Planejamento de Experimentos

1.0 Introdução

"Chamar um especialista em estatística depois que o experimento foi feito pode ser o mesmo que pedir para ele fazer um exame post-mortem. Talvez ele consiga dizer do que foi que o experimento morreu."

Sir Ronald Fisher

Introdução à análise de experimentos

Inserindo o experimento num Data Frame

Importando as bibliotecas

Pandas

https://pandas.pydata.org/

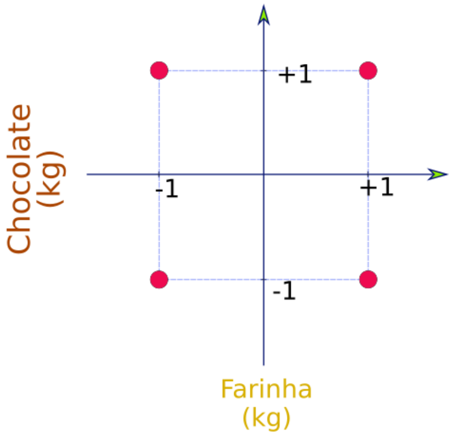
In [1]:
```
import pandas as pd
```

Numpy

http://www.numpy.org/

In [2]:
```
import numpy as np
```

Ensaios realizados na forma normalizada



Ensaio (-)	Farinha kg	Chocolate kg	Porções* (-)
1	-1	-1	19
2	$+1$	-1	37
3	-1	$+1$	24
4	$+1$	$+1$	49

*Quantidade de cupcakes produzidos

Construindo uma matriz representando todos os ensaios realizados:

In [3]:
```
ensaios = np.array([ [-1, -1], [1, -1], [-1, 1], [1, 1]])
```

pyDOE2

https://pypi.org/project/pyDOE2/

In [4]:
```
import pyDOE2 as doe
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-4-97646f0a695f> in <module>
----> 1 import pyDOE2 as doe

ModuleNotFoundError: No module named 'pyDOE2'
```

Costruindo um planejamento fatorial de 2^2

In [5]:
```
ensaios = doe.ff2n(2)
```

In [6]:
```
ensaios
```

Out[6]:
```
array([[-1., -1.],
       [ 1., -1.],
       [-1.,  1.],
       [ 1.,  1.]])
```

Incerindo o planejamento em um Data Frame

In [7]:
```
experimento = pd.DataFrame(ensaios, columns=['Farinha', 'Chocolate'])
```

In [8]:
```
experimento
```

Out[8]:

	Farinha	Chocolate
0	-1.0	-1.0
1	1.0	-1.0
2	-1.0	1.0

	Farinha	Chocolate
3	1.0	1.0

Inserindo coluna com os resultados

In [9]:
```
experimento['Porcoes'] = [19, 37, 24, 49]
```

In [10]:
```
experimento
```

Out[10]:

	Farinha	Chocolate	Porcoes
0	-1.0	-1.0	19
1	1.0	-1.0	37
2	-1.0	1.0	24
3	1.0	1.0	49

Conclusão: Temos, por fim, nosso experimento representado por um *DataFrame* do Pandas. Usaremos este *DataFrame* para iniciarmos a análise do nosso experimento.

Analisando graficamente o experimento

Importando o Seaborn

https://seaborn.pydata.org

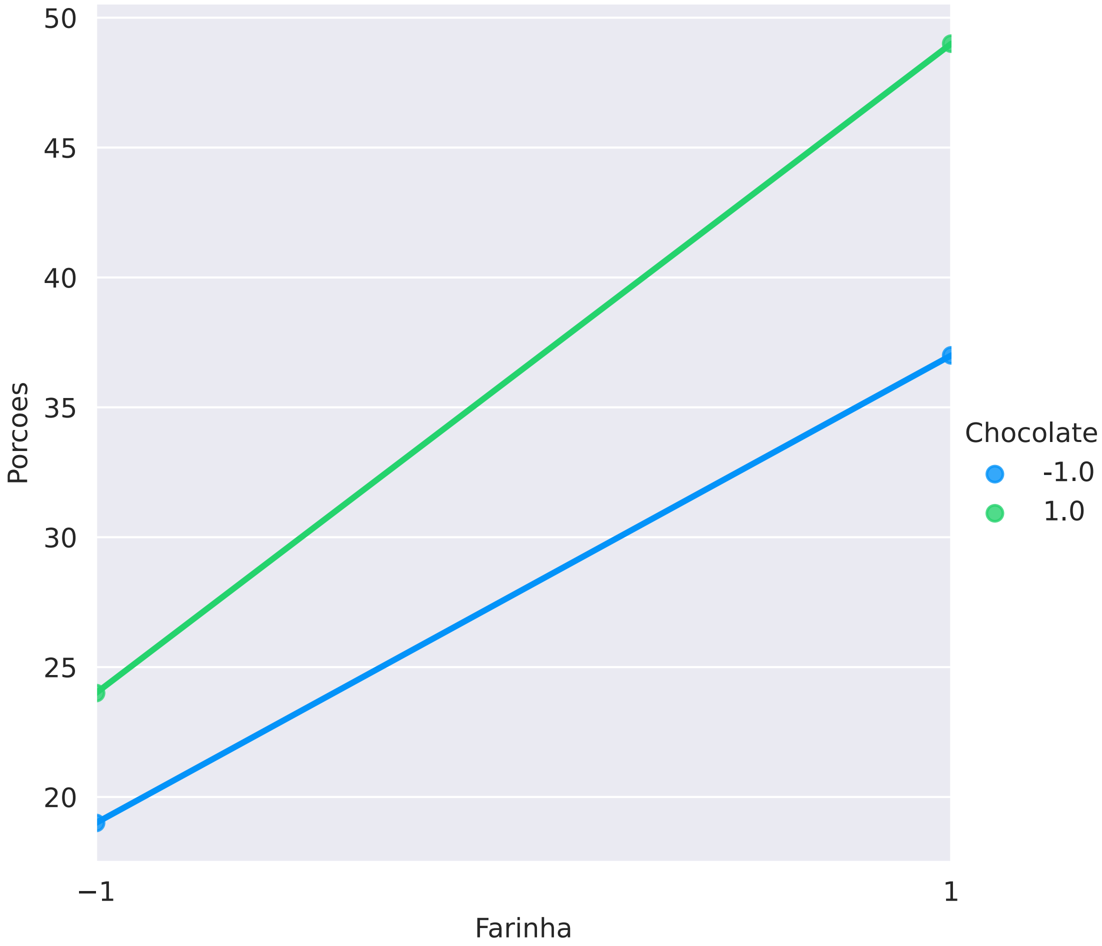
In [11]:
```
import seaborn as sns
```

In [12]:
```
# paletas -> Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap
sns.set_palette('terrain')

# estilo -> white, dark, whitegrid, darkgrid, ticks
sns.set_style('darkgrid')
```

