In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import altair as alt
alt.renderers.enable('notebook')
alt.data_transformers.disable_max_rows()
from pandas.api.types import CategoricalDtype
```

In [4]:
```
import seaborn as sns
sns.set()
import numpy as np
```

In [5]:
```
df_02_Cap200ab = pd.read_csv('datos_encuesta_2017/02_Cap200ab.csv')
# y1: Production volume per district
# y2: Yield (Kg/he)
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044:
DtypeWarning: Columns (11,14,35,36,37,38,39,40,41,46,51,57,62,72,74,76,79,81,8
4,90,107,109,111) have mixed types. Specify dtype option on import or set low_
memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Filter by "se vendió" y "papa blanca, amarilla, avocado y cafe"

In [6]:
```
df_base = df_02_Cap200ab[df_02_Cap200ab["P220_1_VAL"] > 0][df_02_Cap200ab["P204
_NOM"].str.contains("PAPA BLANCA") | df_02_Cap200ab["P204_NOM"].str.contains("P
APA AMARILLA") | df_02_Cap200ab["P204_NOM"].str.contains("PALTO") | df_02_Cap20
0ab["P204_NOM"].str.contains("CAFE")].reset_index(drop=True)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:1: UserWarning: B
oolean Series key will be reindexed to match DataFrame index.
  """Entry point for launching an IPython kernel.
```

In [7]:
```
df_04_Cap200b = pd.read_csv('datos_encuesta_2017/04_Cap200b.csv')
# y21: [P225] Distancia desde la parcela mas importante hacia la capital distri
tal en horas
```

In [8]:
```
df_base_02 = pd.merge(df_base, df_04_Cap200b, how='left', on=['CCDD', 'CCPP',
'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
```

In [9]:
```
df_07_Cap200E = pd.read_csv('datos_encuesta_2017/07_Cap200E.csv')
# y3: [P235_VAL] Cost of seed per he
# y4: [P237_VAL] Cost of manure per he
# y5: [P239] Cost of fertilize per he
# y6: [P241] Cost of pesticide per he
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044:
DtypeWarning: Columns (11,14) have mixed types. Specify dtype option on import
or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [10]:
```
df_07_Cap200E_base = df_07_Cap200E[df_07_Cap200E["P234_NOM"].str.contains("PAPA
BLANCA") | df_07_Cap200E["P234_NOM"].str.contains("PAPA AMARILLA") | df_07_Cap2
00E["P234_NOM"].str.contains("PALTO") | df_07_Cap200E["P234_NOM"].str.contains(
"CAFE")]
```

In [11]:
```
df_base_03 = pd.merge(df_base_02, df_07_Cap200E_base, how='left', on=['CCDD',
'CCPP', 'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
df_base_03 = df_base_03[df_base_03["P204_NOM"] == df_base_03["P234_NOM"]]
```

In [12]:
```
df_08_Cap300ab = pd.read_csv('datos_encuesta_2017/08_Cap300ab.csv')
# y9 - y20
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044:
DtypeWarning: Columns (11) have mixed types. Specify dtype option on import or
set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [13]:
```
df_base_04 = pd.merge(df_base_03, df_08_Cap300ab, how='left', on=['CCDD', 'CCP
P', 'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
```

In [14]:
```
df_15_Cap800 = pd.read_csv('datos_encuesta_2017/15_Cap800.csv')
# y22: [P801] Pertenece a una asociacion o cooperativa
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044:
DtypeWarning: Columns (11,25,27,38,39,40,41,42,43,44,45,46,64,65,66,67,69,70,7
1,72,79,80,81,82,84,85,86,87) have mixed types. Specify dtype option on import
or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [15]:
```
df_base_05 = pd.merge(df_base_04, df_15_Cap800, how='left', on=['CCDD', 'CCPP',
'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
```

In [16]:
```
df_17_Cap1000 = pd.read_csv('datos_encuesta_2017/17_Cap1000.csv')
# y7: [P1001A_1] Cost of land lease (per he?)
```

In [17]:
```
df_base_06 = pd.merge(df_base_05, df_17_Cap1000, how='left', on=['CCDD', 'CCPP'
, 'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
```

In [18]:
```
df_01_Cap100_2 = pd.read_csv('datos_encuesta_2017/01_Cap100_2.csv')
# [P105_N]: numero de parcela, hay que sumar por UA
# [P105_SUP_ha] Superficie de la parcela en he
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044:
DtypeWarning: Columns (11,21,23) have mixed types. Specify dtype option on imp
ort or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [19]:
```
df_01_Cap100_2["Land_area_he"] = df_01_Cap100_2["P105_SUP_ha"].groupby([df_01_C
ap100_2["CCDD"], df_01_Cap100_2['CCPP'], df_01_Cap100_2['CCDI'], df_01_Cap100_2
['CONGLOMERADO'], df_01_Cap100_2['NSELUA'], df_01_Cap100_2['UA']]).transform('s
um')
# df_01_Cap100_2[["CCDD", "CCPP", 'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA', 'P105
_N', 'P105_SUP_ha', "Land_area_he"]]
```

In [20]:
```
df_base_07 = pd.merge(df_base_06, df_01_Cap100_2, how='left', on=['CCDD', 'CCP
P', 'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA'], suffixes=('', '_y'))
```

In [21]:
```
df_02_Capitulo_IV_NACIONAL = pd.read_csv('datos_censo_mercadodeabastos_2016/Cap
itulo_IV_NACIONAL.csv')
# x7: [P39_1] Selling vegetables spots
```

In [22]:
```
df_base_08 = pd.merge(df_base_07, df_02_Capitulo_IV_NACIONAL, how='left', on=[
'CCDD', 'CCPP', 'CCDI'], suffixes=('', '_y'))
```

In [23]:
```
df_base_final = df_base_08
# df_base_final[df_base_final["P204_NOM"] == "CAFE PERGAMINO"][['CCDD', 'CCPP',
'CCDI', 'CONGLOMERADO', 'NSELUA', 'UA', "P204_NOM", "P234_NOM", "P801", "Sale_p
rice_per_kg"]]
```

In [24]:
```
# Removing NULL VALUES OF DISTRITO FIELD
df_base_final = df_base_final[df_base_final['DISTRITO'].notnull()].reset_index(
drop=True)
```

In [25]:
```
# df_base_final.count()
```

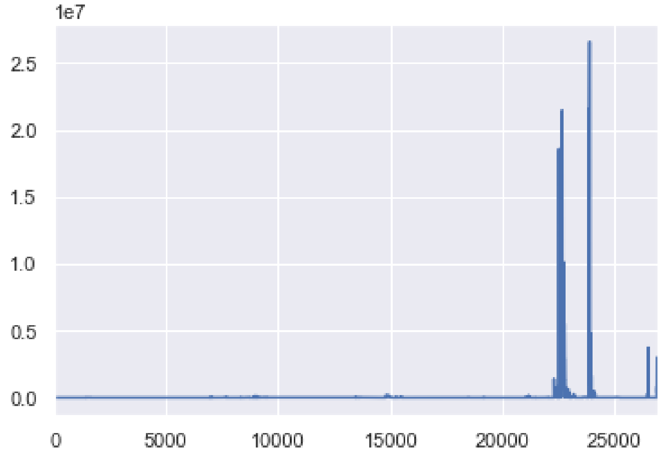
In [26]:
```
# df_02_Cap200ab["P204_NOM"]
# df_07_Cap200E["P234_NOM"]
```

Creando nuevas variables

In [27]:
```
# x2: Sale price per kg
df_base_final["Quantity_for_Sale"] = df_base_final["P220_1_CANT_1"] + df_base_f
inal["P220_1_CANT_2"].apply(pd.to_numeric, errors='coerce')/1000
df_base_final["Sale_price_per_kg"] = df_base_final["P220_1_VAL"] / (df_base_fin
al["P219_EQUIV_KG"] * df_base_final["Quantity_for_Sale"])
df_base_final["Quantity_selled_kg"] = df_base_final["P219_EQUIV_KG"] * df_base_
final["Quantity_for_Sale"]
```

In [28]:
```
df_base_final["Quantity_selled_kg"].plot()
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x11c378b50>



In [29]:
```
# y2: Yield (kg/he)
df_base_final["Harvested_production"] = (df_base_final["P219_CANT_1"] + df_base
_final["P219_CANT_2"].apply(pd.to_numeric, errors='coerce')/1000)*df_base_final
["P219_EQUIV_KG"]
df_base_final["Yield_Kg_per_he"] = df_base_final["Harvested_production"] / df_b
ase_final["P217_SUP_ha"]
```

In [36]:
```
# ((df_base_final["Harvested_production"] - df_base_final["Quantity_selled_k
g"]) < 0).value_counts()
```

In [37]:
```
# y1: Production volume per district (kg)
df_base_final["Volume_Kg_per_District"] = df_base_final['Harvested_production']
.groupby(df_base_final["NOMBREDI"]).transform('sum')
```

In [38]:
```
# x1: Market Connection (Mercado local [P223_1], Mercado regional [P223_2], Mercados de Lima [P223_5], Agroindustria [P223_4], Mercado Exterior [P223_3], No sabe [P223_6])
df_base_final["Market_Connection_01_Local_Market"] = (df_base_final["P223_1"] == "Mercado local (feria local, centro de acopio local)?").astype(int)
df_base_final["Market_Connection_02_Regional_Market"] = (df_base_final["P223_2"] == "Mercado regional (feria regional, centro de acopio regional)?").astype(int)
df_base_final["Market_Connection_03_Lima_Markets"] = (df_base_final["P223_5"] == "Mercados de Lima?").astype(int)
df_base_final["Market_Connection_04_Agroindustry"] = (df_base_final["P223_4"] == "Agroindustria?").astype(int)
df_base_final["Market_Connection_05_Outside_market"] = (df_base_final["P223_3"] == "Mercado exterior?").astype(int)
df_base_final["Market_Connection_06_doesnt_know"] = (df_base_final["P223_6"] == "NO SABE").astype(int)
```

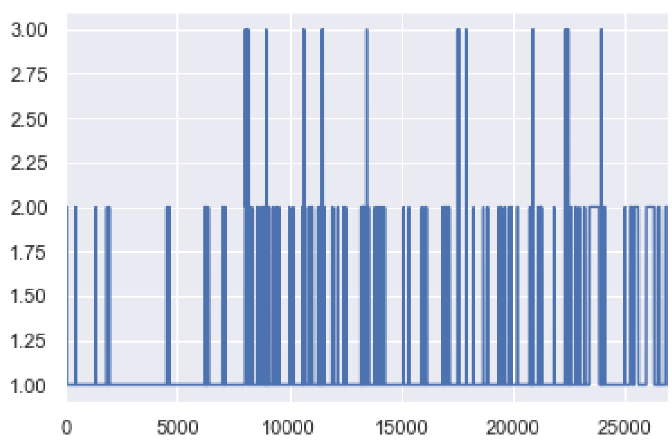
In [39]:
```
# y7: Cost of land lease per he
df_base_final["Cost_of_land_lease_per_he"] = (df_base_final["P1001A_1"] / df_base_final["Land_area_he"])
```

In [40]:
```
choices_market_connection = ["Market_Connection_01_Local_Market", "Market_Connection_02_Regional_Market", "Market_Connection_03_Lima_Markets", "Market_Connection_04_Agroindustry", "Market_Connection_05_Outside_market", "Market_Connection_06_doesnt_know"]
```

In [41]:
```
# Number of market connection choices
df_base_final["Number_of_markets_connections"] = df_base_final[choices_market_connection].sum(axis=1)
```

In [42]:
```
df_base_final["Number_of_markets_connections"].plot()
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x119cbbe10>



In [43]:
```
df_base_final["Number_of_markets_connections"].value_counts()
```

Out[43]:
```
1    24459
2     2344
3      110
Name: Number_of_markets_connections, dtype: int64
```

In [44]:
```
df_base_final["Number_of_markets_connections"].value_counts() / len(df_base_final["Number_of_markets_connections"]) * 100
```

Out[44]:
```
1    90.881730
2     8.709546
3     0.408724
Name: Number_of_markets_connections, dtype: float64
```

In [45]:
```
# df_base_final[df_base_final["Number_of_markets_connections"] == 2][choices_ma
rket_connection]
df_base_final["Array_of_market_connections"] = df_base_final[choices_market_con
nection].apply(lambda x: x.index[x.astype(bool)].tolist(), 1)
```

In [46]:
```
# df_base_final["Array_of_market_connections"]
```

In [47]:
```
def getConnectionModality(x):
    last_one = x[-1]
    if last_one == "Market_Connection_01_Local_Market":
        val = "Local market"
    elif last_one == "Market_Connection_02_Regional_Market":
        val = "Regional market"
    elif last_one == "Market_Connection_03_Lima_Markets":
        val = "Lima markets"
    elif last_one == "Market_Connection_04_Agroindustry":
        val = "Agroindustry"
    elif last_one == "Market_Connection_05_Outside_market":
        val = "Outside market"
    else:
        val = "Doesnt know"
    return val
```

In [48]:
```
df_base_final["Market_connection"] = df_base_final["Array_of_market_connection
s"].apply(getConnectionModality)
```

In [49]:
```
df_base_final["Market_connection"].value_counts()
```

Out[49]:
```
Local market       15208
Regional market     5452
Lima markets        3346
Outside market      1682
Doesnt know         1145
Agroindustry          80
Name: Market_connection, dtype: int64
```

In [50]:
```
df_base_final["Market_connection"].value_counts() / len(df_base_final["Market_c
onnection"]) * 100
```

Out[50]:
```
Local market       56.508007
Regional market    20.257868
Lima markets       12.432653
Outside market      6.249768
Doesnt know         4.254450
Agroindustry        0.297254
Name: Market_connection, dtype: float64
```

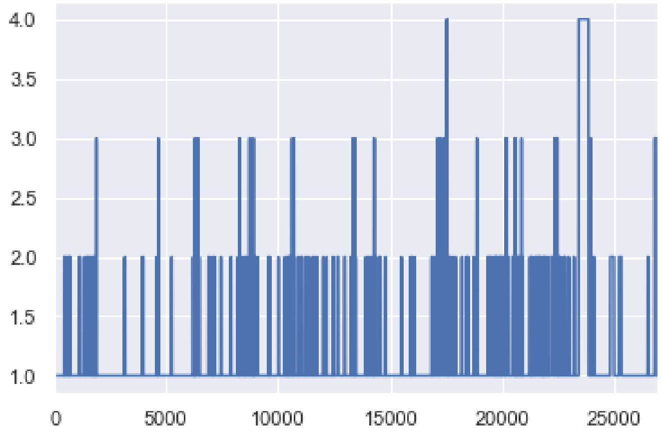
In [51]:
```
df_base_final["Market_Connection_01_Collector"] = (df_base_final["P222_1"] ==
"Acopiador").astype(int)
df_base_final["Market_Connection_02_Wholesaler"] = (df_base_final["P222_2"] ==
"Comerciante mayorista").astype(int)
df_base_final["Market_Connection_03_Retailer"] = (df_base_final["P222_3"] == "C
omerciante minorista").astype(int)
df_base_final["Market_Connection_04_AssociationCooperative"] = (df_base_final[
"P222_4"] == "Asociación / cooperativa").astype(int)
df_base_final["Market_Connection_05_CompanyAgribusiness"] = (df_base_final["P22
2_5"] == "Empresa / agroindustria").astype(int)
df_base_final["Market_Connection_06_FinalConsumer"] = (df_base_final["P222_6"]
== "Consumidor Final").astype(int)
df_base_final["Market_Connection_07_doesnt_know"] = (df_base_final["P222_7"] ==
"NO SABE").astype(int)
```

In [52]:
```
choices_next_market_connection = ["Market_Connection_01_Collector", "Market_Connection_02_Wholesaler", "Market_Connection_03_Retailer", "Market_Connection_04_AssociationCooperative", "Market_Connection_05_CompanyAgribusiness", "Market_Connection_06_FinalConsumer", "Market_Connection_07_doesnt_know"]
```

In [53]:
```
df_base_final["Number_of_next_markets_connections"] = df_base_final[choices_next_market_connection].sum(axis=1)
```

In [54]:
```
df_base_final["Number_of_next_markets_connections"].plot()
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x119a7e850>



In [55]:
```
df_base_final["Number_of_next_markets_connections"].value_counts()
```

Out[55]:
```
1    23777
2     2485
4      449
3      202
Name: Number_of_next_markets_connections, dtype: int64
```

In [56]:
```
df_base_final["Number_of_next_markets_connections"].value_counts() / len(df_base_final["Number_of_next_markets_connections"]) * 100
```

Out[56]:
```
1    88.347639
2     9.233456
4     1.668339
3     0.750567
Name: Number_of_next_markets_connections, dtype: float64
```

In [57]:
```
df_base_final["Array_of_next_market_connections"] = df_base_final[choices_next_market_connection].apply(lambda x: x.index[x.astype(bool)].tolist(), 1)
```

In [58]:
```
def getNextConnectionModality(x):
    last_one = x[-1]
    if last_one == "Market_Connection_01_Collector":
        val = "Collector"
    elif last_one == "Market_Connection_02_Wholesaler":
        val = "Wholesaler"
    elif last_one == "Market_Connection_03_Retailer":
        val = "Retailer"
    elif last_one == "Market_Connection_04_AssociationCooperative":
        val = "Association / Cooperative"
    elif last_one == "Market_Connection_05_CompanyAgribusiness":
        val = "Company / Agribusiness"
    elif last_one == "Market_Connection_06_FinalConsumer":
        val = "Final consumer"
    else:
        val = "Doesnt know"
    return val
```

In [59]:
```
df_base_final["Market_next_connection"] = df_base_final["Array_of_next_market_c
onnections"].apply(getNextConnectionModality)
```

In [60]:
```
df_base_final["Market_next_connection"].value_counts()
```

Out[60]:
```
Wholesaler                  9326
Collector                   7407
Retailer                    5562
Final consumer              3218
Company / Agribusiness       922
Association / Cooperative    478
Name: Market_next_connection, dtype: int64
```

In [61]:
```
df_base_final["Market_next_connection"].value_counts() / len(df_base_final["Mar
ket_next_connection"]) * 100
```

Out[61]:
```
Wholesaler                  34.652398
Collector                   27.522015
Retailer                    20.666592
Final consumer              11.957047
Company / Agribusiness       3.425854
Association / Cooperative    1.776093
Name: Market_next_connection, dtype: float64
```

In [62]:
```
# Costs variables
# df_base_final["Cost_of_land_lease_per_he"]
df_base_final["Cost_of_seed_per_he"] = df_base_final["P235_VAL"]
df_base_final["Cost_of_manure_per_he"] = df_base_final["P237_VAL"]
df_base_final["Cost_of_fertilize_per_he"] = df_base_final["P239"]
df_base_final["Cost_of_pesticide_per_he"] = df_base_final["P241"]
```

In [63]:
```
# Quality variable: Certified seeds
df_base_final["Certified_seed"] = df_base_final["P214"].astype('category').cat.
rename_categories({'¿No certificada?': 0, '¿Certificada?': 1})
```

In [64]:
```
# Good practices
df_base_final["Good_practice_01"] = df_base_final["P301A_1"].astype('category')
.cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_02"] = df_base_final["P301A_2"].astype('category')
.cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_03"] = df_base_final["P301A_9"].astype('category')
.cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_04"] = df_base_final["P301A_10"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_05"] = df_base_final["P301A_11"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_06"] = df_base_final["P301A_12"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_07"] = df_base_final["P301A_12A"].astype('categor
y').cat.rename_categories({2: 0, 1: 1})
df_base_final["Good_practice_08"] = df_base_final["P301A_13"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_09"] = df_base_final["P301A_14"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_10"] = df_base_final["P301A_15"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_11"] = df_base_final["P301A_16"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
df_base_final["Good_practice_12"] = df_base_final["P301A_17"].astype('category'
).cat.rename_categories({'No': 0, 'Si': 1})
```

In [65]:
```
# df_base_final[["P301A_1", "P301A_2", "P301A_9", "P301A_10", "P301A_11", "P301
A_12", "P301A_12A", "P301A_13", "P301A_14", "P301A_15", "P301A_16", "P301A_1
7"]]
# df_base_final[["Good_practice_01", "Good_practice_02", "Good_practice_03", "G
ood_practice_04", "Good_practice_05", "Good_practice_06", "Good_practice_07",
 "Good_practice_08", "Good_practice_09", "Good_practice_10", "Good_practice_1
1", "Good_practice_12"]]
```

In [66]:
```
# Demand connecting variable: Time to district capital
df_base_final["Time_to_district_capital"] = df_base_final["P225"]
```

In [67]:
```
# Belongs to an association or cooperative
df_base_final["Belongs_to_an_association_or_cooperative"] = df_base_final["P80
1"].astype('category').cat.rename_categories({'No': 0, 'Si': 1})
```

Removing "Doesnt know" rows of Market connection variable

In [68]:
```
_df_base_final = df_base_final[df_base_final["Market_connection"] != "Doesnt kn
ow"].reset_index(drop=True)
```

In [69]:
```
_df_base_final = _df_base_final[_df_base_final["Market_next_connection"] != "Do
esnt know"].reset_index(drop=True)
```

In [70]:
```
_df_base_final["Market_connection"].value_counts()
```

Out[70]:
```
Local market       15208
Regional market     5452
Lima markets        3346
Outside market      1682
Agroindustry          80
Name: Market_connection, dtype: int64
```

In [71]:
```
_df_base_final["Market_next_connection"].value_counts()
```

Out[71]:
```
Wholesaler                    9037
Collector                     7055
Retailer                      5245
Final consumer                3081
Company / Agribusiness         922
Association / Cooperative      428
Name: Market_next_connection, dtype: int64
```

In [72]:
```
ordered_market_connection = ["Local market", "Regional market", "Lima markets", 
"Agroindustry", "Outside market"]
ordered_next_market_connection = ["Collector", "Wholesaler", "Retailer", "Assoc
iation / Cooperative", "Company / Agribusiness", "Final consumer"]
```

In [73]:
```
cat_type = CategoricalDtype(categories=ordered_market_connection, ordered=True)
_df_base_final["Market_connection"] = _df_base_final["Market_connection"].astyp
e(cat_type)
```

In [74]:
```
cat_type_next = CategoricalDtype(categories=ordered_next_market_connection, ord
ered=True)
_df_base_final["Market_next_connection"] = _df_base_final["Market_next_connecti
on"].astype(cat_type_next)
```

In [75]:
```
_df_base_final["Market_next_connection"][:5]
```

Out[75]:
```
0      Collector
1      Collector
2      Collector
3     Wholesaler
4     Wholesaler
Name: Market_next_connection, dtype: category
Categories (6, object): [Collector < Wholesaler < Retailer < Association / Coo
perative < Company / Agribusiness < Final consumer]
```

In [76]:
```
_df_base_final["Market_connection_codes"] = _df_base_final["Market_connection"]
.cat.codes + 1
_df_base_final["Market_connection_codes"] = _df_base_final["Market_connection_c
odes"].astype("category", ordered=True)
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3291:
FutureWarning: specifying 'categories' or 'ordered' in .astype() is deprecate
d; pass a CategoricalDtype instead
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [77]:
```
_df_base_final["Market_connection_codes"][:5]
```

Out[77]:
```
0     1
1     1
2     1
3     1
4     1
Name: Market_connection_codes, dtype: category
Categories (5, int64): [1 < 2 < 3 < 4 < 5]
```

In [78]:
```
_df_base_final["Market_next_connection_codes"] = _df_base_final["Market_next_co
nnection"].cat.codes + 1
_df_base_final["Market_next_connection_codes"] = _df_base_final["Market_next_co
nnection_codes"].astype("category", ordered=True)
```

In [79]:
```
_df_base_final["Market_next_connection_codes"][:5]
```

Out[79]:
```
0    1
1    1
2    1
3    2
4    2
Name: Market_next_connection_codes, dtype: category
Categories (6, int64): [1 < 2 < 3 < 4 < 5 < 6]
```

In [80]:
```
def getCropName(name):
    if name == "PALTO":
        val = "Avocado"
    elif name == "CAFE PERGAMINO":
        val = "Coffee"
    elif name == "PAPA AMARILLA":
        val = "Yellow potato"
    elif name == "PAPA BLANCA":
        val = "White potato"
    else:
        val = "-1"
    return val
```

In [81]:
```
# df_02_Cap200ab["P204_NOM"]
# df_07_Cap200E["P234_NOM"]
_df_base_final["P234_NOM"].value_counts()
```

Out[81]:
```
PAPA BLANCA       12261
CAFE PERGAMINO     5921
PALTO              5866
PAPA AMARILLA      1720
Name: P234_NOM, dtype: int64
```

In [82]:
```
_df_base_final["Crop_name"] = _df_base_final["P204_NOM"].apply(getCropName)
```

In [83]:
```
_df_base_final["Crop_name"].value_counts()
```

Out[83]:
```
White potato     12261
Coffee            5921
Avocado           5866
Yellow potato     1720
Name: Crop_name, dtype: int64
```

In [84]:
```
# _df_base_final.count()
# _df_base_final[["ESTRATO", "RESFIN", "REGION", "DOMINIO", "FACTOR", "CODIG
O"]]
```

Variables

In [85]:
```
_df_base_final['Selling_vegetables_spots_per_district'] = _df_base_final['P39_
1'].groupby(_df_base_final["DISTRITO"]).transform('sum')
```

In [86]:
```
# _df_base_final["Selling_vegetables_spots_per_district"].value_counts()
```

In [87]:
```
_df_base_final["UA_ID"] = _df_base_final["CCDD"].astype(str) + _df_base_final[
"CCPP"].astype(str) + _df_base_final["CCDI"].astype(str) + _df_base_final["CONG
LOMERADO"].astype(str) + _df_base_final["NSELUA"].astype(str) + _df_base_final[
"UA"].astype(str)
```

In [88]:
```
# _df_base_final["Harvested_production"]
```

In [89]:
```
_df_base_final["Land_owner"] = (_df_base_final["Cost_of_land_lease_per_he"] <=
0).astype(int)
_df_base_final["Land_owner"].value_counts()
```

Out[89]:
```
1    18454
0     7314
Name: Land_owner, dtype: int64
```

In [90]:
```
# _df_base_final.to_csv(r'db_white_yellow_potato_avocado_coffee.csv')
_df_base_final.to_csv(r'db_white_yellow_potato_avocado_coffee_vegspots.csv')
```

In [91]:
```
_columns = ["Unnamed: 0", "UA_ID", "ANIO", "CCDD", "NOMBREDD", "CCPP", "NOMBREP
V", "CCDI", "NOMBREDI", "CONGLOMERADO", "NSELUA", "UA",
            "RESFIN", "REGION", "DOMINIO", "FACTOR", "CODIGO",
            "Crop_name", "Harvested_production", "Volume_Kg_per_District", "Yie
ld_Kg_per_he", "Cost_of_seed_per_he", "Cost_of_manure_per_he",
            "Cost_of_fertilize_per_he", "Cost_of_pesticide_per_he", "Cost_of_la
nd_lease_per_he",
            "Good_practice_01", "Good_practice_02", "Good_practice_03", "Good_p
ractice_04", "Good_practice_05",
            "Good_practice_06", "Good_practice_07", "Good_practice_08", "Good_p
ractice_09", "Good_practice_10",
            "Good_practice_11", "Good_practice_12", "Certified_seed", "Time_to_
district_capital",
            "Belongs_to_an_association_or_cooperative", "Market_connection_code
s", "Market_next_connection_codes", "Sale_price_per_kg", "Land_owner",
            "Selling_vegetables_spots_per_district", "Quantity_selled_kg"]
```

In [92]:
```
_df_base_final[_columns].to_csv(r'db_white_yellow_potato_avocado_coffee_filtere
d_by_variables_vegspots.csv')
```

In [93]:
```
df2 = _df_base_final[_columns].rename({'UA_ID': 'id', 'ANIO': 'year', 'CCDD':
'ccdd', 'NOMBREDD': 'department',
            'CCPP': 'ccpp', 'NOMBREPV': 'province', 'CCDI': 'ccdi', 'NOMBREDI':
'district', 'CONGLOMERADO': 'conglomerate',
            'NSELUA': 'nselua', 'UA': 'ua', 'RESFIN': 'resfin', 'REGION': 'regi
on', 'DOMINIO': 'domain',
            'FACTOR': 'factor', 'CODIGO': 'code', 'Crop_name': 'crop_name', 'Ha
rvested_production': "y0", 'Volume_Kg_per_District': 'y1',
            'Yield_Kg_per_he': 'y2', 'Cost_of_seed_per_he': 'y3', 'Cost_of_manu
re_per_he': 'y4',
            'Cost_of_fertilize_per_he': 'y5', 'Cost_of_pesticide_per_he': 'y6',
'Cost_of_land_lease_per_he': 'y7',
            'Good_practice_01': 'y8', 'Good_practice_02': 'y9', 'Good_practice_
03': 'y10', 'Good_practice_04': 'y11',
            'Good_practice_05': 'y12', 'Good_practice_06': 'y13', 'Good_practic
e_07': 'y14', 'Good_practice_08': 'y15',
            'Good_practice_09': 'y16', 'Good_practice_10': 'y17', 'Good_practic
e_11': 'y18', 'Good_practice_12': 'y19',
            'Certified_seed': 'x1', 'Time_to_district_capital': 'x2', 'Belongs_
to_an_association_or_cooperative': 'x3',
            'Market_connection_codes': 'x4', 'Market_next_connection_codes': 'm
e', 'Sale_price_per_kg': 'x5', 'Land_owner': 'x6',
            'Selling_vegetables_spots_per_district': 'x7', "Quantity_selled_kg"
: "qtykg"}, axis=1)
```

In [94]:
```
# df2.to_csv(r'db_wypotato_avocado_coffee_semmodel_variables.csv')
df2.to_csv(r'db_wypotato_avocado_coffee_semmodel_variables_vegspots.csv')
```

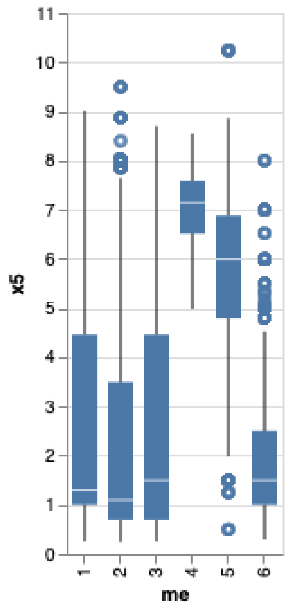
In [95]:
```
df2['id'] = df2['id'].astype(str)
```

In [96]:
```
df2["crop_name"].value_counts()
```

Out[96]:
```
White potato     12261
Coffee            5921
Avocado           5866
Yellow potato     1720
Name: crop_name, dtype: int64
```

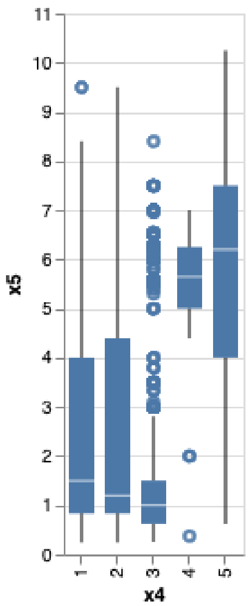
In [97]:
```
alt.Chart(df2[['me', 'x5']]).mark_boxplot().encode(
    x='me',
    y='x5:Q'
)

# df2[['id']].dtypes

# df2['id'].value_counts()
```

Out[97]:



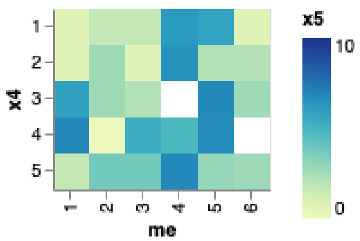
In [98]:
```
alt.Chart(df2[['x4', 'x5']]).mark_boxplot().encode(
    x='x4',
    y='x5:Q'
)
```

Out[98]:



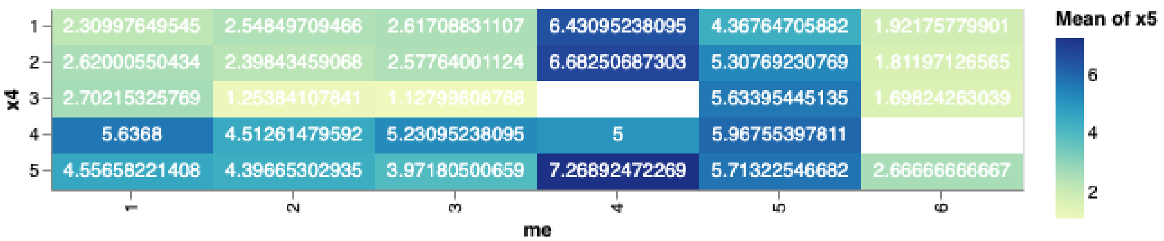
In [99]:
```
alt.Chart(df2[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='x5:Q'
)
```

Out[99]:



In [100]:
```
# Configure common options
base = alt.Chart(df2[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(x5):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap + text
```

Out[100]:



In [101]:

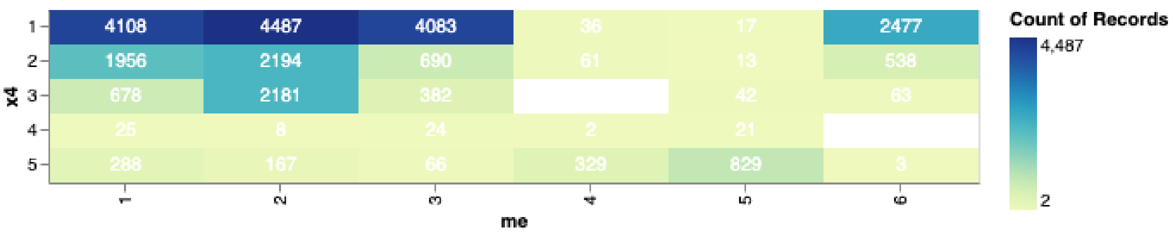
```
# Configure common options
base2 = alt.Chart(df2[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(x5):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap2 + text2

# REvisar el valor de (4,2) Box plot de cada combinacion
```

Out[101]:

In [102]:

```
# Configure common options
base3 = alt.Chart(df2[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(x5):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text3 = base.mark_text(baseline='middle').encode(
    text='count(x5):Q',
    color=alt.condition(
        alt.datum.x5 > 1000,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap3 + text3
```

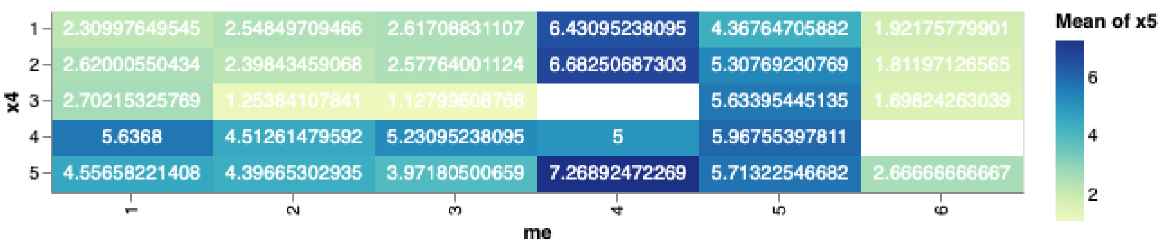
Out[102]:



In [103]:

```
heatmap + text
```

Out[103]:



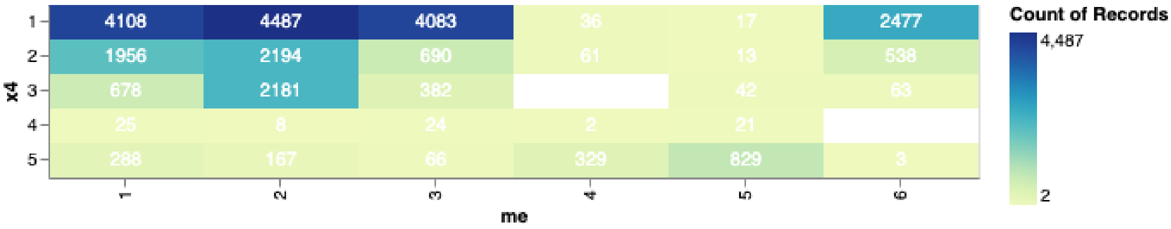
In [104]:

```
heatmap2 + text2
```

Out[104]:

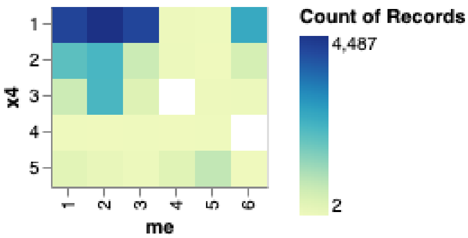
In [105]:
```
heatmap3 + text3
```

Out[105]:



In [106]:
```
alt.Chart(df2[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count():Q'
)
```

Out[106]:



In [107]:
```
dummy_me = pd.get_dummies(df2["me"],prefix='me')
dummy_x4 = pd.get_dummies(df2["x4"],prefix='x4')
```

In [108]:
```
# df2["qtykg"].value_counts()
```

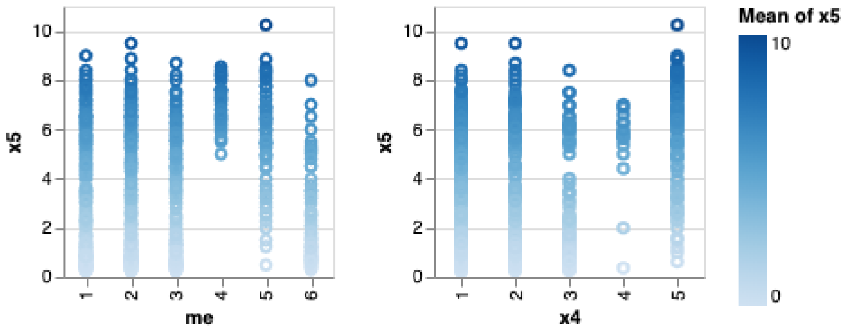
In [109]:
```
df2 = pd.concat([df2, dummy_me, dummy_x4], axis=1)
```

In [110]:
```
df2.columns
```

Out[110]:
```
Index(['Unnamed: 0', 'id', 'year', 'ccdd', 'department', 'ccpp', 'province',
       'ccdi', 'district', 'conglomerate', 'nselua', 'ua', 'resfin', 'region',
       'domain', 'factor', 'code', 'crop_name', 'y0', 'y1', 'y2', 'y3', 'y4',
       'y5', 'y6', 'y7', 'y8', 'y9', 'y10', 'y11', 'y12', 'y13', 'y14', 'y15',
       'y16', 'y17', 'y18', 'y19', 'x1', 'x2', 'x3', 'x4', 'me', 'x5', 'x6',
       'x7', 'qtykg', 'me_1', 'me_2', 'me_3', 'me_4', 'me_5', 'me_6', 'x4_1',
       'x4_2', 'x4_3', 'x4_4', 'x4_5'],
      dtype='object')
```

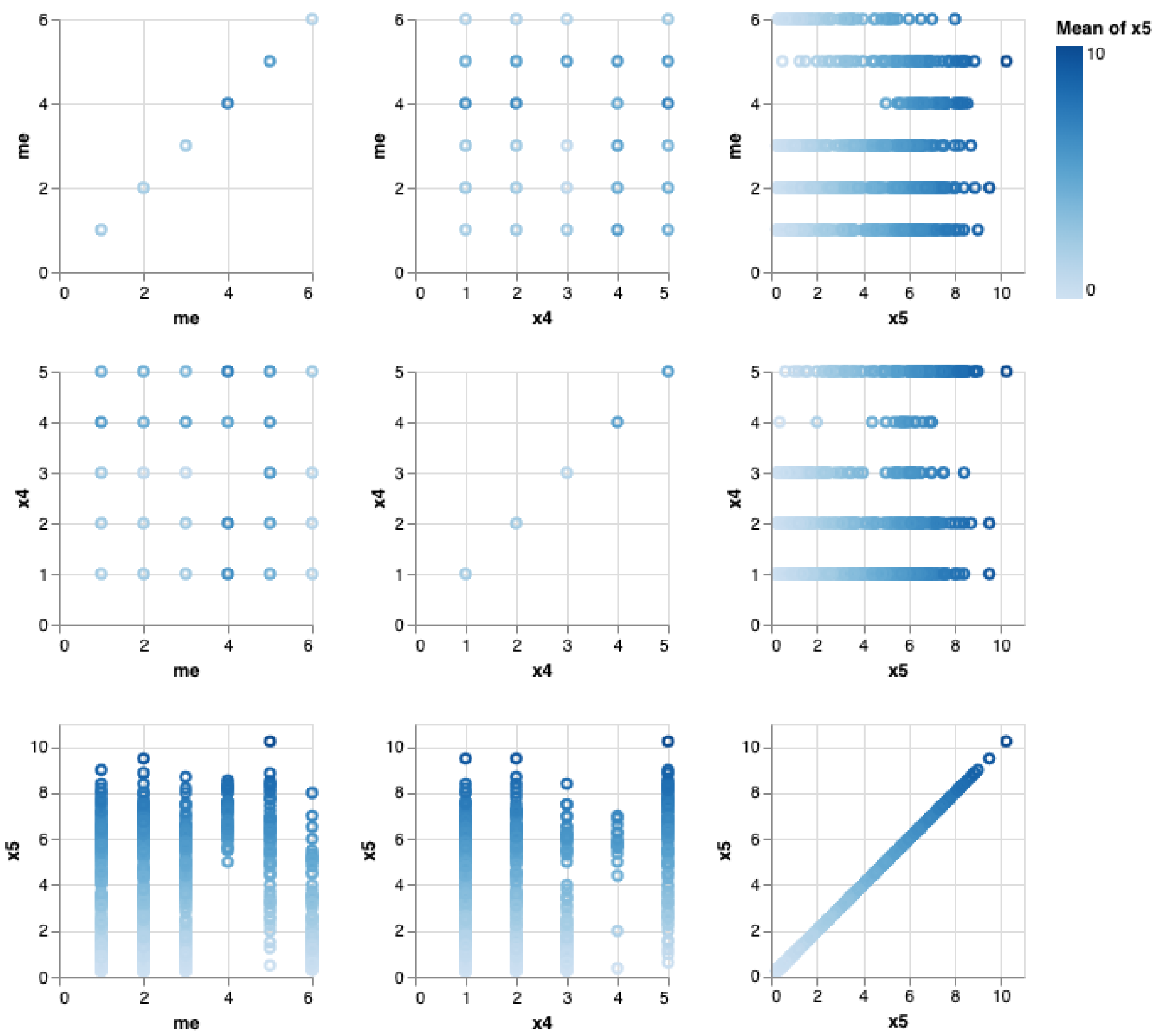
In [111]:
```
alt.Chart(df2).mark_point().encode(
    alt.X(alt.repeat("column"), type='nominal'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='mean(x5):Q'
).properties(
    width=150,
    height=150
).repeat(
    column=['me', 'x4'],
    row=['x5']
)
```

Out[111]:



In [112]:
```
alt.Chart(df2).mark_point().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='mean(x5):Q'
).properties(
    width=150,
    height=150
).repeat(
    column=['me', 'x4', 'x5'],
    row=['me', 'x4', 'x5']
)
```

Out[112]:



In [113]:
```
# alt.Chart(df2).mark_boxplot().encode(
#     alt.X(alt.repeat("column"), type='quantitative'),
#     alt.Y(alt.repeat("row"), type='quantitative'),
# ).repeat(
#     column=['me', 'x4', 'x5'],
#     row=['me', 'x4', 'x5']
# )

# #  x='x4',
# #     y='x5:Q
```

Spliting db to each Crop type

In [114]:
```
df2_wpotato = df2[df2['crop_name'] == 'White potato'].reset_index(drop=True)
df2_ypotato = df2[df2['crop_name'] == 'Yellow potato'].reset_index(drop=True)
df2_avocado = df2[df2['crop_name'] == 'Avocado'].reset_index(drop=True)
df2_coffee = df2[df2['crop_name'] == 'Coffee'].reset_index(drop=True)
```

In [115]:
```
# df2_wpotato['y1'].value_counts()
```

In [116]:
```
_variables = ['id', 'y0', 'y1', 'y2', 'y3', 'y4', 'y5', 'y6', 'y7', 'y8', 'y9',
'y10', 'y11', 'y12', 'y13', 'y14', 'y15',
              'y16', 'y17', 'y18', 'y19', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6',
'x7', 'me', 'qtykg']
```

In [117]:
```
df2_wpotato[_variables].to_csv(r'db_wpotato_semmodel_variables_vegspots.csv')
df2_ypotato[_variables].to_csv(r'db_ypotato_semmodel_variables_vegspots.csv')
df2_avocado[_variables].to_csv(r'db_avocado_semmodel_variables_vegspots.csv')
df2_coffee[_variables].to_csv(r'db_coffee_semmodel_variables_vegspots.csv')
```

In [118]:
```
df2_avocado['crop_name'].value_counts()
```

Out[118]:
```
Avocado    5866
Name: crop_name, dtype: int64
```

In [119]:
```
len(_df_base_final["Market_connection"])
```

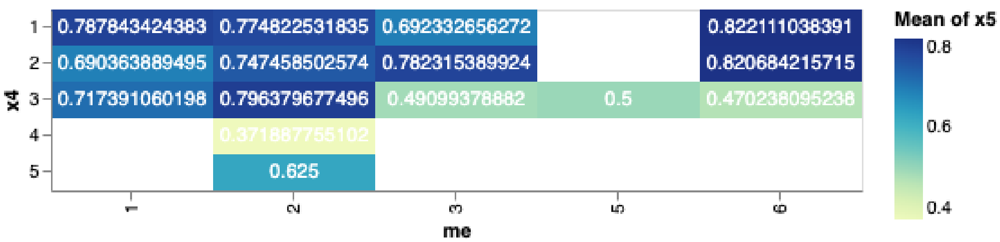
Out[119]:
```
25768
```

In [120]:
```
# 12261 - df2_wpotato.count()
n = df2_wpotato[df2_wpotato['x2'].notnull()].reset_index(drop=True)
# 9464 - n.count()
n1 = n[n['y4'].notnull()].reset_index(drop=True)
# 8294 - n1.count()
n2 = n1[n1['y5'].notnull()].reset_index(drop=True)
# 7637 - n2.count()
n3 = n2[n2['y6'].notnull()].reset_index(drop=True)
# 7379 - n3.count()
n4 = n3[n3['y3'].notnull()].reset_index(drop=True)
# 7352 - n4.count()
# # n4[_variables].to_csv(r'db_wpotato_semmodel_notemptyvariables.csv')
n4[_variables].to_csv(r'db_wpotato_semmodel_notemptyvariables_vegspots.csv')
```

In [121]:

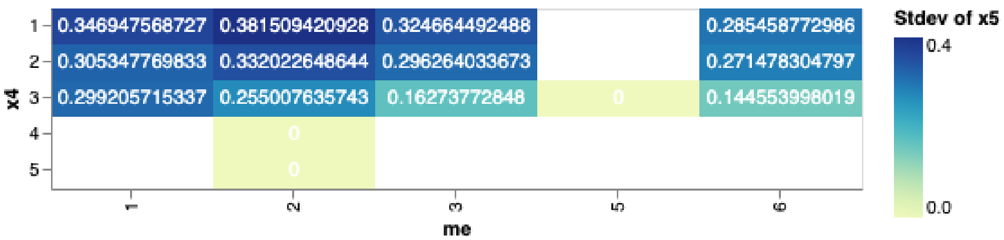
```
# Configure common options
base = alt.Chart(n4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(x5):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap + text
```

Out[121]:



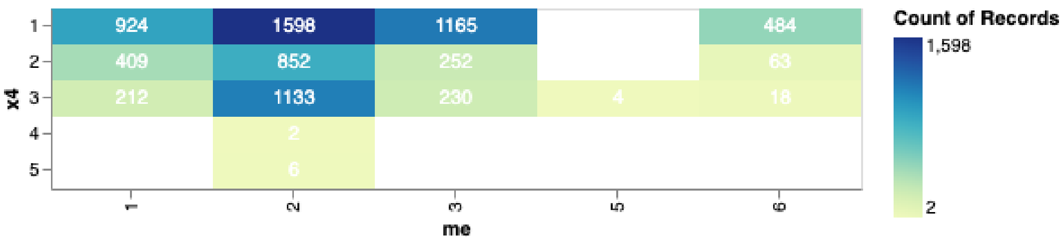
In [122]:
```
# Configure common options
base2 = alt.Chart(n4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(x5):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap2 + text2

# REvisar el valor de (4,2) Box plot de cada combinacion
```

Out[122]:



In [123]:
```python
# Configure common options
base3 = alt.Chart(n4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(x5):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text3 = base.mark_text(baseline='middle').encode(
    text='count(x5):Q',
    color=alt.condition(
        alt.datum.x5 > 1000,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap3 + text3
```

Out[123]:



In [124]:
```python
n4["x4"].value_counts()
```

Out[124]:
```
1    4171
3    1597
2    1576
5       6
4       2
Name: x4, dtype: int64
```

In [125]:
```python
n5 = n4.drop(n4[(n4["x4"] > 3) | (n4["me"] == 3) | (n4["me"] == 4) | (n4["me"]
== 5)].index).reset_index(drop=True)
```

In [126]:
```python
n5["me"].value_counts()
```

Out[126]:
```
2    3583
1    1545
6     565
5       0
4       0
3       0
Name: me, dtype: int64
```

In [127]:
```python
# n5
```

In [128]:
```python
n5[_variables].to_csv(r'db_wpotato_semmodel_notemptyvariables_vegspots_3x3.csv'
)
```

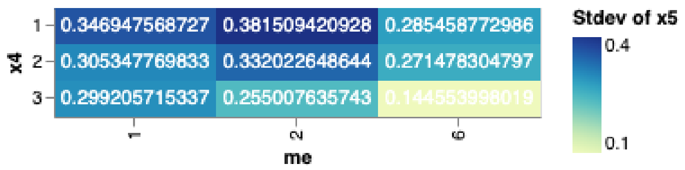
In [129]:
```
# Configure common options
base = alt.Chart(n5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(x5):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base2 = alt.Chart(n5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(x5):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base3 = alt.Chart(n5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(x5):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)
```

In [130]:
```
heatmap + text
```

Out[130]:

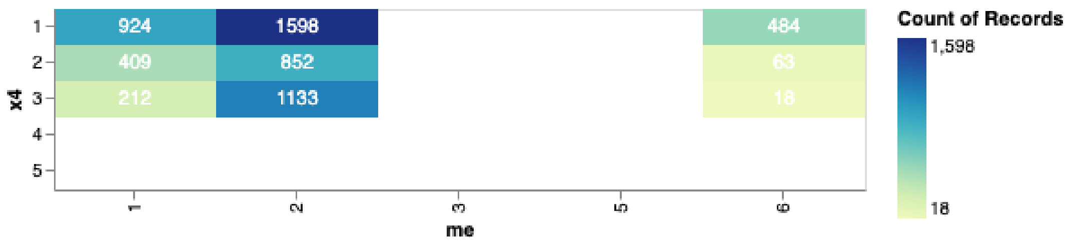
In [132]:
```
heatmap2 + text2
```

Out[132]:



In [133]:
```
heatmap3 + text3
```

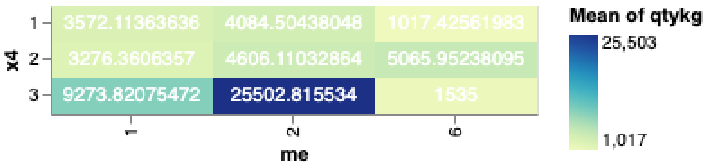
Out[133]:



In [134]:
```
# Configure common options
base = alt.Chart(n5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(qtykg):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(qtykg):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base2 = alt.Chart(n5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(qtykg):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(qtykg):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base3 = alt.Chart(n5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(qtykg):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)
```

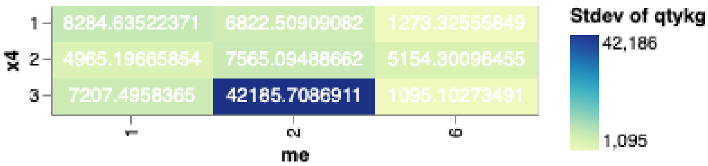
In [135]:
```
heatmap + text
```

Out[135]:



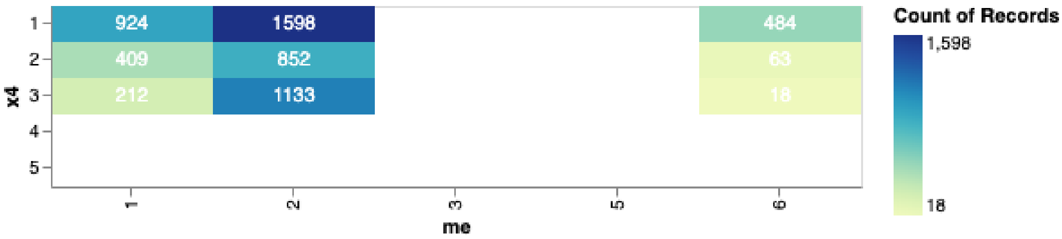
In [136]:
```
heatmap2 + text2
```

Out[136]:

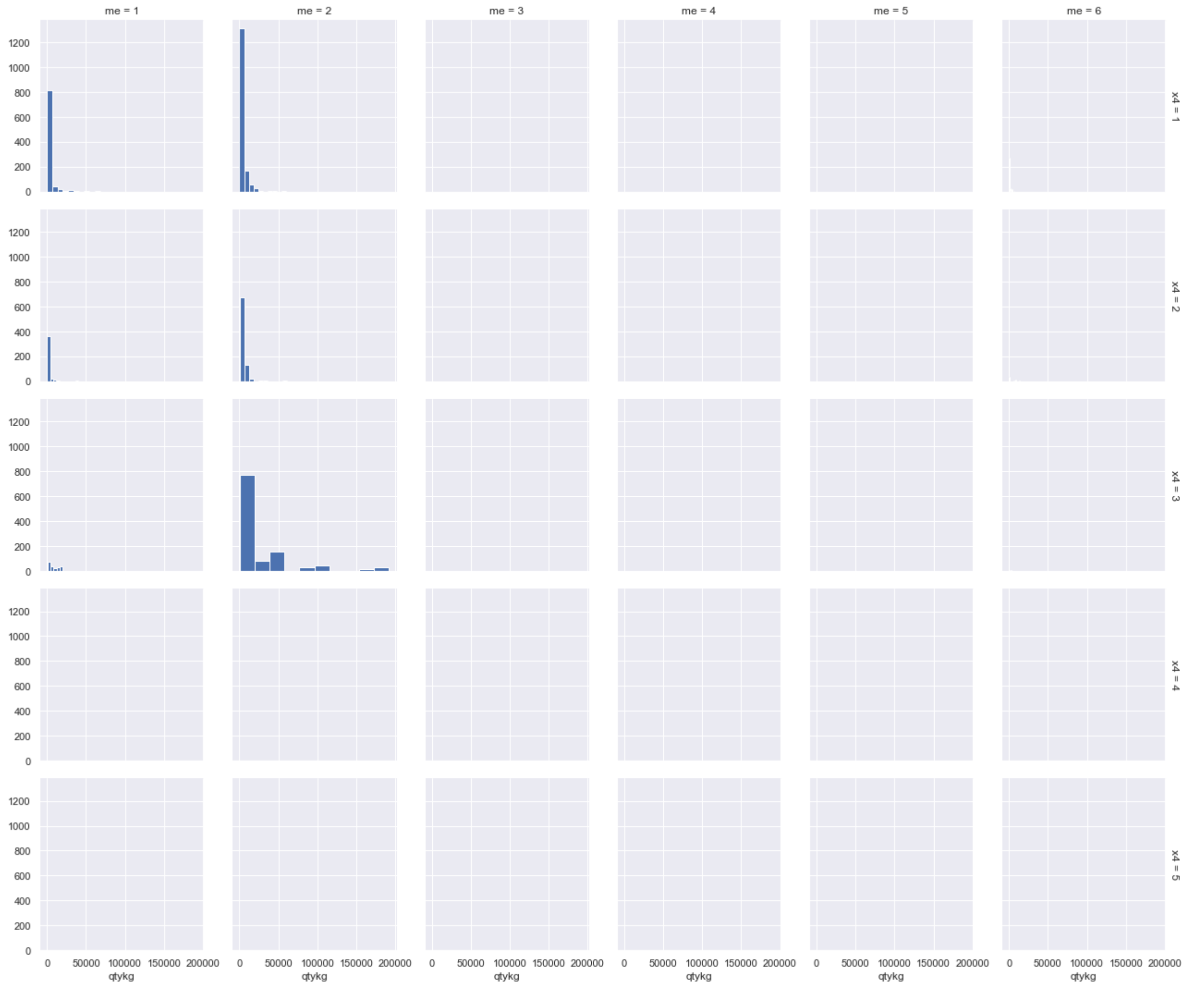


In [137]:
```
heatmap3 + text3
```

Out[137]:



In [138]:
```
grid_n5 = sns.FacetGrid(n5, row="x4", col="me", margin_titles=True)
grid_n5.map(plt.hist, "qtykg");
```

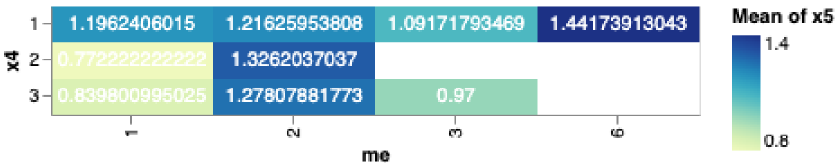


In [139]:
```
# 1720 - df2_ypotato.count()
yp = df2_ypotato[df2_ypotato['domain'].notnull()].reset_index(drop=True)
# 1517 - yp.count()
yp1 = yp[yp['y5'].notnull()].reset_index(drop=True)
# 1331 - yp1.count()
yp2 = yp1[yp1['y4'].notnull()].reset_index(drop=True)
# 1171 - yp2.count()
yp3 = yp2[yp2['y6'].notnull()].reset_index(drop=True)
# 1109 - yp3.count()
yp4 = yp3[yp3['y3'].notnull()].reset_index(drop=True)
# 1096 - yp4.count()
# yp4[_variables].count()
# # yp4[_variables].to_csv(r'db_ypotato_semmodel_notemptyvariables.csv')
yp4[_variables].to_csv(r'db_ypotato_semmodel_notemptyvariables_vegspots.csv')
```

In [140]:

```
# Configure common options
base = alt.Chart(yp4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(x5):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap + text
```

Out[140]:



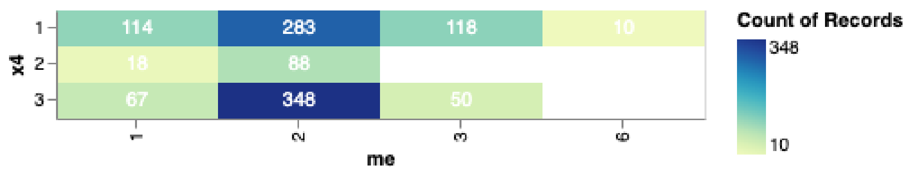
In [141]:

```
# Configure common options
base2 = alt.Chart(yp4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(x5):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap2 + text2

# REvisar el valor de (4,2) Box plot de cada combinacion
```

Out[141]:

In [142]:
```
# Configure common options
base3 = alt.Chart(yp4[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(x5):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text3 = base.mark_text(baseline='middle').encode(
    text='count(x5):Q',
    color=alt.condition(
        alt.datum.x5 > 1000,
        alt.value('black'),
        alt.value('white')
    )
)

# Draw the chart
heatmap3 + text3
```

Out[142]:



In [143]:
```
yp4["x4"].value_counts()
```

Out[143]:
```
1    525
3    465
2    106
5      0
4      0
Name: x4, dtype: int64
```

In [144]:
```
yp5 = yp4.drop(yp4[(yp4["x4"] > 3) | (yp4["me"] == 3) | (yp4["me"] == 4) | (yp4
["me"] == 5)].index).reset_index(drop=True)
```

In [145]:
```
yp5["me"].value_counts()
```

Out[145]:
```
2    719
1    199
6     10
5      0
4      0
3      0
Name: me, dtype: int64
```

In [146]:
```
# yp5
```

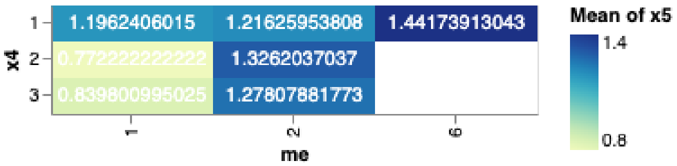
In [147]:
```
yp5[_variables].to_csv(r'db_ypotato_semmodel_notemptyvariables_vegspots_3x3.cs
v')
```

In [148]:
```
# Configure common options
base = alt.Chart(yp5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(x5):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base2 = alt.Chart(yp5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(x5):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(x5):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base3 = alt.Chart(yp5[['me', 'x4', 'x5']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(x5):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(x5):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text3 = base.mark_text(baseline='middle').encode(
    text='count(x5):Q',
    color=alt.condition(
        alt.datum.x5 > 1000,
```

```
            alt.value('black'),
            alt.value('white')
        )
)
```

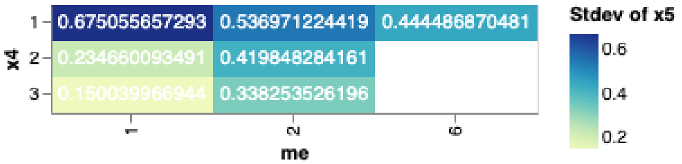
In [149]:
```
heatmap + text
```

Out[149]:



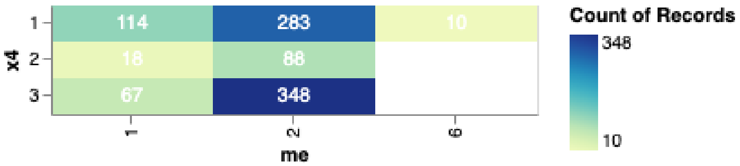
In [150]:
```
heatmap2 + text2
```

Out[150]:



In [151]:
```
heatmap3 + text3
```

Out[151]:



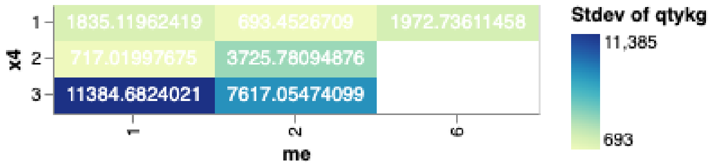
In [152]:
```
# Configure common options
base = alt.Chart(yp5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='mean(qtykg):Q'
)

# Configure heatmap
heatmap = base.mark_rect().encode(
    color=alt.Color('mean(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text = base.mark_text(baseline='middle').encode(
    text='mean(qtykg):Q',
    color=alt.condition(
        alt.datum.num_cars > 100,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base2 = alt.Chart(yp5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='stdev(qtykg):Q'
)

# Configure heatmap
heatmap2 = base.mark_rect().encode(
    color=alt.Color('stdev(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)

# Configure text
text2 = base.mark_text(baseline='middle').encode(
    text='stdev(qtykg):Q',
    color=alt.condition(
        alt.datum.num_cars > 5,
        alt.value('black'),
        alt.value('white')
    )
)

# Configure common options
base3 = alt.Chart(yp5[['me', 'x4', 'qtykg']]).mark_rect().encode(
    x='me:O',
    y='x4:O',
    color='count(qtykg):Q'
)

# Configure heatmap
heatmap3 = base.mark_rect().encode(
    color=alt.Color('count(qtykg):Q',
        legend=alt.Legend(direction='vertical')
    )
)
```

In [153]:
```
heatmap + text
```

Out[153]:

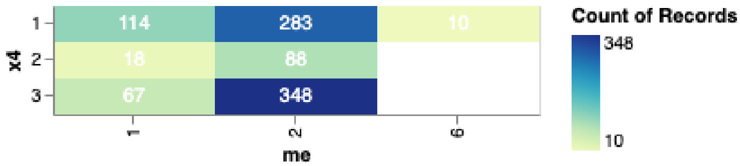
In [154]:
```
heatmap2 + text2
```

Out[154]:

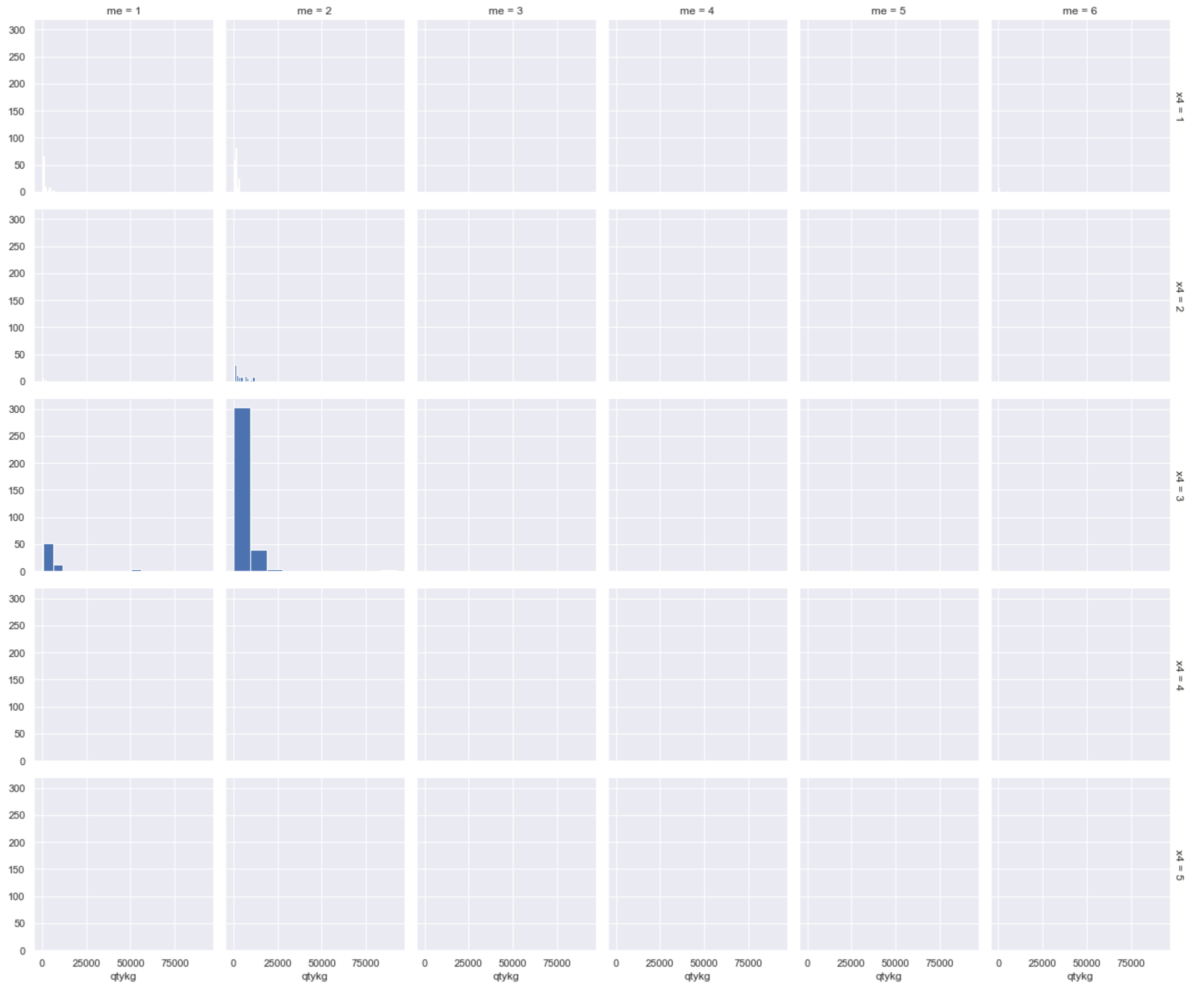


In [155]:
```
heatmap3 + text3
```

Out[155]:



In [156]:
```
grid_yp5 = sns.FacetGrid(yp5, row="x4", col="me", margin_titles=True)
grid_yp5.map(plt.hist, "qtykg");
```



In [131]:
```
# 5866 - df2_avocado.count()
av = df2_avocado
# 5866 - av.count()
av1 = av[av['domain'].notnull()].reset_index(drop=True)
# 4443 - av1.count()
# av2 = av1[av1['x2'].notnull()].reset_index(drop=True)
# # 7428 - av2.count()
# av2[_variables].count()
# av2[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).to_csv(r'db_avoc
ado_semmodel_notemptyvariables.csv')
av1[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).to_csv(r'db_avocad
o_semmodel_notemptyvariables_vegspots.csv')
```

In [133]:
```
# av2[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).count()
av1[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).count()
```

Out[133]:
```
id       4443
y0       4443
y1       4443
y2       4443
y7       4443
y8       4443
y9       4443
y10      4443
y11      4443
y12      4443
y13      4443
y14      4443
y15      4443
y16      4443
y17      4443
y18      4443
y19      4443
x2       4443
x3       4443
x4       4443
x5       4443
x6       4443
x7       4443
dtype: int64
```

In [143]:
```
# 5921 - df2_coffee.count()
co = df2_coffee
# 5921 - co.count()
co1 = co[co['domain'].notnull()].reset_index(drop=True)
# 5733 - co1.count()
co2 = co1[co1['x2'].notnull()].reset_index(drop=True)
5721 - co2.count()
# co2[_variables].count()
# co2[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).to_csv(r'db_coff
ee_semmodel_notemptyvariables.csv')
co2[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).to_csv(r'db_coffee
_semmodel_notemptyvariables_vegspots.csv')
```

In [144]:
```
co2[_variables].drop(["y3", "x1", "y4", "y5", "y6"], axis=1).count()
```

Out[144]:
```
id       5721
y0       5721
y1       5721
y2       5721
y7       5721
y8       5721
y9       5721
y10      5721
y11      5721
y12      5721
y13      5721
y14      5721
y15      5721
y16      5721
y17      5721
y18      5721
y19      5721
x2       5721
x3       5721
x4       5721
x5       5721
x6       5721
x7       5721
dtype: int64
```

In [164]:
```
n4[_variables]['y2'].value_counts()
```

Out[164]:
```
200.000000      502
400.000000      286
300.000000      237
120.000000      230
320.000000      217
133.333333      207
160.000000      179
100.000000      165
150.000000      151
240.000000      144
166.666667      141
140.000000      138
266.666667      125
250.000000      112
800.000000      109
600.000000      104
700.000000       97
156.250000       95
106.666667       94
180.000000       89
125.000000       72
500.000000       71
280.000000       69
26.250000        64
175.000000       61
33.000000        48
80.000000        48
116.533139       42
260.000000       40
30.000000        38
                ...
440.000000        2
35000.000000      2
168.000000        2
410.526316        2
916.666667        2
7647.058824       2
108.108108        2
644.230769        2
585.714286        2
148.148148        2
322.580645        2
370.000000        2
180.018002        2
1560.000000       2
145.000000        2
226.666667        2
177.935943        2
114.285714        2
30000.000000      1
35353.535354      1
88.000000         1
5.000000          1
217.391304        1
1333.333333       1
474.666667        1
650.000000        1
4687.500000       1
235.294118        1
252.000000        1
206.176471        1
Name: y2, Length: 389, dtype: int64
```

In [158]:
```
max(n4[_variables]['y2'])
```

Out[158]:
```
46153.84615384615
```

In [ ]: