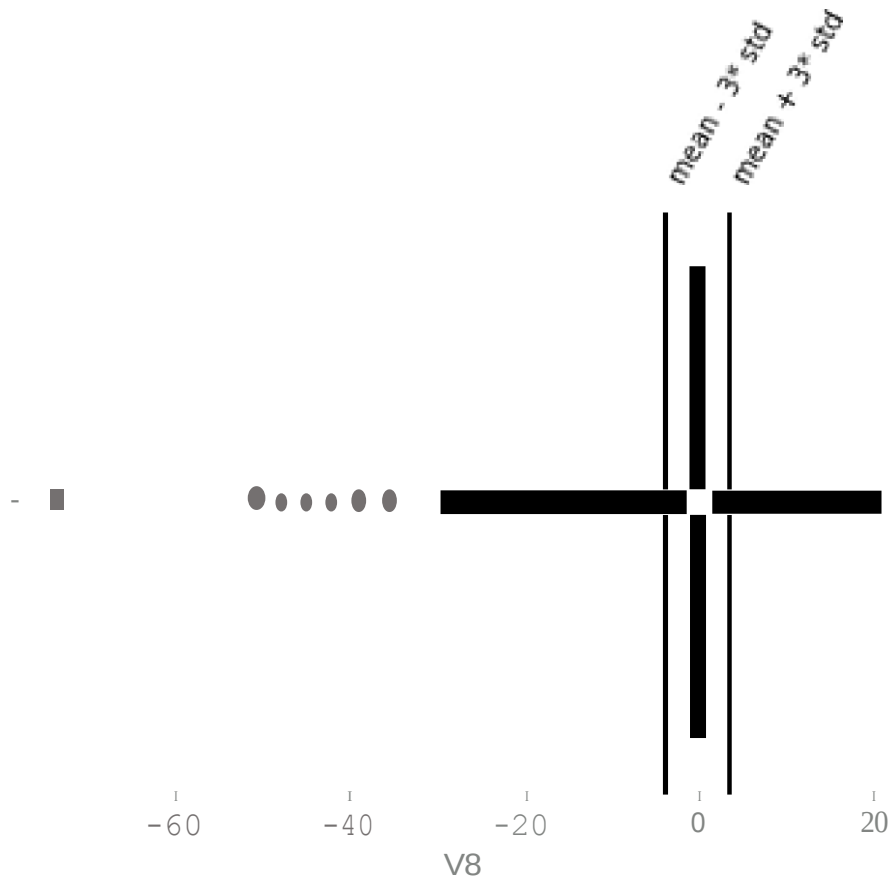

mean - 3* std
mean + 3* std
-60
-40
-20
0
20
V2

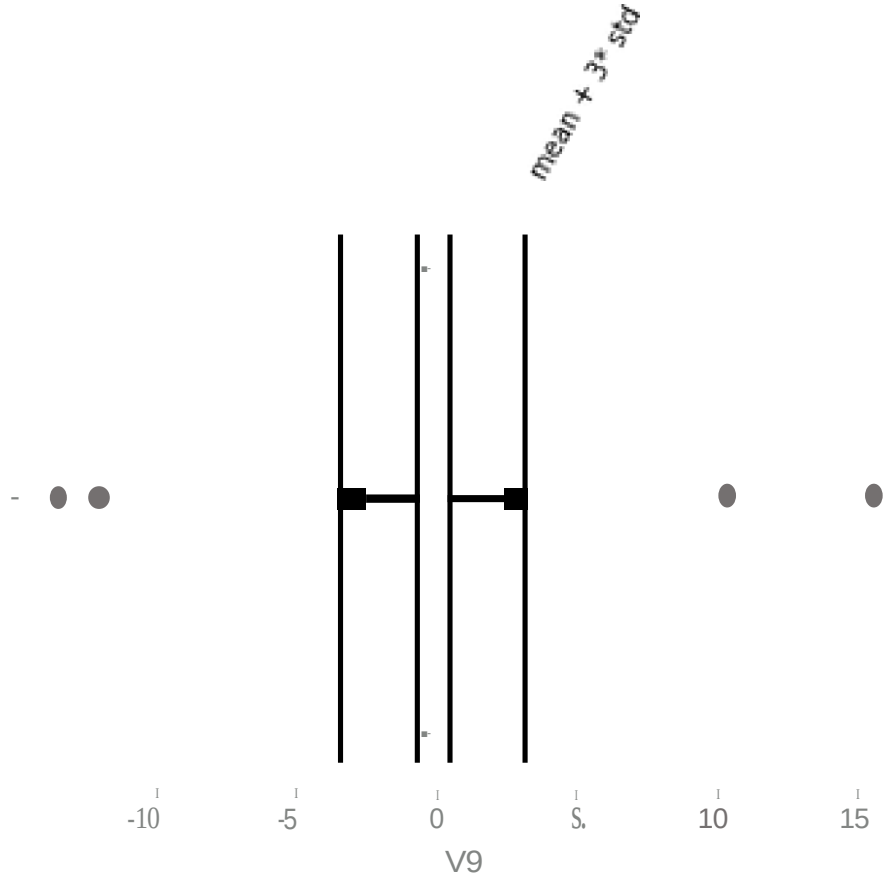

mean - 3* std
mean + 3* std
-50
-40
-30
-20
-10
0
10
V3

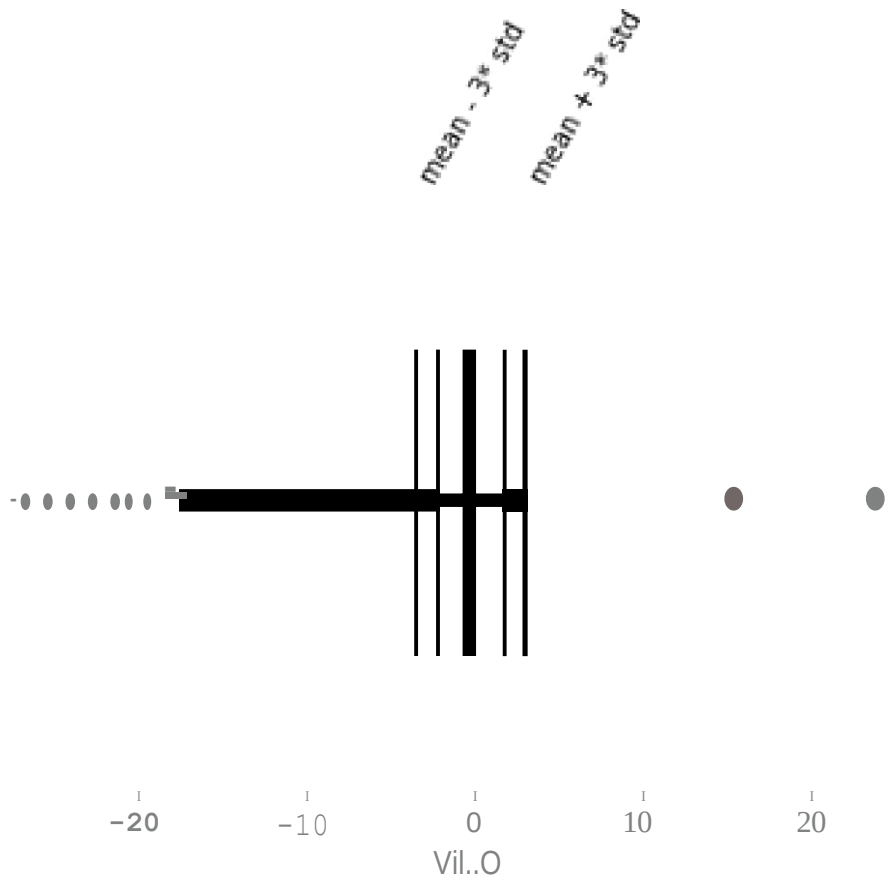

+ 3* std
-5
0
5
10
15
V4

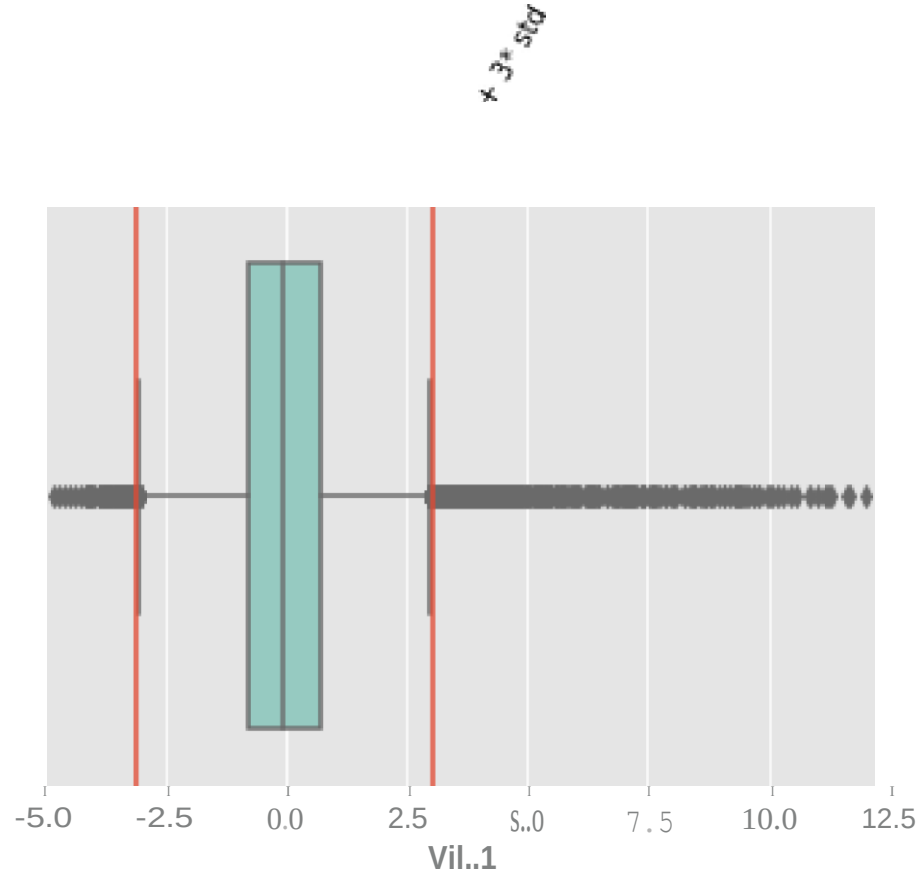

mean - 3* std
mean + 3* std
-120
-100
-80
-60
-40
-20
0
20
40
\15

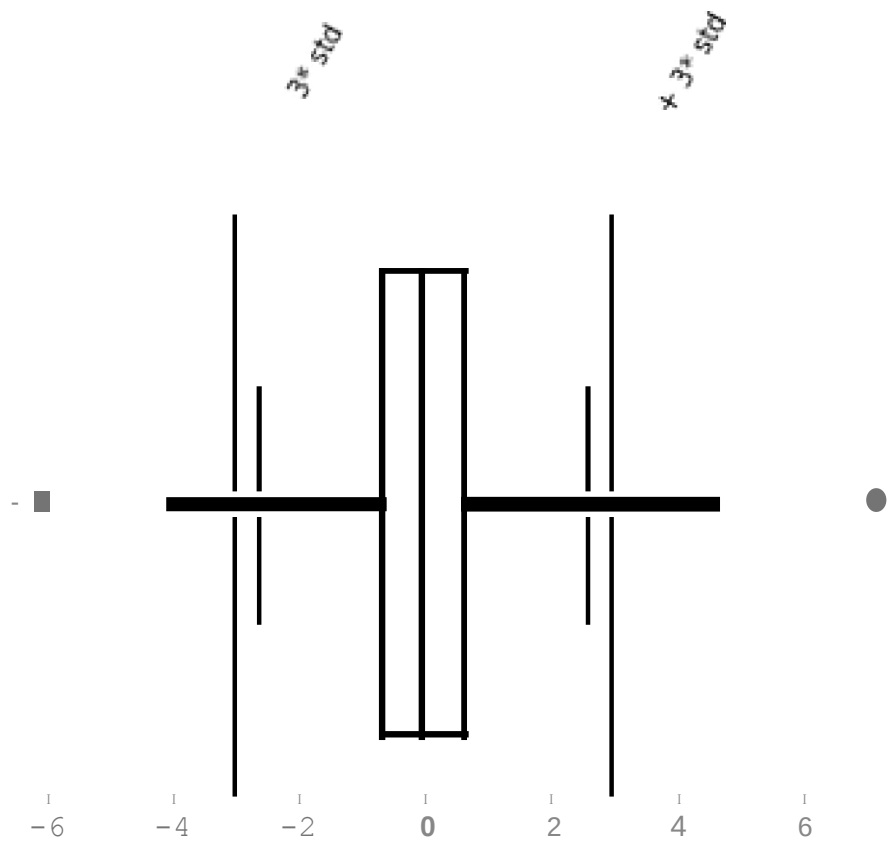

mean - 3* std
mean + 3* std
-20
0
20
40
60
V6


mean - 3* std
mean + 3* std
-60
-40
-20
0
20
V8

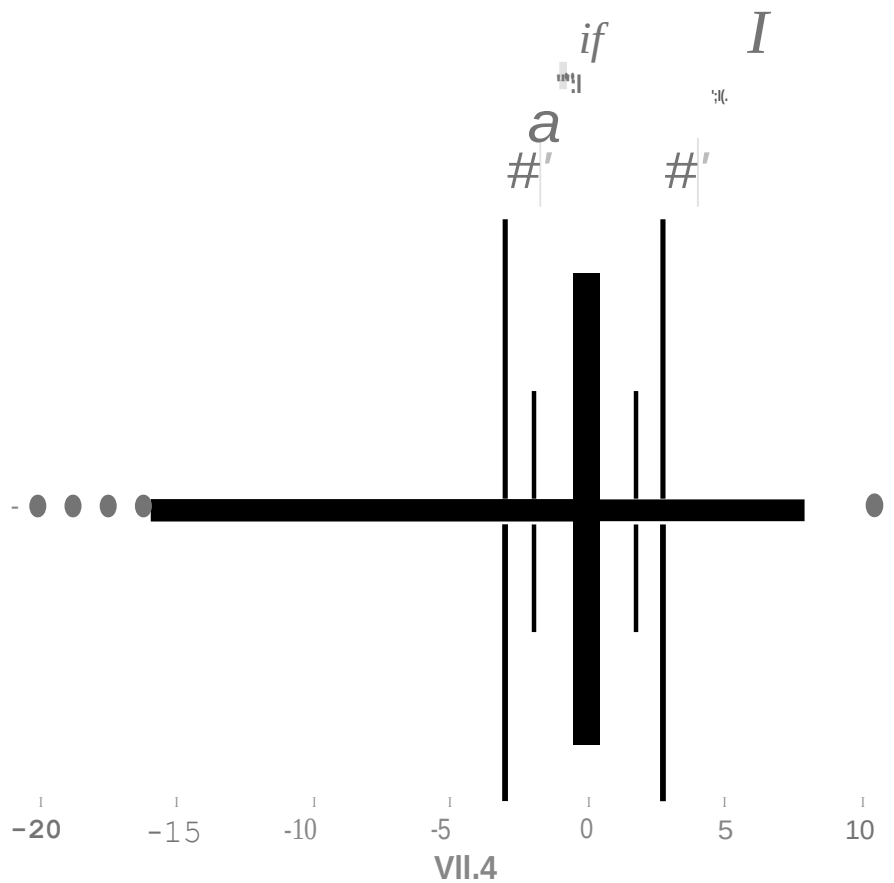

mean + 3* std
-10
-5
0
10
15
V9


mean - 3* std
mean + 3* std
-20
-10
0
10
20
Vil..O

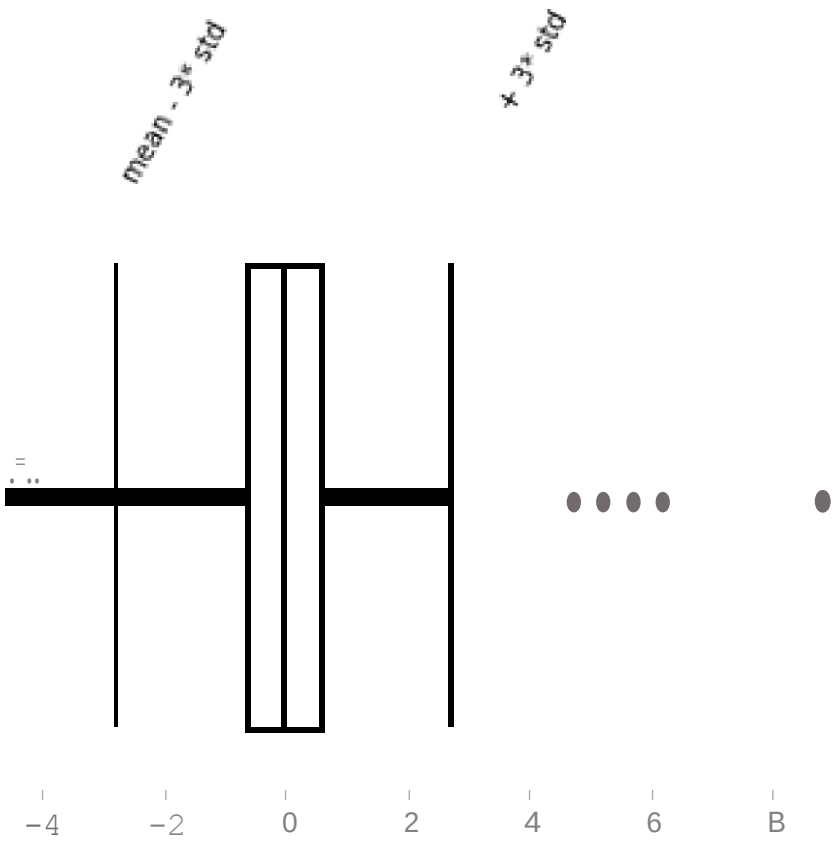

+ 3* std
-5.0
-2.5
0.0
2.5
5.0
7.5
10.0
12.5
Vil..1

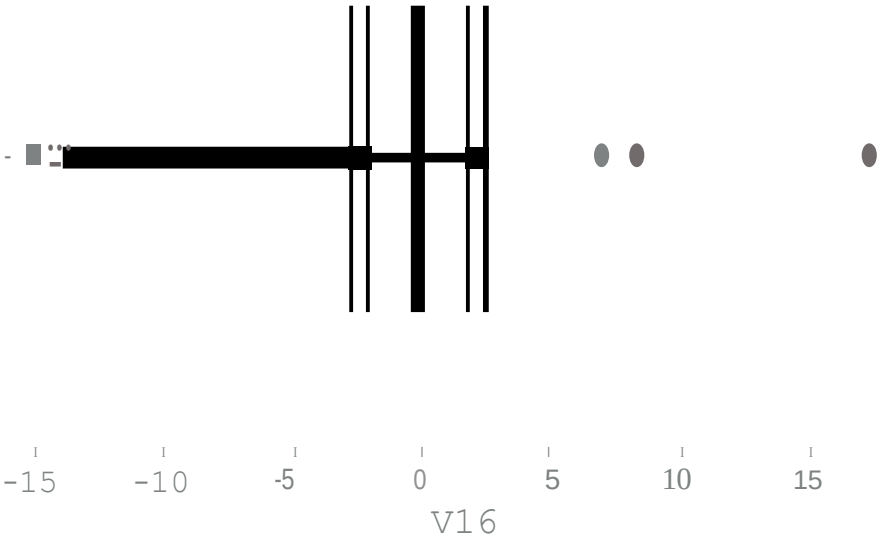


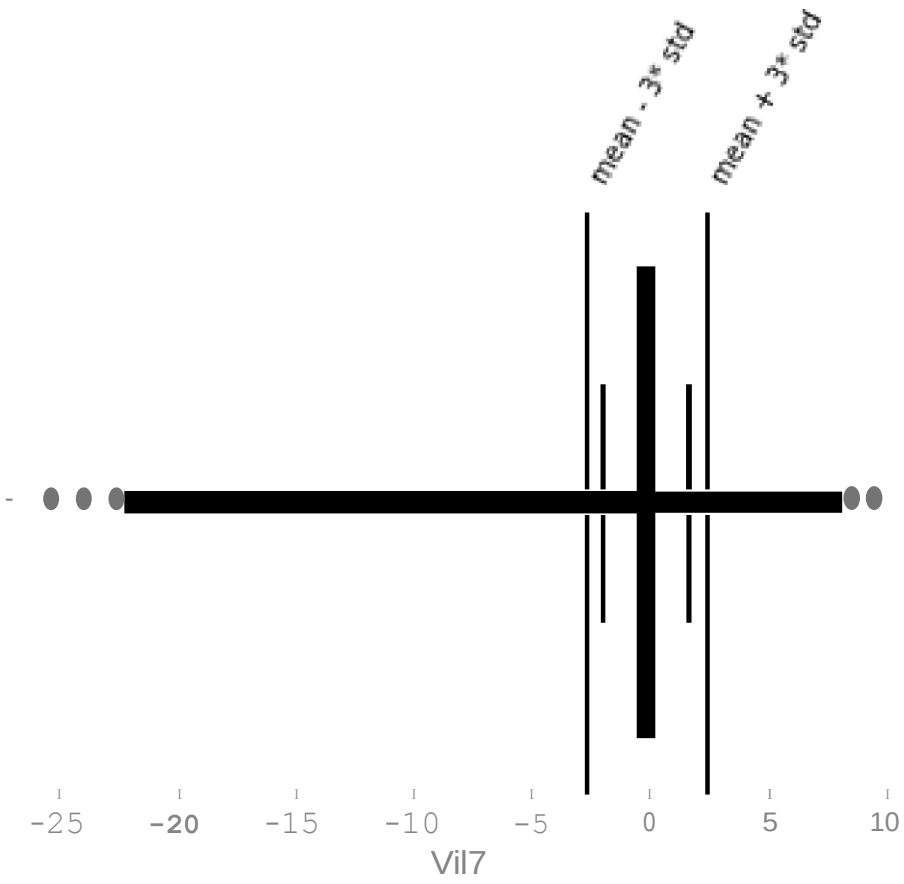
Vil..3

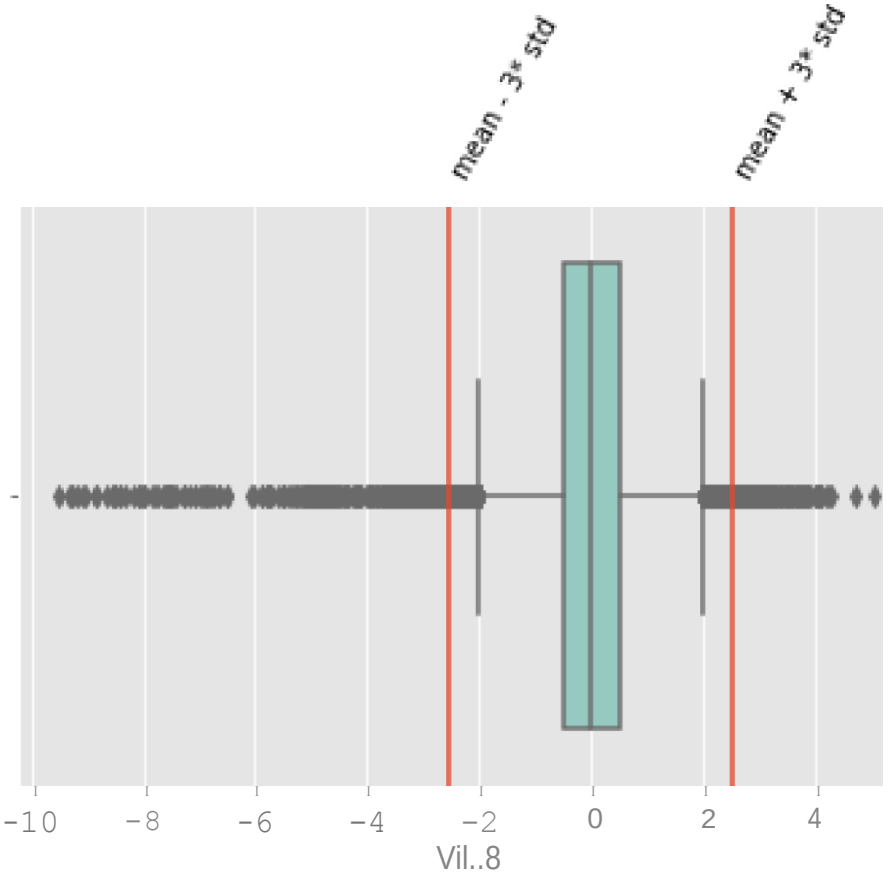


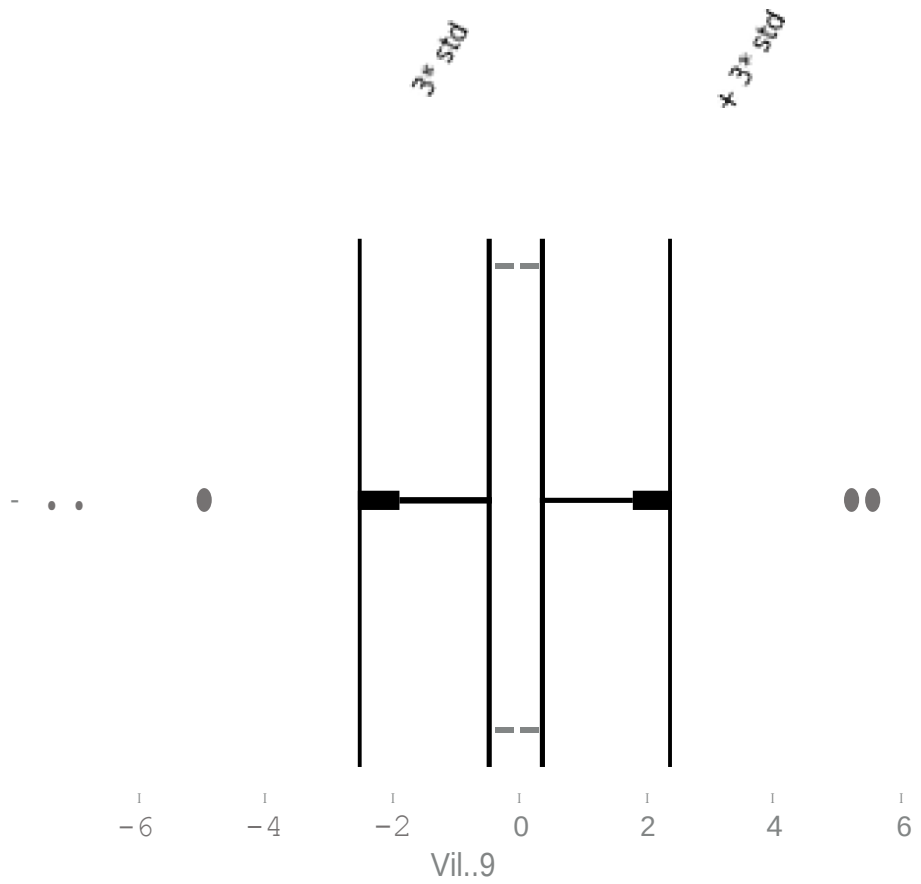
VII.4


mean - 3* std
+ 3* std
−4
−2
0
2
4
6
B
Vil..5

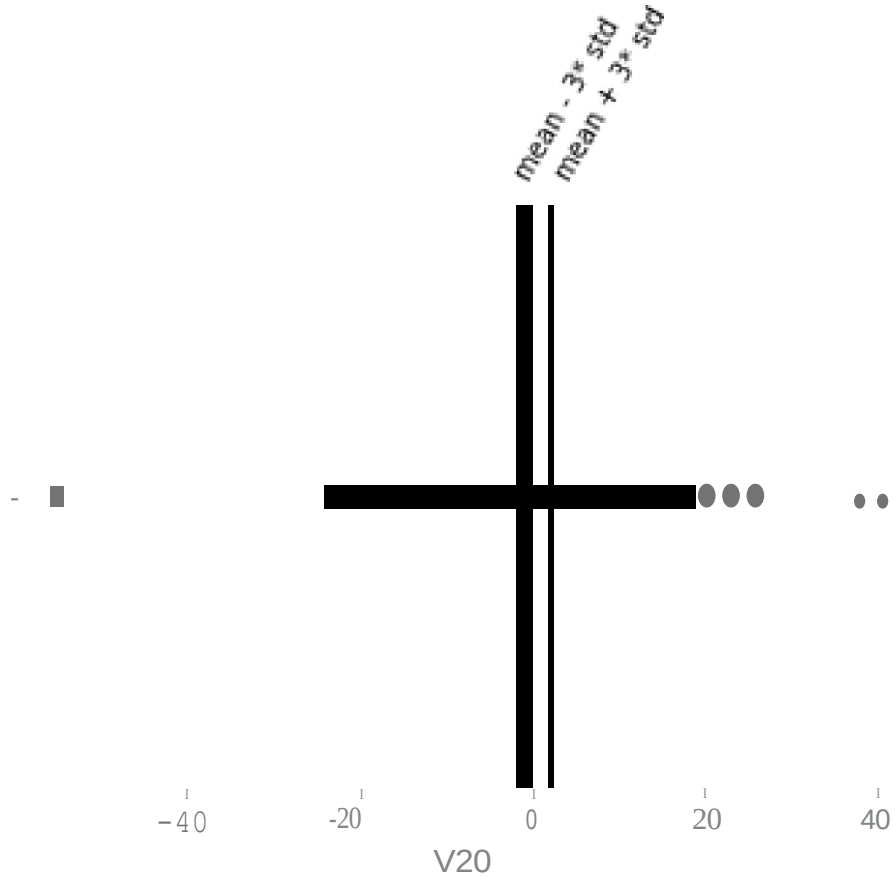

-15
-10
-5
0
5
10
15
V16


mean - 3* std
mean + 3* std
-25
-20
-15
-10
-5
0
5
10
Vil7

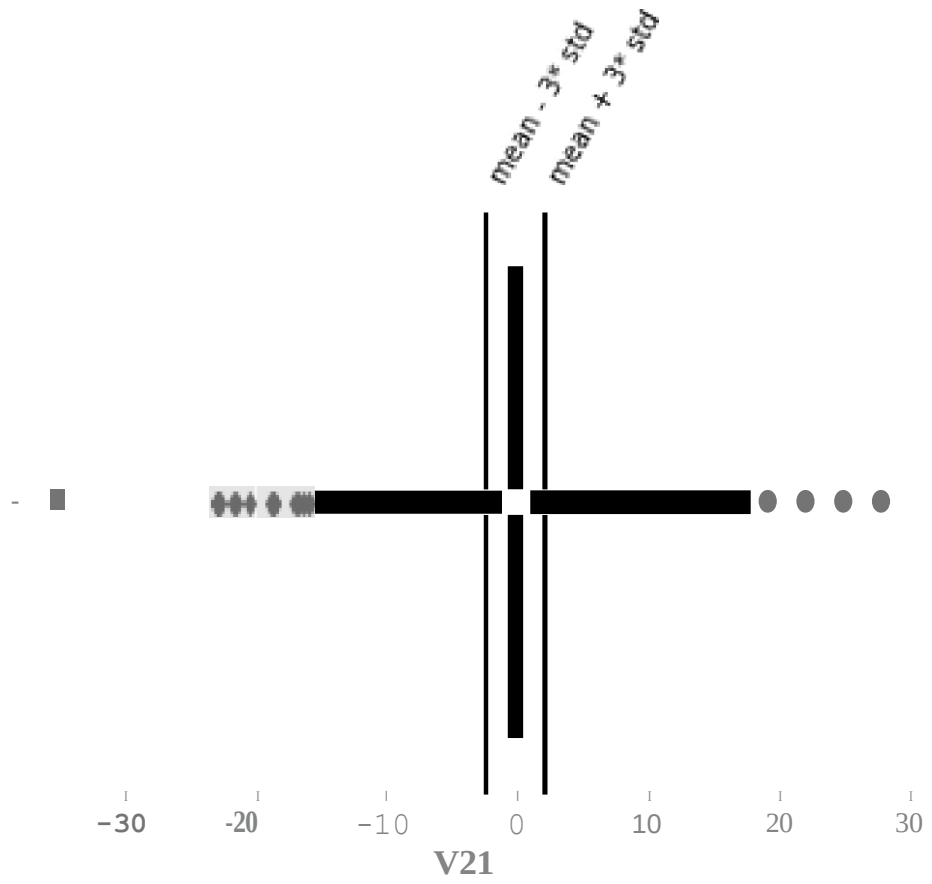

mean - 3* std
mean + 3* std
-10
-8
-6
-4
-2
0
2
4
Vil..8

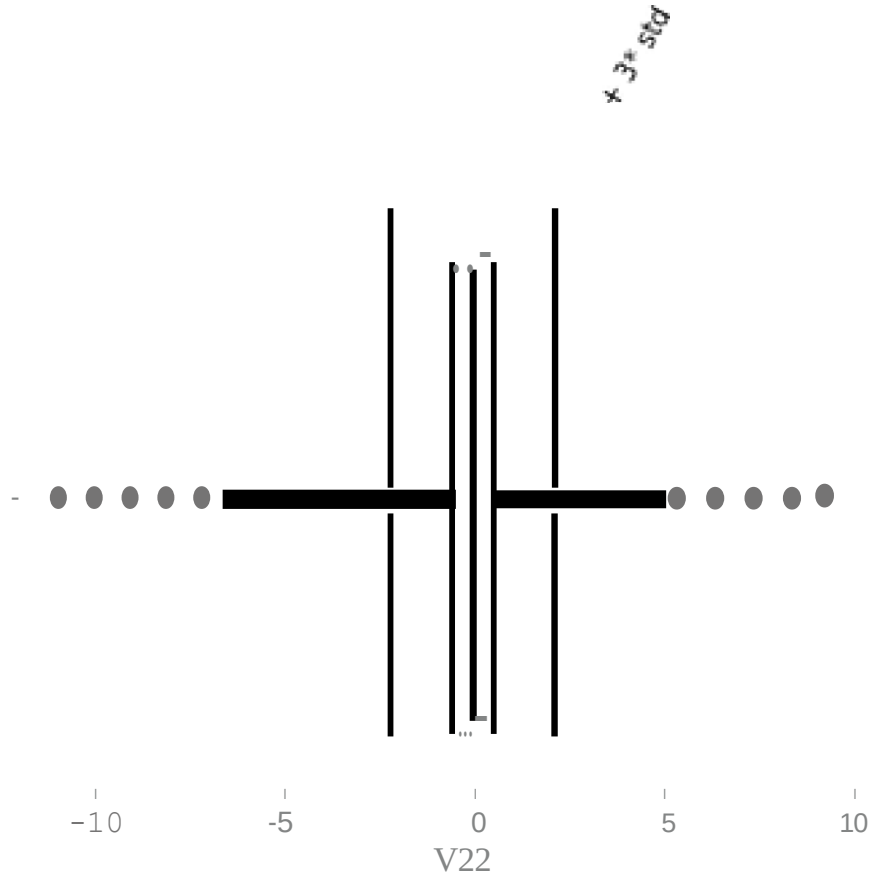


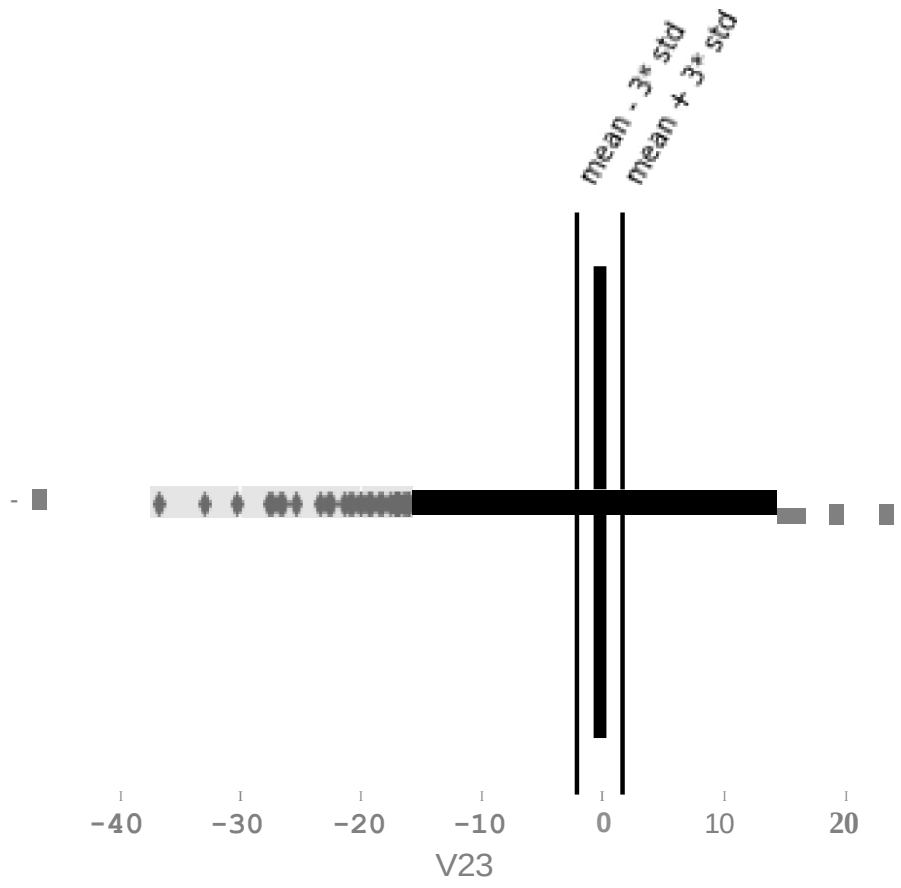
Vil..9

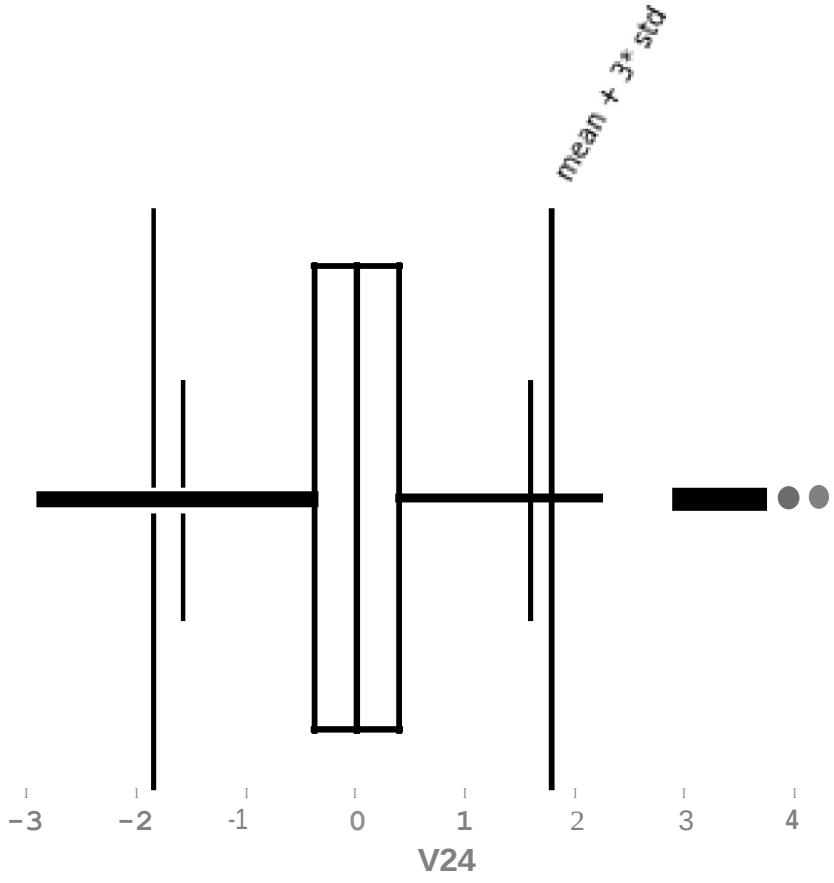


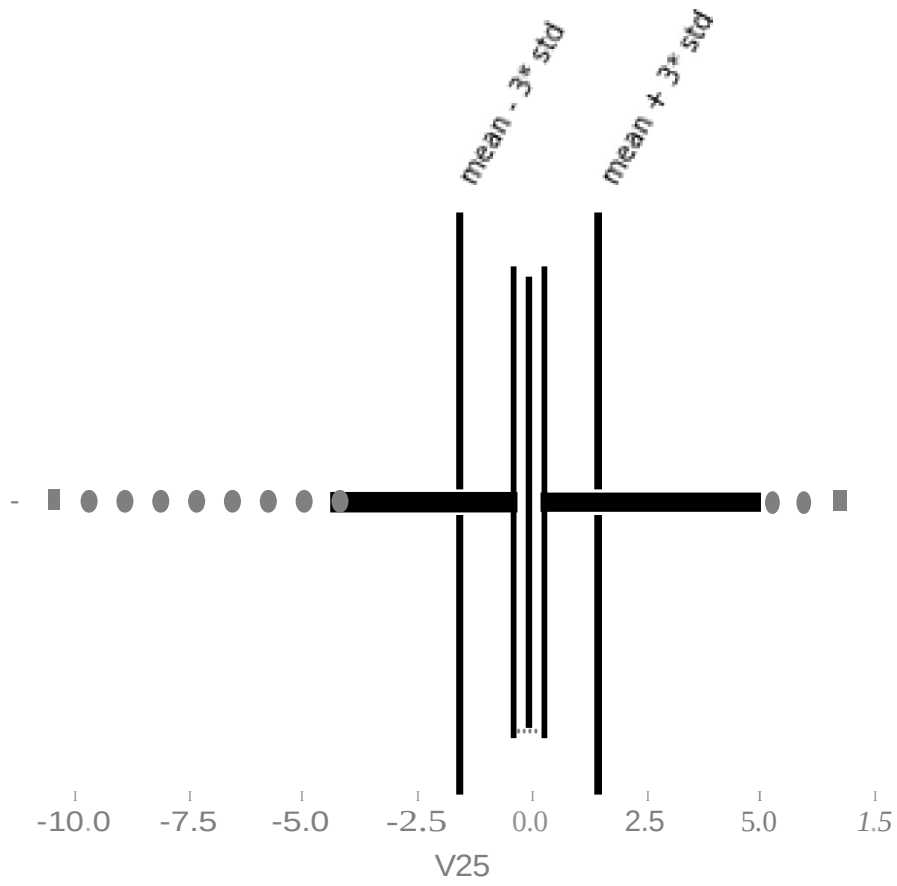
V20

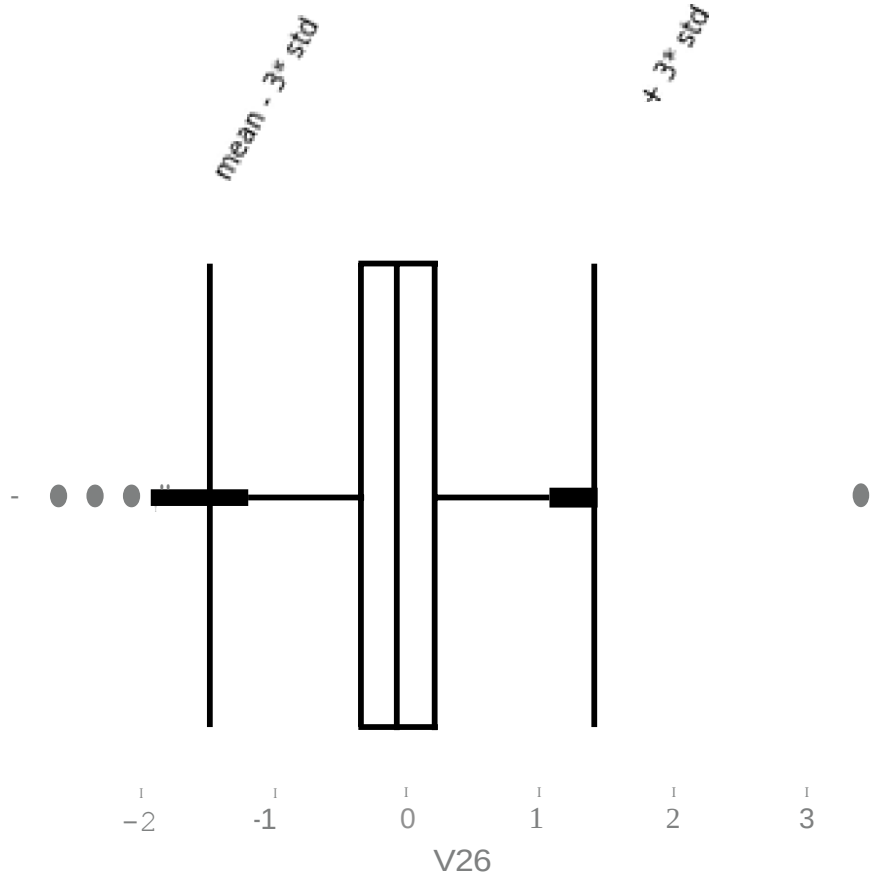

mean - 3* std
mean + 3* std
-30
-20
-10
0
10
20
30
V21

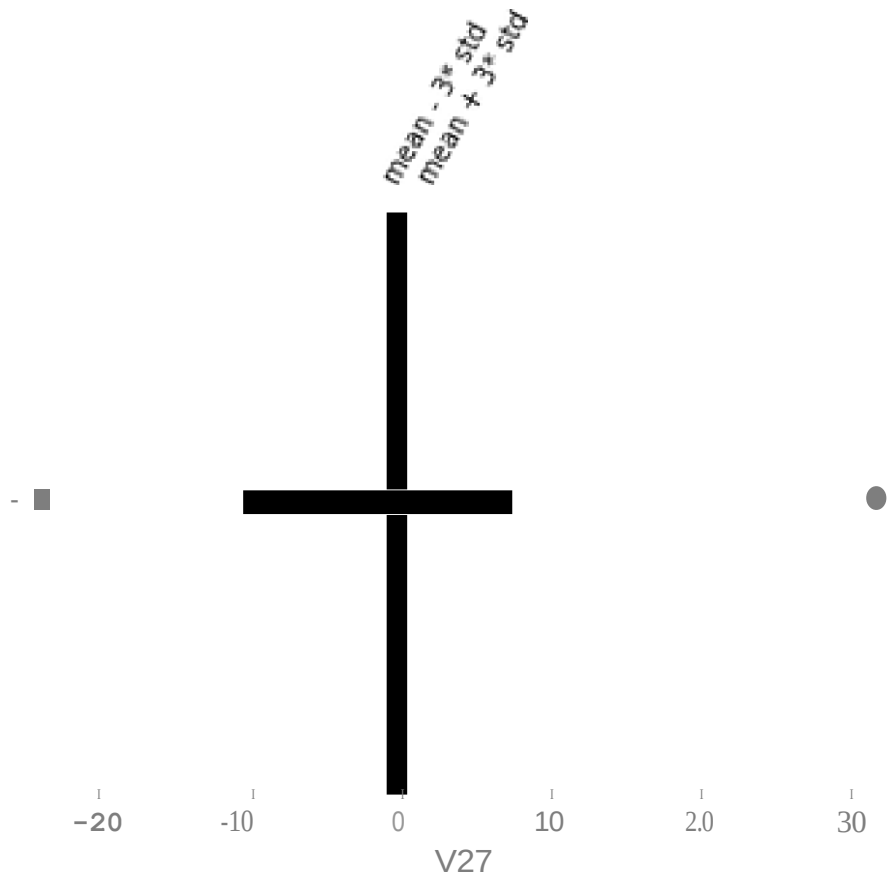

+ 3* std
-10
-5
0
5
10
V22

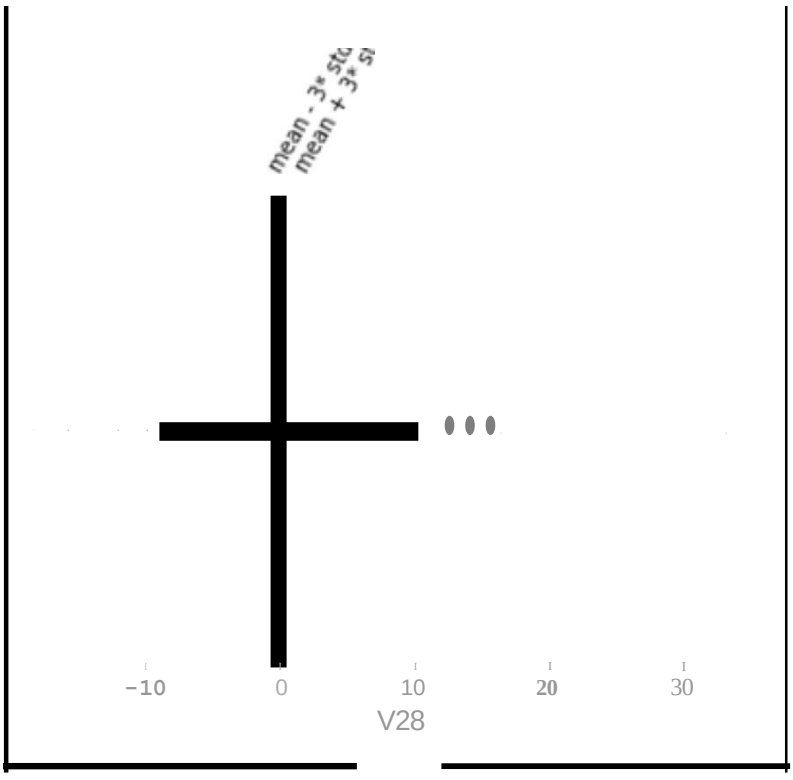

mean - 3* std
mean + 3* std
-40
-30
-20
-10
0
10
20
V23


mean + 3* std
-3
-2
-1
0
1
2
3
4
V24


mean - 3* std
mean + 3* std
-10.0
-7.5
-5.0
-2.5
0.0
2.5
5.0
1.5
V25


mean - 3* std
+ 3* std
−2
-1
0
1
2
3
V26


mean - 3* std
mean + 3* std
-20
-10
0
10
2.0
30
V27


mean - 3* std
mean + 3* st
-10
0
10
20
30
V28