Para a farinha

In [13]:
```
ax1 = sns.lmplot(data = experimento, x = 'Farinha', y = 'Porcoes', hue = 'Chocolate', ci = None
ax1.set(xticks = (-1, 1))
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x7f9931d9bf28>

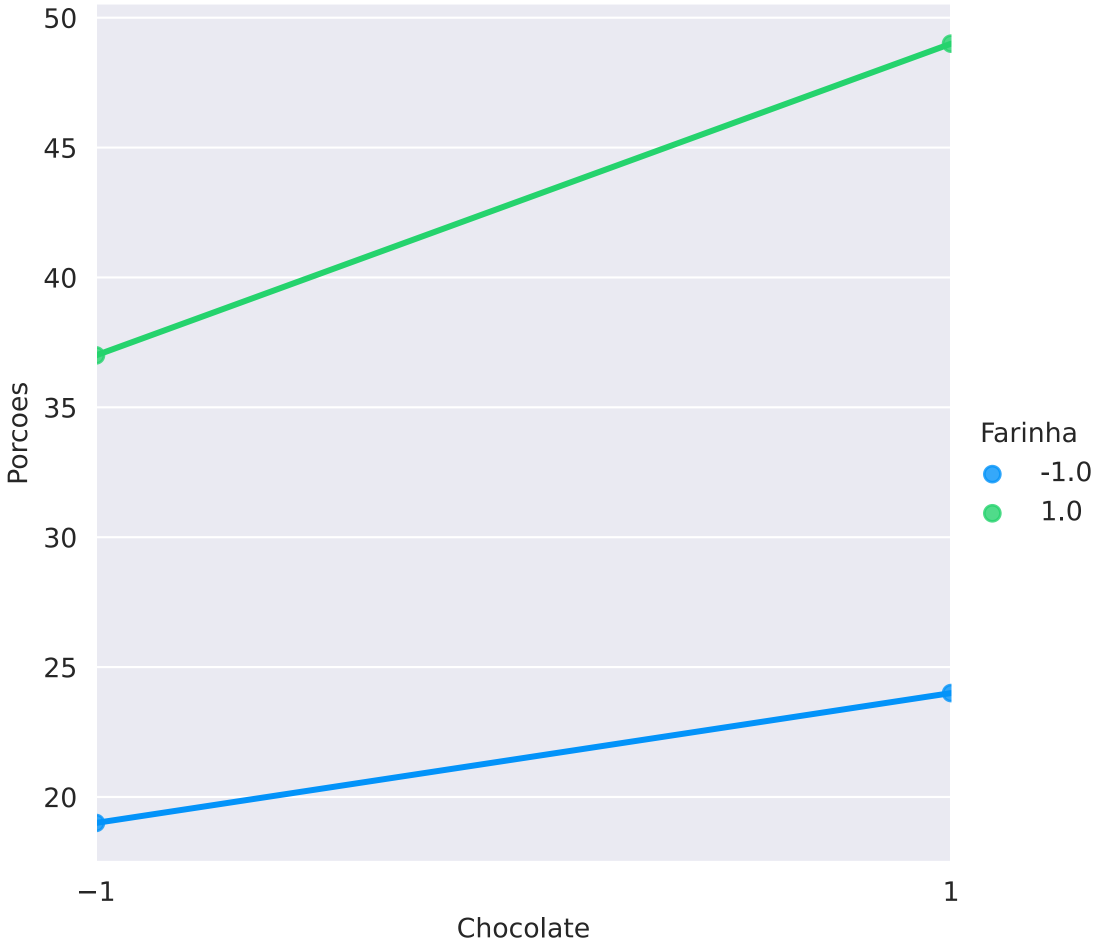


Para o chocolate

In [14]:

```
ax2 = sns.lmplot(data = experimento, x = 'Chocolate', y = 'Porcoes', hue = 'Farinha', ci = None
ax2.set(xticks = (-1, 1))
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x7f99314cb5c0>



Ajustando o modelo estatístico

Modelo estatístico

$$P = \underbrace{\beta_0}_{\text{Intercepto}} + \underbrace{\beta_1.x_{farinha} + \beta_2.x_{chocolate}}_{\text{Efeitos isolados}} + \underbrace{\beta_3.x_{farinha}.x_{chocolate}}_{\text{Efeito interação}} + \underbrace{\epsilon}_{\text{Erro}}$$

Bibliotecas Stats Model

In [15]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

.

In [16]:

```
modelo = smf.ols(data=experimento, formula='Porcoes ~ Farinha + Chocolate + Farinha:Chocolate')
```

In [17]:

```
modelo_ajustado = modelo.fit()
```

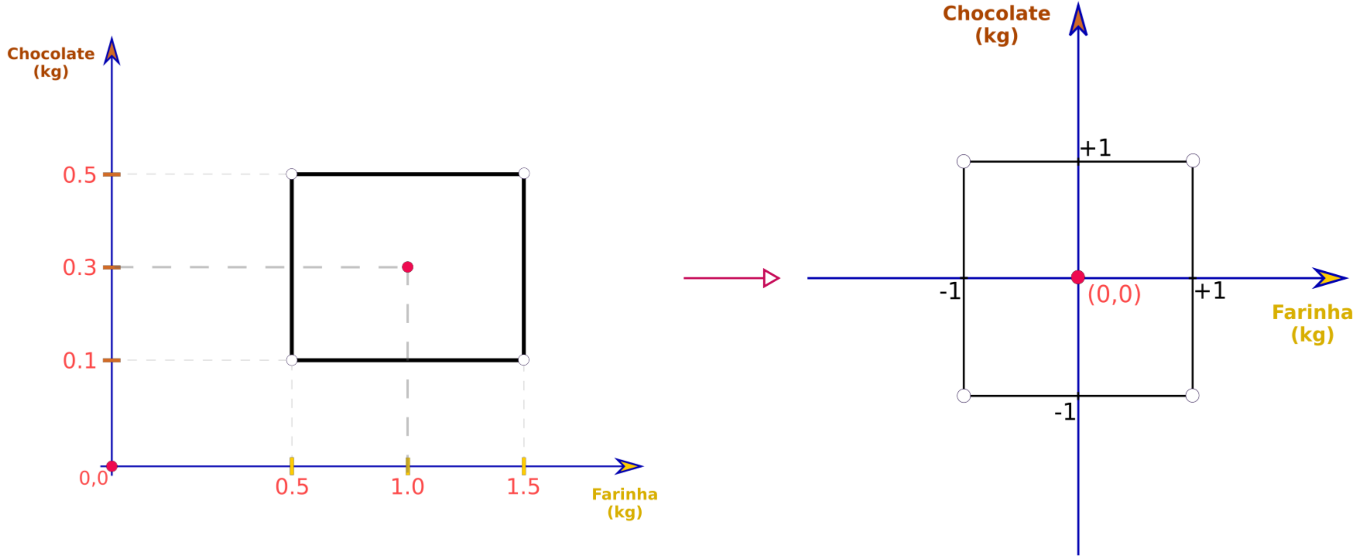
In [18]:

```
print(modelo_ajustado.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Porcoes   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                    nan
Method:                 Least Squares   F-statistic:                       nan
Date:                Thu, 13 May 2021   Prob (F-statistic):                nan
Time:                        15:35:36   Log-Likelihood:                 126.02
No. Observations:                   4   AIC:                            -244.0
Df Residuals:                       0   BIC:                            -246.5
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            32.2500        inf          0        nan         nan         nan
Farinha              10.7500        inf          0        nan         nan         nan
Chocolate             4.2500        inf          0        nan         nan         nan
Farinha:Chocolate     1.7500        inf          0        nan         nan         nan
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   1.500
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.167
Skew:                           0.000   Prob(JB):                        0.920
Kurtosis:                       2.000   Cond. No.                         1.00
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/statsmodels/stats/statto
ols.py:75: ValueWarning: omni_normtest is not valid with less than 8 observations; 4 samples we
re given.
  "samples were given." % int(n), ValueWarning)
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/statsmodels/regression/l
inear_model.py:1728: RuntimeWarning: divide by zero encountered in true_divide
  return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/statsmodels/regression/l
inear_model.py:1729: RuntimeWarning: invalid value encountered in double_scalars
  * (1 - self.rsquared))
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/statsmodels/regression/l
inear_model.py:1650: RuntimeWarning: divide by zero encountered in double_scalars
  return np.dot(wresid, wresid) / self.df_resid
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/statsmodels/base/model.p
y:1452: RuntimeWarning: invalid value encountered in multiply
  cov_p = self.normalized_cov_params * scale
```

Aumentando os Graus de liberdade

Replicatas no centro



Ensaio (-)	Farinha kg	Chocolate kg	Porções (-)
5	0	0	29
6	0	0	30
7	0	0	29
8	0	0	30

In [19]:
```
centro = np.array([ [0, 0, 29],
                    [0, 0, 30],
                    [0, 0, 29],
                    [0, 0, 30] ])
```

In [20]:
```
centro_dataframe = pd.DataFrame(centro, columns=['Farinha', 'Chocolate', 'Porcoes'], index=[4,5
```

In [21]:
```
centro_dataframe
```

Out[21]:

	Farinha	Chocolate	Porcoes
4	0	0	29
5	0	0	30
6	0	0	29
7	0	0	30

.

In [22]:
```
experimento = experimento.append(centro_dataframe)
```

In [23]:
```
experimento
```

Out[23]:

	Farinha	Chocolate	Porcoes
0	-1.0	-1.0	19
1	1.0	-1.0	37

	Farinha	Chocolate	Porcoes
2	-1.0	1.0	24
3	1.0	1.0	49
4	0.0	0.0	29
5	0.0	0.0	30
6	0.0	0.0	29
7	0.0	0.0	30

Análise de significância estatística

In [24]:

```
modelo = smf.ols(data=experimento, formula='Porcoes ~ Farinha + Chocolate + Farinha:Chocolate')
```

In [25]:

```
modelo_ajustado = modelo.fit()
```

In [26]:

```
print(modelo_ajustado.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                Porcoes   R-squared:                       0.971
Model:                            OLS   Adj. R-squared:                  0.950
Method:                 Least Squares   F-statistic:                     45.21
Date:                Thu, 13 May 2021   Prob (F-statistic):            0.00152
Time:                        15:35:39   Log-Likelihood:                -14.155
No. Observations:                   8   AIC:                             36.31
Df Residuals:                       4   BIC:                             36.63
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            30.8750      0.710     43.494      0.000      28.904      32.846
Farinha              10.7500      1.004     10.708      0.000       7.963      13.537
Chocolate             4.2500      1.004      4.233      0.013       1.463       7.037
Farinha:Chocolate     1.7500      1.004      1.743      0.156      -1.037       4.537
==============================================================================
Omnibus:                        4.655   Durbin-Watson:                   0.841
Prob(Omnibus):                  0.098   Jarque-Bera (JB):                1.080
Skew:                          -0.180   Prob(JB):                        0.583
Kurtosis:                       1.237   Cond. No.                         1.41
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/scipy/stats/stats.py:160
4: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=8
  "anyway, n=%i" % int(n))
```

Teste de hipótese

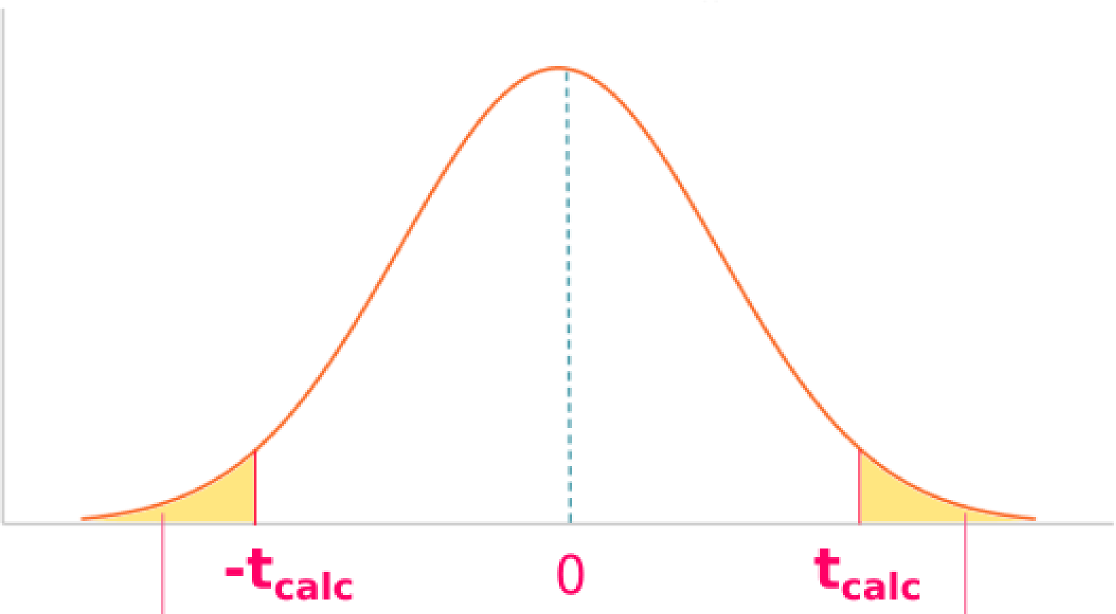
Temos 4 parâmetros: β_0 , β_1 , β_2 e β_3

H_0 --> $\beta_i = 0$ Não significante

ou

H_A --> $\beta_i \neq 0$ Significante

Distribuição t



$$P > |t|$$

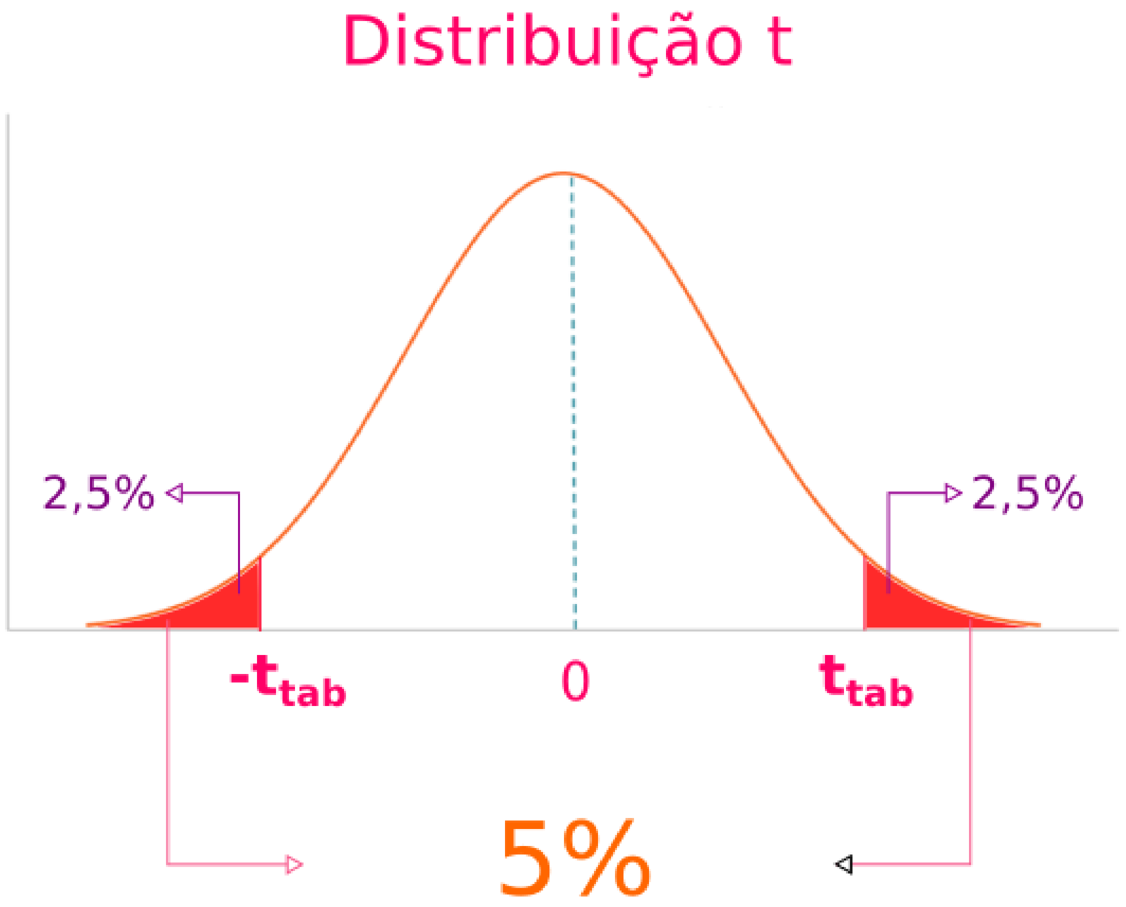
Avaliando significância

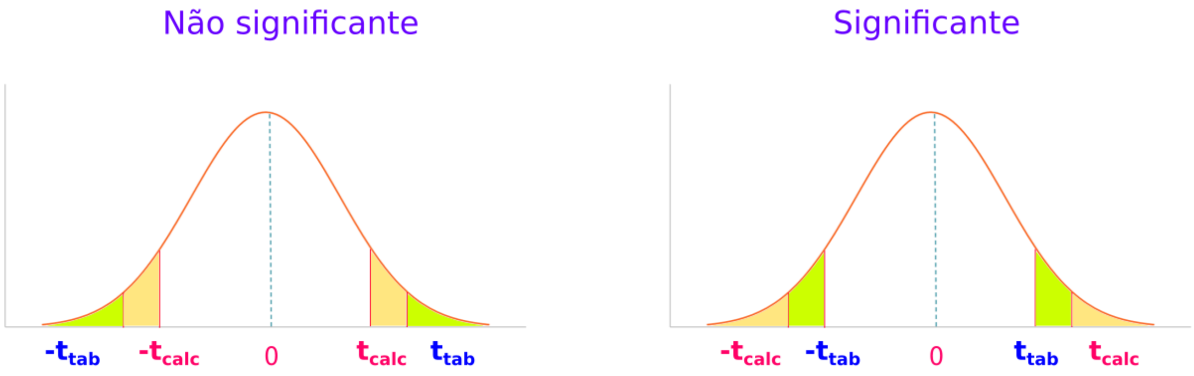
$(P > |t|) \geqslant \alpha$ **-->** $\beta_i = 0$ Não significante

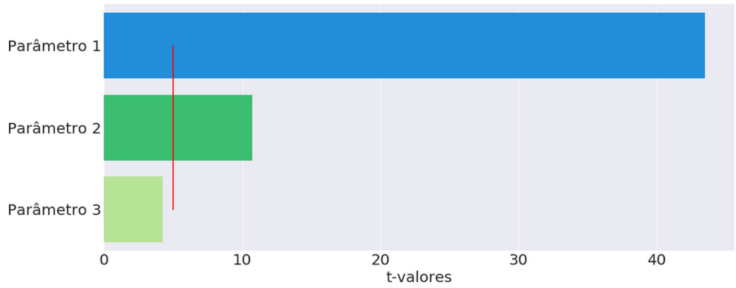
ou

$(P > |t|) < \alpha$ **-->** $\beta_i \neq 0$ Significante

Teste de significância estatística usando o t







In [27]:
```python
t_valores = modelo_ajustado.tvalues
```

In [28]:
```python
t_valores
```

Out[28]:
```
Intercept            43.494275
Farinha              10.708252
Chocolate             4.233495
Farinha:Chocolate     1.743204
dtype: float64
```

In [29]:
```python
nome = t_valores.index.to_list()
```

In [30]:
```python
nome
```

Out[30]:
```
['Intercept', 'Farinha', 'Chocolate', 'Farinha:Chocolate']
```

-

In [31]:
```python
from scipy import stats
```

-

In [32]:
```python
distribuicao = stats.t(df = 4)
```

In [33]:
```python
distribuicao.ppf(q = 1 - 0.025)
```

Out[33]:
```
2.7764451051977987
```

In [34]:

```
limite = [distribuicao.ppf(q=1-0.025)]*len(nome)
```

In [35]:

```
limite
```

Out[35]:

```
[2.7764451051977987,
 2.7764451051977987,
 2.7764451051977987,
 2.7764451051977987]
```

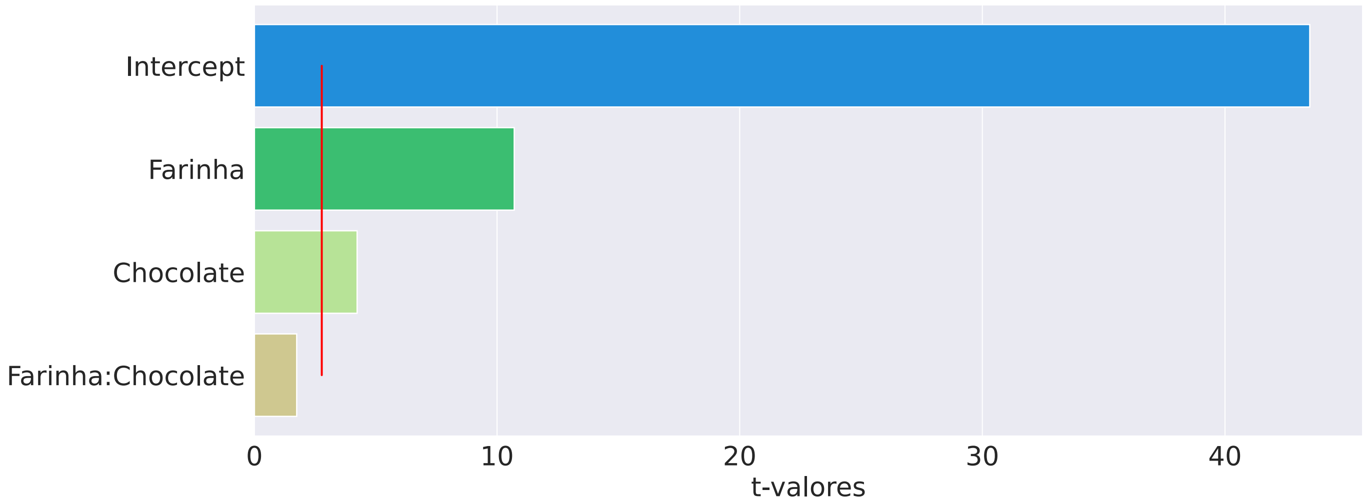
Plotando o gráfico

In [36]:

```
pareto = sns.barplot(x = t_valores, y = nome)
pareto.figure.set_size_inches(15,6)
pareto.tick_params(labelsize=20)
pareto.set_xlabel('t-valores', fontsize=20)
pareto.plot(limite, nome, 'r')
```

Out[36]:

```
[<matplotlib.lines.Line2D at 0x7f992fbd7be0>]
```



Propondo um novo modelo

Modelo estatístico

$$P = \underbrace{\beta_0}_{\text{Intercepto}} + \underbrace{\beta_1.x_{farinha} + \beta_2.x_{chocolate}}_{\text{Efeitos isolados}} + \underbrace{\beta_3.x_{farinha}.x_{chocolate}}_{\text{Efeito interação}} + \underbrace{\epsilon}_{\text{Erro}}$$

Modelo estatístico atualizado

$$\underbrace{}_{}P = \underbrace{\beta_0}_{\text{Intercepto}} + \underbrace{\beta_1.x_{farinha} + \beta_2.x_{chocolate}}_{\text{Efeitos isolados}} + \underbrace{\epsilon}_{\text{Erro}}$$

In [37]:
```python
modelo_2 = smf.ols(data = experimento, formula='Porcoes ~ Farinha + Chocolate')
```

In [38]:
```python
modelo_ajustado_2 = modelo_2.fit()
```

In [39]:
```python
print(modelo_ajustado_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Porcoes   R-squared:                       0.950
Model:                            OLS   Adj. R-squared:                  0.929
Method:                 Least Squares   F-statistic:                     47.09
Date:                Thu, 13 May 2021   Prob (F-statistic):           0.000571
Time:                        15:35:41   Log-Likelihood:                -16.416
No. Observations:                   8   AIC:                             38.83
Df Residuals:                       5   BIC:                             39.07
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     30.8750      0.842     36.658      0.000      28.710      33.040
Farinha       10.7500      1.191      9.025      0.000       7.688      13.812
Chocolate      4.2500      1.191      3.568      0.016       1.188       7.312
==============================================================================
Omnibus:                        2.106   Durbin-Watson:                   1.850
Prob(Omnibus):                  0.349   Jarque-Bera (JB):                1.245
Skew:                           0.868   Prob(JB):                        0.537
Kurtosis:                       2.153   Cond. No.                         1.41
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/scipy/stats/stats.py:160
4: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=8
  "anyway, n=%i" % int(n))
```

Gráfico Padronizado de Pareto do novo modelo

In [40]:
```python
t_valores = modelo_ajustado_2.tvalues
```

In [41]:
```python
t_valores
```

Out[41]:
```
Intercept    36.658022
Farinha       9.025173
```

```
Chocolate     3.568092
dtype: float64
```

In [42]:
```python
nome = t_valores.index.to_list()
```

In [43]:
```python
nome
```

Out[43]:
```
['Intercept', 'Farinha', 'Chocolate']
```

.

In [44]:
```python
distribuicao = stats.t(df = 5)
```

In [45]:
```python
distribuicao.ppf(q = 1 - 0.025)
```

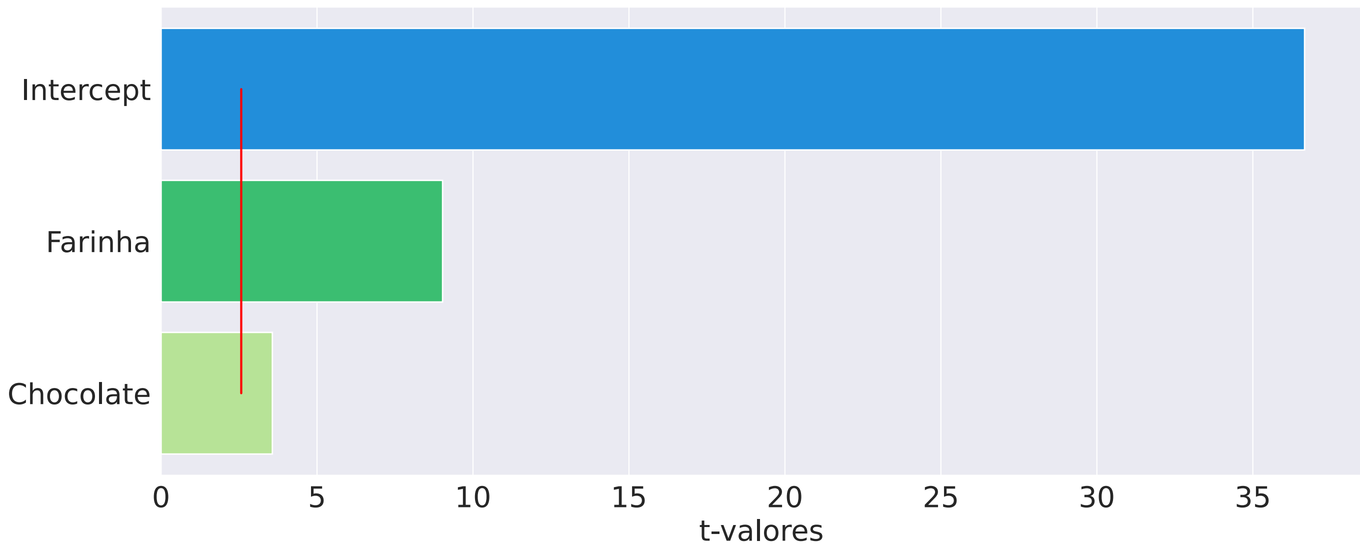
Out[45]:
```
2.5705818366147395
```

In [46]:
```python
limite = [distribuicao.ppf(q=1-0.025)]*len(nome)
```

Plotando o gráfico

In [47]:
```python
pareto = sns.barplot(x = t_valores, y = nome)
pareto.figure.set_size_inches(15,6)
pareto.tick_params(labelsize=20)
pareto.set_xlabel('t-valores', fontsize=20)
pareto.plot(limite, nome, 'r')
```

Out[47]:
```
[<matplotlib.lines.Line2D at 0x7f992f709ba8>]
```



Preditos por observados

In [48]:
```python
observados = experimento['Porcoes']
```

In [49]:
```python
observados
```

Out[49]:
```
0    19
1    37
2    24
```

```
3    49
4    29
5    30
6    29
7    30
Name: Porcoes, dtype: int64
```

.

In [50]:
```python
preditos = modelo_ajustado_2.predict()
```

In [51]:
```python
preditos
```

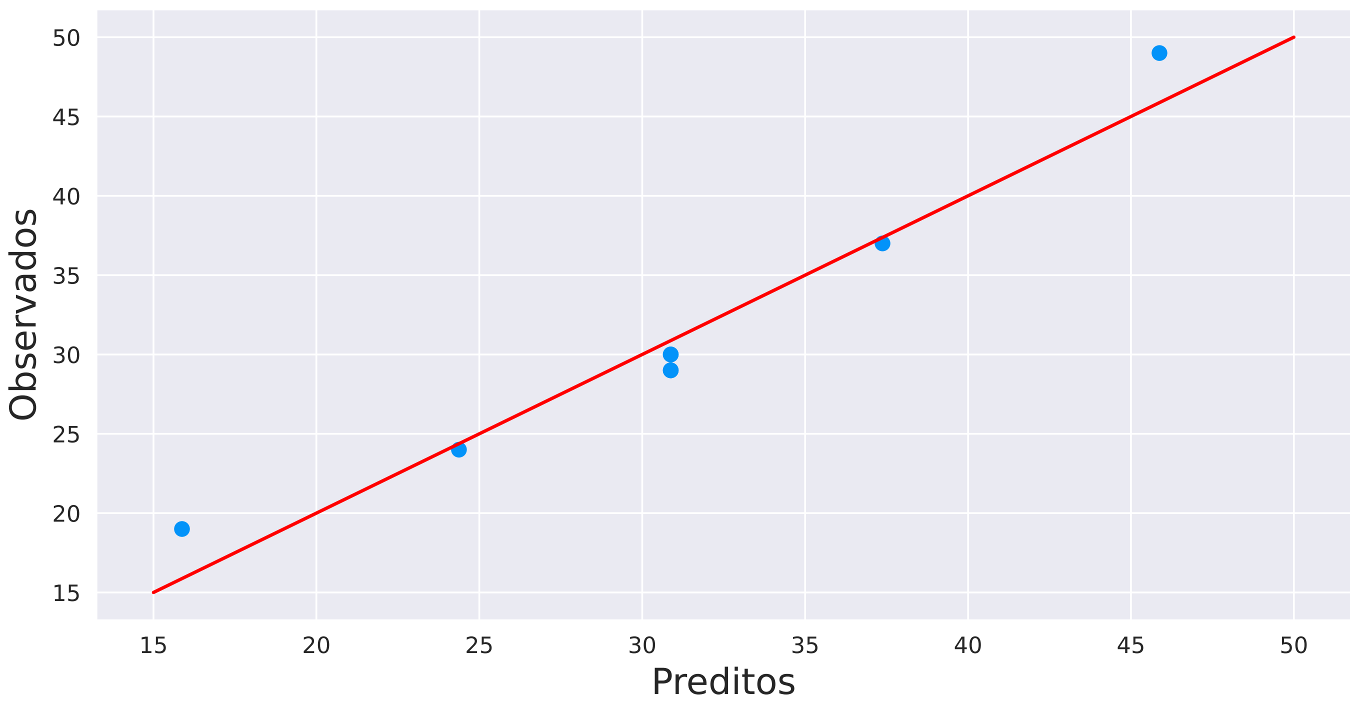
Out[51]:
```
array([15.875, 37.375, 24.375, 45.875, 30.875, 30.875, 30.875, 30.875])
```

.

In [52]:
```python
import matplotlib.pyplot as plt
```

In [53]:
```python
plt.figure(figsize=(10,5))
plt.xlabel('Preditos', fontsize = 16)
plt.ylabel('Observados', fontsize = 16)

#Linha de guia
x = np.linspace(start=15, stop=50, num=10)
y = np.linspace(start=15, stop=50, num=10)

plt.plot(x, y, 'r')

#Comparacao
plt.scatter(preditos, observados)
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x7f992f61f400>
```



In [54]:
```python
print(modelo_ajustado_2.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                Porcoes   R-squared:                       0.950
Model:                            OLS   Adj. R-squared:                  0.929
Method:                 Least Squares   F-statistic:                     47.09
Date:                Thu, 13 May 2021   Prob (F-statistic):           0.000571
Time:                        15:35:46   Log-Likelihood:                -16.416
No. Observations:                   8   AIC:                             38.83
Df Residuals:                       5   BIC:                             39.07
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     30.8750      0.842     36.658      0.000      28.710      33.040
Farinha       10.7500      1.191      9.025      0.000       7.688      13.812
Chocolate      4.2500      1.191      3.568      0.016       1.188       7.312
==============================================================================
Omnibus:                        2.106   Durbin-Watson:                   1.850
Prob(Omnibus):                  0.349   Jarque-Bera (JB):                1.245
Skew:                           0.868   Prob(JB):                        0.537
Kurtosis:                       2.153   Cond. No.                         1.41
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/home/edcarlos/anaconda3/envs/data_science/lib/python3.6/site-packages/scipy/stats/stats.py:160
4: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=8
  "anyway, n=%i" % int(n))
```

.

Explorando o modelo

In [55]:
```python
parametros = modelo_ajustado_2.params
```

In [56]:
```python
parametros
```

Out[56]:
```
Intercept    30.875
Farinha      10.750
Chocolate     4.250
dtype: float64
```

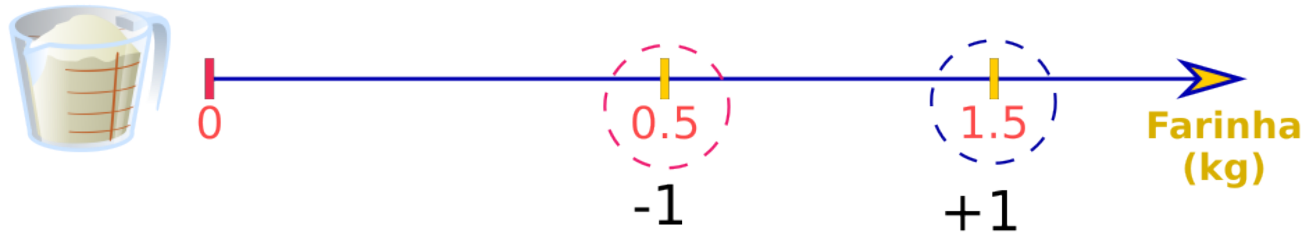
.

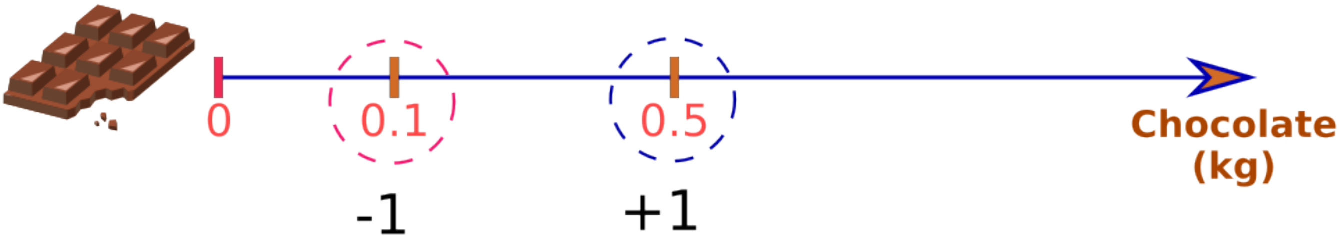
Definindo a função

In [57]:
```python
def modelo_receita(x_f, x_c):

    limite_normalizado = [-1, 1]
    limite_farinha = [0.5, 1.5]
    limite_chocolate = [0.1, 0.5]

    x_f_convertido = np.interp(x_f, limite_farinha, limite_normalizado)
    x_c_convertido = np.interp(x_c, limite_chocolate, limite_normalizado)

    porcoes = parametros['Intercept'] + (parametros['Farinha'] * x_f_convertido) + (parametros[
    return round(porcoes)
```

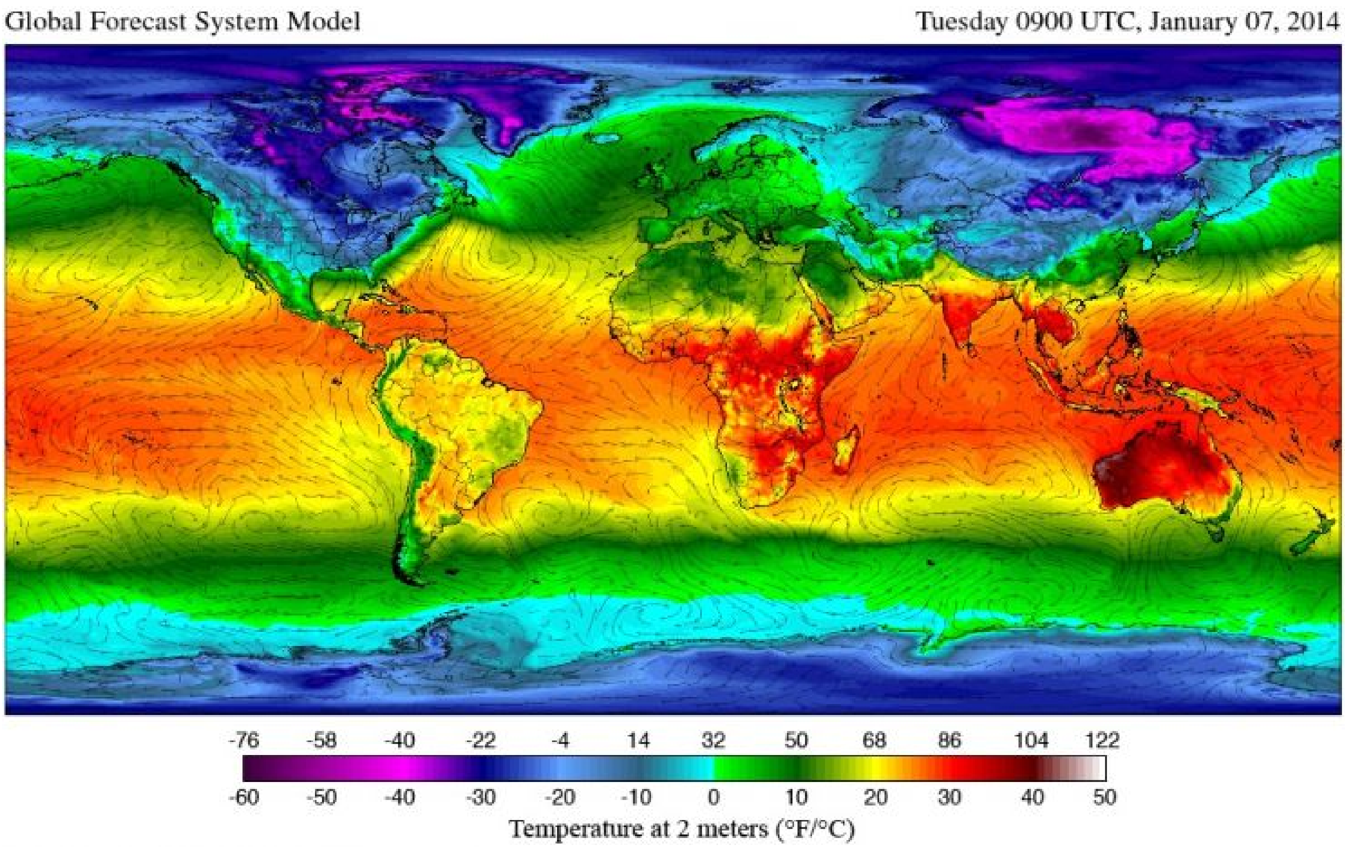
In [58]:
```python
modelo_receita(0.6, 0.1)
```

Out[58]: 18

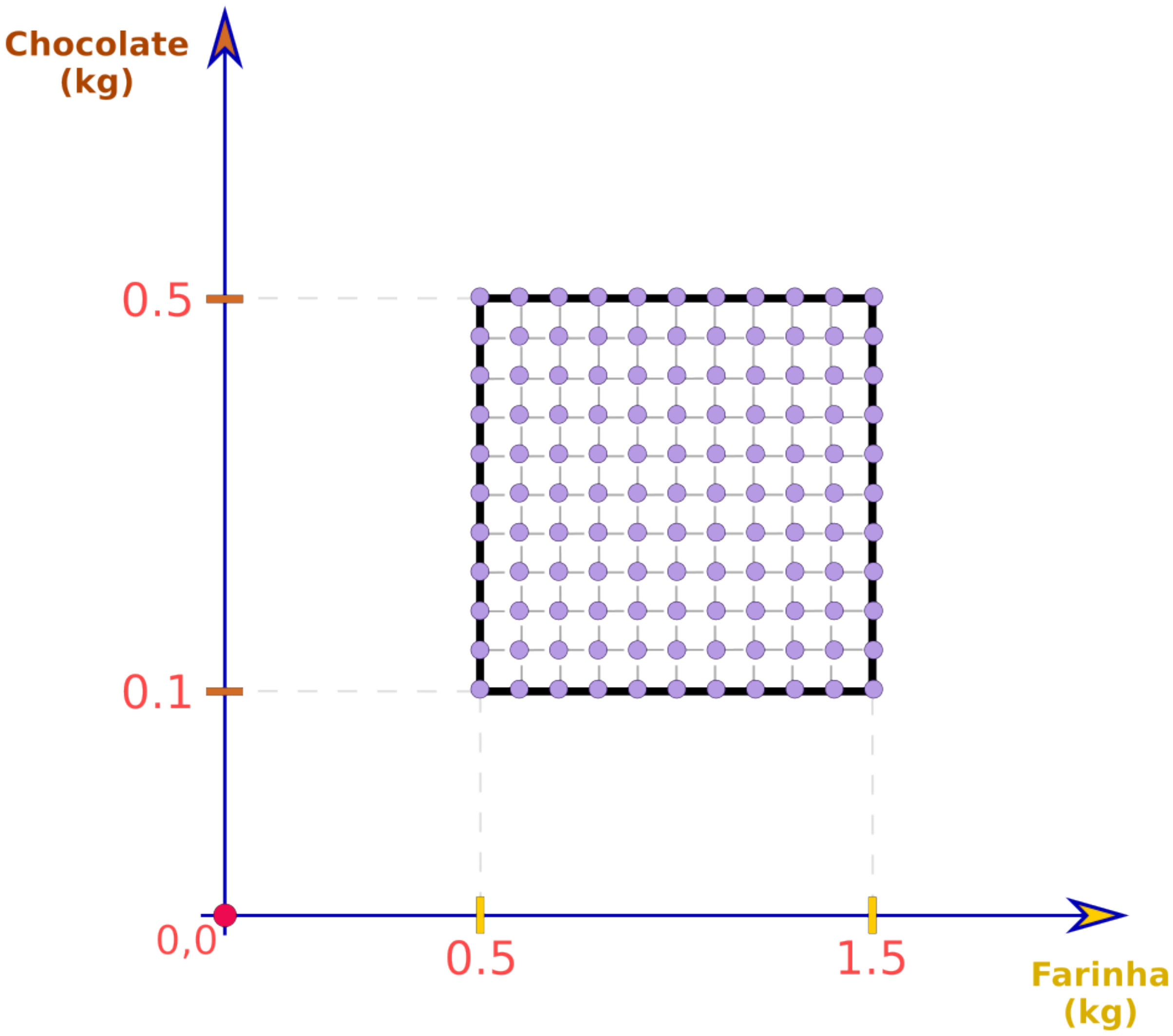




Mapa de cores



Fonte: National Centers for Environmental Prediction



In [59]:
```
x_farinha = np.linspace(start=0.5, stop=1.5, num=10)
```

In [60]:
```
x_farinha
```

Out[60]:
```
array([0.5       , 0.61111111, 0.72222222, 0.83333333, 0.94444444,
       1.05555556, 1.16666667, 1.27777778, 1.38888889, 1.5       ])
```

.

In [61]:
```
x_chocolate = np.linspace(start=0.1, stop=0.5, num=10)
```

In [62]:
```
x_chocolate
```

Out[62]:
```
array([0.1       , 0.14444444, 0.18888889, 0.23333333, 0.27777778,
       0.32222222, 0.36666667, 0.41111111, 0.45555556, 0.5       ])
```

.

In [63]:
```
pontos = []
for cont1 in x_farinha:
    temp = []
    for cont2 in x_chocolate:
        temp.append(modelo_receita(cont1, cont2))
    pontos.append(temp)
```

In [64]:
```
pontos
```

Out[64]:
```
[[16, 17, 18, 19, 20, 21, 22, 22, 23, 24],
 [18, 19, 20, 21, 22, 23, 24, 25, 26, 27],
 [21, 22, 23, 23, 24, 25, 26, 27, 28, 29],
 [23, 24, 25, 26, 27, 28, 29, 30, 31, 32],
 [25, 26, 27, 28, 29, 30, 31, 32, 33, 34],
 [28, 29, 30, 31, 32, 33, 33, 34, 35, 36],
 [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
 [33, 34, 34, 35, 36, 37, 38, 39, 40, 41],
 [35, 36, 37, 38, 39, 40, 41, 42, 43, 43],
 [37, 38, 39, 40, 41, 42, 43, 44, 45, 46]]
```

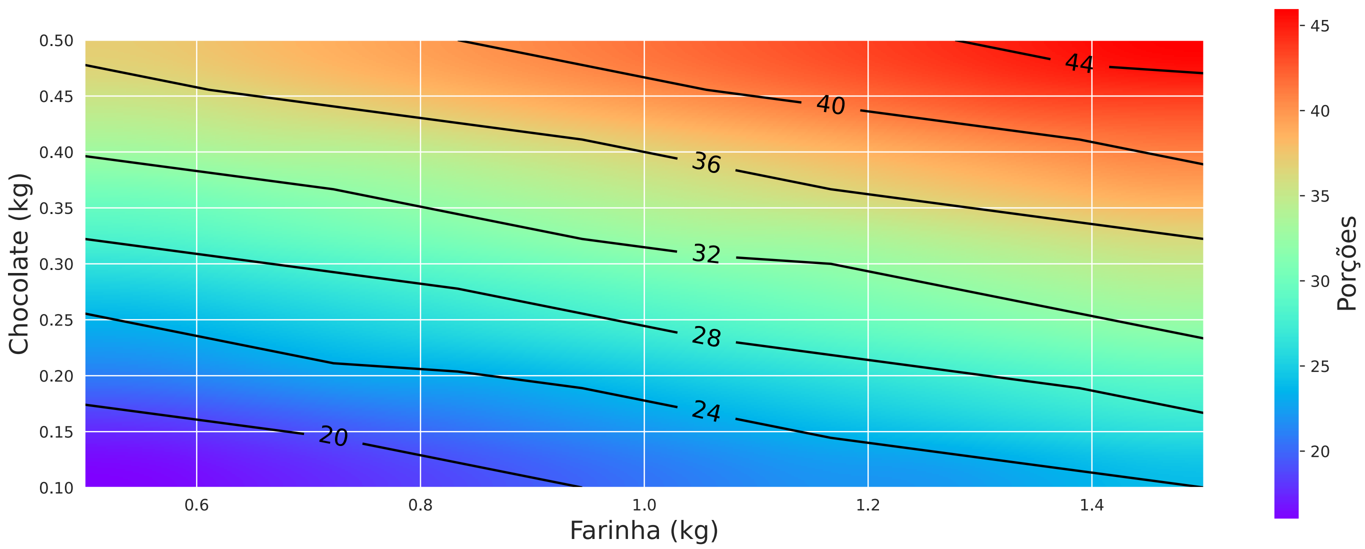
Construindo a superfície de resposta

In [65]:
```
import matplotlib.cm as cm
```

https://matplotlib.org/users/colormaps.html

In [66]:
```
plt.figure(figsize=(16, 6))
plt.xlabel('Farinha (kg)', fontsize=16)
plt.ylabel('Chocolate (kg)', fontsize=16)

mapa_cor = plt.imshow(pontos, origin='lower', cmap=cm.rainbow, interpolation='quadric', extent=

plt.colorbar().set_label('Porções', fontsize = 16)

linhas = plt.contour(x_farinha, x_chocolate, pontos, colors='k', linewidths=1.5)
plt.clabel(linhas, inline=True, fmt='%1.0f', fontsize=15, inline_spacing=10)
```

Out[66]: <a list of 7 text.Text objects>



In [ ]: