INSTITUTO FEDERAL DO ESPÍRITO SANTO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
PROBABILIDADE E ESTATITICA

DAVID VILAÇA
LUCAS GOMES FLEGER
PAULO RICARDO VIANA FERREIRA

RELÁTORIO DO TRABALHO DE ESTATITSICA

Serra, ES
2018

Introdução

Os dados analisados neste trabalho, são oriundos da base de dados disponibilizada pelo portal da transparência do governo federal. Analisamos a base de pagamentos totais de pagamentos do bolsa família, relacionando o quanto cada município recebe de parcela referente ao mes de dezembro do ano de 2017. Onde o objetivo é gerar um gráfico do tipo histograma dos pagamentos.

Os dados possuem uma abrangência nacional. Os dados estão disponibilizados para uso público, podendo ser acessado por meio do site do portal da transparência do governo federal (link).

Os dados retornados pelo portal são arquivos no formato csv, que tem em seu conteúdo o município e a parcela recebida de verba do governo federal.

No trabalho iremos extrair os dados para que seja feito uma análise estatística dos mesmos.

Desenvolvimento

Usamos o Github hospedar nosso código de forma livre e versionado no endereço https://github.com/lukasg18/trab-estatistica. Para que fosse possível trabalhar com os dados o primeiro passo foi ler o arquivo, cujo qual possui um conjunto de dados no formato csv. Foi retirado previamente do arquivo informações desnecessárias, como por exemplo o nome do beneficiado.

Criamos uma função de leitura e captação dos dados do arquivo e as demais para efetuarmos os cálculos conforme especificação.

Tratamento de arquivo de amostra

In [2]:
```
def getAmostraCsv(filename):
    amostra_csv = pd.read_csv(filename)
    # dividindo todos valores por 100
    amostra_csv["Valor Parcela"] = amostra_csv["Valor Parcela"].map(lambda x : x / 100)
    return amostra_csv
```

Após tratamento dos dados, começamos os cálculos estatísticos, começamos a fazer os cálculos estatísticos.

Sendo eles:

Média:

A média é uma medida que mostra para onde se concentram os dados de uma distribuição como o ponto de equilíbrio das frequências e é obtida pela soma total dos termos dividida pelo número total de termos.

Para calcula a média do vetor de dados, desenvolvemos uma função que recebe como parâmetro o vetor de pagamentos em reais, e calcula na função o somatório com todos os valores que estão no vetor e dividimos pelo tamanho do vetor, que equivale a quantidade de elementos da amostra. (Segue abaixo a imagem com o código da função):

Média

In [3]:
```
def calculaMedia(listaDados):
    somatorio = 0
    media = 0
    tam = len(listaDados) #quantidade total da amostra
    for valor in listaDados:
        somatorio = somatorio + valor;
    #fim  for
    media = somatorio/tam

    return media
#fim função media
```

Moda

Com o resultado obtido pela função que calcula a média, podemos concluir que a parcela media recebida é de: `179,39`.

Moda (Mo):

É o valor mais frequente num conjunto de dados. Para verificar qual valor aparece com mais frequência no conjunto de dados, optamos por varrer todo vetor de dados e armazenar cada valor encontrado como um índice de um dicionário e atribuir em cada índice a quantidade de vezes que o valor aparece na série.

A função de intitulada de calculaModa trabalha da seguinte forma, tem como parâmetro de o vetor de dados, que tem dado por dado verificado e testado para verificar se o valor já é uma das chaves do dicionário, caso essa verificação seja verdadeira o número de ocorrência é incrementado e caso não esteja no conjunto de chave do dicionário ele é adicionado e incrementado.

Por fim é o dicionário e também a lista com todas as chaves do dicionário são passados como parâmetro para função calculaOcorrencias, que é a função responsável por percorrer todo o dicionário e verificar quem aparece mais, a função retorna o valor com maiores ocorrências para a função de calculaModa.

Moda

In [4]:
```
def caculaOcorrencias(listaChaves, dicDados):
    maior = 0
    moda = 0

    for chave in listaChaves:
        if dicDados[chave] > maior:
            moda = chave
            maior = dicDados[chave]
        #fim if
    #fim for

    return moda
#fim função

def calculaModa(listaDados):
    dicDados = {}       #dicionario de ocorrencias

    for valor in listaDados:
        if valor in dicDados.keys():
            dicDados[valor] = dicDados[valor]+1
        else:
            dicDados[valor] = 1
    #fim for

    listaChaves = dicDados.keys()
    moda = caculaOcorrencias(listaChaves, dicDados)

    return moda
#fim função moda
```

Mediana:

Mediana (Md) é o valor que medeia os valores presentes num conjunto ordenado numericamente.

Para calcular a mediana, primeiro foi necessário ordenar a lista de dados em ordem crescente para que seja verificado se o tamanho é par ou ímpar, se for par é a média aritmética dos dois valores centrais e se for ímpar a mediana vai ser o valor na posição central.

Mediana

In [5]:
```
def ehPar(listaDados):

    if len(listaDados)%2 == 0:
        return True
    #fim if

    return False
#fim função

def calculaMediana(listaDados):
    # Verifica se o tamanho da lista é par
    if ehPar:
        pos1 = listaDados[int((len(listaDados)/2))]
        pos2 = listaDados[int((
            (len(listaDados) + 2) /2
        ))]
        media = calculaMedia([pos1,pos2])
        return media
    #Se for impar
    else:
        pos = listaDados[int(len(listaDados)/2)]
        return pos
#fim função mediana
```

Variância:

A variância, uma medida de dispersão que mostra quão distantes os valores estão da média. Para calcular a variância utilizei a seguinte fórmula (Var = (sum[(valCorrente - media)^ 2])/(n-1)) , que equivale ao somatória do quadrado do valor corrente menos a média, sobre o tamanho da amostra menos 1.

A função CalculaVariancia tem como parâmetro a lista de dados, cujo o qual será calculada a média utilizando a função de tirar a média e em seguida pego também o tamanho da lista, que equivale ao tamanho da amostra. Em seguida varremos toda lista para calcular o somatório do quadrado do valor corrente menos a média.

Variância

In [6]:
```
def calculaVariancia(listaDados):
    media = calculaMedia(listaDados)
    somatorio = 0
    tam = len(listaDados)
    variancia = 0

    for valor in listaDados:
        somatorio = somatorio + ((valor - media) ** 2)
    #fim for
    variancia = somatorio/tam-1

    return variancia
#fim função variancia
```

Desvio Padrão:

O desvio padrão (dp) é simplesmente o resultado positivo da raiz quadrada da variância. Na prática, o desvio padrão indica qual é o “erro” se quiséssemos substituir um dos valores coletados pelo valor da média. Para calcular o desvio padrão eu chamei a função de calcular a variância, pegamos o valor da variância e elevamos ao quadrado e obtive o desvio padrão.

Desvio padrão

In [7]:
```
def verificaDesvioPadrao(listaDados):
    variancia = calculaVariancia(listaDados)
    variancia = variancia ** (1/2)
    return variancia
##fim função Desvio padrão
```

Primeiro Quartil:

O primeiro quartil é o valor aos 25% da amostra ordenada. Para obter o primeiro quartil, primeiro usamos a função do python para ordenar a lista, para garantir que os dados estariam ordenados depois de algumas manipulações, pegamos também o tamanho da lista de dados, que equivale ao número total da amostra e foi somado 1, além disso, multiplicamos por ¼ devido um quartil representar ¼ da amostra. Ou seja, valor contido na posição da amostra ordenado = ((tamanhoAmostra + 1) *0,25).

```
"def medidas_boxplot(arr):\n",
"    return 0"
"    # quartil 1\n",
"    q1 = percentil(arr, 25)\n",
"    # quartil 2\n",
"    q2 = percentil(arr, 50)\n",
"    # quartil 3\n",
"    q3 = percentil(arr, 75)\n",
"    \n",
"    aiq = q3 - q1\n",
"    val_max = q3 + (1.5 * aiq)\n",
"    val_min = q1 - (1.5 * aiq)\n",
"    return (val_min, q1, q2, q3, val_max)"
```

Segundo Quartil:

O segundo quartil é o valor aos 50% da amostra ordenada, ou seja, equivale ao mesmo valor da mediana.

Terceiro Quartil:

O segundo quartil é o valor aos 75% da amostra ordenada. Para obter o terceiro quartil, primeiro usamos a função do python para ordenar a lista, para garantir que os dados estariam ordenados depois de algumas manipulações, peguei também o tamanho da lista de dados, que equivale ao número total da amostra e foi somado 1, além disso, multiplicamos por ¼ devido um quartil representar ¼ da amostra. Ou seja, valor contido na posição da amostra ordenado = ((tamanhoAmostra + 1) *0,75).

```
"def medidas_boxplot(arr):\n",
"    return 0"
"    # quartil 1\n",
"    q1 = percentil(arr, 25)\n",
"    # quartil 2\n",
"    q2 = percentil(arr, 50)\n",
"    # quartil 3\n",
"    q3 = percentil(arr, 75)\n",
"    \n",
"    aiq = q3 - q1\n",
"    val_max = q3 + (1.5 * aiq)\n",
"    val_min = q1 - (1.5 * aiq)\n",
"    return (val_min, q1, q2, q3, val_max)"
```

Amplitude Interquartil:

O intervalo interquartil, também denominado por média espalhada, média de 50% ou, mais tecnicamente, propagação de H, é uma medida de dispersão estatística igual à diferença entre os percentis 75 e 25 ou entre o quartil superior e o quartil inferior.

Momento:

Em Estatística, a expressão genérica de esperança, o enésimo momento ou momento de ordem n de uma variável aleatória X é dado por: E[x^n]. Os momentos são muito importantes em Estatística para caracterizar distribuições de probabilidade. Por exemplo, a distribuição normal é caracterizada apenas pelo primeiro e pelo segundo momentos. Os primeiros, segundos, terceiros e quartos momentos caracterizam a tendência central, dispersão, assimetria e curtose, respectivamente, de uma distribuição de probabilidades.

Momentos

In [10]:
```
def momentos(arr, m, a=0):
    acc = 0
    for i in arr: acc = acc + ((i-a)**m)
    return acc / len(arr)
```

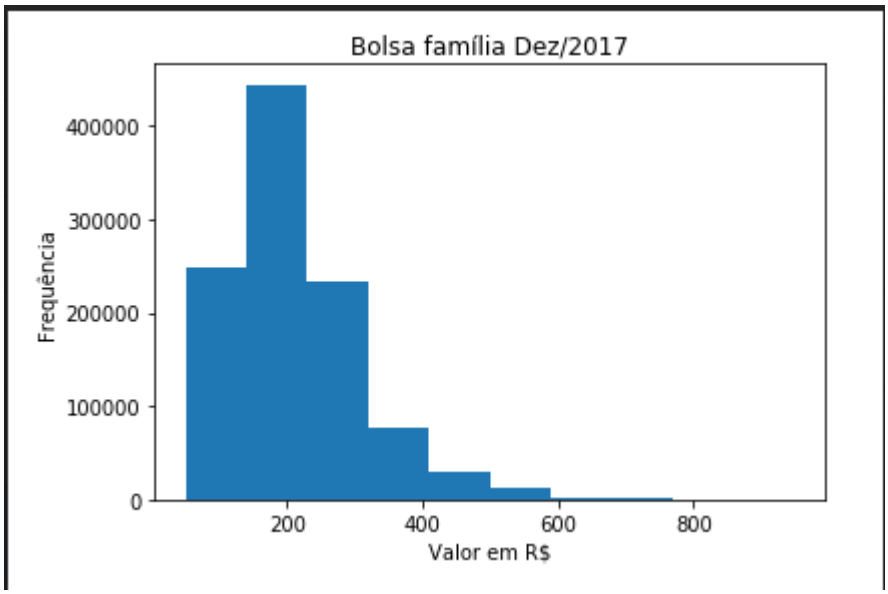
Função de cálculo de assimetria:

```
"def simetria(arr):\n",
"    media = calculaMedia(arr)\n",
"    moda = calculaModa(arr)\n",
"    desvioP = verificaDesvioPadrao(arr)\n",
"    simetria_valor = (media - moda) / desvioP\n",
"    s = ''\n",
"    if simetria_valor < 0.15:\n",
"        s = 'Moderadamente Simétrica'\n",
"    elif 0.15 <= simetria_valor and simetria_valor < 1:\n",
"        s = 'Moderadamente Assimétrica'\n",
"    else:\n",
"        s = 'Fortemente Assimétrica'\n",
"    return s"
```

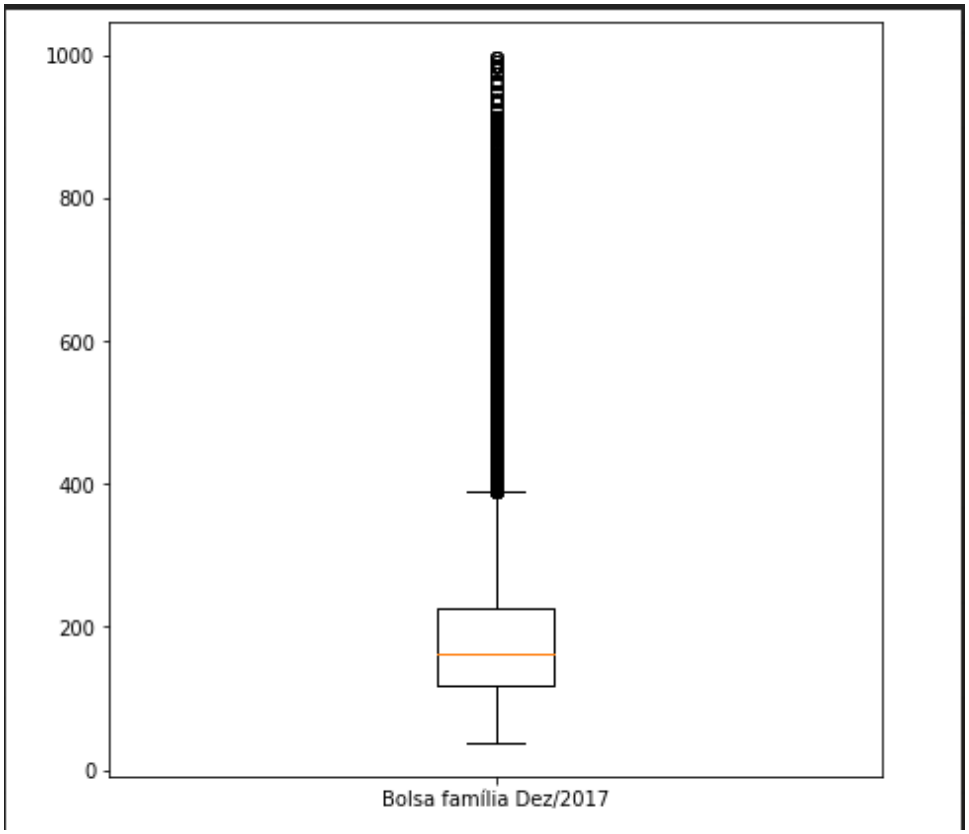
Histograma, Box Plot e resultados:

Para imprimir o histograma e o Box plot utilizei uma biblioteca padrão do python, que recebe como parâmetro o conjunto de dados e imprime os gráficos.

Histograma:



Boxplot:



Resultados Gerais:

```
Média: 179.39
Moda: 124.00
Mediana: 163.00
Variância: 10615.37
Desvio padrão: 103.03
Simetria: Moderadamente Assimétrica
Medidas do boxplot: (-46.5, 117.0, 163.0, 226.0, 389.5)
Curtose: Leptocúrtica
```

Concluímos que de acordo com os dados acima informações a respeito dos pagamentos do bolsa família por município brasileiro do mês de dezembro de 2017.