```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 3. Models"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 114,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import cross_val_score, KFold,
StratifiedKFold,ShuffleSplit, cross_validate\n",
    "from sklearn.datasets import make_blobs\n",
    "from sklearn.ensemble import RandomForestClassifier\n",
    "from sklearn.ensemble import ExtraTreesClassifier\n",
    "from sklearn.tree import DecisionTreeClassifier\n",
    "from sklearn.model_selection import train_test_split \n",
    "import pandas as pd\n",
    "from babel.numbers import format_currency\n",
    "from sklearn.ensemble import AdaBoostClassifier\n",
    "from sklearn.discriminant_analysis import
QuadraticDiscriminantAnalysis,LinearDiscriminantAnalysis\n",
    "import numpy as np\n",
    "import seaborn as sns\n",
    "from matplotlib import pyplot as plt\n",
    "from tqdm import tqdm\n",
    "from sklearn.neighbors import KNeighborsClassifier\n",
    "from sklearn.neighbors.nearest_centroid import NearestCentroid\n",
    "from sklearn.semi_supervised import LabelPropagation, LabelSpreading\n",
    "from sklearn.svm import SVC  \n",
    "from xgboost import XGBClassifier\n",
    "import random\n",
    "from sklearn.metrics import classification_report\n",
    "from sklearn.metrics import confusion_matrix, accuracy_score\n",
    "from sklearn.linear_model import LogisticRegression, SGDClassifier,
RidgeClassifier\n",
    "from sklearn.neural_network import MLPClassifier\n",
    "from sklearn.neighbors import RadiusNeighborsClassifier\n",
    "from sklearn.ensemble import GradientBoostingClassifier\n",
    "from sklearn.linear_model import PassiveAggressiveClassifier\n",
    "from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB\n",
    "from sklearn.svm import NuSVC\n",
    "from sklearn.metrics import roc_curve, precision_recall_curve, auc,
make_scorer, recall_score, accuracy_score, precision_score, confusion_matrix\n",
    "import warnings\n",
    "from imblearn.over_sampling import SMOTE\n",
    "from sklearn.preprocessing import MinMaxScaler\n",
    "warnings.filterwarnings('ignore')\n",
    "\n",
    "from sklearn.model_selection import GridSearchCV\n",
    "from sklearn.metrics import make_scorer, average_precision_score,
precision_recall_curve\n",
    "from mlxtend.classifier import EnsembleVoteClassifier\n",
    "\n",
    "seed = 0\n",
    "\n",
    "def confusion_mx(y, y_predict, labels = [0,1]):\n",
    "    cm = confusion_matrix(y, predictions, labels)\n",
    "    plt.figure(figsize=(4,3))\n",
```

```
    "    ax= plt.subplot()\n",
    "    sns.heatmap(cm, annot=True, ax = ax,fmt='g')\n",
    "    # labels, title and ticks\n",
    "    ax.set_xlabel('Predicted labels')\n",
    "    ax.set_ylabel('True labels')\n",
    "    ax.set_title('Confusion Matrix')\n",
    "    ax.xaxis.set_ticklabels([0, 1])\n",
    "    ax.yaxis.set_ticklabels([0, 1])\n",
    "    plt.show()\n",
    "def predict_metrics(y, predictions):\n",
    "    print(classification_report(y,predictions)) \n",
    "    print('>>> Accuracy:',np.round(accuracy_score(predictions,
y),3),'<<<')\n",
    "    confusion_mx(y,predictions)\n",
    "\n",
    "'''\n",
    "def get_profit(clf, x_test):\n",
    "    # Define thresholds and profit/cost per answer\n",
    "    thresholds, c = np.arange(0, 1, 0.025), 1\n",
    "    revenue_answer, expense_answer = 11, 3\n",
    "    \n",
    "    # predict with model\n",
    "    y_prob = clf.predict_proba(x_test)[:,c]\n",
    "    revenues = []\n",
    "    dict_thresholds = {}\n",
    "    i=0\n",
    "    for t in thresholds:\n",
    "        y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "        cm = confusion_matrix(y_test, y_pred)\n",
    "        revenue = cm[1][1] * revenue_answer\n",
    "        expenses = cm[:, 1].sum() * expense_answer\n",
    "        net_revenue = revenue - expenses\n",
    "        revenues.append(net_revenue)\n",
    "    \n",
    "    # plot  \n",
    "    plt.figure(figsize=(10,7))\n",
    "    plt.plot(thresholds, revenues, marker='.', label =
clf.__class__.__name__)\n",
    "    plt.plot([0, 1], [0, 0], 'k--')\n",
    "    plt.xlabel('\\\"Probability\\\" threshold')\n",
    "    plt.ylabel(\"Net Revenue\")\n",
    "    plt.title('Profit curves on unseen data')\n",
    "    plt.legend(loc='best', title=\"Models\")\n",
    "    plt.show()\n",
    "\n",
    "    t = thresholds[np.argmax(revenues)]\n",
    "    profit_dict = dict(zip(thresholds,revenues))\n",
    "    max_profit = profit_dict[t]\n",
    "\n",
    "
print(\"-----------------------------------------------------------------\")\
n",
    "    print(\"The classification threshold wich maximizes the profit:
{:.2%}\".format(t))\n",
    "    print(\"Profit:\",format_currency(max_profit, 'EUR', locale='de_DE'))\
n",
    "
print(\"-----------------------------------------------------------------\")\
n",
    "'''\n",
    "def scale_data(x_train, x_test, scaler = MinMaxScaler()):\n",
    "    x_train_scaled = scaler.fit_transform(x_train)\n",
    "    x_train_scaled = pd.DataFrame(x_train_scaled, columns=x_train.columns,
index=x_train.index)\n",
```

```
    "    x_test_scaled = scaler.transform(x_test)\n",
    "    x_test_scaled = pd.DataFrame(x_test_scaled, columns=x_test.columns,
index=x_test.index)\n",
    "    return x_train_scaled, x_test_scaled\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 115,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>AcceptedCmp1_01T</th>\n",
       "      <th>R_NumStorePurchases_01T</th>\n",
       "      <th>R_Mnt_NumWebPurchases_01T</th>\n",
       "      <th>R_Mnt_NumStorePurchases_01T</th>\n",
       "      <th>R_Mnt_NumCatalogPurchases_01T</th>\n",
       "      <th>R_MntWines_01T</th>\n",
       "      <th>R_MntMeatProducts_01T</th>\n",
       "      <th>R_MntFishProducts_01T</th>\n",
       "      <th>R_DealFrq_01T</th>\n",
       "      <th>RFM_01T</th>\n",
       "      <th>...</th>\n",
       "      <th>AcceptedCmp2_01T</th>\n",
       "      <th>Marital_Status_01T</th>\n",
       "      <th>AcceptedTot_01T</th>\n",
       "      <th>AcceptedCmp3_01T</th>\n",
       "      <th>Days_as_cust_01T</th>\n",
       "      <th>AcceptedCmp4_01T</th>\n",
       "      <th>Income_01T</th>\n",
       "      <th>AcceptedCmp5_01T</th>\n",
       "      <th>Education_01T</th>\n",
       "      <th>Response</th>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>ID</th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>
```

```
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>67</th>\n",
       "      <td>0</td>\n",
       "      <td>0.816497</td>\n",
       "      <td>0.704801</td>\n",
       "      <td>0.724756</td>\n",
       "      <td>0.0</td>\n",
       "      <td>0.588731</td>\n",
       "      <td>0.481838</td>\n",
       "      <td>0.0</td>\n",
       "      <td>0.801784</td>\n",
       "      <td>0.1574</td>\n",
       "      <td>...</td>\n",
       "      <td>0</td>\n",
       "      <td>1</td>\n",
       "      <td>0.0</td>\n",
       "      <td>0</td>\n",
       "      <td>0.166024</td>\n",
       "      <td>0</td>\n",
       "      <td>0.028395</td>\n",
       "      <td>0</td>\n",
       "      <td>1</td>\n",
       "      <td>0</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "<p>1 rows Ã 27 columns</p>\n",
       "</div>"
      ],
      "text/plain": [
       "    AcceptedCmp1_01T  R_NumStorePurchases_01T  R_Mnt_NumWebPurchases_01T  \\\n",
       "ID                                                                         \n",
       "67                 0                 0.816497                   0.704801   \n",
       "\n",
       "    R_Mnt_NumStorePurchases_01T  R_Mnt_NumCatalogPurchases_01T  \\\n",
       "ID                                                               \n",
       "67                     0.724756                            0.0   \n",
       "\n",
       "    R_MntWines_01T  R_MntMeatProducts_01T  R_MntFishProducts_01T  \\\n",
       "ID                                                                 \n",
       "67        0.588731               0.481838                    0.0   \n",
       "\n",
       "    R_DealFrq_01T  RFM_01T    ...     AcceptedCmp2_01T  Marital_Status_01T  \\\n",
       "ID                            ...
```

```
\n",
       "67       0.801784   0.1574    ...                    0
1   \n",
       "\n",
       "    AcceptedTot_01T  AcceptedCmp3_01T  Days_as_cust_01T
AcceptedCmp4_01T  \\\n",
       "ID
\n",
       "67              0.0                 0          0.166024
0   \n",
       "\n",
       "    Income_01T  AcceptedCmp5_01T  Education_01T  Response  \n",
       "ID                                                       \n",
       "67    0.028395                 0              1         0  \n",
       "\n",
       "[1 rows x 27 columns]"
      ]
     },
     "execution_count": 115,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "train = pd.read_excel('df_02.xlsx', index_col=0)\n",
    "test = pd.read_excel('df_test_02.xlsx', index_col = 0)\n",
    "train.head(1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 116,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>AcceptedCmp1_01T</th>\n",
       "      <th>R_NumStorePurchases_01T</th>\n",
       "      <th>R_Mnt_NumWebPurchases_01T</th>\n",
       "      <th>R_Mnt_NumStorePurchases_01T</th>\n",
       "      <th>R_Mnt_NumCatalogPurchases_01T</th>\n",
       "      <th>R_MntWines_01T</th>\n",
       "      <th>R_MntMeatProducts_01T</th>\n",
       "      <th>R_MntFishProducts_01T</th>\n",
       "      <th>R_DealFrq_01T</th>\n",
```

```
"      <th>RFM_01T</th>\n",
"      <th>...</th>\n",
"      <th>AcceptedCmp2_01T</th>\n",
"      <th>Marital_Status_01T</th>\n",
"      <th>AcceptedTot_01T</th>\n",
"      <th>AcceptedCmp3_01T</th>\n",
"      <th>Days_as_cust_01T</th>\n",
"      <th>AcceptedCmp4_01T</th>\n",
"      <th>Income_01T</th>\n",
"      <th>AcceptedCmp5_01T</th>\n",
"      <th>Education_01T</th>\n",
"      <th>Response</th>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>ID</th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"      <th></th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>2895</th>\n",
"      <td>0</td>\n",
"      <td>0.745356</td>\n",
"      <td>0.657711</td>\n",
"      <td>0.663696</td>\n",
"      <td>0.460157</td>\n",
"      <td>0.306529</td>\n",
"      <td>0.608015</td>\n",
"      <td>0.695128</td>\n",
"      <td>0.534522</td>\n",
"      <td>0.836929</td>\n",
"      <td>...</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0.0</td>\n",
"      <td>0</td>\n",
"      <td>0.127624</td>\n",
"      <td>0</td>\n",
"      <td>0.039571</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"  </tbody>\n",
```

```
       "</table>\n",
       "<p>1 rows Ã 27 columns</p>\n",
       "</div>"
      ],
      "text/plain": [
       "      AcceptedCmp1_01T  R_NumStorePurchases_01T  \\\n",
       "ID                                                \n",
       "2895                 0                 0.745356   \n",
       "\n",
       "      R_Mnt_NumStorePurchases_01T  R_Mnt_NumCatalogPurchases_01T  \\\n",
       "ID                                                                \n",
       "2895                     0.663696                       0.460157   \n",
       "\n",
       "      R_MntWines_01T  R_MntMeatProducts_01T  R_MntFishProducts_01T  \\\n",
       "ID                                                                   \n",
       "2895        0.306529               0.608015               0.695128   \n",
       "\n",
       "      R_DealFrq_01T   RFM_01T    ...     AcceptedCmp2_01T  \\\n",
       "ID                               ...                       \n",
       "2895       0.534522  0.836929    ...                    0   \n",
       "\n",
       "      AcceptedTot_01T  AcceptedCmp3_01T  Days_as_cust_01T  \\\n",
       "ID                                                          \n",
       "2895              0.0                 0          0.127624   \n",
       "\n",
       "      Income_01T  AcceptedCmp5_01T  Education_01T  Response  \n",
       "ID                                                           \n",
       "2895    0.039571                 0              0         0  \n",
       "\n",
       "[1 rows x 27 columns]"
      ]
     },
     "execution_count": 116,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "test.head(1)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Nr. features to keep"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 117,
   "metadata": {},
```

   "outputs": [],
   "source": [
    "def top_features(df, top):\n",
    "   tmp = df.iloc[:,:top]\n",
    "   tmp = pd.concat([tmp,df.Response],axis=1)\n",
    "   return tmp"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 118,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Original Data Size:\t2240\n",
      "Current Data Size:\t1732\n",
      "Nro. of columns:\t27\n",
      "Deleted 22.68% of Data\n"
     ]
    }
   ],
   "source": [
    "print('Original Data Size:\\t2240\\nCurrent Data Size:\\t{}\\nNro. of columns:\\t{}\\nDeleted {:.2%} of Data'.format(train.shape[0],train.shape[1],1-train.shape[0]/2240))\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# To balance or Not"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 119,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAf4AAADTCAYAAABgKnF5AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpYi5vcmcvqOYd8AAAEpxJREFUeJzt3X2wXHddx/
H3h8YWUbEtvcWSpKTaiFbkoVxLFWUYgiVFJB2GSqvYWDoTH6qoKFrUoQoywFBqlgnkNAHsaXiQ6PW1kxAO0pbGtrSR6F3CtNcUsit6QNaHgx8/WN/sUtyk25v7u7m3vN+zezsOd/
z2z3f25n0s789Z89JVSFJkrrhSeNuQJIkjY7BL0lShxj8kiR1iMEvSVKHGPySJHWIwS9JUocY/JIkdYjBL0lShxxj8kiR1yJJxNzAMq1evrmuuuuuWbcbUiSNEoZZNCinPE/
8MAD425BkqSD0qIMfkmSNDuDX5KkDjH4JUnqEINfkqQOMfglSeqQRflzvmF6wZsuHXcL0gH75sEGb8kSR1i8EuS1CEGvyRJHWLwS5LUIUML/
iQbk+xIcscs234ZSU5qq0nyYVJppLcluTEvrFrk9zTHmuH1a8kSV0wzBn/xcDAqPYYtJlgM/
DtzXVz4VWNke64CL2tgjgfOBFwInAecnOWKIPUuStKgNLfir6r6r6jpg5b3gP8FlB9tXApdvzA3B4kmOA
lwObq2pnVd0PGdbq2pnVT0IbGaWDxxOSJGkwIz3Gn+Ra4FvAl/wRwX2ZR7C5AD0WGh+AyhOkCqV3dbNXyyewZyX0AUMfKLQNRTHV6jQG
HGPySJHWIwS9JUocY/JIkdYjBL0lShxj8kiR1iMEvSVKHGPySJHWIwS9JUocY/
JIkdYjBL0lShwzz7nwbk+xIckdf7d1J/

jPJbUn+LsnhfdvenGQqyaeTvLyvvrrVppKcN6x+JUnqgmHO+C8GVu9R2ww8u6qeA3wGeDNAkhOAM4Afa
K/
58ySHJDkEeB9wKnACcGYbK0mS5mBowV9V1wE796j9S1Xtaqs3AMva8hrgiqr6alV9FpgCTmqPqaq6t6q
+BlzRxkqSpDkY5zH+1wP/3JaXAtv6tk232r7qe0myLsnWJFtnZmaG0K4kSQvfWII/
ye8Cu4AP7S7NMqz2U9+7WLW+qiaranJiYmJ+GpUkaZFZMuodJlkLvBJYVVW7Q3waWN43bBmwvS3vqy5J
kp6gkc74k6wGfht4VVU92rdpE3BGksOSHAesBD4B3ASsTHJckkPpnQC4aZQ9S5K0mAxtxp/
kcuAlwFFJpoHz6Z3FfxiwOQnADVX1C1V1Z5IrgbvoHQI4t6q+3t7nl4FrgUOAjVV157B6liRpsRta8Ff
VmbOUN+xn/NuBt89qq4eh5bkySps7xynyRJHWLwS5LUIQa/
JEkdYvBLktQhBr8kSR1i8EuS1CEGvyRJHWLwS5LUIQa/JEkdYvBLktQhBr8kSR1i8EuS1CFDC/
4kG5PsSHJHX+3IJuT3NOej2j1JLkwyVSS25Kc2PeatW38PUnWDqtfSZK6YJgz/
ouB1XvUzgO2VNVKYEtbBzgVWNke64CLoPdBgd7tfF8InAScv/vDgiRJeuKGFvxVdR2wc4/
yGuCStnwJcFpf/dLquQE4PMkxwBaz94cJSZI0oFEf4396Vd0P0J6PbvwlwLa+cdOttq/
6XpKsS7I1ydaZmZl5b1ySpMXgYDm5L7PUaj/1vYtV66tqsqomJyYm5rU5SZIWi1EH/xfbV/
i05x2tPg0s7xu3DNi+n7okSZqDUQf/JmD3Dmfl5rgav66me1s/
tPBh5uhwKuBU5JckQ7qe+UVpMkSXOwZFhvnORy4CXAUUmm6Z2d/
07gyiTnAPcBp7fhVwOvAKaAR4GzAapqZ5K3ATe1cW+tqj1PGJQkSQMaWvBX1Zn72LRqlrEFnLuP99kIb
JzH1iRJ6qyD5eQ+SZI0Aga/
JEkdYvBLktQhBr8kSR1i8EuS1CEGvyRJHWLwS5LUIQMFf5Itg9QkSdLBbb8X8EnyZOAp9K6lCn
qcAzhtybJEmaZ4935b6fB36NXsh/kseC/
xHgfUPsS5IkDcF+g7+q3gu8N8mvVNWfjqgnSZI0JANdq7+q/
jTJjwAr+l9TVZcOqS9JkjQEg57cdxlwAfCjwA+1x+Rcd5rk15PcmeSOJJcneXKS45LcmOSeJB9Ocmgbe
1hbn2rbV8x1v5Ikdd2gd+ebBE5od9E7IEmWAm9o7/
flJFcCZ9C7Le97quqKJH8BnANc1J4frKrjk5wBvAt47YH2IUlSFw36O/47gO+ax/
0uAb41yRJ6vxq4H3gp8JG2/RLgtLa8pq3Ttq9KEiRJ0hM26Iz/
KOCuJJ8Avrq7WFWveqI7r7kAuA/
4MvAv9H4x8FBV7WrDpoGlbXkpsK29dleSh4GnAQ880X1LktR1gwb/78/
XDtv1ANYAxwEPAX8NnDrL0N2HFWab3e91yGGSdcA6gGOPPXZeepUkabZrJ6MkfTgalbWZFrzzx1xp9LJS
wRyxRJ0PBskrvtArQ4CwH4Yl3lF8yKRmFMM+rUNLiL6UJkKLYNdZX7ThIixSFrWGoP+Gtj21Ebf7+
cbC1TY+B3AkV/5IXZyvsgD5cHuh3GERMC3O2gswTdJHN4jnlFGVi6d7fX0G/dMhiNvZLzmZmZN1LHP0B1
9o7Z2GrqkQ2UXGM/KVtWyHH0PUMHBSDhI26pjbH/
KxzbLDOQGb5N3IF8lBwfYWRG9rQhfdpS6xqMnPc15uhmY2FQJ5IE5qgUBgFc7aY6QvYgDOPSNn2oxkLqsa
6+R1NF6XEj5oafVAKA2+d/pIRJPtbmRnkIBBKpJ9kg2v2dLUwMExSRQC4R85BpDhAd0vp+8ZCqxS5SIa8c
qZ1c/x0cw7KB6VzjlkLE2TEOPpzGEhjt9nAfuhUw/BSavhjnlvSg2gD3EUg9C6GVjt6MMD0rP/
8CIHQhfAQ/mXjfGo4j6d6XgaAm6KAfqW8H/
S9Ij99f4DEmZcOU7AQBa3uFbwKpvyQuKpjkcnqBZxWWVp19hFFNuqKIcTRqDIrrmbjyI7O9wTY/

```
IMk6YF1b/e8knx5Rb5p/RwEPjLuJxSwXrB13Czo4+W9vFM7PsN75mqpa/XiDFkrwz/
Zfqb5ppWo9sH407WiYkmytqslx9yF1jf/2umGhfNU/
DSzvW18GbB9TL5IkLVgLJfhvAlYmOS7JocAZwKYx9yRJ0oKzIL7qr6pdSX4ZuBY4BNhYVXeOuS0Nj4ds
pPHw314HpKoef5QkSVoUFspX/ZIkaR4Y/JIkdYjBr4OKl2aWRi/JxiQ7ktwx7l40fAa/
Dhpemlkam4uBx73wixYHg18HEy/
NLI1BVV0H7Bx3HxoNg18Hk9kuzbx0TL1I0qJk8Otg8riXZpYkHRiDXwcTL80sSUNm8Otg4qWZJWnIDH4
dNKpqF7D70sx3A1d6aWZp+JJcDlwPPCvJdJJzxt2ThsdL9kqS1CHO+CVJ6hCDX5KkDjH4JUnqEINfkqQ
OMfglSeqQJeNuQNLoJfk6cDu9/wd8FvjZqnpovF1JGgVn/FI3fbmqnldVz6Z3c5Zzx92QpNEw+CVdT9/
NkJK8KclNSW5L8get9m1J/
inJp5LckeS1rf65JO9K8on2OL7Vn5lkS3uPLUmObfWLk1yY5ONJ7k3ymlY/
Jsl1SW5t7/9jrX5KkuuT3Jzkr5N8+4j/
20iLjsEvdViSQ4BVtEsjJzkFWEnvFsnPA16Q5MX07tW+vaqe274luKbvbR6pqpOAPwP+pNX+DLi0qp4D
fAi4sG/8McCPAq8E3tlqPw1cW1XPA54LuRkKOD3gJdV1YnAVuCN8/
n3S11k8Evd9K1JbgX+CzgS2Nzqp7THLcDNwPfR+yBwO/CyNrv/sap6uO+9Lu97/uG2/MPAX7Xly+gF/
W5/X1XfqKq7gKe32k3A2Ul+H/jBqvoScDJwAvAfrde1wDMP+C+XOs7gl7rpy212/UzgUB47xh/
gHe34//Oq6viq2lBVnwFeQO8DwDuSvKXWofy+yj/
tW+5QBU1XXAi4HPA5clOatt29zXywlV5TXkpQNk8Esd1mbubwB+M8m30LtB0ut3H0tPsjTJ0UmeATxaV
X8JXACc2Pc2r+17vr4tf5ze3RUBfgb49/31keSZwI6qej+wob+wob3/
DcCL+s4beEqS7z2gP1iSP+eTuq6qbknyKeCMqKeMqrosyfcD0+c4x+AX+97AX+5XisCQ30ptInNlqbwA2JnkA
gdcDxwLG/cc/ThtvAR4U5L/L/
bfs8q6pmkvwccHmSw9q43wM+c0B/
sNRx3p1P0pwl+RwwWVUPjLsXSYPxq35JkjrEGb8kSR3CV+SpA4x+CVJ2hCDX5KkDjH4JUnaEFUNO+0PjqQO
+T++9gju8LwGTQAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<Figure size 576x216 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.figure(figsize=(8,3))\n",
    "plt.gca().spines['right'].set_visible(False)\n",
    "plt.gca().spines['top'].set_visible(False)\n",
    "ax = sns.countplot(x=\"Response\", data=train)\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 120,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Sets\n",
    "x_train = train.drop(columns='Response')\n",
    "y_train = train.Response\n",
    "\n",
    "x_test = test.drop(columns='Response')\n",
    "y_test = test.Response"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 121,
   "metadata": {},
   "outputs": [],
   "source": [
    "# scale Train and Test\n",
    "x_train, x_test = scale_data(x_train, x_test, scaler = MinMaxScaler())"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Models\n",
    "---"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**1. Standard**\n",
    "\n",
    "* Logistic\n",
    "* Multinomial NB\n",
    "* Bernoulli NB\n",
    "* GaussianNB\n",
    "* Passive Aggressive Classifier\n",
    "* Label Propagation\n",
    "* Label Spreading\n",
    "* NUSVC\n",
    "* SVC\n",
    "* Random Forest Classifier\n",
    "* SGD Classifier\n",
    "* Decision Tree Classifier\n",
    "* XGB Classifier\n",
    "* Gradient Boosting Classifier\n",
    "* RidgeClassifier\n",
    "* KNeighborsClassifier\n",
    "* QuadraticDiscriminantAnalysis\n",
    "* LinearDiscriminantAnalysis\n",
    "* Nearest Centroid\n",
    "\n",
    "**2. Extras**\n",
    "\n",
    "* Boosting\n",
    "* Voting\n",
    "\n",
    "**3. Neural Networks**\n",
    "\n",
    "* MLP\n",
    "* Keras Deep Learning\n",
    "\n",
    "**4. Hyperparameter Tunning**\n",
    "* Grid Search\n",
    "* Random Search\n",
    "* Keras with GA\n",
    "\n",
    "**5. Try with decomposed features**\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "#
----------------------------------------------------------------------------
----------"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {},
```

```
   "outputs": [],
   "source": [
    "# models\n",
    "n_jobs = -1\n",
    "models = [LogisticRegression(random_state = seed, n_jobs = n_jobs),\n",
    "          MLPClassifier(random_state=seed)\n",
    "         ]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "metadata": {},
   "outputs": [],
   "source": [
    "# grid\n",
    "\n",
    "def sampling(data, column, seed):\n",
    "    #random.seed(seed)\n",
    "    \n",
    "    num_of_1=len(data.loc[data[column]==1])\n",
    "    idxs=random.sample(set(data.loc[data[column]==0].index), num_of_1)\n",
    "    new_data_0 = data.loc[data.index.isin(idxs)]\n",
    "    \n",
    "    sample = pd.concat((new_data_0, data.loc[data[column]==1]), axis=0)\n",
    "    \n",
    "    y=sample[column]\n",
    "    x=sample.drop(columns=column)\n",
    "    \n",
    "    return( x , y)\n",
    "\n",
    "\n",
    "x_t, y_t = sampling(train, 'Response', seed = 0)\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "\r",
      "  0%|          | 0/2 [00:00<?, ?it/s]"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 5 folds for each of 2 candidates, totalling 10 fits\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent
workers.\n",
      "[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    3.6s finished\
n",
```

```
      " 50%|âââââ     | 1/2 [00:03<00:03,  3.66s/it][Parallel(n_jobs=-1)]: Using
backend LokyBackend with 4 concurrent workers.\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 5 folds for each of 2 candidates, totalling 10 fits\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    1.2s finished\
n",
      "100%|ââââââââââ| 2/2 [00:05<00:00,  2.99s/it]\n"
     ]
    }
   ],
   "source": [
    "grid_dict={\n",
    "    'LogisticRegression':{'C':[10, 50]},\n",
    "    'MLPClassifier':{'hidden_layer_sizes': [(1),(2)]}\n",
    "    }\n",
    "\n",
    "# Add MLP to best models\n",
    "n_splits = 5\n",
    "CV = StratifiedKFold(n_splits=n_splits,
random_state=seed)#KFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = ['accuracy','precision','recall','f1']\n",
    "\n",
    "grid_search_dict={}\n",
    "for model in tqdm(models):\n",
    "    model_gs = GridSearchCV(model,grid_dict[model.__class__.__name__],cv =
CV,n_jobs=-1,scoring='f1',verbose=1)\n",
    "    model_gs.fit(x_t,y_t)\n",
    "    grid_search_dict[model.__class__.__name__]=model_gs.best_estimator_"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 27,
   "metadata": {},
   "outputs": [],
   "source": [
    "from mlxtend.classifier import EnsembleVoteClassifier\n",
    "from imblearn.pipeline import make_pipeline, Pipeline\n",
    "smote_ = SMOTE(random_state=seed)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 33,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "\r",
      "  0%|          | 0/6 [00:00<?, ?it/s]"
```

```
 ]
},
{
 "name": "stdout",
 "output_type": "stream",
 "text": [
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n"
 ]
},
{
 "name": "stderr",
 "output_type": "stream",
 "text": [
  "\r",
  " 17%|ââ        | 1/6 [00:12<01:00, 12.09s/it]"
 ]
},
{
 "name": "stdout",
 "output_type": "stream",
 "text": [
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n",
  "Fitting 2 classifiers...\n",
  "Fitting clf1: logisticregression (1/2)\n",
  "Fitting clf2: mlpclassifier (2/2)\n"
 ]
},
{
 "name": "stderr",
 "output_type": "stream",
 "text": [
  "\r",
  " 33%|âââ      | 2/6 [00:26<00:51, 12.86s/it]"
 ]
},
{
 "name": "stdout",
 "output_type": "stream",
 "text": [
```

```
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n"
  ]
 },
 {
  "name": "stderr",
  "output_type": "stream",
  "text": [
   "\r",
   " 50%|âââââ     | 3/6 [00:38<00:38, 12.67s/it]"
  ]
 },
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n"
  ]
 },
 {
  "name": "stderr",
  "output_type": "stream",
  "text": [
   "\r",
   " 67%|âââââââ   | 4/6 [00:50<00:24, 12.37s/it]"
  ]
 },
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
   "Fitting 2 classifiers...\n",
   "Fitting clf1: logisticregression (1/2)\n",
   "Fitting clf2: mlpclassifier (2/2)\n",
```

```
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "\r",
      " 83%|âââââââââ | 5/6 [01:02<00:12, 12.16s/it]"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "100%|âââââââââ| 6/6 [01:13<00:00, 12.01s/it]\n"
     ]
    }
   ],
   "source": [
    "scoring = ['accuracy','precision','recall','f1','profit', 'profit_norm']\
n",
    "\n",
    "vote_df = pd.DataFrame()\n",
    "\n",
    "x = x_train\n",
    "y = y_train\n",
    "\n",
    "for score in tqdm(scoring):\n",
    "    \n",
    "    v_claf = EnsembleVoteClassifier(clfs=models, voting='soft', verbose=1)\
n",
    "    pipeline = make_pipeline(smote_, v_claf)\n",
    "    \n",
    "    if score != 'profit' and score != 'profit_norm':\n",
```

```
    "        entries = []\n",
    "        model_name = v_claf.__class__.__name__\n",
    "        accuracies = cross_val_score(pipeline, x, y, scoring= score,
cv=CV)\n",
    "        for fold_idx, accuracy in enumerate(accuracies):\n",
    "            entries.append((model_name, fold_idx, accuracy))\n",
    "        cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx',
score])\n",
    "        vote_df[score] = cv_df.groupby('model_name')[score].agg('mean')\n",
    "    \n",
    "    else:\n",
    "        profits = []\n",
    "        profits_norm = []\n",
    "        revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "        revenues = []\n",
    "        revenues_norm = []\n",
    "\n",
    "        for fold_train, fold_valid in CV.split(x,y):\n",
    "            pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
    "            y_prob = v_claf.predict_proba(x.iloc[fold_valid])[:,1]\n",
    "            t = 0.5\n",
    "            y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "            cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
    "            revenue = cm[1][1] * revenue_answer\n",
    "            expenses = cm[:, 1].sum() * expense_answer\n",
    "            net_revenue = revenue - expenses\n",
    "            r_real = np.sum(y.iloc[fold_valid].values)*8\n",
    "\n",
    "            revenues.append(net_revenue)\n",
    "            revenues_norm.append(net_revenue/r_real)\n",
    "        profits.append(np.average(revenues))\n",
    "        profits_norm.append(np.average(revenues_norm))\n",
    "        if score =='profit':\n",
    "            vote_df[score] = profits\n",
    "        else:\n",
    "            vote_df[score] = profits_norm"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 34,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
```

```
       "      <th></th>\n",
       "      <th>accuracy</th>\n",
       "      <th>precision</th>\n",
       "      <th>recall</th>\n",
       "      <th>f1</th>\n",
       "      <th>profit</th>\n",
       "      <th>profit_norm</th>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>model_name</th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>EnsembleVoteClassifier</th>\n",
       "      <td>0.870083</td>\n",
       "      <td>0.543933</td>\n",
       "      <td>0.806637</td>\n",
       "      <td>0.648048</td>\n",
       "      <td>227.8</td>\n",
       "      <td>0.552027</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "                          accuracy  precision    recall        f1  profit \\\\\n",
       "model_name                                                                  \n",
       "EnsembleVoteClassifier  0.870083   0.543933  0.806637  0.648048   227.8 \n",
       "\n",
       "                          profit_norm  \n",
       "model_name                             \n",
       "EnsembleVoteClassifier     0.552027  "
      ]
     },
     "execution_count": 34,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [],
   "source": [
    "test_x = test.drop(columns='Response')\n",
    "test_y = test.Response"
   ]
  },
  {
   "cell_type": "code",
```

```
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 43,
   "metadata": {},
   "outputs": [],
   "source": [
    "y_scores = pipeline.predict_proba(test_x)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 110,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n",
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n"
     ]
    }
   ],
   "source": [
    "max_profits = []\n",
    "\n",
    "best_t = []\n",
    "revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "\n",
    "thresholds, c = np.arange(0, 1, 0.025), 1\n",
    "revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "for fold_train, fold_valid in CV.split(x,y):\n",
    "    #\n",
    "    v_claf = EnsembleVoteClassifier(clfs=models, voting='soft', verbose=1)\
n",
    "    pipeline = make_pipeline(smote_, v_claf)\n",
    "    \n",
    "    pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
"\n",
    "    revenues = []\n",
    "    \n",
    "    \n",
    "    y_prob = v_claf.predict_proba(x.iloc[fold_valid])[:,1]\n",
    "    \n",
```

```
    "    for t in thresholds:\n",
    "        y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "        cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
    "        revenue = cm[1][1] * revenue_answer\n",
    "        expenses = cm[:, 1].sum() * expense_answer\n",
    "        net_revenue = revenue - expenses\n",
    "        revenues.append(net_revenue)\n",
    "        \n",
    "        \n",
    "    t = thresholds[np.argmax(revenues)]\n",
    "    profit_dict = dict(zip(thresholds,revenues))\n",
    "    max_profits.append(profit_dict[t])\n",
    "    #max_profits.append(max_profit)\n",
    "    best_t.append(t)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 87,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[(0.625, 289), (0.4, 264), (0.6000000000000001, 186),
(0.5750000000000001, 260), (0.525, 230)]\n",
      "0.545\n",
      "245.8\n"
     ]
    }
   ],
   "source": [
    "print(list(zip(best_t,max_profits)))\n",
    "print(np.mean(best_t))\n",
    "print(np.mean(max_profits))\n",
    "\n",
    "mean_t = np.mean(best_t)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 95,
   "metadata": {},
   "outputs": [],
   "source": [
    "def adjusted_classes(y_scores, t):\n",
    "    \"\"\"\n",
    "    This function adjusts class predictions based on the prediction
threshold (t).\n",
    "    Will only work for binary classification problems.\n",
    "    \"\"\"\n",
    "    return [1 if y >= t else 0 for y in y_scores]\n",
    "\n",
    "def precision_recall_threshold(p, r, thresholds, t=0.5):\n",
    "    \"\"\"\n",
    "    plots the precision recall curve and shows the current value for each\
n",
    "    by identifying the classifier's threshold (t).\n",
    "    \"\"\"\n",
    "    \n",
    "    # generate new class predictions based on the adjusted_classes\n",
    "    # function above and view the resulting confusion matrix.\n",
    "    y_pred_adj = adjusted_classes(y_scores, t)\n",
```

```
    "    print(pd.DataFrame(confusion_matrix(y_test, y_pred_adj),\n",
    "                       columns=['pred_neg', 'pred_pos'], \n",
    "                       index=['neg', 'pos']))\n",
    "    \n",
    "    # plot the curve\n",
    "    plt.figure(figsize=(8,8))\n",
    "    plt.title(\"Precision and Recall curve ^ = current threshold\")\n",
    "    plt.step(r, p, color='b', alpha=0.2,\n",
    "             where='post')\n",
    "    plt.fill_between(r, p, step='post', alpha=0.2,\n",
    "                     color='b')\n",
    "    plt.ylim([0, 1.01]);\n",
    "    plt.xlim([0, 1.01]);\n",
    "    plt.xlabel('Recall');\n",
    "    plt.ylabel('Precision');\n",
    "    \n",
    "    # plot the current threshold on the line\n",
    "    close_default_clf = np.argmin(np.abs(thresholds - t))\n",
    "    plt.plot(r[close_default_clf], p[close_default_clf], '^', c='k',\n",
    "             markersize=15)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 99,
   "metadata": {},
   "outputs": [],
   "source": [
    "# fit dados todos"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 111,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 2 classifiers...\n",
      "Fitting clf1: logisticregression (1/2)\n",
      "Fitting clf2: mlpclassifier (2/2)\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "Pipeline(memory=None,\n",
       "     steps=[('smote', SMOTE(k_neighbors=5, kind='deprecated',
m_neighbors='deprecated', n_jobs=1,\n",
       "   out_step='deprecated', random_state=0, ratio=None,\n",
       "   sampling_strategy='auto', svm_estimator='deprecated')),
('ensemblevoteclassifier',
EnsembleVoteClassifier(clfs=[LogisticRegression(C=1.0,
class_weight...verbose=False, warm_start=False)],\n",
       "            refit=True, verbose=1, voting='soft', weights=None))])"
      ]
     },
     "execution_count": 111,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
```

"source": [
 "pipeline.fit(x,y)"
]
},
{
 "cell_type": "code",
 "execution_count": 112,
 "metadata": {},
 "outputs": [],
 "source": [
 "y_scores = pipeline.predict_proba(test_x)[:, 1]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 113,
 "metadata": {},
 "outputs": [
 {
 "data": {
 "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAmcAAAG5CAYAAADLbpPTAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzt3Xu8XfOd//
HXR4S4lhJtCRItKoggTRO3pkNTUqVaVH7uow1TpkzplI66lNLO0JqLMpSJlkYUbWWj41KKpcW9CpCRVKQ
kpJSLuoSKf3x97HXZOTs7Ziex9vpzX8/HI4+z9Xd+11mftdRJv3+/
aa0VmIkmSpDKs0N0FSJIk6W2GM0mSpIIYziRJkgpiOJMkSSqI4UySJKkghjNJkqqSCGM4kARA1/
xMR8yLi3ojYOSIe7u661HoRcVpEXN7dEXeMeheLiJkRMT8iXo6ip6twtfoybbm4n4FNAv8wcmpn/
l5mbt9vXbsulcL1nRMTYiDizu+uQ3ksMZ9K732czc3fd8YjKvGS/v2uSSltaQMsVkOoeHZZXKVAuObdEKjHJnUp3iBPbhfmvtXPubZ6P0GG/+E9ZVyiVwRsjuNdrcp8Bcjhz6AOf8vBIz1X7vbK1P+OMK4mrwNPk3DhjxxvUKJWY/rXn1EOu1Ia3iGn8sOCKKtsTyczvJb+TC9bhv8/GJNsA9xf73JYbfbnmTBJA+QQlk3TMBT1gvgdJYD9c8qFuw5e/Z4Abmss0b7UQb+Wkf0Tw3Wt9H8W8Ag9w/R4jDXsnrBDpk3+ulxKTxgqkbYqgt+CjX9u3+nCXRnSzJxC8+7+2p3Kx3Rty+7Ib9eHGuTMrUc5Z+EfIE+tQE9uwHj1N4/cTgWE3gQYkOhvpEUj2xdJ4NiCq2ULrrlVOaBdKA/fm8/aWQl3Cz1C5qJqsvgYhBzwdGaRrWa/vE+LE9mt7/u6Z72Pz5/Ku9ZvgMJq3C3+YTRxjYZkbJqL+YVHT6OoXE8CMt3KqhYUMfXTo5tbSQhmZ7yv6Y0R/TAwt0vt6p/YQhX2j8xDTTLxxzFtHb77xB4ZyGJMbcLuKGL9/jtU6NLGo5jtiW3h6p1sZLGqLtWCNzMXgLDy/GK19V2sg+nwn0zAVw9N1lbLuhTOcT/HG/t7gNnSVyVBUDzKl7MO3JJ3ZVzZhZqpHDDxrTuTw2BUPnk+ZbOW6JAsGuYtVEhRnKYxbzHTKT1jmn1QaC/DTaL3mmJNiEhmJZEWN46rHsozqXfmJHzC+WtoHnUXHgXVEnT7cN9gtPd1tedADEQAhR8EK5Oj3jk4zZwXfJwjMsunvTBe4JUZlrOLQ70QojbFz5DFfqNjqIRwj25+0MSDVI/V8Z62+2ZCxDMyJ14hyuNddtbyZTDWOnpfB8EGPEMBAHbvWPY2kpGphL6l4T6pr8VF8WDfuL1TIxJ7/g9KxxCr1zQrsc2V7S4XDh96XPLqTqdutoVWMLq/cKJFNebdVYvcPaeObuDTXDZbnewc/2LExxhQQrCC3b7qdCPm+3yDKs6EYMtSsVR8+5sgiYHy9QvBKJZyt1fdRafB9SeIVW5KRgxe/
u7rwpMEEbRA/s4mm2FXtbSxwAhzwXBRvK6HrTjkmjPGq2rp+iLlohgV8yQVj5vA6Pq6aU3G7ztSfI6gzd/
CpkqLEbxV5MgJY+3+19BNgjtKzS08+YNnM8C5zV6uzOekvAw7D6Wd0wVqCAJr3Y+mlzLgtv8WnKZXe2r3m4e
RUBbYzt7CK2wqn8yVyW0GJXQgq+uLYybtc86o54AxEycBDYGRi37Rl4dm4q2MOKEtT0dPojUOjKuvg1eNVe8V
BDbqfbZl8Zx4IBUTRVb2LEG5pN1ZSu9W5jNxCvX1FWAwqqf3COyqnzkeKJNvmKWdVqxEyu0VwjzX+x7GD+Do
x9SyhpzSGlzqOVjjyiRTRn9AoZ1bmhs+nvTS9PqP7kgHxyJFc5BFyBNaPOVIF1gSUpwbsVxgC9eKSt4Ax1Mo
pA/X9Bm1Q8AzFnjm2Hyv+UVzOmfMJqJh5X1mAOe3ko9wHVOc3aYhhy1sjwOi3jLF3sA1ZGXAZNwt82zl6+VCH
kuWRY9GyOQhYWt1rSD1TVJFFv8E74v8ATHB/Pv86Bh2DJwWgX/fUrqLwDnKn0Pkw4/hhwHTEsQs7zugkfnN8
sY+mT+O+1ZlTA01jWMLVa+bRfTYjZ+MdNZPUyeO3fLgk4lRzHhwXbdYgrodBIp0sH9yNaNUBbQsGk0pV5t3ba
D0xN4f4wEfNuTEyJvx9ZH6nGOqjJVcD69r4Kxm9E7mKkXMIXc8uDvdp52E77aSW3pYjW8XwBZRrc2qP2SM8g
Q2Jr1iZPkqVYyb4cvhbt/3jWuB4A5LtCUx2mTQ5rHQ8Fnux+6WOqAuNu7wtncxPcVBz0VShE8ZYcmdo6PfIdj
4Ybq8ZWBYThiZm2vaWHkDlP0XA0tn41ZzXoRawnGOoRw61j9zqcHHRAJMtnr8Ms5Wxfrj+lHrMA5lbHjsSmuT
Bcb5sr3BGvKhcV/pwMIC5GkRvq+P/e9kCAkgoLYhpKBGzbtIy9pmmjX/
vxcuV20hPKI2XOdPbczlTq7DZ3rshdtE9PHrtTtKZ33X9Ag3mOijQrOAWZk/
vJWHvqgX0I8aNCJcnbRrzNfoBOz+Wm64R/
tOrgnKJGLI7lKbY2aUcxlR01pHNlp5DMfs53UJ6yUUXz9VoLMjzebdcJTbzhzfN9Zx1OmnYAdbO+w5Vzs1Y
QRCYbXt9A4XWfrUSi9cN4KEYj/
Hw3M6Ob3LnN0O2XO+jJ9ycN78+bpt/
AbfMvbHfnS2LGF4MfmzLz4iMZIHgd+wn/
sHUO5WeqHOvn+fUTQKPCdMRe83y9JPIdMHQcI/
JzxGtnoQ7cLdKurF/9iE8RNC/
wH8oAn8WucODRR5hwi/8u3X7Q3ArtNPK4G/ad0GQtJW7/q7FpJ5CfGWmx+RJrMdB/eTYvtEt9/
zhUO0hVtS71DnPRzLYy3l1sEMLjBMFu6mWzBBGTXF3+LHG1ZZkDQsv/
zBY+J1P3hxYqKxHx+CBbBqHEnOEaztaRowzc8Q5ATXVIsIkFBqGZi6k2o/AJjkt0/dBwi1pkcSbOjwmFb8gLJTTFTVG/
NE4NUFrnxLb+1cSxe98/
wfF5uJWEd1zwmsm00I7QV4RhHZ0wO+57CoWZuDFJPG/pqYzs1jgP48BqlyR1dnGqUUUmLpINbhCG8Ekmvvz/
T8DNjY4cu1tr3vBbrEg8fWcJJlkAx51gQ0iatzz8PV0y6p/
uwNFQqCrDszoTlE0bZi7ZC+CdOjr3OG8P0DkW0n2sstI2IjmcyJz1JBobv8hQrtGlDcGh/
GCZRe1XpnU4wqOAN45t/qN/
9sz/5oHh6dbBpB61xRmDzcySOx/bpcbKc8lQnW9l/dHqh9bm2ZI0bcfVf1eIZU7/
e/Yl9cwYE3IwwOdHWG9vaxLqYLeUqWOtkXSrM6Sv1nPUy3jRhf75PtkY3cedRk/
3ObfrHdhDOSiH1gn2+mdKOfX2grvHZdzsrW86mrtZdxGzdmYJzYEk5RfUt3+eAeqz5IS6EGyB/
PQpT6j3GTM7jR6sVWY5vYbpXDWIgK3vw6Qf34RfPYtUmlNt8xGF0T9L+jSMzZFQW5e22Kedl/jfwIyrVuE4f
xiMWjzj7ekSA+Ko1v/
CnH/2ABlnbU3g2ZQbnnIlUl/
h+J1yLlJb/CGVP5tcNPBjUKzf7SYbsdSz7KaHtUV2trkN5TP9uH/5o/wUCv+t+mrjybbdwTyZG/
kEpNxIDSCZmNuoHtWFzsvb6L+C/TZlz91sR2wzLXJdYpMyM/
kJBiQ/5pDxdr6K1kFbXUnHvGvWmmC+f9dlqv8xTnjcpXXWb9YHWE9Xod8dzfh5z3lA5yxtxwf7Pdm/
NgfZfmW9dufc6c/Stc8fkpdUVCVhu4lmUA0E0K8Qrbr4Zw/5P74zdwhO3+t2PCSXgwKEP/
a07xbnhP4qRLK0HPzmX4Mi8fk3O9yR/NJ/1u9h/r6dt4ouoMRo7NuD/
8z7LhRKq5wy20QRVqLFq7HCiv3m2/
YBNfuA1afPxbJwKdy8bSzfQS6wr0sVy14s6MuSSzh+wGHuGqAFmJrjX3mTRZj/
gSPe+uyHm8KvlDeQ7eyY9qjz+F5ue/pvh6ZwNlx0g7ssL7N3fGVoIt3ooI/
wqrp6ndjdwgc4xEHeNCwDf24ciERkr5KnrjFBjnJIozwuR5zxuv8AI2/ReCa/
KLYJ1mXKwZA+Dh7FHoDmJDNq4pm3ZDnn0Nv3SA6hKVIwjG1OKJEV0QL2HsN1+9kUkHFTsMvRbRfNqp3IoHJH
rkVCt37F1wSpJ0RrTj/ZRK2/
4DIWhh+DHQwwAa58RqwUUaOmEIlAWnCT4cKCn/
jJJc5ySDOvc6XHPM/3JybGHCa+TrevMYmTLBtN2p5MsGhN0v30BN0OPL6rzw4DE69Hwgi1qjqmCbg/
IJRnjX53YhpTf/tutsMb/
CQlxuFPvuMgcxYxC1m5FX7dZ+qT+PG+9+V3/GtlbJyyLEK3BX8Az9xFAAV/ia/
hEu+ZNz4KmdwNWIJSxvZpfgKc+uq/lFbsFHzUaRZgjDoLhRqWrUUpiDn/
HENO0dwjH5Vr1VMaY1HgLaH5f1OOOuPC8lRTZ9DMJmT4UIAQMBNdx0Qat3qBoMbQcMZkd4Bvs12fWm9ixP+fs
u9H/
p8b5Z7HGoIUuh5i8KSVRrhmCbq5WYxu1vUGrTSz8sFcgwOcQa2eqx7c8l6lZ0Vgr3rSt7LUgsrUaN9uXQyuvO
9b48GwfZvRZzJ1hI/
u8M7wx7bmdmj4ZkW2fkEwlHu5fnyVRb+AlHxFeWhw0HYxgzbsBmbz1Z4WYDx6Qx0r0qzT4HX0uB9T8KftiZx+
t7TwoSP0iMVYLX6/bxYfzzJXOoLcMr3/vfsKNG32J0mRY7z0/iu17Pp/
fn9hcKB0vbmeTE8yu5KvttBxX0LzKPb9ioT6kpXf0WJxX0mvSSaqBrnGdvzGX8XEvXl6Oe4ObuIqrX9cWXLH
jCFzhLC7YHQMxS/
9HNydnRg82wqEmLW4mqxvyVS5xl+N5mG2jL+gqV4uhwV5GnqgrNzl14ncTJO36vnLXiRdv6thp7K6DvGmBBt
d/N3vcQ3mhBfPd7ciDq8e71q1LXSzCZ7kGm22+Pb4Lf9I66b2S1pIGvdycZJaW2PK04A/
0pGe8HxXR8QY2Vt7l7Wua76k5A/
BBPkk0Qjau5HoufeWsbrgrmRUQnzYWqVgscP4f6GbpFmqZzUDDv37BWTBd8BRB+dgxPJAD+q1dWSTOyxoxE/
7T1GJQa0W3ku0Hw9T/e5PrjrMDn09gL+gLIxvnWx8tXdFcLU/
7ZCUw1lEDnFWkV9UUGhn+H7tz23CVYNVhEJyJ3N2A7yN7H/
VqSLz5BaVk6xq11mlAqoPBU8wThWFgkEQaNvADBIT3qnNb7O9TdbfxxzZeLdGdrmd1JhDwtUrs7Xmu5BMcI
pSVjSvwrZGtd8QsnVVfS0ufGi3Lc9bJfoYrkpQ15J7nU2ZzqmwOi7ou2DHXQZHPk39n4KveuXduNUuRGpbJe
+Z/
MB5exS1Aw7+wUEIpwcr7r65nbRNaV5MqEcdpTUCEEAEM1RSGBdoxoYPIR+IVAJYijhVjKq6Ua5QcjNQvxNL
7Fi7vdewJQhyaG1V4DM8SXjvZ62/hvnP9nFfGtP3KJ3UyMEJH0AHC5CcgBF18g/qn/
AicuYHv3Oo0v4HgPdnWFlu1VM8Mmgnbax97/
IKYE2gPdvu7avVYx5x7YU04XGnfLLnmdVs2a3vQQXC0rCzyv5Z0lRU+dHsb4P9i3lbAtv6tSeEwOY0gF6m0x
a26bnOSTjEOl5vEnr/9lZcnNVvEyXXHPmPeb4GPY72Cjf+Ssdcnp8ARvPDj/
kkfVMKnNIWLnxvtZL8zMOxd8ybmiBaQKngJRg4pDxeZRxtZHn+ioHYK5kkwhnRAW2tjkeM9kaP+nDa5VdXCf
Tny4WnjBZhsMvWTUf2kYT0rWTa2p51n6nT5KNR90D2TF2iTF7ZHKQ2IiqIy0Ww98F/+y63rvxxDsTHc6HNd/
U4qZ9G6/UAOaD3xs0nDW1rvy4rBJZ5q3nh7zFT/
oFw4mWQQFtJvU6RWsNNP8V99+UdHnNFXL20yO0cybl6IU+4E/Ik+M8qYaD8vl38Nh/
BuCM+LjvNc3nn53thTfr/wEVK7QkkZ0KqJbmHnsmY/
CBVOzRQqnasvY3PTVMPE1jmK2fcA2J3/AvWzVvMmrXgAGNhfEXg+Cog8BuSVvRHfYmfgz/
zWQ/J8Xm2q24+GuAwQYPTtAxFjA8IeZ1x5u6jK9c5d5MyTYR+G5Tf+ZEsZ/
tMBo2jsZ6aeLqb4KMj+vkSHkEtsAFfCNNA4nQpEgNtTKKKcnfXdnAkpbnnzTtY+Xrh28y3P2rQ3xFt9MRw8L
tyqpFVFqJDjj14zd27zeGB5jTrvvOPxp5XcXGHV3p0bwJ7EXfYcV+Q0G2hU+aTrgZP+p0npYc6nAJ3ULUJGx
t2WnH+wFvldDH/
sNoD5ePFH4mt49IxbVJwBn9y3dJMZp0zjbU3TdJ9DuHBl8uAVuqnc8TIMEsM8lVxdZjd5WP/
DX5PMz6X/
9OBHOWvuXjMSg7X1/
VRMiE70kC7k8y2c9Wnu+XKgv0qOLk0BIHjtfXgGrOGaqy2hhSwPgr0PRmJaMPnPPn9s2jD5GZ9ZfTtGiE20u
ldqDzLGGUrX8rUdTk0pSSQhRVmEfr3aXEfEqKGvRBSWBGIXKUIEBKYmdbsjhaVmWw4YZdImWIyZkhDKVxqvO
sbJ96w2WDzLm2i2BUJrCWMw52PYVN3s5RTzgHB7rXlzLYedoAsUo/
NYyrjGNBHk6cLLbYwQw7X+OGiozHpvAWxd5aLCdNrk+Mzg8s9LEyFg29vRh9BtmbtPqDbxQcz2CiQK0yMi3zD
9oD90UxPIBVSDywKw+C5HR6vXT1lQkGXB2Mkrok0mnWgC9M2YxBTwSkYJv6zQkQs6tuKyCJDZFV8WgYjOxC
0xcivXsYO8zLaD2W/1Dxsr7TVVZafaTeUq7/
uzdk+w7Pza9j9NPQgFjxcfg6NUw13SIcsnjn7QL1aIUcHbb/
s7u/AAkp5WnaW9mbXXl/YS9ZXxEbfeEbBhPNl+pMU6W+/lxQzoE0OwuNiWVUyIk4/
UT7Px5qnh0ROcQPXPjd23e2/2+NnOtUkH8/8IfHwdIOBA5Zk69tQw4nKctjAJhL5dDK+ktLS0amJqWMdH/
bEjThcVz+HBDyR0xJEq5t6bRHSoatCTMMkdF9LLH/r5bV/EaUKakHFEhbbcfUF+6nOl/
bkOpTHFbtKwj9a8c9lWfn2ukXKFb+AuNc9Oq+gnzbGr6XepWxc+Xrh7bO62KxADYkEAndpR8Tn1a/x0ae/
eCrt8Z5/fEVb+FHt8r6RG0w6KbXGrjc94MDiAASEHeJy/
TqdnG/b8eYJHuhtXzVRqu0NjcE7GQ8V/
gOrcdxKbpTwAHK+/1j2k+u4WWq0RfzAsIbAsAJoB29ItwR2TEpYYeXqgBh1cqZpi4C8V6EkxB2cL4R+u91nxu
e8Y5+YVjUDq+w+tm0rnMx3q2nd7S8i/YSbFBWxMaq6xLecaW1w6k3oZkEmrX6Ay4ZfuqgUfd+5a+vZ0pW8Pc
TYnFrP8D5x8pNQX7t6szbSt8Nz8MfmNLj4BJo1J3iAr2K1gKNobyQtumzhLjuIPbnLPD8RDoSrOsDjDCnq9h
kN8ijnHnFEpCmLvfKhR6Z4hHkEHR9NgMNzkMqMGDW9OhbSL4kI/
4Y37nBHVYfwXXmLqt6A3ZGL1PCR8tOF4S7wzyI5E4SKJMXBNGBKI6Zu9YAfP7h2nYh+bATNS77H4HMvRW4m5R
DAwBhyw0AnoDzR2kjBMDmmTLRKYpMpTR8Dc5HA4ZwgnH6+wkkmbHqnKy4MmJGoTEg58MCX6Qe9pxg4jAKhJC
g80a7S0WeVDdXq9Mz+4t7MFMODrYE/lVJ0pB1DxfpV3CnyS/zH9UaCbHVR9zGPnazFRETQo6MSc79r2dp7D/
/wkNcHKZp8jAZDgDZjzOSjn/TNZ2yCKP1O6bnGjZh1L+j/HaUA/
AXJq+rJHIFvyJCNHA1PgBaWvWBJqNcafFUvnAWw4SmWmFWGTl5vKFfOYyuoQg4mgLAxsp19mYIhHr8F2TVO5Q
jjTLhCwYTQ0T4O7nNzrFgoXVAL+DTFmTxSFe4/g2HJs5Jd8ApsVQcnXh7TJFBa/
sGz+S2ZTz+7iZMzeYttmKJ2V0K0fTN1EGdpCZkIsQiINgTjN0ZUD2VWDBzN9HCVNbywkxtIYHOxZuKszsXnOd
xRaFFQdiNDxaAHz0b6aBnwTthRL1r9HPFhKfqu+xPw89hnj1bPcpPAI/
e5R11vqwLkXKgaZD+7uMM7Fev7dTknpZmWtlZ1h4F8t6Zy3GfbaZ80/KlzXkVSUfYZGE0qsxn+JaeyBRTu/
eJHh3ExW2EZX68H6GkW9fiA5I8RmQt+0oamsPkpkN0+Pfb+RvRHRtS6ZoKYUhS7DU/
DX6WeJxgTRkHEwzUYDMHwGEAYa7r3Wr+MrA4y9wBzOqVSFPSSUCxYF6e8Di2DEMq+fQZj6S4D4XkR1BRoJQa
Fi+6W9L+fhRN0/Qob9h8ZyC2wPORmKI5wVdZL5MNM6Hy7nKUWrdOg5egWvQh4dTmEUOWyPH9ewvIUqaFLD0vI
R/gsd+wAhnbW2mrVKoDtHuyOyFcPSB0Pq0Gv/
xRJiSFY5aW+mi6gArXz+7DcR+AWc7c+xLrD+9tN1etDL5OWIp+c+D76W3vlIMVa+QNmDNHfmWOhiGaTRa5IfUc
Oxkpfw/
wexRfJLc14aWzg4Gq4+5S/5NRsLtuFlg/+/
jqGcUMbIxDw3eXCLeWwYXcPTHeuTHJMkycgQ9m8DukfgjFmJJaFcn6I+w/
ULGbM6ZIl1s6ngjj+fFhrdHycq4Hm1gWydW0jDIqI5ptHKiHCwufSmdYE+mjc+o/
hLiI+4lfpuLA0i/wA+Gb+Y8bAX0H7G57i6pTmBqCv06gLR7HKqsntzvHTEj48kZXzPdfNYFtPyoBeh/
6H4D+o2lT6Es9p0Qz5V7U/fQbwVTBQdR57fyxt9a0MzH+ixqqL5y0o6MvWPBwoR+FVUb+/
HwO8NNYvhr/fl6SI8mbj5y0SR8j7VNf1/Jf1rcDwa8xJWoUR7QYn2OsYm/
HdgHygYsyKzx3+T8x0H8F8xgX1jJKOBvPrzOsJ0sjOX9n92/
fJeLpkYfyKeFEyeDe2PmPlsflDhZ6WfUwDq9Y6gUhnKw6Mm8Y7jiNHxqKNO0qo4/
aibuQdZAkrnLc3DQzCL29RWA2wpRgumrd1qYRcVtTqxmNpHCRm3mAhCGq3QYHlm1/AnnJUBfk3+JEEJCxJU3/
jewVPqsCP2B3iEhBg/
Lwoea9H2XBEmfXoBWoRKF6bXQ9Pv8AQKd1eIvBl3fXz2p4Cd6AqyWHmRHCrQNrwQ+j9XeBI8vZ5e2bCtS3PB4L
aBzHrlg2J4yOOCXrnNr8d1U61g2IMlJMlP4o3qJaAq6KBiq0zTmlfZb2/
D5dXHOoF48pH1R9sB9zfXkYKkE5uVmWD5KiKEf9nJxvRHEfiSROIpqCiefSFiMvdgvZL1iFKyQ6Xa27rRbGHu
/7mOaWbQaxqsr/
0xWJyy9G6RLf9RUd4R2rmZuqZBwdvOxc0JXfbzKq1r0ZXBZuG3LBFGaMVn0j3n4BVHfpBaRpZ6RFRi/
bxpzBjFGlTYi4WvxPP0V2HiNwZW8CiQJSQ5RovWvh4D3uidKgNZ+uGVvqIeqfcFvxg3BEGfZm8I21UnD8UcQ/
40vCJFtQuGebPMm8yDc+VCTNrcEZ6dRZyoEWGS2T34ggU0Q9Af+KgfAAlYaQJ2+M1aDh+AvVYQpwM47ed/
Ysh8SnS/AlhiNeV8VHE7nONkiI0aoo+M78LZo2KRqvnczGuq6N4mrTkRJLp3MJPhZTkF+b1AqM4yFOu3/
IZpPbTvrN1wy+rKzq7KQf+hG1Shu8NQbnnLmWNedNY+wzCpKCcPfiGb+q5xYfaHvb5Jqo4tRGYTiGXqhtKkE
UOtijPAtb1EOrZqbBmc0h2fVLQPNAfeqN/c+fpQwlNR2Kt3gF2LSP4Mgh/BgiF5HkW4HrXajqHs/
2aYyDLkoCzxbcNKTLGXtLfyp5IWB1OAjiSlH3IOWhkyAefWgm3L2l5LGOs9dHSpKqa4Ljz3SJ6cdE5LvW2uq/
nvVbOnO8qnr3ziBJRuzc20Iq5jWsX2B1Hzm5JbpNjy4SoLvhyEnE/
6GUbdj6V2oZM9jkNw5mVtPsTlNhXZ0mK27YAUVC7t+vF5Du8RhsYrH3+v8iQVYCLrLffbHlPmIxo6t19F0xOf
+f3n8qZ8y3cGSSwuTXv+zwvc4cDp3R7pdoyOn/
bSXJ1hNdLdzUc5NyW/I9rMvm1wunTrIGg+tu8IUQ1fUPVk5c8RanoPAm7WOQb5WvQHKWm6Jz/
wzGq0YN8t25h8ssNf7RsK7/1eavZTL7dbMvWlUe/
HqU7chsTqDNo75PmG5dzUpq9OGSZ8jaZeBBQO1yjcrD3B8tlb8jwYfDJTNCYMwZvZlBgs0XvqZ7pgvwOJ5CZn
u/AUh5Dp+fTHBi04WPT4Ny72TNbOA3HmZO9wTAwwLBHIP3O4ZrhuoErONWp5eP8lNsgmm5XBifHmHRa5IuA/
Q9/Nn9L5flxfWO98Q6gaSjNsevL4vWVTwrcBnQMgL1XRiBXBA3iF1UARwlPXIbZbCgD8bsFLZjxAT6QtZl1bS
3kd/P/uv1fHv5tphNZhoJ98DD8tvVHfajIZV3L+Cjgfn8/4w8lMOLpeJk3O44hhHiwOA6m1BEe/
TX9Tob8Q5UoJUIiIdDqBpCa6IJcI1bCDeHvDtOkXVvlrN5MIN06U/
wn52A9cTE0VvFMLdOtmwJfQp0QQADFyXmklXxkKlzPWXkm16n0aG/
0Oq7bCWrRJWlaNRzrUq0Ba0E9YiAJFD4l2tAg9U94+y3l7LfNwp2G4nJEvfTIWMAnSDrEtpu6oQ7KafoAzAIO
4U3nGvBS0e3SVJRNCRgXEm8bJ5lUyhm7CuDY1CulpbW0D0jR0VPbp2Px1pEi0o0+k7AxZJIj/
kllHggV7MeEvnmsfsQJbf8mjg2FVEr9MJLYmEbOrKwBQ91+fDm/qb9sTnRn3/
RxybOzB6P4SHbdR6pNd5G7cvPaGmD9Ia35u3b6AHZd5wbTDHDSNRc6xtVLzXIfYF6O7Jt0tU6G2ZgV4gItTX
WgJ3Gdo9+lRfQThgKbu4NMPUmB9afsMTKoWhDx3kJV4xQ5Bmyg+6VK/
uKm5P+gR9kEhqtKp0YnI7XzpLlh7Ojzj6DRm6X6oj+/
3d6M0Ug7KDUSGVX8MbfUWmLePjO+pxjnQI0QCMQNnDJOu0IX6XpumgaELYbORl0VaQvSQPfDTXw1AILmx6mPU
3uuJ8xhHU7dnGkmQ/
KYjM3ja6rh+JbW0Ulw1LHk5YX0UjRjCZBZn2KhPg3w5Ofs3UzI07kTpK2Xb0Ohpb+eGq4vYDiucTBzhtPmfcs
c9qfvGP/D/Ke/m3Ndz+yoU6ZHVu6OXqRJeHo1/
5IeZBlHtOw+kSTtLhuqELT3tANAJVmVK8W0cTnvucHCYFzr1WUA7JM5TRqhWmc6a0bC4AAO4Q1x7smN5X6y
ZmJIe5hWbfgvcrYtefmZ9J5NPu5q+BnnKdZq+DjGl2lgyDVzoLgNeXHrgKXaB1D6lK1BMZL/
EJ5O3qngDuZjZjRqa0xbPNh/HJx5lRcePTY3yChnXYCzgi47KlZ7QzAHMzu8sRU4J+/
D7wAYSQb2ILmmIV2dy4VO3bPvfpDOv7c8ULjTIXzHkj5xgxHO0TuLOFnsNfzRwSdgVn+pMYaOdbmoORWB3bVB
J7NWq7A2+GQXv7sROe36ULptU4zHUUfWbDn8A+4Ak6ybXsjS6MKv9GfBCqOyTUv0xm1+vjc2HuNLl+O7eEWfM
yl8C8BU+vOKa+vx0/BwlGFYYAH9BNVdw7gO6zh2a6f2g/
5S5y0ZYGNk5+NNHHWSVW6xpcWA1Zxu53ucEdRK8ld3Ju6bpp3SPwL6nvsPc6eEl7h7yEDwAk6iVJPeZb3/
R5fOHn7i/
gg5WtlA9OsMGx5DW4EqfuwJ3Qvt8tOsDg6e8P9PAOgVvOJhgE9jRpbRK4pzHFfUZYQPAJi3+2TIEWUnkOH9Y5H
Ts+GqfZfVdx9afoZxPDcz+Zfuq05vKp+FPdKI25OVtfSGSE5QvKfNhFjUW7tW9nBD/
cCM2SGUjuvRXdbs0xXKv7OJ0sGX0QpyxYMlDTL9wT/L1wS6ENbtSFZZLh/
EBfX4ZutYxJi+zeZaDvrPEcEbTuwBrHt3F5VvE2HXq1NgA2UGiD9Myw8QfyQ6ny8CMQu3jv5TzbrUJGcnrNeu/
sZU29QxBydeDrMP52lo5rA3E72LrrbRHctMNQ7fvJoJH+LGOy6uvT6SmiAqhoVEUm7X5zHuHFHxxHWoW4Jwgm
lUnIN8DklV44X2ZwJS8OvCyoaqMOqZZUc/
Q1Ba49e9CVbnMojcuVI4fA9iNvHmR0+6g1T9F1TNYtQuxCJvmVyOaSBPs5ym1vjCDp6b62/
lRvW5tVmYcW8onkE1aIYYzyxmO40p9DWRm8aTwWodmHKzaK8sahRu69oEqK4A1WZWcvS0PkkxRg+bpQa6F+
YqzSfMlvz6vEmW9HGdqUvy7VJ4Y3BThN5WIPGOv4b9Zk+E7MnsqeY4FpHuKRdthJ9N8DaVPg1j/
P4Ti/1tTGoCw8X4ZyM7Kbgu/rdCH2c/
cSatf4PR8Rv8jMacC5wF3UPo+tgTuWsS6pR4rMrmYqJEmS1CqOnEmSJBXEcCZJklQQw5kkSVJBDGeSJE

kFeVff52zdddfN/v37d3cZkiRJXZo8efKzmdm3q37v6nDWv39/
Jk2a1N1lSJIkdSkiZnXdy2lNSZKkohjOJEmSCmI4kyRJKsi7+pozSZLUOm+88QazZ8/
mtdde6+5SitanTx/
69etH7969l2l9w5kkSWrI7NmzWWONNejfvz8R0d3lFCkzmTt3LrNnz2bAgAHLtA2nNSVJUkNee+011ll
nHYNZJyKCddZZ5x2NLhrOJElSwwxmXXunn5HhTJIkqSCGM0mS1K09Gi0ogoMPPvit5/Pnz2Xddddl9913X6bthw4dyvjTGPz8+Sy4Y+z7SheB47WLpXrBO3
nHYNZJyKCddZZ5x2NLhrOJElSwwxmXXunn5HhTJIkqSCGM0mS1G0igoMPPvit9wsWLKBv377sueeeeS7W
d/v378+yzz77jPiUwnEmSpG6z2mqr8eCDDDzJ//
nwAbrrpJjbYYINurqp7Gc4kSVK32mOPPfj1r38NwLhx4xg9evRby5577k+97nPMWjQIIYNG8bUqVMBm
Dt3LiNHjmTbbbflyCOPJDPfWufyyy9n6NChDB48mCOPPJI333zzkf298sorfOYzn2bgbbbZhq622Yvz48
S04ysYZziRJUrc64IADuPLKK3nttdeYOnUqH//4x99aduqpp7LttsydepUzjrrLA455BAATj/
9dHbaaSfuv/9+9tprLx5//z//HEApk+fzvjx47njjjjuYMmUKvXr14oorrlhkf7/5zW9Yf/
31eeCBB3jwwQfZfffW3ewDfA+Z5IkqVsNGjSImTNNmMm7cOEaNGrXIsttvvvv51rrrkGgL/
7u79j7ty5vPDCC9x2221ce+21AHzmM59h7bXXXBuCWW25h8uTJfOxjHwNg/
vz5rLfeotsc+utt+aEE07g9/4BnvuuSc777xzsw9xqRjOJElSt9trr7044YTmDhxInPnzn2rvX66s
k3brSo6umVFZnLooYdy9tlnL3Ffm222GZMnT2bChAmcdNJJjBw5klNOOWU5HMXy4bSmJLXI5FnzzOP/
WGUyeNa+7S5GK8/d///
eccsopbL311ou077LLLm9NS06cOJF1112XNddcc5H2G264Xnzan+vdt11V66+
+mqeeeYZoHbN2qxZsbZ5pNPPsmqmqq67KQQcdxAknnMB9993X7MNbKo6cSVILTJ41jwN/dDd/
W7CQlVZcgSu+NIztN167u8Gv8uSitGvXz+OPfbYj2Juu+8OPfbYxdpPO+00Dj/
8cAYNGsSqq67KZZddBtSuRRs9ejTbbcn/jEJ9hoo40AGDhwIGeeeSYjR45k4cKF9O7dm/
PPP5+NN974rW3+4Q9/4Otf/zorrLACvXv35oILLLmjNQTYoOhoufLcYMmRITpo0qbvLkKQunX/
rDM698WEWJvQK+NrIzTn6kx/
p7rKkpTJ9+nS22GKL7i7jXaGGzjzoiJmfmkK7WdVpTklpg2CbrsNKKK9AroPeKKzBsk3W6uyRJhXJaU5J
aYPuN1+aKLw3j7kfnMmyTdZZSlLREhjNJapHtN17bULREtSpSpIkFcRwJkmEdO0J
jV69eDB48mK222or99tzmbc5YsQI2m7NNWrUKJ5//
vnlUuyMpxJkqRJ3jVVWWYUpU6bw4IMP0tPTw0XXXbRXn3VyrHs0zTtKj1FYx7OYy4tts
3///jz77LPMnDmTLbbYi9+/+ctsueWWjBw5kvnz57nwPw5sPw5z//md13z5ww6vpwS5z//md13353tt9+enXfemT/+8Y/+8Y/
L9bi8lYYkSVpqp//
u2VD+qT/YOWaYP/+u+D/k6P8+bYoC/7odgBEoE/
9yZqL9+Vl89Qjg+PfkTq//dirYAlcQRBkkszBCmMDfgdqQGq+ku5/+usM/DN7zbFG7Wrnm8+rk/
u2VD+1+wYAE33HADu++++2LL7r33XqZNm8bGG2/M7rvvvzrXXXsuIESM488wzufnmm1lttdX43ve+x/
e///1OH3j+yCOPMG7cOC6++GL2339/
rrnmGg466CDGjBnDhRdeyKabbsoo999999zDV77yFX772982VHcjDGeSJKkpXnxtAQurp0Nz7+szJzI+bP
n8/gwYOB2sjZEUccvifoUOHsskmmwAwevRobr/
9dvr06cO0adAdPYcccdAfjb3/7G8OoO93XgAED3trX9ttvz8yZM3n55Ze588+OF+hTjxo1reF
rrr7+jY2rPcCZJkpZaIyNck2fN48Af3c0bCxay4oor8LUV+PcDtn3HN2Juu+asMxGx2PVMmzuU6iaYYVSzi2Yrz8
8rr7zyW6979erF/NnzWbhwIWuttVaWuttVaXNbwTXnMmSZKou2xZVh8buTlQe/Qce+99/
LYY4+xcOFCxo8fz04778wwbJrwSYcDLY8i4444477cSwYcO0444473rpG7dVXX+VPf/
rTUm97zTXXXZMCAAfzsZ8DIEfzsZz8DIDN54IEHutzWmmuuyYABA/jZz34G1I7r+yd9f7/85XKtzuW6lYYMGZJt
+yzD3379mXs2LGMHj2aQMGYWGxXw2aQYMGMWYsGGW+kP+KK67gkkzuYZtttmLJzuW6wlYYMGZJt9
yWRJEnNNX36dLbbbrsVNjeHy6H8aS666LMHJKZQ/
v1jo6+qwiYnJmDulqXUfOJEmSCuIXiAJ0nvGiBSRJ0Ao4Wx1NMKaA0D4Im61WbTQZiLiDERMSkiJs2ZM1ZRKwOXAMcl5kvAHA
HwYGA08B57Z17WD1XKwh86LMHJKZQ/
r27dukqiVJkrpHU8NZRPSmFsyuyM7RPSmFsyuyMxrATLz6cx8MzMXAhfz9tTlbGDDfdqvV82qntttmSvEz0KcvkOCw99UTdeD9Ww86ZHEVsrUBQ3t2zEf7ZlGIAU82sz5JkqqTSNPPbmgFcAk
zPzO/
XtX+orts+wIPV6UO6uAji5rRJiYgJsIRfn+tDM9Fx2bNtnd7esNQdeg8nfV8J1m1YPmfYbxtjQ5W3tq5YQ+lLj0I8Inm9s86Lbn/z7AAQk0aQMwpGEZGbGhtCxb7N06Ejm6AAtOaXfcwn88+xwKIlkqQl6nx9NY6Y8uf7gLAzlxDB6zqVEGaYq7+4oQ+aRl
sgQzn3BzXIM+5
rCIcIskAV53fSu9mfhnhz7LsZR/Kx+oAAw3Rzkc2uM1bXfrw4kW6yM2KrgLNx9DwK+WPy5Z7cW
QpDJADrddqEx0+IT7Y48LJkxD8P7Pgh9u+/
2J0bqZL+fOKTNTJQ3AAAhI8IfOE1B+9gbkrIkuLkGgy4uLUaBmvVhqUFkrKJkbn2lPOAbqZJqGnJNLf
cnRsxLm+4+F4Yi3+uWG2kRMxqVqtiIGk0QPNtBLA+YqkRcqGYSVmlUxAabDNPrtUEdTNy+vMX9lTpzoerxbYRiOC+vaDEqtLAK4cewCEXMnJkxkHuR6JlMj6XVqs
VUXX7g5rJKr2ktoKZKk4f2Ge7kOo5bYdfL9OJM2bZ5e98C+/
TN+bdHmSUkVpdcPH8f4htXVNkIV03nN5I8JsWG3LjmldFWO0WiDoUJRqUkLmsKCmWAsUnGGc5+LjJcD/
9YM1kEjcLWFjPo3p1kSJBTGcSZIkFcRwJkmEMOZJEl1QwWiOXQhg+WQXSoyw/EZoHLS1y1T7lEJpoNUkpOd+1/PxlrFZR8x8vbDnt8vy85cvVNQ1HJ3u1EMq7rDyKKp8FBq9dm7yXp
vwSPk4Kqme0Kha6llpaoDrm7bkd3lGv2qAwkuA9Z2DU5v5hvZuXoGbZ+aMcL64U18XQG2U+rDvD1yhlfPLl9KM5IHulSCmKrKUbkuOUcrE4jkIP3M26jcjFqzFbgK0U1FZ2qb7vSu5vbb5zvDnE8HkRJMvY5s+M1AtmX4snrYIGcU2AhAA
YEWOZG+OIuoG4W86qK2lMe3kDoAYF80UIFwZwJ32EJrc+QJkgAB8kYYSwsNyONDVlE2QUEYi5NhnPhwfNf7cEJl0MCUlDPRgQGxYFmbzKfXFGPsOU0JE7OQ3WdwPZkfuGzMl9UXEkPOR7nnkEiqaqxdTOyJXfAY8CP/bYmB1
JIMuOOGj3RNfI6U4Cd1fehGOcu0+/xc8PIZDZwZLRPr1p2hSRgT82AKTkpNM/xA7Iek+FQMgDJuhEZPsXqXz4DmwexeMlLGoK6QAJeRVLyzRRp6JDPPTmwAA3nTv5BtkeN7OEsj0u9+mmQ2q/

UkEMZ5IkSQUxnEmSJBXEcCZJklQQw5kkSVJBDGeSJEkFMZxJkiQVxHAmSZJUEMOZJElSQQxnkiRJBTGc
SZIkFcRwJkmSVBDDmSRJUkEMZ5IkSQUxnEmSJBXEcCZJklQQw5kkSVJBDGeSJEkFMZxJkiQVxHAmSZJU
EMOZJElSQQxnkiRJBTGcSZIkFcRwJkmSVBDDmSRJUkEMZ5IkSQUxnEmSJBXEcCZJklQQw5kkSVJBugxn
EbFZRNwSEQ9W7wdFxMnNL02SJKnnaWTk7GLgJOANgMycChzQzKIkSZJ6qkbC2aqZeW+7tgXNKEaSJKmn
ayScPRsRHwYSICL2BZ5qalWSJEk91IoN9DkauAj4aET8BXgMOKipVUmSJPVQXYazzHwU2C0iVgNWml9ym
ml+WJElSz9RlOIuIU9q9ByAzv92kmiRJknqsRqY1X6l73f73QfYE5jenNcjenHIkSZJ6tkamNc+tfx8i
SZKkHmxZnhCwKrDJ8i5Ek1iRJjV1z9geq22gAvYC+gNebSZIkNUEj15ztWfd6AfB0ZnoTWkmSpCbocloz
M2cBs6k9vqXsH5EbNTswiBJknqiRqY1/xE4FXgaWFg1JzrCoiXVJkiT1SI1Max4LbJ6J6Zc5tdjCRJUk/
XyLc1nwBeWNoNR8SlEfFMRDxY13ZaRPwlIqZUf0bVLTspImZExMR8eml3Z8kSdJ7QSMjZ48CEyPi18D
rbY2Z+f0u1hsL/
Bfw43btP8jMc+obImIgcACwJbA+cHNEbJaZbzZQnyRJ0ntGIyNnjwM3ASSsBa9T96VRm3gY812AdewNXZ
ubrmfkYMAMY2uC6kiRJ7xmNPCHgdICIWC0zX+mqfwOOiYhDgEnA8Zk5D9gAuLuuz+yqbTERMQYYA7DRR
n5pVJIkvbd0OXIWEcMjYhrV8zQjYpuI+OEy7u8C4MPAYOApoO3RUNFB3+ygjcy8Kne37LmMZk
iRJZWpkWvM84NPAXIDMfADYZVl2lplPZ+abmbkQuJi3py5nAxvWde0HPLks+5AkSXo3a+jZmpn5RLumZ
bpQPyI+VPd2H6Dtm5zXAQdExMoRMQDYFLh3WfYhSZL0btbItzWfiIgdgIyIlYCvUk1xdiYixgEjgEjgHUjY
ja1G9mOiIjB1KYsZwJHAmTmQxFxFTCN2iOijvabmpIkqSeKzA4v7Xq7Q8S6wL8Du1G7NugSbko7Z
MiQnDRpUneXIUmS1KWImJyZQ7rq18jIWWTmgcuhJkmSJHWhkvO7oyIGyPYGyPiiIhYq+kVSZIk9WBdhrPM3
BQ4mdrd++
+LiOsj4qCmVyZJktQDNfptzXsz82vUnMfcJHQtdFAoRq5Ce2aEXFoRNwA3Ent5rE+WkmSJKkJGv
lCwAPAL4BvZ+ZdTa5HkiSpR2sknG2SmRkRqzW9GkmSpB6ukWvOhi3HZ2tKkiSpEy19tqYkSZI619Jna0
qSJKlzTXu2piRJkpZeIyNyRwFHIyNnRwFHAxsAs4HB1XtJkiQtZ12OnGXms8Ais0EBFDqu
lMImK9iDgLeKQl1UmSJPUwSwxnEXEcMAX4T+DuiDiU2rVmqwDbt6Y8SZKknqWzac0xwOaZ+VxEbATMAH
bJzLtbU5okSVLP09m05muZ+RxAZj4O/MlgJkmS1FydjZz1i4jv171/f/37zPxq88qSJEnqmToLZ1/v4P3k
Xv37zPxq88qSJEnqmToLZ19v935yMwuRJElSJ+EsMy9r8PFNkiq8SJElq8PFNkiRJag3DmSRJUkG6DGcRs
bZIkSXrnGhk5+88G2yRJkvQOLfELARExHNgBWDsijqtbtDqwQrMLkyRJ6ok6+7bmSsAHqj6961q/BOzbzKIkSZJ6qs6+rXk7cHtEXJqZfwaIiF7ABzLzpVYVKEmS1JM0cs3ZWRGxekSsBkwDHoqI9ve+kiRJ0nLQSDgbWI2UjQauBz4GHNzUqiRJknqoRsLZShGxErVwdl1mvgFkc8uSJEnqmRoJZxcCM4HVgN9HxMbUvhQgSZKk5ayzb2sCkJnfB+ofgP7niPhk80qSJEnqubocOYuItSLi+xFxf0RMjoh/B/q0oDZJkqQep5Fpzf8C5gCfoXZ/sznA1c0sSpIkqafqcloTWCMz/7nu/RkRMbpZBUmSJPVkjYyc3RYRB0REr+pnP+DXzS5MkiSpJ+rs2ZovU7tlRgDHAVdUi1YA5gOnNr06SZKkHqazxzf1XtIySZIkNUcj05oARMTfRcS+EdH+IejtW1SRJktRTdSEMZWSSSJEnJLEzkLNRlWQVCRzDMdw6f9mVpxBSJSiu01JOTKIeqQwDR2sd76E/Dg+wV7z4ekaFeN7BstRXrANS54a48aeBkNzYUg7+HJmGU9U1kHyBy
zSk2KRsXxCdIB7YaJmJTQkz2uZ0qeJawbEpK1xDMyVyc0r2yHaUL7Y2bN17YMeHIKDRBZP1Mq2DWJuzyaqZSzbbMc+W/
YKUgwHZWo00AW90Slr+nu/
vpjIL0u9wv0EWjiKH1jXgEZpn7TqGxpFQmmAgQxlqY2tD1ZpT5ePWbNoY2Ix+3B8x8YTfh2Axf0yf/
iOBL4tk7s9Gv1oPyfrQcXYH1l00r8OlXgvN5EXUEyzR7H/uFiCgHXu2P4stkX6szDYnXs8qwNGM3g+pGpDQDBnR4DARTrSzDBNeAJ+DQ40P8PnRbn0tODk/iQxNjfwnwZ6v7WJkbdgr1u3e44XY/
d6fCcunkSqFaaXdvfVFZQUhhZRHSdAo36rdFQHKgbWv6Xw/sxNvZtG9xUqqrvf3Q2BSZ9F8XOp0ek03xnZT/
c5A/1Vbck0fLX7iR0m0nADaxAoPRPRlhZmzaKQFN8/qCqtyYSj3ee9NbQZqpJUyJR6Xc6PlkyN8NH+MftLCEMIsvkC6cy0BZovfrDk2/ag0HZvQZIcfSxPvVjNgXEF2Cx5GwT/
72RmHU+K4mI8JfohaDqV+y6ZJAwj4yVgfb/I8OD0kP5zkqTnzAygTw2gPaNYJqA0aunDE5ZG+FEgyyf1kOTNdzFlo1OTE11NsNFqgdhY7OWBdFLTPRTzpt8P6BzsyKH/QwFQ4xJUW/
PoDr5VFEb2kz5yMYRxnTrfMRDfmKlGNeBZpDb+3WqDAE7ECqBKpCh+gB/eZV1Ck7kFFiXVsO6xjc/3hVB4HTvD7Nuuij+6n+F3iiZSMGCZ3ILl3oY9R/
IxAsK2udP6GxVWcRtg11sX5ilKg+ktljHUNaaF+WMUy2XBi9c/Cl4RhvU27TaH3Q8fZmyB6RzWr6t8h9/
FPxdNzyNkNr8B7qMuyh8AKujvZZtOvv07ZPNNTaQ2ROwY2yQbOnvr5VbNZU/
/pDL5TTbGbXdEa3cGyhsKGKbF1JQlRrBlukeLZfW9Ovjgx1QV2Fq93Qa8u03WZCXtl+M/
pnWg3Y2xHLQsDBBqwcMhHUgFGtyZFWOvh8KgYOGk7kQxsI7Gp8lZQTo/1oD2iRXWylWy/AsP3pW4PGYcSp+C8AoZ2BdpjNbYl5+qpPAOV+GRW0mHHZVKoaJ1Wq6lmBzUZEi86m+HXwbVn1Ao6S4WunW/
I5KqsRzaI8PE0HBxhmyBHqv3vdVWaOQYAmTS8k17TYOWSQ+wGsBZkHQhjNcCMLEfHxnt1Z0ECJpfYVZZuChiU0jT/
KpGT3eoohY3ygIAdBNW5MMKW7U9ik4kE/
XwbUm0VdX4tuLk2k17a8tL14EupzoyeMtaKOa5sk+XNhy8fVP7BvpuzF3ncZDxgqlfnEY+3MnfFb8SVQ16T3+bxTgOZZMvmRUDohPoL+tZ5vCrPGlIRwrT7xtI0IV36/
yf0t0ZIzpy3vK9y2cGUGHxgmnmHGaQUTzOHKQsYZhtNzWOdgp8vE7GmqrDVtkt7yjkxvGj53R03YNoWXKaBsq95a0eWg5ThCk4SxjrNyP0MxrK1vXUuOl0btsyB

```
bj88ssBOO6445gyZcoiyzfbbDMuuugiAMaMGcOf/
vSnRZYPHjyY8847D4CDDjqI2bNnL7J8+PDhnH322QB84QtfYO7cuYss33XXXfnWt74FwB577MH8+fMXW
b7nnntywgknAP7u+bvn7149f/
f83Wv2714pShs5iw7acpEOEWMiYlJETJozZ06LypIkSWqNyMyue7VIRAwHTsvMT1fvTwLIzLM76j9kyJ
CcNGlSCyuUJElaNhExOTOHdNWvtJGz3wObRsSAiFgJOAC4rptrkiRJapmirjnLzAURcQzwv0Av4NLMfK
iby5IkSWqZosIZQGZOACZ0dx2SJEndobRpTUmSpB7NcCZJklQQw5kkSVJBDGeSJEkFMZxJkiQVxHAmSZ
JUEMOZJElSQQxnkiRJBTGcSZIkFcRwJkmSVBDDmSRJUkEMZ5IkSQWJzOzuGpZZRMwBZrVgV+sCz7ZgP2
qc56Q8npMyeV7K4zkpUyvOy8aZ2berTu/
qcNYqETEpM4d0dx16m+ekPJ6TMnleyuM5KVNJ58VpTUmSpIIYziRJkgpiOGvMRd1dgBbjOSmP56RMnpf
yeE7KVMx58ZozSZKkgjhyJkmSVBDDmSRJUkEMZ5WI2D0iHo6IGRFxYgfLV46I8dXyeyKif+ur7HkaOC9
fi4hpETE1Im6JiI27o86epKtzUtdv34jIiCjiq+nvdY2cl4jYv/r78lBE/
LTVNfY0Dfz7tVFE3BoR91f/ho3qjjp7koi4NCKeiYgHl7A8IuI/
qnM2NSK2a3WNYDgDICJ6AecDewADgdERMbBdtyOAeZn5EeAHwPdaW2XP0+B5uR8YkpmDgKuBf21tlT1L
g+eEiFgD+CpwT2sr7JkaOS8RsSlwErBjZm4JHNfyQnuQBv+unAxclZnbAgcAP2xtlT3SWGD3TpbvAWxa
/RkDXNCCmhZjOKsZCszIzEcz82/
AlcDe7frsDVxWvb4a2DUiooU19kRdnpfMvDUzX63e3g30a3GNPU0jf1cAzqAWlF9rZXE9WCPn5cvA+Zk
5DyAzn2lxjT1NI+ckgTWr1+8DnmxhfT1SZt4GPNdJl72BH2fN3cBaEfGh1lT3NsNZzQbAE3XvZ1dtHfb
JzAXAC8A6Lamu52rkvNQ7ArihqRWpy3MSEdsCG2bm9a0srIdr5O/
KZsBmEXFHRNwdEZ2NHuida+ScnAYcFBGzgQnAP7amNHViaf+70xQrtnqHhepoBKz9PUYa6aPlq+HPPCI
OAoYAn2hqRer0nETECtSm/Q9rVUECGvu7siK1qZoR1EaY/y8itsrM55tcW0/
VyDkZDYzNzHMjYjjwk+qcLGx+eVqCIv5b78hZzWxgw7r3/
Vh8ePmtPhGxIrUh6M6GRvXONXJeiIjdgH8B9srM11tUW0/
V1TlZA9gKmBgRM4FhwHV+KaDpGv037JeZ+UZmPgY8TC2sqTkaOSdHAFcBZOZdQB9qD99W92novzvNZji
r+T2waUQMiIiVqF2YeV27PtcBh1av9wV+m97Bt9m6PC/VFNp/
UwtmXkPTfJ2ek8x8ITPXzcz+mdmf2nWAe2XmpO4pt8do5N+wXwCfBIiIdalNcz7a0ip7lkbOyePArgAR
sQW1cDanpVWqveuAQ6pvbQ4DXsjMp1pdhNOa1K4hi4hjgP8FegGXZuZDEfFtYFJmXgdcQm3IeQa1EbMD
uq/inqHB8/
JvwOrAz6rvZzyemXt1W9HvcQ2eE7VYg+flf4GRETENeBP4embO7b6q39saPCfHAxdHxD9Rmzo7zP/
pb66IGEdtan/d6lq/U4HeAJl5IbVr/0YBM4BXgcO7pU5/
DyRJksrhtKYkSVJBDGeSJEkFMZxJkiQVxHAmSZJUFMwMiP4RMbF6PyIiXoiI+yNi
ekScupTbGxERS/XIp6qGxW7aGRFHRcQh1euxEbFv9fpHbQ+fjohvNrD9/
hExsaptbAfLB0fEqLr3p0XECUtzDI2oP4YG+/ePiAeXsGyiNwWWymY4k7Q8/
V9mbkvtUVoHRcT29Qurp2s0XWZemJk/7qD9S5k5rXrbZThrwGBq90RqWHVzS//tlbRE/
gMhaVnNoXYz08UeY5aZrwCTgQ9HxGER8bOI+BVwYxVO/
i0iHoyIP0TEF+tWXTMifh4R0yLiwrYQExEXRMSkiHgoIk5fqGzWKiO8Cq0TEl
Ii4IiLOiIhj6/p9JyK+Wnd8fwNeaLetlYBvA1+sttN2HAOr/TxabaNtJGt6RPwQuA/
YMCJGRsRdEXFf9fmsXvX9bnX8UyPinLpd7hIRd1bbbRsJ7OyzbKtzlYi4streeGCV9n0klcUnBEhaJpn
5serl59svi4h1qD1X8wzgY8BwYFBmPhcRX6A24rQNtecI/
j4ibqtWHQoMBGYBv6m2fTXwL9W6vYBbImJQZk6t1nkxM4dW05jnAXs2UPuJEXFMZg6u6u0PXAv8exUID
wCGVnfQbzu+O9tt428RcQowJDOPqbZzGvBRao9JWgN4OCIuqFbZHDg8M79STcWeDOyWma9ExDeAr0XEf
wH7ABAdybmY9m5pvAu
APwamYOiohB1MKhpII5ciZpedo5Iu4HbgS+m5kPVe03ZWbbCNtOwLjMfDMznwZ+Ry3AAdybmY9m5pvAu
KovwP4RcR9wP7AltQDXZlzdz+HLUnRmzgTmRu1ZrSOB+9/Bo41+nZmvZ+azwDPAB6r2WZl5d/
V6GLVjuCMiplB7bu/
GwIvAa8CPIuLz1B4f0+YXmbmwmpZt22Znn2WbXYDLq+OcCkxFUtEcOZO0PP1fZnY0cvVK3evoZP32z5P
LiBgAnAB8LDPnVRfm91nCOu/keXQ/Ag4DPghg+g6283rd6zd5+9/Z9p/
BTZk5uv3KETGU2sOwDwCOAf6ug+1Gu59d8Tl90ruII2eSWu02atdp9YqIvtRGdu6tlg2NiAHV1OIXgdu
BNakFmxci4gPAHu2298W6n3ctRR1vRETvuvc/
B3anNvL0vw1u4yVq05dL625gx7pr5FaNiM2q687el5kTgOOoTVl2prPPsr7PgdV+tgIGLUO9klrIkTNJ
rfZzatOPD1Ab0fnnzPxrRHyUWrj6LrA1tVDx88xcWE2VPgQ8CtZRbnsrR8Q91P5nc7GRqE5cBEyNiPsy
88DqGrJbgeeradVG3AqcWE1Nnt3ojjNzTkQcBoyBoyLiJZ2QcBoyLiJWr5pOphb1fRkQfaqNi/
9TFppb0Wfav63MB8D8RMRWYwuLhTVJhItPRbkmqRuvuA/
bLzEe6ux5JPZfTmpJ6vKjdmHYGcIvBTFJ3c+RMkiSpII6cSZIkFMZxJkiQVxHAmSZJUFMwMiPAQUxnE
mSJBXk4gyZcoiyzfbbDMuuugiAMaMGcOf/
```

```
      "text/plain": [
       "<Figure size 720x504 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "name": "stdout",
     "output_type": "stream",
```

```
     "text": [
      "----------------------------------------------------------------------\n",
      "The classification threshold wich maximizes the profit: 54.50%\n",
      "Profit: 281,00Â â¬\n",
      "----------------------------------------------------------------------\n"
     ]
    }
   ],
   "source": [
    "# Define thresholds and profit/cost per answer\n",
    "thresholds, c = [mean_t], 1\n",
    "revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "# predict with model\n",
    "y_prob = pipeline.predict_proba(x_test)[:,c]\n",
    "revenues = []\n",
    "dict_thresholds = {}\n",
    "i=0\n",
    "for t in thresholds:\n",
    "    y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "    cm = confusion_matrix(y_test, y_pred)\n",
    "    revenue = cm[1][1] * revenue_answer\n",
    "    expenses = cm[:, 1].sum() * expense_answer\n",
    "    net_revenue = revenue - expenses\n",
    "    revenues.append(net_revenue)\n",
    "\n",
    "# plot  \n",
    "plt.figure(figsize=(10,7))\n",
    "plt.plot(thresholds, revenues, marker='.', label =
pipeline.__class__.__name__)\n",
    "plt.plot([0, 1], [0, 0], 'k--')\n",
    "plt.xlabel('\\\"Probability\\\" threshold')\n",
    "plt.ylabel(\"Net Revenue\")\n",
    "plt.title('Profit curves on unseen data')\n",
    "plt.legend(loc='best', title=\"Models\")\n",
    "plt.show()\n",
    "\n",
    "t = thresholds[np.argmax(revenues)]\n",
    "profit_dict = dict(zip(thresholds,revenues))\n",
    "max_profit = profit_dict[t]\n",
    "\n",

"print(\"----------------------------------------------------------------------\")\
n",
    "print(\"The classification threshold wich maximizes the profit:
{:.2%}\".format(t))\n",
    "print(\"Profit:\",format_currency(max_profit, 'EUR', locale='de_DE'))\n",

"print(\"----------------------------------------------------------------------\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 105,
   "metadata": {},
   "outputs": [],
   "source": [
    "p, r, thresholds = precision_recall_curve(test_y, y_scores)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 106,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "     pred_neg  pred_pos\n",
      "neg       357        24\n",
      "pos        30        37\n"
     ]
    },
    {
     "data": {
      "image/png": 
"iVBORw0KGgoAAAANSUhEUgAAAfUAAAHwCAYAAAC/hfaiAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzt3XmYXFd95//3V5sltTa35d2ShddgjGODDY8xDMOYHyRiHwIQfAThE
icETxKWEEgmkPAjDoQkkyEJmQECnpgATsI+IRrHjIfNOIANljeMV+liwvakvWWVt2SWjq/
P85tU91Vra5u9e3qPv1+PU89qrrn6ta3blX1p+69554bKSUkSdLMN6fbBUiSpUhqhEuhlqEuSVAhDXZKkQhjq
kiQVwlCXJKkQhrokSYUw1DVpIuKuiLhwjHlWR8SuiIIg7RWVNSER8MiL+tNt1NIuI6yPiN6r7l0XEt7td
00RFxMsi4v+LiKXdrmWmmr3PCJSRJwygf+3pvq/80e80ZpvyIi/vHQK9RoDPPZ2BHIm4Dbpkh2+iSm+uGHhkZ5VoGbkHaKPwD9Vh4ajeoJRi3HuUskIJTH3OxlyqIeeIhIlgmKgDfX6+vqn
iIqlk/08KaXnpJSuH2OeR1JKS1JK+yf7+d0cd1f7U+d0TEHRHxym7GXNVEEwIuB/
wVcDPxLRCzockieTrvkH2ijtBw1HaTSG+uzx8ymlJcJcDzgJ8C3jNyhsj8THTmxmp9rgA+Cnw2IlZ0uaZJV
UegRMZwOeBXwIuALYDV0/
Xz12778RM+J7MhBpVD9/0WSal9CjwFeBMeGaL4QMR2R2gAZwUEcsj4jqIeyjIho2IP23eXR4Rb4qIeyJ
iZ0TcHRHPq6Y/
HBEvr+6XFxHrqi3ZJyLir6vpx4d4rIs4d7bVFxN9GxMbgw+EfYvJFhgOfpbyKyDoGfw6ZyJV4rfZUXlUhyb
OW+ptgg7WlZEnBMRt1ZtnwMBXAz3AqU3LOz8ivhsRT1db8hc2tfVg7mWb0s2R8S1RkIFc0UgY0Wb+
n67meWFTvS3LPEjdB6vxlIj4Vksj4i+6nPf7OUR8aPquT4S2/Q4tXB3OD8hQ+DRbRBg/
zYCJiVUT8r6q2pyLiw9X0X0YbuA23wm230nxvU9GVqvEfHB6r18KCJeUb9VfqeWjvAfPqp5j
Z0R8FVjZ2bunCUspeSv8BjwMRM+r+X8R8tR/
AKcCJbZ7nRuA/VfeXOASdX99cAV2BTBA3k07F/
hz4KaDvM43AEdUrUr+OdwOPAwrGWBSwANgC/
W73+1wL7gD8d5XkXBnYCxxVTTTseeKp6q6yFLiw9X0YbuA23wm230nxvU9GVqvEfHB6r18KCJeUb9VfqeWjvAfPqp5j
```

wGJgKfAF4MtNz3s98Bsja2hT32pgJ3BptfwjgLNHLqPdcqr35qtAL7CIvDW9EYiq/
XCgHziuem23AO+t1uFJwIPAf5jkz+9HgadHuf1glP8zF7gD+Bvy53gh8NNNn4V/bJp3DcM/
k9fT+p1oN+1g35PLqs/Qm6pafgvY3LQeh70PbeofVtM4ljmeGj8D/
FH1Pj6zfpo+B9eQ90StJn8vL6rafh1YX73fS8iHUq4eZV3eCPw1cFj1WdrZvO69Tf6t6wV4m4I3OYftr
uqP4Ibqj+Siqu164H1N8x4N7Blqr6ZdCnyzun8d8DsHeZ6hUL8B+BNg5Yh5nvnSk39g7AeWNrX/
OfDJ6v4VwNea2s4A+sfxurcBPznWsqo/Ns/8caymfZeDh/
pgtT73kUPudU3tfzD0R65p2nXArwLHAgeAwzuo/2xgW9Pj6+ks1N8N/Msobc8so91yqvfm/
2l6HOSguKB6/CbgG9X9FwCPtHnuf5gGn/
kXkoNoXpu2Kxg71N834v8Mm8bY35PLgPVNbYur5zim3fvQpsZhNY1jmeOp8dPAlcAJbZ4/
MTzkPw+8q7r/deC3m9pOJ38P5jH8+72a/
D3paZr3nzHUa725+332+I8ppRUppRNTSr+dUupvatvYdP9E8i/
8x6pdt0+Tf+kfVbWvAh7o4PneCJwG3BsRN0f7jmTHAVtf7jmTHAVtTSjubpm0gb+kOebzpfgNYGKP3rH1n5MMC26u
6lzN8d99oyzoOeDRVf3Wa6jiYm1JKK8hbrmvJu1OHnAj84tD6q2r5aXKgr6pe87Y29S+OiI9XuzV3kH8
YrYjxnynQ6Xs0mmc+D9U6+Sw5DCAfC/+n6v6JwHEjXucfksOk21YBG1JKgxP8/
xvHmDbW9wSaPm8ppUZ191A7qI61zPHU+F/IP9q+H/lw1K+P9lzk78vQ8xzH8O/HBnKIj3zfjyP/
KN09Yl7VyJ6VgvzLeshG8q/
7laP8QdwInDzmAlP6EXBp5M46rwG+GBFHjJhtM9AbEUubgn01eff+uEQ+fv4HwMuAu1JKByJiG/
mP1lgeA46PiGgK9tV0EIwppV0R8dvAAxHxiZTSbeR1dHVK6U0j54+IY8mveUVK6ekRze8kb/
W8IKX0eEScDdzW4WtothE4b5S23eQtvCHHtJln5KUbPwP834j4C/
LW+S80Pc9DKaVTqVFEfIx8aKWdDSml57SZvhOJ7IORk4b63sylrEujznRy2d2XGNK6XHynh
ci4qeBr0XEDSml9WM8x2byD4YhQ1vkTwDNfUAeAw6PiJ6mYF/NxF+bOuCWuoZJKT0G/F/
gryJiWdUp5uSIeEk1y98DvxcRz4/
slIg4ceRyIuNEXFkyh3JhsJr2GlsKaWN5N3cfx4RCyP3jH4jP94SHI+l5D8sW4B5EfFeYFmH//
+LSLmRcRrGD0UW6SUniKvl/dWk/
4R+PmI+A8RMbd6bRhGxGN6bhf9w5qhf4405rGOGfyJ2cXle9niOqHwgAtwOvqf
YKnEJe32O9vtvI6/Xvgeuafox8H9gREX8QEYuq13pmRPZUBOse7fl/M+XTINvd2gX6UG2PAX8RET3Ve/
Ciqu124ILIYyYsJx8yGG9NY31PxvIE+Zj0aLaQD9McbJ5DqjEifjF+3BFzGzlsOznV9DPA71ad4JYAfw
Z8buQPh5TSBmAd8CcRsaD64fDzE3096oyhrnZ+hdzx6W7yl/
2L5F3HpJS+AHyAfGxsJ7kjTm+bZVwE3BURu8g9my9JKQ20me8G4zcC/
AH+cUvrqBGq+jhyW95N38Q3Qfhdqi5TSXvLehMvIr/f15M4/4/Eh4OKIOKv6sfJq8q7oLVUdv8+Pv2//
iXwM8l7gSeDtTctYBPQBNwH/Z5w1DL2eR8id9N4JbCWH2E9WzX9D7tT3BPApOv8B9Rng5eT3feh59pP/
SJ8NPFTV/ffkwx5d1VTbKeQ+AZvI7yvV5+tzwA/IHf2umeDTjPo96cDfAq+tepb/
9zb1N8jfs+9Uu87Pr7HGnwK+V31H15L7yjzUwTI/QT7j4wby+z4AvHWUeX+JvHdnK/
lH6qcn9jLUqaFek5IkaYZzS12SpEIY6pIkFcJQlySpEIY6pIkFcJQlySpEIa6JEmFMNQlSSrEjBt8ZuXKlWlLleSpGkpIjoeXr
lxyy239KWUjuxk3tpCPSI+AbwSeDKljGab9brJLLleSpGkpIjoeXr
fO3e+fJA9AMppXkC9VeSpwOfB3NdYiSVLxattSTyndEPn6yaN5gymdCbJiRUQcWw1tBAbj//
kksdIbq7YWVXplYktSkm8fUj2f4MJ6bqmktoR4Rl5O35jnqqJO4884pqW/
aGhiA5cvh534OYryX+pAkFaubod4ujtqOWZtSupJ83V9OP/3ctHIlzJnF/
fY3boTHHoOUDHVJ0o91M9Q3ka95POE8kU9xrRwIcybcf32J8/
ChdBojD2fJGl26eb27lrgV6rLd54PB/
reLokSRpdnae0fQa4EFgZEZvIl92bcD5BS+hhwLfl0tvXkU9p+ra5aJEmaDers/
X7pGO0JeHNdzy9JkjSbOCKcJEmFMNQlSSqEoS5JUiEMNQlSSqEoS5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBJhag11CPiooi4oLyLWR8S72rkSQmqhJUIoyqZSqzebXXCOJVkPkdUBcqOGnoKeIbpi9iohqZi4l13UmSJYLG26OMdQgRkCvj4AjOTu2IwTyHxcTE67aEhNJSN4wyBChsjKT7njZf7OVp5tdknrEmT5kEXMmO9RaJw4jnoYkTkBlyzhEeJEEgCzQUxvrxkTl6NSS7TrHOjGfXoGnz1YsMhwfhO1Bz5f6OWxcGmGNq3VRYxmOBfREwEkm63OTXdsqb
EuSVAhDXZKkQhjqkiQVwlCXJKkQhrokSYUw1CVJKoShLklSIQx1SZIKYahLklQIQ12SpEIY6pIkFcJQlySpEIa6JEmFMNQlSSqEoS5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBJhTDUJUkqhKEuSVIhDHVJkgphqEuSVAhDXZKkQhjqkiQVwlCXJKohsoh0ZK40xCo2Csq2Snu5KQzKAz72mSSaSBIhWtpCKvjJu4SikE5RwB5zM2SxqIzY2S8xIQRSVOVqhKGkIUhXQskl6SqUMQCuSSlJKqSkqAfKrUSqhHVXDJhgpyhWEAphQjkikQVwlCXJKkQhrokSYUw1CVJKoShLklSIQx1SZIKYahLklQIQ12SpEIY6pIkFcJQlySpEIa6JEmFMNQlSSqEoS5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBJhTDUJUkqhKEuSVIhDHVJkgphqEuSVAhDXZKkQhjqkiQVwlCXJKkQhrokSYUw1CVJKoShLklSIQx1SZIKYahLklQIQ12SpEIY6pIkFcJQlySpEIa6JEmFMNQlSSqEoS5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBJhag11CPiooi4oLyLWR8S72rkSQmqhJUIoyqZSqzebXXCOJVkPkdUBcqOGnoKeIbpi9iohqZi4l13UmSJYLG26OMdQgRkCvj4AjOTu2IwTyHxcTE67aEhNJSN4wyBChsjKT7njZf7OVp5tdknrEmT5kEXMmO9RaJw4jnoYkTkBlyzhEeJEEgCzQUxvrxkTl6NSS7TrHOjGfXoGnz1YsMhwfhO1Bz5f6OWxcGmGNq3VRYxmOBfREwEkm63OTXdsqb

ieZ7J5eEDSEENdk26snueTbWSP+Yk6cCBvkW9ucwWCefPG3wN/
qjz1FKxb1/7HzcBA3jNiqEuzg6GuSffII3Drre07o82b19pjf7oYHMw/SJ71rPbty5a1nz6axx/
P66Kd3l44+eTJWxf9/fDc57ZO/
+EPc7BLmh0MdU26ffvyVuPpp7dvH9m572AaDdi+HW6+OQfhyLY6TNYgNw8/DDfdBEuWDJ+
+Z0/+wXP88Z33Lejry3sQ2r3moWnt1mu7DomSymWoz0A9Pfl48cjR16aTiPGF98E0GnDvve23lMe79Tz
VliwZfUCd8YydODS87ebNMH9+a/t4+w9IKpOhPgOtXp1vs8mqVXDSSfU+R2/
v+HdVD43u953vtLYdrPPaaA7WH2HhQjjttNbOepI0xFDXhPT1waOPth/
oZrIHkpkqp5zSfrCag0kpv9577mnfftRR41ve44/nc/nbbY3Pn9+6K1+SmhnqmpCtW/
Nx7tGMPP5dspQmrwf+4GA+x/
6MM9q3H3bY5DyPpDIZ6pqwOXPgrLPq7c3e05M7lB15ZH3PMdWGdtnfd19rR7ndu/
P6HHmxmal0sPPePeddmt4MdU1r073/wJIlExvedteufG55ux9E3d7FvmULbNzY+sOivz8fHjj//
PGNnS9p6hjq0iGY6I+OwcHc8W86dnobGIAHH2w9X3/Tpnw63nOeM3mn/UmaXIZ6YRqNPHTpD3/
Y2tbbC8cd1/my+vryrd1V1eo6R1xTY2jc/HYdHYeu/jdyN/
vu3RPr0S9p6hjqhWk08m3nzuHT9+zJx29f//p8nLoTQ+dGb9rUfjex1yWfmJ6evO66edx861a4/
fZ8GKDdezvdz/
+X1J6hXpiU8rHPCy8cPn0iA55APna6Zs3s6s1et6nsJ9Bo5C3sdhepmTMn97Lv5o8LSZPLUJcKtndv3s
3ermObnd2k8hjqUsEGBvIu9jVrWtsWLPC8d6k0hro0C0zHXvaSJp+hLp58Etavbz3e3t+ft/
Tcmpu5VqzwTAVpNjHUxeOPwy23wOLFrW1Ll9pJbiY777xuVyBpKhnqhZnoqUiLFsHZZ09uLZKkqWWoF+
b5z+92BZKkbpnT7QIkSdLkMNQlSSqEu99niaGRxX74w9YOcdu3d6cmSdLkMtRnkV274NZb25+i1u3LfU
qSDp2hPoscOAAnn9x69S1JUhk8pi5JUiEMdUmSCmGoS5JUCENdNdkqRCGOqSJBXCUJ8lenry+O4LF3a7Ek
lSXTylbZZYvTrfpIlqNPLleNevh6eeam3v7fV0SanbDHVJHdu9GzZuhB07hk/h/78/Xbr/
ggu7UJSlz97ukju3bB3v2wJw5w29PPPQUPNDt6iS5pS5pXBYvzlvlzbZtg507u1OPpB9zS11SR4Y6W86
f3+1KJI3GLXVJHbGzTT9uaUuUnSVIhDHVJkgphqEuSVAhDXVJkgphqEuSVAhDXVJkgphqEuSVAhDXVJkgphrokSYUw1CVJKoS
hLklSIQx1SZIKYahlbZZYvTrfpIlqNPLleNevh6eeam3v7fV0SanbDHVJHdu9GzZuhB07hk/h/78/Xbr/
ggu7UJSlz97ukju3bB3v2wJw5w29PPQUPNDt6iS5pS5pXBYvzlvlzbZtg507u1OPpB9zS11SR4Y6W86
f3+1KJI3GLXVJHbGzpTT9uaUuSVIhDHVJkgphqEuSVAhDXZKkQhjqkiQVwlCXJKkQhrokSYUw1CVJKoS
hLklSIQx1SZIKYahLklQIQ12SpEIY6pIkFcJQlySpEIa6JEmFMNQlSSqEoS5JUiEMdUmSCmGoS5JUiHn
dLkDSzNdo5Nv997dv7+2FlSuntiZpNjLUJR2ylKC/H/H/
7931vb9uyB5z4XXvziqa9Lmm0MdUmTor8fXvrS1u3d3e/d29PdJs5DF1SZIKYahLOmRLl8KiRd2uQp
K73yUdshNPzDdJ3eWuiRJhTDUJJJkqhKEuSVIhDHVJkgphqEuSVAhDXVJkgphqEuSVAhDXVJkgphqEuSVAhDXVJkgphrqkrn
jiiSd4yUtewqOPPtrtUqStRiGOqSuKDH/wg3/3t3n3u9/9/FPvGN3bw4Q//
d7VKkYtQa6hFxUUTcFxHrI+Jdo8zzuoi4Luioh/rrMeSVNv50547DH43vd+fPvGN3bw4Q//
HQcOHOALX/
gi9913X7fLLlIpQ29jvETEX+AjwM8Am4OaIWtSurptpnlOBwEdvMvSilti4ij6qpHUnc0GvDIIzAw8ONp11
77d+zfnwDYt28v73znOO7nmmmu6VKFUjjq31M8D1qeUftP3t9zKDxcvSO4xCB7DxB4d0L39GaCZaJHnDzK6rs8C
CcfXa+nXHGHr761b9g374GAPv37+cb3/
gG69at63KV0sxXZ6gfDzxQ+f5Qc9vqdPsc19ObRRzcPdpsYDR3R6HsadtVW1oqV1IdFi0aflNFYq3fbTm2oqV1IdFi0aflnWa6
+9msHBwWHzDAMM8Na3vnWYRK5tpacTjecCpwIXR8SKlv+vUYqYcyX37kpBcqqqqT
7Pf36+Qd4q//jHr6C/f9eweVJK3HnnnXztz2sd1/
rQoVSOeoM9U3AgsrPywAdw3bS4y/MnJ2MHcS3zLmlfSw4D8rymzybCD2ZbKlxKzuSuZfefr8z4zqvf1IcsH+nFPzqbzwBPcqxFLdEuzq
cmMLqpLLUGeo3A6dGxLMiYgFwCbB2xDxfBl4KSRE1/60pe
msCqpLLWFekppEHgLcB1wD/DDlNLaiDg3IpqbNrlwDSKJ3jL8MzOpy+rkKpPf9lt7+gGvHmLo09J/b4TA2idJ/ozwJYzB7tKGmnxpv/vRxLKwr/
D5lNJdEfG+iHFwTdeU3Ux0/Xk...

eij4bjjpr4uaaI63f3+D8AtwAurx5uALwCGuqQZbWAAbrkFDhtxkm6jAWvWGOqaWToN9ZNTSq+PiEsBU
kr9ERE11iVJU2L/fnjOc2DJkuHT77knt0kzSae93/
dGxCKqoWIj4mRgT21VSdIU6OmBRYvATRSVotMt9T8G/
g+wKiL+CXgRcFldRUnSVFi9Gnp7c7h3qq8PtmyBwcH27StXwrHHTk590niNGerVbvZ7gdcA55N7v/
9OSqmv5tokqXYjd7uPZetWuO++HOwjt/AHBnKoX3LJ5NUnjceYoZ5SShHx5ZTS84F/m4K
aJKnrGo0c2vff3zp9wQI45RRYtmx424MPQr/DcqmLOt39flNE/
FRK6eZaq5GkaWJwEJ5+Gu66q7VtzhxY7EDZmoY6DfWXAr8ZEQ8Du8m74FNK6ay6CpOkbhochO3b4bnPb
d++aNHU1iN1otNQf0WtVUjSNDWeTnRStx001CNiIfCbwCnAncBVKaVR+nxKUjlWroSUul2FND5jbal/
CtgH/Dt5a/0M4HfqLkqSuu300/OtJH19ufd+O729+YeMZraxQv2MlNJzASLiKuD79ZckSarDxo35NrI/
QH9/PrXvpS91IJ6ZbqxQf+ZKbCmlClMlWBb4319OdR/4ieGT9+2DZ54Al78Ypg/v/
4aVZ+xQv0nI2JHdT+ARdXjod7vy0b/r5KkqfbUU7BhQ+sFaiD36F+5ElasGD59+/
bRfwhoZjloqKeU5k5VIZKkQzc4mEN99erWtjlz4IQTpr4mTZ1OT2mTJE0TfX2waVP70et2787HxcfT6a
3RyMu6//7W3e/bt8POne2H0z38cDj55PxjQdODoS5JM8zbAAA3n8+Xlt/
ooffvj4l9nfDzfd1NpRrtHIt97e4dP37Mk/AI46CpYvH//
zqR6GuiRNU1u25I5tIzUaMHdu7vC2dOnkPNeePXDaabBw4fDp3/52bjvnnOHT77knd7Bbv761hp074cC
B9lv3vb1w9NGTU7MaGeqSNE1t3Ajf/W5r0EzAIOzAULJud5hq4rP29e670o/
YkfcmjNyV39+fB/Q599z2hwcVpBmx2SjPKvqY9Y9IyLOAVallK6psTvSkZ5pjJgD/
A1w2ZgLirgcuBzg6KO9xI8kKZ/
bvmMH3HZb+8u5HnnkLsqXJ2hvglY1fT4BGBz0+OlwJnA9ZF7OhwDrIZ2IV6WUhl24Vmxxw9s
eeCCf8z7b1BbqKaXBiHgLcsC6ltLau5+/hESumuMuxKVJ2PFPT+5cd9JJcMwxwb9s
wZw7MnTu8bWAAVq0y1CVJmjZ6evIW97Jl+X6zRYtyqK9Y0XoFt/
vvh+3bp67OiTLUJUmzxurVMGtAaZACyJBkzLkPrfvX/QvjMfPCm1EhAQoCfMyVEhWO+XQGqtMRXv6t
tfzQbQ9Al/QIRDBaDcW3R8jRkaKBsH8cAQvC9aKzKrdKqZDPFcPxKQS9YdIkasfXFHu3bvT/
iVhb3Ihg5Xim4PUSqyk6uJCtQlpSFjBoS1xRPqbCIHgChsSa4RrEmgsTO3vrGcqr+vCKFFyeyQsC9N
/ezoCsTGgs1S8KMJAr59kkf2Pw8mm6/

UmSCmGoS5JUCENdkqRCGOqSJBXCUJckqRCGuiRJhTDUJUkqhKEuSVIhDHVJkgphqEuSVAhDXZKkQhjqk
iQVwlCXJKkQhrokSYUw1CVJKoShLklSIQx1SZIKYahLklQIQ12SpEIY6pIkFcJQlySpEIa6JEmFMNQlS
SqEoS5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBXCUJckqRC1hnpEXBQR90XE+oh4V5v2wyLic1X79yJiT
Z31SJJUstpCPSLmAh8BXgGcDVwSaAh8BXgGcAVwaEWeMmO2NwLaU0inA3wD/
ta56JEkqXZ1b6ucB61NKD6aU9gKfBV49Yp5XA5+q7n8n8reERI01SZJUrHk1Lvt4YGPT403AC0abJ6U0
GBHbgSOAvoMteM8Ma/0SwgwxpbW30GfBV49Yp5XA5+q7n8ReFlERI01SZJUrHk1Lvt4YGPT403AC0abJ6U0
GBHbgSOAvoMteM8eGBycxEolSZomUpr4/60z1NttcY8stZN5iIjLgcurR3tf8pKlDxxqcRrNvsNh/
rZuV1E213G9XL/
1cx3Xa+8K2Lu5acKJnf7POkN9E7Cq6fEJwOZR5tkUEfOA5cDWkQtKKV0JXAkQEetS2nluLRWrWr8Drt8
auY7r5fqtn+u4Xnn9pgmt3zqPqd8MnBoRz4qIBcAlwNoR86wHSZJmr9q21Ktj5G8
BrgPmAp9IKd0VEe8D1qWU1gJXAVdHxHryFvolddUjSUFkS72ykkoTaNz/
APjFcS72ykkoTaNz/
dbPdVwv12/9XMf1mvD6Dfd2S5JUBoeJlSSpENM21B1itl4drN93RMTdEfGDiPh6RDR8SoWysdZx03yvj
YgUEfYmHodO1m9EvK76HN8VEf881TOZB38jVgdEd+MiNuqvxMXd6POmSoiPhERT0bED0dpj4j4UX6/
0FEPK+jBaeUpt2N3LHuAeAkYAFwB3DGiHl+G/hYdf8rnum3Dpcvy8FFlf3f8v1O/
nruJpvKXADcBNwbrfrnim3Dj/DpwK3AYdXj4/qdt0z5dbh+r0S+K3q/hnAw92ueybdgAuA5wE6
HKX9YuAr5PFczge+18lyp+uWukmvM9ZtS+mZKqVE9vIk8zoA618lnGOD9wF8CA1NZXAE6Wb9vAj6S
UtoGkFJ6coprnMk6Wb8JWFbdX07rOCQ6iJTSDbQZI6XJq4FPp+wmYEVEHDvWcpRq4FPp+wmYEVEHDvWcqdrqLcbYvb40eZJKQ0C
Q0PMamydrN9mbyT/
YlTnxlzHEXEOsCqldM1UFlaITj7j7DpwGnRcR3IuKmiLhoyqqb4iITeSznN46NaXNGuP9Ow3U
fErbIZi0IWbVVsfrLiLeAJwLvKTWispz0HUcEXPIVya8bKoKkkwnn+F55F3NP17RJyZUnq65tpK
0Mn6vRT4ZErpryLiheQJMKR2ov7xZYUIZN1231MczxCwHG2JWbXWyfomIlwN/
BLwqpbRnimorxVjpeHev0b8S/
ppT2pZQeAu4jjh7zG1sn6fQmAu4jh7zG1sn6fSPweYCU0o3AQmDllFkFQ3O3T0d3qk6RrqDjFbrzHXb7Vr+OPkQPdY5PgddB2n
lLanlFamlNaklNaQ+y28KqW0rjvzjid/I34MrnDJxGxkrw7/
sEprXLm6mT9PgK8DCAink8Ak0O9S1TWmXZ1gK/UvWCPx/YnlJ6bKz/NC13vLmKHG+vDYgE1SCH-v3vw9FLgC9U/
Q8fSSm9qmtFzzAdrmNNUIfr9zrgZyPibmA/8Psppae6V/XM0bN9fOvd8kOE6UPVo5jT2H6YNUV/
Hw5m4XJGlyGOrS7HYjTReJiIjfj4ibq+s3/0nT9F+ppt0REVdX034ADVO2XU/
7axFxdBfql9Rkw4oJ6l+ETGXPMznVdXjnyWPjX4e+WISayPiAuAp8jUAXpRS6ouI3moR3wbOTymliHgT
N4L+QRxmT1CWGujT7LIqI24E1wC3AV6vpP1vdbqseLyGHD/7H/
E8CX0wp9QGklPtYnXOHQC8LnqGs8LgIempHpJo3L3uzT79KeUzgZOJDH1AP68Ot5+dkrpLJTSVdX0d
uNJ/w/gwyml5wL/mXxBD0ldkZKhLs1RKaTvwNuDhN3l
+eIdvx4RSwAi4vOvC6iDiimj60+3058Gh1/1eR1HXufpdmsZTSbREFxB3BJSunq6hHPnnkaSGsn
vKG6OtcHgG9FxH7y7vnLV/F7lHzp2Gd14zVI+jGv0iZJUiHc/
S5JUiEMdUmSCmGoS5JUCENdkqRCGOqSJBXCUJckqRCGuiRJhTDUJUkqxP8P8TOlB0VD9BUAAAAASUVOR
K5CYII=\n",
      "text/plain": [
       "<Figure size 576x576 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "precision_recall_threshold(p, r, thresholds, t=mean_t)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },

```
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "#
--------------------------------------------------------------------------------
----------"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "n_jobs = -1\n",
    "models= [LogisticRegression(random_state = seed, n_jobs = n_jobs),\n",
    "         MultinomialNB(),\n",
    "         BernoulliNB()]#,\n",
    "         '''\n",
    "         GaussianNB(),\n",
    "         #PassiveAggressiveClassifier(random_state=seed, n_jobs = n_jobs),
# parece ser bom\n",
    "         #NuSVC(random_state=seed, gamma='scale',probability=True), # bom\
n",
    "         LabelPropagation(n_jobs = n_jobs, alpha = 0, kernel='knn'),\n",
    "         LabelSpreading(n_jobs = n_jobs, kernel = 'knn'),\n",
    "         RandomForestClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         SGDClassifier(random_state = seed,n_jobs = n_jobs,loss='log'),\n",
    "         DecisionTreeClassifier(random_state = seed),\n",
    "         XGBClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         GradientBoostingClassifier(random_state = seed),\n",
    "         #RidgeClassifier(random_state = seed),\n",
    "         SVC(random_state = seed,kernel='linear',probability=True),\n",
    "         KNeighborsClassifier(n_jobs = n_jobs),\n",
    "         #NearestCentroid(),\n",
    "         QuadraticDiscriminantAnalysis(),\n",
    "         LinearDiscriminantAnalysis()]\n",
    "    '''\n",
    "#RadiusNeighborsClassifier(n_jobs=-1, radius=10, weights ='uniform')
#podre\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 131,
   "metadata": {},
   "outputs": [],
   "source": [
```

```
    "n_splits = 5\n",
    "CV = StratifiedKFold(n_splits=n_splits, 
random_state=seed)#KFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = ['accuracy','precision','recall','f1'] #accuracy #precision 
#recall #f1\n",
    "#entries = []\n",
    "\n",
    "x = x_train\n",
    "y = y_train \n",
    "\n",
    "from imblearn.pipeline import make_pipeline, Pipeline\n",
    "smote_ = SMOTE(random_state=seed)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 132,
   "metadata": {
    "scrolled": false
   },
   "outputs": [
    {
     "data": {
      "image/png": 
"iVBORw0KGgoAAAANSUhEUgAABZgAAAHwCAYAAAArRQrgAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzs3WdgFNUaxvH/
7KYXQgqhJtSgICBNujQpAQsWREBFEQRE9OoVEUEUQQEFuTYsgIigKIqoqEhTAZVeBEV6Dy0JKZCe7M79
EIyEqLRkZ0me36fkTHs2gezuu2feY5imiYiIiIiIiIiIiIjIxbJZHUBERERERERERERErkwqMIuIiIiI
iIiIiIjIJVGBWUREREREREREQuiQrMIiIiIiIiIiInJJVGAWEREREREREREREREREREERkUuiArOIiIiIiI
iIiIXBIVmEVEREREREREREREETkkqjALCIiIiIiIiKXRAVmEREREREREREREREREREEbkkHlYHKESm1QFERERE
REREREignjQnbSDGYREREREREREREREREuSQqMIuIiIiIiIiIjIJVGBWUREREREREREREREREREREQuiQrMIiIi
IiIiIiIiInJJVGAWEREREREREREREREREREERkUuiArOIiIiIiIiIiIiIiIXBIVmEVEREREREREREREREREREETkkqjALCIiIi
IiIiIiKXRAVmEREREREREREREREREREREEbkkKjCLiIiIiIiIiIiIiIiIyCXxKMqTG4YRDbwG4TG4YRDbwG2IHppmlOOGd7ZWAGUAZI
AO4xTTPmzLb7gGfO7PaZofFVWEREREREREEbkynDx5kldfn05sfAoethyyMuJ56aWEAkJsTqaiEiJ
U2QzmA3DsANGC5AbaCXYRi1z9ltHTBj2z9ltEjDLNM16wBjgrjwQ2QC/ZwZKZAAE+A5wzCC/
+160dHRhfsARERERERERERERMTtJCYm8uBDz+Bfvhf1Wo6i6oWvmvAaEWaowS1Ghw3fJRR8fkdDAwKuH7KKFmb
2d1NBGREqkoW2Q0AfaYprnPM0M0s4BOg2zn71Aa+P/
P1j2dt7wwsNU0zwTTNRGAp8K8V5Pj4+EILLiIiIiKulZ2dzaS33+O/
wyk12MPseLnn6yOJCJy2RwOB2++N5M7hwyn5yNP8d2yH6yOVCy8+db7NG7zJIGlygkelwd3knT7gGEboxR6YRgDDMPYYBjqGhGEDDMPGYYBjGhri4uEILLi
IiIiKuY5omdw0ZwBzfWI7dWp8DN9Xh8W8/YWYYBjGhri4uEILLi
IiIiKuY5omdw0ZwBzfWI7dWp8DN9Xh8W8/YMbHH1kdTUTksvR9YiSzUioS1+VJjnUayrM/
7GXyu+9bHeuKd/
R4IkHB5fON2T088fYrxdp9LJBsk3hzCJmyE39c92rvU29QGR4U3ycH9j2r2Fqf6/Dcgw3RMEjkz/R2oBfJPpyDf6tT7DmpyLiJpUffg/WxpumAa9iOWVWk5RATTe1htzvmRkN17SHiuR2MhiYFwrkyMeqpMSXQMxAAmTyOY2uTqj3gApvT0uJiCY5rUqpTx6mimncAvdmwrUB/9b1ua2auT2ZAsUIAVxuGASiYhG6a0G6Y8GBGcBtnAYG3dnKW8YLNQmmIbOyWy6A4
wk+PFXxfYKUjnRXMp8ZsbGDbXdkm5gRrkbi/TdlOOYTB94gIPxzOgAYWdQX4Q1IGcfO/f5nJRTjzNgAkyCWG
pJXmf/qLrYZ/7bUDbjN4eiHRZq11OLWA/+yPdMMuQUI9ZZw7HTqQeVXZRBeHmKJSKgR7KRQoIXBLuiCmP3R3V
LLdxRuq9mjcX3kWzeiJPI+dePi4q5kt+QXZubBRpGYS2M6n/hhtz1RU4LrfK0gNf47Gjip5dZnVQ8wRfoxnl0N6PxZ
nhbklgFJUwHs7gGLzzsExRDDVaPGSYaWUO3ToGhTrrZFhRr2hJHLhlYhH/Q7bCKFgU8IvKrs4wsJ0Afwjt7/3DnpWB
0Akcx4YTqQN9Ejfc6PtjlEERiCXFeICONwVhyaV3RTZjPyuPAa54PnHNGpkt2ZPdhbdhEgnlRZBWXEXEwqm5LxSb8b
8M2BbNPcydLbXaH8YUE+0cAZgL3dcEYfTD+7TvA==
```

AndcXGHf6eGKaJoZxQXf1iYiIiIjIJVKB2U2cr2dUTk4Os6ZOY+
+mLZg2G9E976R1+3YWJBUREZHiLvZQDLEvz7I6xgXJSTxJZp0IAq+NyjeedfA4uybOtiiViIgUho0bNz
Pjg3nk5BhUq1qWRx7uh5+fn9WxRETkHCowu4k5c+awbds25syZw5AhQwpsHz5wMHcGlufuqEY4nU4+mj
mXmH376d3/
AQvSioiISHEWHlnJkh7MlyLK4WDhmzPwjgjHKyQI0zRJWrqeWm2ac1WbZi7NkqgezCIihea775bx0ecb
adL6KewensSd2McDDw7lww9ex8NDpQwREXdiszqA5M5eXrp0KaZpsmTJEhISEvJt37RxI/
UcntQvn7tius1m495rGrF58TIcDocVkUVERETcgs1up2P/
uzEWbyZhxkKSp39LzdBy1HJxcVlERArXR59+R/P2j2L38ASgTNlqRFzVnS++/
OaCjjdNk+3bt7NlyxacTmdRRhURKfH0sZ8bmDNnTt4TntPpLDCLefPqtbQqW6nAcRFefsTHx1O2bFmXZ
RURERFxN97+frTo2c3qGCIiUogcTp8CY5HVmrJm7cvc2f3Wfz123779DH/
mFYLKNMDu4U38kWkMH9qPxo21+KuISFFQgdkN/Pjjj+Tk5AC5vZZ/
+OGHfAXm2vWv5dfNc6kSEpbvuKNZ6X/br1lEREkSkJEmKOU5OVhahVSph2HSDnohcvpW/
rObdT78i0wnNrq7Kow/2xcvLy+pYlnt6xCji4xPOv+MFOnxo7z9ui0/2pFXn/GOH9m1gxfKlREev/
dfzHovL4YH/fI6HR+7vzGx4O/0Hdcd0JOHpASHBAdgu4PkiIrL6+R/
EBQgLC2H8uLGFci4RExekArMbaNeuHYsXLyYnJwcPDw/at8/f87BZyxZ8/
Na7XJsQT9UzReZv9/5BtRZN8PT0tCKyiIiIiOVOx8Xz09wF2GtUAC9Pshcuo1GntpSvWc3qaCJyBXv/
k3m8te4wfjf8F8Nu57NDO1g1eCifT3sNwzCsjmep+PgEuvV8wSXXWv3zd6xZ8T5NW9+PYRicPhXH5jWz
GfnCF3h6/nOx/9jR/az8aU1ecRnAMAxa3jAIpzOHgMBQNq2eTZ/
+wygdXMYVD4WvPnnGJdcREbGKCsxuoHfv3ixduhTI7a/
cu3fvfNsNw+DlGVN5e+IrxG9fi8OAZl06MaBHDyviioiIiLiFnz/
9mtCBt2Dzyv3A3WzfiA1T5tO1aiR2T73MFZGLZ5oms5f8gv8dfxUEfSOvJuZEE35c+TPt21xvYTrrHT6
0lzdf7uWy6yWfSuX3jfPx9PIj8WQM/
gHBjBnZG097BiHBgX97TGpqKhVqFGyhYffwxG56UKVGE8pVrMXEF+6gfFn/
on4IIiIlgl55u4HQ0FA6duzIwoUL6dSp09+2vfDx8eHxUSMtSCciIiLifk7HncSoFJZXXIbcD+X9WtXl
0K/bqHrdtRamE5ErVWpqKmnepTm3+69HzetYuX5RiS8wR0RWd9kM5rPNmfU/
wiMGE1WrDaZpsm3zQrLT93HrHQ8W2Nc0Td58dSROpwObzZ439vumhXTrlZvdxzeQyjWaM+Chp12SXzOY
RaS4U4HZTfTu3ZuDBw8WmL0sIiIi4mrhYWHEfvKD1THycTgc7Dq4n7iMFLw9PCnnE4BZPbjgjjYbp1ds
IXHvSZdnDA8LO/
9OInJJho54hrhC7P37T0zTJO5gAhHnjKfu3MDPm1Zw366df3vcfQMGF3m2S1EmLIRJ4wqvIBwWFuLyYm
lqagqp2RFE1WoD5H6YWKfhjXw5ZyifzRqW1xv78KG9eT2TS/mdZvZbval3XQ/sHt7s/
H05jVv0yNc242TcPpc9lrAwrZ0kIsWbCsxuIjQ0lEmTJlkdQ0RERIRJ4yZYHSEf0zS5dcB9OO5pQ4XK5
TAdDk4v2YjfjhhMhwPDbs/bN2DzQb6b/
yU+PufOPxSRK1lcfALl7xrmkmvVWPIFx37+kuCW3TAMg8z4o2RuXkqjRyf97UKi5V2S6tIcm/
tyoZ7PioXqFn63iGVrCvZcjqrVmgd716RBgwYAREdHM23qlLztTqeTDRs2sG/
ffhJjQ6ka1SRv294dP3L/vbfwYL8+Rf8ARERKABWYRURERMStLf5+GYdrhuBTuRwAht2OT5cmOA7G4/
XBcpJqlcPpZSfg9yOM7jNQxWURuSwNOt2G/
9rl7J49GtPuQenAALoOGPa3xWW5PNHR0efdJz09neDy7ah+VfN847v++ImhQ6fmW/
j+n86XnJzKjNc2UDq0Mqmn48hOP862UH8+/2zOv1570aJFF/AoRETyW7tqDV/
NnIuRY1KpTnUeeHgA3t7eVscqUiowi4iIiIhbW7F+DR51zr1hHZxhpVg0aTpbtmwhLSOdVo+2zLtVWkT
kctRs2paaTdtaHaPYu9AC7iOPjeTo4d+oEFEXgIN71tC0YUVeHPvmRV0vOzsbDw8PDMO46KwiIhdiwaf
zOfD5WkbUugUPuwc7Dx5kaN+Hef2jacX6b48KzCIiIiJSpC5khtq/iUtKIMPWlKAmtfONH9+
+h1tvvTXvxfql3giuGWoiIu7t1VfG8MaUaWxcPh/
DgKbX1eLBfsMv+jxnz3YWESlspmmyYt4iXqx7b97YVWGV6ZhSmx+WLOOGzh0tTFe0VGAWERERkSJ1uQV
cp9NJlwd6E181Ce8ypTFNk6yffuPp+wYy+L4HCimliIi4K7vdzmOPDrI6hojIv8rOziYwp+DddM3K1Wb
WqvUqMIuIiIiIWMVms/H5m9N5/
tVJbF+5Hi9s3Nv1Vm7repPV0USkBDJNk5htm4jZvY2ykdWp2qC5+jOLiBQTl3PnnWma+MVmQ4P84+uOb
GP+tgUsW7vyMtO57513KjCLiIiIiNsLCAhg4jOjrY4hIiWcIyebRVMnYlzdnIDmPdi17ze2vjGGrgOex
MvX3+p4IiJymS63gPvxe7OYu2I5PWq0wTAMjp6K59vTW/lpzS/
Y7fZCSul+VGAWERERERERuQBbln2FT9ve+FfLXWzOq2F7MivXYs1XH9G65wCL04mIiNV69evD4vCFPD/
/M/7Y8jutbryBiR9MKdbFZQDdxyMiIiIiIiJyAU4cickrLv/JO7Q8p1LTLEokIiLupvPNXZnw/
htkhXszbOwzBAQEWB2pyGkGs4iIiIiISDFimiYzP/
6E79asAxPu7NCOO7vdYnWsy3Li0D5OTLR+kbfTJ9Mo7cjBsP/1VtrpdJK0YwPrx/
XD09MTgJycHGISU8n0LoWRnUY5H4NS/n5WxRaREuTIkSO898p0shKz8An1od8T/
SlfvrzVsaSYU4FZRERERESkGHl01HOsCa6I7029MU2Tl9avYuuuSYx9cqjV0S5Z2chqlL9rmNUxKLNjC
xuXzCK8ywMAJG9fz/HFsyh1XVeOZKbim3WaNnf157vpkwm+
+zF8wiNwZmdy4qu3iaxXlyr1mrg077G5L7v0eiJirYMHDjJ58EServUkQRVKkZiexLh+LzDivWdUZJYi
pQKzlEgOh4PZ095m14ZfMO0edOp+L+06drI6loiIiIjIZTl69CjrUjLxbd0QAMMw8GvSku/
nf8iTp05RqlQpixNe2SpefS3J8SfYNXMUTu9SpMQepuZ/XscwDAAy44/
yzZtjKX3TQHzCIwCweXpT7o7/sHXmKJcXmEWkZJnxynuMumY4/
l65i44G+5ZmZK1hTJ04lWcnP2dxOinO1INZSqQRgx/
gqt1f81xtB89GZXDw4wnMeucNq2OJiIiIiFyWX7dsIaNyVIHxjIqV2bNnjwWJip/
arTrRbeCTVAwOIOLOx/
KKywDeYRXI9PTDv1q9fMcYhoHTWy0yRKRoOZJz8orLfwryKUVWQpZFiaSkKNIZzIZhRAOvAXZgummaE8
7ZHgl8AJQ+s89w0zQXGoZRBdgO7Dyz6xrTNK1vuCXFwtYtW6iZdZiGFcMAsNkMetUJ49kfF5Ddb1Be3z

QRERERkcIWHR1dpOdPT08nrfo1BF1zbb7xpF83MPTHb/
Dy8vrX4xctWlSU8S5ZmbCQQm33cOLQvss+x7G4JEKadf/bbSn7fyewev4ic/r+3/
n1AvtIl42sdtn5IPfnJiJXhsJ4fkg/
nEpG+Ufw8fDJG0vNSmXF+hWXfX53fX4Q91BkBWbDMOzAFKAjEAOsNwxjgWmaf5y12zPAp6Zpvm0YRm1g
IVDlzLa9pmnWL6p8UnL9unYVzcp5Fxiv7OMgNjaWihUrWpBKRERERERERqCwnyDHh0d/bfn6/
vEk2zbvwefqjUAyNj+G7fUvZpXR88ptGu72qRxL1gdoYD9+/fT45WP8eoyMG/
MmZ1Jw4pBJK2eQ3Z4BJ6BwZhOJ+k/zuaVp/9D95u7WphYRNxZYTw/7NqxiwlDJ/P0NU/g7eFNRk4GE/
54hVlfz6ZqtaqFkFLk7xXlDOYmwB7TNPcBGIbxCdANOLvAbAJ/
NgELAo4WYR6XM02TBfM+Z83iZYDJdR3ac9tdPfLdQiWF42I+iUs5lUyfyiY1ygTmG1+9+whf9u2LzVb4
nWP0SZ+IiIiIuMr0lycw6e13+OWrDzGAjo2u5eFHR1kdq9ipWrUqd9UO57OvX8dZrxNmciz+v33H6+NH
EhAQwOj/
vcWBxDR8yGFQj27c0KaV1ZFFpJireXVNHpjYn27de3B9g1YQaDDo1YdVXJYiZ5imWTQnNozuQLRpmv3P
fH8v0NQ0zSFn7VMeWAIEA/
5AB9M0N55pkbEN2AWcAp4xTfOnf7te48aNzQ0bNhTFQ7lkk0ePocaxZDpVvQqAHw7u5rcQH556cazFye
Sx+
+5kQKU0qofm9iZavCeRmKrtGPzkiH+cCSIiIiIi4k70utU9nDhxgq+XfE+F8DJ07tAeu91udSQRKeH0/
OAeisnv4YJmyRblDOa/
C3BuNbsXMNM0zVcMw2gOzDYMow5wDIg0TfOkYRiNgC8Nw7jGNM1T+S5gGAOAAQCRkZGF/
wguQ0JCAmnb99C54fV5YzdUqcmvm34mNjaW8PBwC9PJS1M/
5J1XJvDh1q04bXau69CHwb3vtTqWANt+38rs6ZNxZJ4mPOJqBj0ynMDAwPMfKIUiKSmJN98YR+yJvXh7
B/HgwKeoWfMqq2OJiIjIWVavW8eUjz9l5+k0xr32Bv8d+CA+Pj7nP1CKRNmyZel/b2+rY4iI/
K2jR48yY/
J7ZJzMwDvEm35P9KdChQpWx5JipigLzDFAxFnfV6JgC4x+QDSAaZqrDcPwAcJM04wFMs+MbzQMYy9QE8
g3Rdk0zanAVMidwVwUD+JS7dy5k2sDQwuMNwwKY/v27SowW8zb25v/jHjO6hhyjlU/L+fzd//
Lva098PaycezkPv7TfzVvfbBQb5pc4PTp0wwedAs3djhNo9peZGQeZMKLvRnyn3do2PA6q+OJiIgI8Ol
XC3h5+Rp8O91BRJe7+OLwQVYNHsKC6VOLpNWbiIhcuWJiYnh5wASerjWU0hWCSMpIZnz/
cTz57lNERESc/wQiF6goC8zrgSjDMKoCR4CewLkf6x4CbgBmGoZRC/
AB4gzDKAMkmKbpMAyjGhAFXP4yv8Co4U+TEB9fGKcCYG/
M4b8dz8zMpJHNj05RtfONrziwm9XPPeeyYln1SoX3ByMkLIyxE8YX2vlEzvXJ+68w6AbPvD7l5UO9ufX
aBD6aNZV+Ax61OF3x99701+nQ+hQhwbmLYPp427njJpNp747j7Xe/
yNvvxIkTvPnGCyQnHcXfP4yHHh7pdneRiIiIFFfTvl6IX49+ed/
7RlTm+NUN+GbxYm7p0sXCZCIi4m7emzSdZ2oPI9A7967g0j5BPFN7GG9Nms7o1563OF3xtn//
flKPJDCq3+P4ly1Nv8cHU6ZMGatjFZkiKzCbppljGMYQYDFgB2aYprnNMIwxwAbTNBcATwDTDMN4nNz2
GfebpmkahtEaGGMYRg7gAAaZpplQGLkS4uN5Ifq2wjjVeU2e+xHrYvbTpFJuM/
VNRw+Cvx9fDBjkkusXtmcWfXH+nUQug92RXGARzKiKPqzavMmiRCXLoYPb6Xi9d74xm83ANJPzvo+Nje
Xx/3Tj1uhsAgM9SEs7yMjht/
P8C59QrVoNV0cWEREpURwOB6dsnnifM+59dV1Wrl+hArOIiOSTnZhFYKX8LScDvQPJicu2KFHJsHPHTq
YNe4kvuo0lwNuP+NQkRj/wX8bMfJXQ0ILdDoqDopzBjGmaC4GF54w9e9bXfwAt/
+a4z4HPizKbKzx2Zy/m/rCUBStyfwSRlSoxtNc9FqeSv7Nnz25mv/
kKzrRkTh89QEJCAiEhIVbHuuJER0df1vGZJw9Au/
y9oPYfy+DHlesu+9xAcWiuf0Eu9WcVH7efBrX9CAv1yhszTZPNm/fknTM+dg/
DHg0iMDD36cPPz84dNznpcWcHwspcfUHXKSm/BxERKR6GjhhBbPxJq2Pkid9/
mIrnjKXu2Mb6tWvpM2CgJZn+TnhYKJPGjbM6hohIkRn11CgS3Oj54VwP9xvM/
m37SCubhp+nX954enY6m7Zt4uF+gy3JFRIWytiXxlpybVeZOfkdnm1wD14engCE+ZdmWO3uzHjtHZ4cM
9LidEWjSAvMJZ3NZqNXh85Wx5Dz+H3rFj4c/QjDrgvD39uDByvWZOQD3Rk/
cz6lS5e2Ot4V5XILh0sXfc0nnz9Hjxae2GwGp1Jz+GxDAN+vXERAQEAhpSz+LvX3kJiYyMODunLHzdn4
+dpxOExmzz3BpMkfcP317QB4dMgdBAbG5DvO29tGsybX8Obb3152dhEREXcTG3+SsLvutzpGnqjlP3D4
+4WEtO+CYRhknYwn9adltP3PcGx2u9Xx8sTOnWl1BBGRIpUQf5Ln24yyOsa/
OnD1QV78+iWebT4Cbw9vMnMyeXHtSzx/
13NULVvFkkzPrSjexWUAe7ojr7j8p7IBIZw65L4fSFwuFZgtkJWdjWEYJKacZvnmjZQODKRd/
UZ4eujXYYXZr7/Mcy3KYrfltmYoG+jDE3WdfPDWa1oI0MU6Rt+Mf0Ap3vvwTbZtWUfT1l15+e0xxb64/
PTTQ4mPj7U6BgCGRwVee3c7vt5ZpGfYcFKBWbPeZ9as9wHYtWMHbZr64uf31xvY7GwnW3/
bw4MP9nF53rCwcMaPn+Ty64qIiFildtv2+Gxcz56ZU3Da7AT6+dJ5wENuVVwWERH3UKVsZXre1JMXfnw
Zu8OGw+6kR9celhWXi8Ko4SPdbib5wR07yal+Gx72v+p8CWmn2Pj7rzzc3z3a5oaEhTJ2wouFdj5VNF3
o2Ml43vliHkGGB0lpqcQknmRYm86kJacz6p03GXhHD6qWr3D+E0mh8sw
6hd3mm28sItiP+N2Fsq6kXKQWrdrQolUboqOjGfPS21bHcYn4+Fi631LK6hhnlALK/
+PW2GaNmDNvJffeVQZvbxvZ2U4+/jyee3o1o3Kk6x/
DvAXuUZgXERFxpWqNrqNao+usjiEiIleAauWqMrzXk1bHKDIJ8ScZ02GA1THy2VZjF+N/
+JDhLe7B0+5BWlYG41bNYmKfpygX7B4L/T27bGqhnk8FZhcxTZNXP/
6Qlzt0w9czt79pfOppJq9YzLiu3WkaUZWnv/
6SMQOs6YHjaqOGDyMhPs5l1zNNk6OHD+I8lXtNe1A45StFYhgGh3bsw1mvNjbbX4vLnTidwcat+3i4f1
+XZbxQIWFlGDvhZatjSAkVHl6Km2+6nk+/
+hWDbJxOO+3aNqNyZPFcqEBEpKRav2kTr384i3SHg2srV2HoQ4Px9fU9/4EiIiIiJdw1VWpi62Dw/
PLZeJgGpofBg937uE1xuSiowOwi63dtp2OVmnnFZYAw/

```
0DKlyrN8dPJlAsMIszTh7TMDPy8fSxM6hoJ8XE81+nCFgQrDG/
NX8qN1UxuiKoFwJKdcWxJPcrAbjfwa1U/XlqxmidbV8PDbiMlM4fxK/Yzqe+thJVyv9YMzy/
ZYXUEKeHKlwvivnvaWB1DRESKyFeLvmPsd9/
g1aUDNg8PDhw9xi+DBvLtjPexqw2DiIiIyHnVioyiVp8oq2O4TIkrMO+NOUyv6a+7/
LqJ8SeZ2LRDgfFQ/
wCS09MoFxjEocST9P3gHWw2m8vzFWdxySl4ZiTRoWa1vLHOV4ezefk+ElPSqB8ViaennWdXrMfDcIDdm
0F33uSWxWURERGRovbW/
Hn49Lg173ufCuWJa1iHz776kp6332FhMhERERH3cuD4YTbs3EpEeAWaXF0fwzDOf1AxVOIKzIsWLbLku
snJyUx84CGaRlbLN77lyGF6XtuEY6eS2HU6gZ9WbbEkX3G2K+YEjcoXLBY3KOfHnqNxXFezMtdUqcg1V
SpakE5ERETEvSRT8E2CT62rWPHDKhWYRURERMhtxfr65zMIzfGhXeWG7Nx7iBErJjCizyME+pW8CYslr
sBslaCgIOrc1ImJC5fRK6oemdnZvPLLUkr5BzJp88+khwbhUy7c6pguszcnB8L6AAAgAElEQVTmGH1mH
HPJtTIyMmjuk0i7qPw/3xX7E9m0ezveP2sxPxERESn+oqOjL2i/o+kpRJ4zlrJjN6t+/
PGCz3EprJoIIiIi/y4rK4tpb89gz7bDGHaTHvd2o1mLpqxbu55PZ32JM8egSs3yDHi4Hz4+xb/
lpwjAyt/
WUtenEjdf1QqAmmGRNK9UhykLPmJoz4EWp3M9FZhdqGff+znauRNfzPkYH58AJj49n+zsbPz8/
AgMDCzSF+zupnql8i7twTz542/ZFJNEw0qlAVh/KBHvUqHMvevK+5mrB7OIiIhcigst4H74+Txe/
X4F3u1bYxgGWYlJhK7bxKKffsbLK3c9kejoaBWERURKANM0eaTfUNpV6UvjeveR48jh62kz+ebLRThOl
OK2hsPxsHtw8MQuhvQbytTZr6vtp5QIa37byOjGffKNhfmXJict06JE1lKB2cUqVKjAw0OfsDpGifPYX
V345Ic1fP79AQAqV6rIo3c2sTaUiBvZtz+eg4fiqFolnCqVQ62OIyIiFrrnju6Eli7N9C+
+INMwqRESxug338orLouISMnx/bIfqV26HVXL5k4Q87B7cEuj/oz+8D6eu3tmXr/ZymVrUv9UNIu/
W0qXGztbGVlKmL0xB7h75giXXzc1JoHMBtn42rzzje+OO2RJHqupwCwlgs1mo3eHFlbHEHE7OTkOZs5a
TuUIBzVr+LBjx2F+XO5Jn3vaYLdr5oGISEnV5YYOdLmh4ALVIiJSsmxc8ysNI3oWGA8rVYnsnCy8PP8q
rtWt3JwVq6epwCwuVb1SFcZ0GODy6+48vJdpKxfwaJM788Y2H9vF9dc25f7oO//
lSPfw7LKphXo+FZhF5Iqx6ucVfPHxO5jOTKrXbka/gY9pNtVl+m7JFjq09SSyUikAIir6smdfGt//
uI1OHepanK54czgcTJ/5Fht+XY5heHBnt750vOHC2/
bs2rWTKdNeIj0rmfJlqvHYwyMIDg4uwsQlz5IflzFjwQdkO3NoXacFQx4YjN1utzqWiMgFi92zm/
jDBylX4ypCIiKsjiMihWTk06M4GZ9QaOc7cHjvP25LTjiF93XVaF4rf9H42MmDeHrkfy/
224G1LFyzgFUblhdaNoAqEdUL5TyhYSG8OH5soZxL5KqI6uyrcYinl79LjaAKHE1NwCvIj8G39jn/
wcWQCswickWYN/cDtv/
wKvc188JuM9h9ZA6PDVzFlBlf5N2WJRfvZHwCkZWC8o3VqObHqnWxFiUqOYb89z78avxGrRt9cDpNPv5
hOHv2beehBx8/77GbNm/ghTcG0uQWAy8fO6cS9vHAw2uY9e5CAgMDXZC++Hvj/beYcWAeXt1CMWwG7+/
6hjWPr+fj12dZHU1E5LxysrJY9t67GDVq4VMtikO//
or394tp2+cBDPVGFbninYxP4OFbXFModTqdjJsygurl6xBeuiKmabLs18+oULEiizd+TMWQ6izf+hWed
i+OJuznpub30q3jXS7JdrGmLBhldQQpZro0bUen61pzPDGO4IAg/
Lx9rY5kGRWY3UBOTg4fvDMVjsczYuBguva+i1Zt2lgdq1g7nnCK9dv3US6sNI1rVlaB0s2Zpsnyb2byc
Ie/br+KqujDkYSDrFzxPW3aXtm38B4+HMv/plhT0E1OTMM0S+X7P2CaJvsOpPG/
KXssyVQSbP1tCxm+24iKyl1l22YzqHO9Nz9+No/+9w/B09PzX49/c9o4WnS3Y7Pl/
t5KhXhRu9Np3pn+Kk8+rhfOF+rkyZOMe+tlDiQcItinNMP7P0GN6jXIyclh7uov8L4nLG9fn5pB7IiJY
eOvm2hUv6GFqUVEzm/DV58TcEsPfMpVBMC/SnVO/baJHSuXU6tte4vTiciVxGaz8d/
+zzD3m5mcSk7BYebQpEELnrx1NF//8ClLN3/K47dPyns/
8fO2hSxeuYDOrW+xOLmIa9htdiqGlrM6huVUYLaYaZoMe3AQvUpX5J5eA3E4nXz43hxi9u2nZ9/7rY5X
LM34djnmqTg6Vg9h/4EYRi1fzRN3dyM4wM/qaPIP0tLS8Pc4zbl/
supXtbFqzYorvsAcERFO91tKWXLtVWtM1m86RJNGf13/
lzXJ3HJTHRo3rGpJposxb8EpqyNcknUbfqFcDUeBcd/
QdI4fP07EeW5jzjaT8orLfwot582erTsKNWdxlpSUxC2P9SCjuz+ewT4cSj9Bz/
EPMOPxKZQJCSM9xMG5ZX7zah9+2bBKBWYRcXtJp04Tdqa4/KdSdRsSM/
MtFZhF5KL5+QbQ984hBcaPnTjKwK7P5Zus0uqarryzeKQKzFKs7Y7Zz5zFX+CHJ1lmDhUrVuK+6O4lev
KiCswWW7dmDY0NX+qVqwSA3Wbjvmsa88x3S+lxXx9suoWtUG3ZG0NIdhL3XV8NgLoVStO6WigTv1zGsH
tK3hPgyOFDORnvvq0QBvXP7V1kmiYn9p+EtmXzbd+yN52fdq5i9x7X9zgKDQvnxQmTXH7dwtaiWRTfLD
zN7E9OUDbczrETDsqXr0DrNu5fXC5sTz09lPhC/P9w5PA/
n+vU6WQim6RTplL+W6h2b42jX79+5/3bfyxxD/VvKZfvBUxyfBa/
rFxPdPSF9XGuGBF+QftdiLCwcF4a757/H/7p57H/
5EH8h9fAKzh3Frnd1wOzdzg3P3AbVwVVI87jCBW6huU7Ju23BOb8+CEL531TaPkWLVpUaOcSkaIRe+gg
sROftzrGRUlLySgwZjocpOzZwR9X2GMRkYIOHN7Lk1N6Wx2D5OOZ3NPi6QLjx+OPuEU+kaJwKu00H379
KS+1fwi7LXd9ljWHf2fW4s+5L7q7xemsowKzxTavWUfHcgVnqpWze5OUlERISIgFqYqv79dvZWSz/
LM5Svt6YWSnW5TIWifjYxl4o7v+G8uf67vl1fhu3RGirwvCMAyOJ2SyYls2owbUKTCT0xXe/dZ9C/
MX66auDcnKyiEhMY32If54eZbMRczi42Np3b0wZ5L/
+7k+nLqU+KMZhFXILXDu2HCKajUi6HhzzfOeec92P1bM20Lr20Ox2Q0y0hys+CyRAcM7EBDoUyjpL8bK
ee77/+GfCrg9/9uHA2XyF2AMDxs1m9RmyRsLePGNCcxb9z3eTXIXTsw8mkr64lhWr99bomcmiJRE4ZGV
CbvrfqtjXBTH0kXEbd1IqXqN8sZO/rCIxr3vI6LutS7JED93pkuuI1ISVYmo7rIezP9m0cov+X3/
WupUbZo3lp6ZSqXykTzat2Dh2WrqwSyF4YufFjOk0e15xWWAZhF1+HrFagtTWU8F5st0oTPF/
smppCQCa9Sne91G+cbX793Jkl69LvtNrLvOjAoJK8PzS1x/K/e+46dJzw7ByyP/
7MBjpzLz8uyNOUb1SuVdnu1ChYSVsTqCJbq0rcsvm/yY/OVe7IYT/1JBPHJ/
J0uKy8WRl5cH5cpa06ajpLrr/
```

vZ8v3Azm5MSMU2DGlGRtLyA4jJAjVoVsHvY+X72Nmx2J3abN7fe1daS4nJhGzriSWLj44r8On/
s2YZnm4p4Bv/
1MzOdJgd+30ufAfcDEHgkh2OLtmF6GARm+1IrpAb3Dexb5NkuVXhYGSaNm2h1DCnmsrOzeX36dHacTqb
3Y48ysHsP2rRqZXUsOUedDp1ZO/9Tjm/
ZgEfZCuQcPURERKTLissiUjJ0aHkTr0x7npSMZBrXbM+BE9v5au00BvU5/6LVIoUlJCyUZ5dNddn19mz
bwYCuHQuMn0pPLZBjb8wBqleq4qJkFyckLLRQz6cC82W63AKuaZo82vterks8SeXgUEzT5Ou9f9Dx3t7
0e+ThQkrpfsZOeNmS6x44cICpw+7jyWZ/
tVrYE59KvfY3MWzsS0DuhwZTpr9vST75dy0bVqdlw+pWxxApFJ5edqJvbXzJx1eNKkvVqLLn3/
EKExsfR+nelYv8Ok0Sw/h+9neEPXhVbnuMHCfxc/bSuFdzStfMvdOlPpWpX+RJCk/
snINWR5BizjRNeg4ZzMHG11Lp+REcdDh44uv5DDp0iP69dSu0OzEMg2Z33EV2RgZpiQkEdOiA/
TwLyIqIXCwPuwfDBo5hza8r+HDVeCLKR/LkQ6Px8fY9/8EihWTshBdder2flq/k2/
dX0a1Gy7wxh9NBULVwJk1/O9++ufWld1yazyoqMFvMMAxenjGNtye+wsk/
1uKwQYsbu9Cv+x1WRyuWqlSpQv2eDzPik+lc7Z/DiQyDnLI1GTnxBaujiYiIC/
kF+9P2ro5s+ngtmWRjz7FxXbvrCK/uvnewiFjtp1W/
cCCiHL5Vcz8EMux2fDq248OPP+eBnj21dkgR27NmFfu3/
YaJQUT1Glzdpt1573b09PEhqHwFFyUUkZLIMAyaN2hL8wZtrY4i4hKt2lzPM58twNzzMzdVbcaRU3FM2
fUtgyYMO++xx44d473Jb5GZkIJXsD/9/juYChWKx/O0CsxuwNfXl/8+
+4zVMUqMm27vQZdud3DgwAHCwsIICgqyOpKIiFggsEwQbe7vZHUMkSvGynXrsNeMKjCe5u/
H6dOn9ZqqCK37Yh7JZcpT+r7BGIbBsS0biJszi9Z332d1NBERkRLFMAxeeHMiP69YyaSvFlE2ogJjn3v
9vK+DYmJieGnQCEbW60npKoEkpZ9m3ICnefLtF4iIKLg225VGBWYpkex2O9Wrq9WCFUzTZMnP29l34Ai
mCfXrVKdFw2pWxxIREXE7Q0c8TWx8vNUx8hw7foLURnXxatks33j87j0MGfqE2yyAGR4WxqRx462OUWg
yU1OJPZ1CuVva5I2VurYxsXt3kRIfR0AJXZ9DRKyRlZ3Jl0s+4fiJY5iY3NAqmjo1G1gdS86Skp7Cqu2
rCfQrRdOa1+kOoyJgGAbXt23D9W3bnH/
nM9575S2erX83gd5+AJT2DeTZ+r15Y9IUnn9tQlFFdRkVmEXEpabPXUnzGtl06xaIaZos3bSbed8l0L3
LpfeiFRERKY5i4+MJuet2q2PkCTZNDr75NplR1fEOL4NpmiT9vJrKdesQ2rngYjdWiZ073+oIhSrpaAx
eVWsUGPe5qg5x+/aqwCwiLuN0Opn07mhua/
IQN9W5ihxHDl+unsbJxDjaNNVdYe7gu3XfsXXb79xYuTOJx5N5ZsWzDLptIJHhV/4M2StdTlIagSF+
+cYCvP1wxmRYlKhwqcAsIi5zNPYUIT4pNK6Zu1qpYRh0ahTEG18dJz0jG18fLT4jIiLirgzDoOODD7Dx
64WcTDmN4XBSrU5tarRubXW0IhUeFkrs3JmWXT8nM5OMDCe0aJtvPP2PX0nYtZXA/
TusCXYe4YW8Or2IWG/
NrytoHnUTlcteBeQu8te91UNMWfiUCsxuICElge1/7GBMi1F5Y+2rtGHkt6MZ3fdZC5NdWaKjo4vkfGk
xCTxe6Ub8vHzytqVnZ7Ji46qLuuaiRYsKNV9hUYFZ5AJ9M/9TVsz/
CG9nBhmegdz9yFM0aHyd1bGuKLsPxHJtVe8C4zXKe3A09hTVI/
VGxB39uuUgGzbuxMvLSWamjZYt61D76uKxEIGIiOTndDhY/
+UCEhKTMG0GvoaN5t1vwycwEMhdNK7Zne4zq9oVJo0bZ8l13501my9+Xk2GTyDex/
aQtHIZpVt3ACDjwF6al/LhuK8vs6a+a0k+ESl5tu/5nW71BxcY9/UshcPhwG63W5BK/
vT9ph+4q2b3fGNedi/KeZYlOTWZIH+tlXAhiqqAu3fvXsY/Po6R9Xvh4+lNZk4W43/9hPc+/
4iov1nj4kqjArPIBVjyzQKOff0uLzQMBbxxOk2eH/
cEoa9+SGRkpNXxLtmBmFiefjfWZddLTU2heaV06lQNyBszTZMlG5KIX72V0iFl8PDQnyV3sn3HUfbt28
4D9+QW/03T5P0P17JuQyg972yOj7dmnYuIFCe/fPIZZosmlKmS+/om53QK378/
i66PDHabHsslwdsfzOL9mAR8uvcFIDgnh/i3J+J/
7ACePj60uLomT748gRtvvNHaoCJiidCwEKYsGHX+HQvZkbijbD+0gUZRbfONH47fzTvfjgbgwOG9VIlw
z/WOQsNCCvV8e4/s4545fQv1nJcjOTaJaxvULTB+POU4Az8fovfaFqtevToDJg1n4v/ewUjNwfTzoP/
EYcWiuAxFXGA2DCMaeA2wA9NN05xwzvZI4AOg9Jl9hpumufDMtqeBfoADeNQ0zcVFmVXk3yz77APG1Pv
rychmM/hvo1CmTZnMyJdetTDZ5alSKZyBNxbuk+z5vDnre3YfSSeqoi9xSVm88ulhOjcOpmxIDj/
9vp8q1arRqdU1Ls10Kd79NsHqCC6xZu127unx178RwzC4t2cF3v/oMO/NWMagAZ2x27VoRFHatGYP23/
fh93LiSPLgzYd6lOpSpjVsUSkGMpKT+eU00HZKn99eO4RGIB3o/
oc+f0PKtV1/+fn4uKLX9bkFZcBbB4elLqrL81j9/
Dsfx+3MJmIuIMXx4+15LoOh4N+vYZQOfwqwoLKY5omy36by90DbuPu+3oCue0A3pk2xZJ8rla9YjWeb+
P6Qv8/
Sc1MY9KsV6hftl7eh8JJGcl4+Hvyyb2zLU73l+dWWPPv1x1E1YzixbdfsTpGkSiyArNhGHZgCtARiAHW
G4axwDTNP87a7RngU9M03zYMozawEKhy5uuewDVABWCZYRg1TdN0FFVeubIVVY+cPwUlHsC4tl7+MV9P
fvxsIT9tOX/fO3ftkWOFQXe344vFm/
lmfTx7D8czaWBVfLxyC5QNagTyzjf7iEuoTJmQgPOcqfgICwtn3gLXzSS/
GMdOpGG3l8o35ulpw9fHTsNrfXjjnd+oFFHVonS5P7vi4tjhRH5Z/htOM4cA/wDaRl/
Lrm0xxKfsJ7pfMABOh8mimWu46bZ2BIf6W5y4eDi0eS+7N+7ENKBcRDlq31Afmz40ETcRe+gwsRNfc9n
10tPToXbBWTReFSuw8533SQlVK6vCcCGvW3cZ3px7j5x3mbLM/N9YVi3JP++msF8H63WriPwTu93OG+
+9zKsvv0X83mSchoOb7uhA567us9BrSebv7cdNbW9k6PKnaRB6LUmZyRzKiuE/
3YdYHU1KgKKcwdwE2GOa5j4AwzA+AboBZxeYTeDPykUQcPTM192AT0zTzAT2G4ax58z5VhdhXrmCFfUL
4RGD+5KefQpfz796Sm0/cYqe/R6i/yOaRXIxPOw27uzaiOxsB1Nnfc20b45imiY5TpNKZXzo1rwU36/
dSfcujayO6jLjx0+yOsI/atSwGqdTcggM+OvpIv5kFgH+HtS+2p/
Y5Fq8PHGqhQmLhwO7T7BmzQba3hmGp7cvySez+HjGMrx9POjS/
6y7J+wGbe4M4efFv3Fzj2YWJi4etizawHHPBEr3r4JhM4jbnsiK95fQrn80GafTWTvvJ9LNDHBCeHg4D
W5uqhYB4lLhkRGE3OW6fsdOh4Oj771fYDx106+0fHwIpcpeGR/

qJcydb3WEf3Uhr1tvGzSEeIcD46x+pum/b2bCMyPocdutRRlPRORfBQYGMmrsU1bHkH/
QKKohDWs0YN+Jffj7BFCudFmrI0kJUZQF5orA4bO+jwGanrPPaGCJYRiPAP5Ah7OOXXPOsRXPvYBhGAO
AAcAV3QdX3N9DT49h5JB7ebSuP1VC/
Nl0JJnZMZ5Mfv4hq6NdltCwcN791pqZs9k5OeyJOcXEgdUI9Mv9U7Rlz2k+XxnP3lPebHx3B1Uque8b2
VA3njlbmDOZgkpX4ZU3d9D/3jAqVfTlwKE0vl0cS9+7I9i9N43ly9dc9PXcdWbUkcOxfPw/a/4/
HEv8g3ufK4fNllu8DAr1osWtpVj20Qkgfxsb/1Ke7Pn9BB8f2WNB0uLDke3gyJEjhD1YM2/
Mv1YwCftOE7//BGu/
+oXgB6vjG+gFwKntiaz97Cea9WhtVWSRImez26lZty675n5O6M1dsHt7k7jyF0rnOK+Y4nJxMXrIIAa8
NBmj6514hoSS9scWKu/YzB2PlYzbzkVE5NIZhkH1cu7ZB1uKr6IsMP/dFB/
znO97ATNN03zFMIzmwGzDMOpc4LGYpjkVmArQuHHjAttFCktERATjZn3Fh1OncHz/fmo17MSr4/
vg5eVldbTL8uIE62bOfvvNlxwL/zWvuAxwbY1AvvglkVmffE2fPn14Z/osy/JdyQq7gJuSksKwoQOZO/
8H6l7jS/8+kWRkOFi3OYyffl6Mt7d3oV7PKhUjwmndvdT5dywCc2ftzSsu/ym8kg/
ZWU5ysp14eP7VsuHo/jQatapC6041XB3zH62cd8rqCBct9eRp7BV8Coz71inN9qW/4d0yBI/Av/
7G+9UKJm7VLhxZOdi9tECKFF9RzZsSFlGJbV8uJCMnm2saNaRi+/
ZWxypxrq1Th28mv8Sr773PkQ1xdG7RnB6PTsF+1oxmERERERXdRlO+QYoCIs76vxF8tMP7UD4gGME1ztW
EYPkDYBR4r4lJBQUE8/OQIq2O4tYuZyZoYd5CX7/UtMB4c6EmvXr3w9PRUT0E3ERAQwFvvfMQPPyxi/
rxpLP0pg9Cw6rz59vPFprgMuf2cV86zZgZz/
BEnpmnma7+QGJuJl1mGz189Tuf7wyhdxotDu1JZ8XEK19aq51ZF3cLshR176DixE44X2vn+icPhIM0ni
eAb898BlbY1AceWRIJurFngGKcn/DF+VbH6dy/
yd4IrVaTV3T2tjlHilSlThheHD7M6hojIRXE4HCxZvJTSwUFcd911ai8mUkIUZYF5PRBlGEZV4Ai5i/
b1PmefQ8ANwEzDMGoBPkAcsACYYxjGZHIX+YsC1hVhVhEpBBdTwP1t6xYWvn
s/
tzTJP24LrM6yL5fphYgbat8+mvbtC7fo705esrAX9o8rljL1s6do1MUDm90gI9XB1kW+LF24HKfTSftO
zWnVph4N67bg5RV9r/i7J/6NKz8IGv2/sSzY+BNejUoDkBmTQr2Uarww9znunv0wlM+/kGL23lS+/
34TNpsWARQRERE51/zPFuCVFs6ehU5Ss/bx1sSZjJ38NBEREec/
WESuaEVWYDZNM8cwjCHAYsAOzDBNc5thGGOADaZpLgCeAKYZhvE4uS0w7jdN0wS2GYbxKbkLAuYAD5um
6SiqrCLienXrXcvHPo34adt6WtbyIT3TydxVDm7pNVzFZSlx2rXpiK+PH+/
Nfo0cUgkOqMi7k8dQunRu4bNcWFXefOUDnE4nq1evIjMrk1Ytry/WhWZXGP34KKrMncX8T7/
GYThpXLkZI18bjpeXF02pzarV2/FpGoIzw0HOt/
H4nrSxYcMGzcYREREROcfJkydZ9tkanu01PW+saXZnXhw5hndm/c/
CZCLiCkXaRNA0zYXAwnPGnj3r6z+Alv9w7IvAi0WZT0Ss9eKkaXy3cAGzlszHzy+QAaP+Q1RUlNWxRCz
RrGlLmjX926dEALb98TvPvDiI8KuTsXuYvDEjkEf7j6Fdm44uTFn83H9XH+6/q0+B8bdeeINvl3zHZ9/
MJzY2lqPpWXj0LUf/
ZU8R8qYXs16YpgWGRURERM746vNvaFOzR74xb08ffB2hnDp1ilKlrFnrRERcQ6vUiIhlDMOg643d6Hpj
N6ujiLg10zQZPeERWvbOwmb3A6B6fQdvvPcsLZu30UzmImAYBjd17kqrpi3oOOxWfPtH8mfX+LQmDga/
+BjfvDvf0owiIiIihaEw1r5JiEvm0U6vFhg/
fuI4t99+Ox4el15+0lo6Iu5PBWYRERE3l5aWRoUKKdjsfxWSDcOgfJ3T/Lj8ezp36mJhuuLt06/
nkdnSG7+zxmxedo77xpGYmEhwcLBl2UREREQKQ2EUcE+fPs3j94+mRoW6ea3EUtNPEVbFl4+nL7vs84u
Ie1OBWUREEpAgUxkyQPxmGQWJCMlAm33hWZg6jRj3L/ya/
dtHnLCkzQS7393A04TjmgDIFxuNOxNG9e3c8PT0v6/
wl5fcgIiIixVtgYCD3PNyNqWGLqyNqW8NJyr0OlKzkojL2cu4V0dZHU1EXEAFZhERkSJQ2IXDnn07kZOViIeXDQ
Cn0yRuRwirfll5WbccFneX/
SM3YUaZebNxHGk5NAy+hi8aGTVVwuYgEBAYy7fyYx7fSj3x/
j++ZeOxe5okHQzi2aGTVVwuYgEBAYy7fxSj3x/
P6eoO7GlQLj6IqePetDqaiIiINux2WzUrVvX6hgi4mJ6VyoiInIFqFy5nCnNmLmP37t1kZWVRu3btVBm
1UrQ6tLmBdq3asm3bNgIDAyGWbf2D9tvUYhsHV1a6mW7ObMQzD6lhSgmkGs4iIiIi
IiIiyBXg/UUfYDtiMrbxs4xt/CwRKRV4df4bVseSEk4FRhERERERERERERERERTeXkp5C0okkbo66MW/
GcuvIVgRlBXI04ajF6aQkU4FRZERERERERERERERERETEze2PPUCdkWEgkl3owi4iIiIjIZXE
6HPy+7AdijxwF06Rmo4ZE1q9dndSwREZH/
s3fn8Z7O5ePHX9fMGaMKmiRtVIHKVIUyXfUpiYpW0aJF9a2vilKSktVKkKilRtvXqvivPkvitKivkOkILopSlS3amCEO2sQ1i
bEnIdv3+uO5Tz3Tt2PozHE+e+7z3bgxNmnZ8gwPL/
u88ADD7DuPuwPmVmPUHT8gHd7fO2nnZ2N8n8zsVzMNyvKZ6/
nxiMUz2BrTVx+R37PMtIOc0qkAACAASURBVEVKyNqXqQKUH0jlRERERHEWWEUNkFTXpiYpW0aJF9a2vitKivkilopIlS3amCEO2sQ1i
bEnIdv3+uO5Tz3Tt2PozHE+c3/OOa/n43Eecz735zgH735z7H5XOfz/Eft9V90w+7/
u88ADD7DuPWuy7dqvesT2P845gxNmn8wPL/
nxiMUz2BrTVx+R37PMtIOc0qkAACAASURBVKkj8nvUX0wwS5IkSXpCtv7tkUf6+kgRUX1l/Dd3H+0f7w/
XOE/7A07d4SdehSZLUE/t9eSSy/Pjix/7ImfedDabrrrk/f/
zOHzuJRwJbZEiSJl6EDEuAaW5dAYZj6F5SAezDRS8Ztm7EzGcC0Fne3kvBCUmA/
Zn32v3Z+baF7Lf
tOiyNc1YwS5IkSRqKw6tlmRv8iz/s9zuseNq7SaszseXxq0nAGHTO6doJjc6Th69fFmBXTsAmIVJicyk/
PUmSpPEuSsJSEuAIz3n4n6rNbf5X7gGOKRwmnz8s28hiIKo2JEkRpWpCRJkEeE/
SPbdeRLjjKqXjIRlRPWS5EYxKKQKvN21Z1mq6icA1r5bVirUIiOJPckkjwMx0Iu+JDT7G1ZnJ7MKiF/
G8Wi3JgRMGxZc5bCr7w6njpAuesuhHyxOwr3EAVtpKSZsG/
zeRGHAi3EyyrTOUXwodZpBB8U5erTq8u9gHjJE6iJvVUVVgswAHN78jSxySVwlbqAe36lHk9HeDSMzGo/
8AQCSuX0fcBsvwqj3SqT0kRP8DPXTOz68zVIAbEzrDhcFhTpjTOMKFq4Qv5dm6Q83u2IJHGd2aqbcZ6/

7YkSZIkSZIkqbeG2oP5O8DXI+IY4FDg95n50OP9QERMBA4EtgDmAOdFxNGZecnAPpm5e2v/
9wMbtX7FvZm54RDjkyRJkiRJkiQtYENtkfE64ChgS+Bo4IaI+GpEPPVxfmZj4MrMnJ2Z9wNHANs8zv7b
A4cPMR5JkiRJkiRJUseGlGDOzKMyc0dgA+APwHLAB4ELI2Kfx/
ix6cD1rcdzmm2PEhFPAVYDTmptXiQiZkbDE2RGx7WP83C7NPjPnzp07lP8VSZIkSZIkSdIIGWoP5ldFxC
+Bq4CXAmcBOwEHAXs+1o/NY1s+xr7bAT8f1HZjlcycAexAtedY41G/
LPPgzJyRmTOWXbZofyvSJIkSZIkSZJGyFB7MP8KuBv4AfCtzLwQICL+Aqz3GD8zB1i59Xgl4Mbb4Hc7
4H3tDZl5Y/Pv7Ig4herPfNUQ45UkSZIkSZIk9dhQezC/H5ieme8zC5DZOZFmbn5Y/
zMecBaEbFaREymkhD94pItYBplBV0QPbpkTEws3304DnAZcM/llJkiRJkiRJUneGmmBOaqE/
ACLibRHxvsfZn8x8ENgVOA64FDgyM2dFxL4R8arWru3DMpkL6ZGD/
zDTBLEmSJEmSJEl9ZKgtMvYDPt56PBAuH/
G7Qtk8NerzPPH7uTGpBQUmSJEmSJElSnxpqBfMEYLnSmW4ycz70X8JEmSJEmSJEnjxFArmM8CPh4R61OJ5
W2BE3sWlRJkiRJkiSp7w01wbwBwSHxHJEmSJEmSJEkChphgzsx7qQZ
GVKCOSJOmsfmzMyXjHA8kiRJkiRJkiSp7wW01wbwb8BvgDc3jy4EP9iQiSZIkSdKoMKQEc2zkBxw8j
7oYjjYCvgfYCLwvczLwLSxAtqMqX3+1ZK4ZiDDW8D4F8Mzz9+T5UtBQaJL0OsUQWD2HYV
Q00wv7X59xvNvEkMHHEI5IkSZIkSZIkjQpDTTvSyWUJUmSJEmSJEkChphgzsx9ehyHJEmSJEmSJGmU
GVKCOSJOmsfmzMyXjHA8kiRJkiRJkiSp7w01wbwB
Q00wv7X59xvNvEkMHHEI5IkSZIkSZIkjQpDTTDvSyWUJUmSJEmSJEkChphgzsx9ehyHJEmSJEmSJGmU
GVKCOSJOmsfmzMyXjHA8kiRJkiRJkqRRYqgtMjabxzZiSJEmSJEnSODbUBPOyre+nXNAPOyre+nAPsAN414NJIk
SZIkSZKkUWPCEPfL1tddwGXzr/9PKSpIkSVK3HnjgAa6//+nruu++
+rkORJElSHxtqBfOtPLolololxmUjHIskSZKkPvDN73fykumTN77ElEdB2aJEmS+sx
QE8x/5D8J9gCSdSkSd35w6mn8N7MMrL2XRN76ahZttx/7lIlb+v5/
wzh3f3GlskiRJ6j9DSjBn5mY9jkOSJEmSJEmSJEkChphgzsx9ehyHJEmSJEmSJGmUGVKCOSJOmsfmzMyXjHA8kiRJkiRJkiSp7w01wbwB
IhtCz9jA4755e9MMEuSJOlRhpRgjohdDgmdZuU/z+DPAMzdkmZjgRgohDgdmZuU/z+DPAapm5Uw9jkyRJkjRCttpqqyHt99e/
38FKr9ziUdv/cvHFj/Q/19Q/
X73/9+RH+fJEmSFozIHIxfNiSWQQNAVAfmlc9ErzlmVZS6SasnItTIuHxsfYdFtfXSkdSdICNqQFOCYM8Zf6dCbyo9Xgz4O/
zGZAkSZKkPvfON+/EU2fP4+/nqpT2fZ2V2g3t6PZvqZ/Pp3ffppjRJkiRJpmCZMmCMChX/
s6F8+axennnncG/
28bNtl4467DkiRJkiRJUp8UTSiRJMmkSZM0S3zoIyIYNnHgdc22xmAxRKyPuSt5VkSHVQ6qbTcG3QUtGW7HASSuvEKu3mMaZVasUx3F4XdUsaJAa4JmAS7X7EKSyHMqg43hi4JfpyQWtFeTcHH
zMecBaEbFaREymkshHD94pItYBpgBntTYfB2wZEVMiYgqwZbNNkiRJkiRJktQnhrTIH/
Ap4B5g+6bx+bx0cD+j/cDmflgROxxKJYYnAt/
PzFkRHoOShJJkiRJkiRJSpP0Qrr/
vYO0WsB3wd2ABYpnmcmTuZOPk0SJEmSJEmSxoJ5rZP3KENTPtxsEQzAnwZCuYmGqmT7DkOSJEmSJEmSWtI
IkSZIkSZIkaZWaaoJ5I+BLzfdNnN2cTiCRJkiRJkiRJ3RJo8JQE8wANzb
bg2sBLxu5MORJEmSJEmSJGmSJEnjOd1k+xX1Vhtx8ezYTQ6nB+9W5gmCMYrsNiRJQQjVxRa0WW5kOW18wGA3Er6PmM2xBN5wIBfXDe4dNnqqkzMFJrvRGTZxDGN
XX7WvFuUSnRJvm6c90JkJ1bXUd66M1b5NVlmwvy1RJaXmSGqDy7ihBHqKkHPTb07gb5SJkrVe2NDbMzxK+jSeJjbmyc
SJEmSJEmSSpNeQRERLmsSKUzYBJZD5trmgA2AQjkm6KXGUWbTQwZn8uWTVkSJkHvbvzZR+AbXwFFV1PO/XVM
9eoXR8RE4EBgC2AOcF5EH2Zl7T2WQvYG3heZt4xf+KTJEmSJEmSJEkChphgzsx9ehyHJEmSJEmSJGmUGVKCOSJOmsfmzMyXjHA8kiRJkiRJkiSp7w01wbwB
Z+b9wBHANoDZZeSdwYGbeAZCZt/
QwHkSJEmSJEmSpteQRERLmsSKUzYBJZD5trmgA2AQjkm6KXGUWbTQwZn8uWTVkSJkHvbvzZR+AbXwFFV1PO/XVM
6SJEmSJEmS9Lh61iIDiHlsy3nBAk6L5YO0OSsDpwl5Y0SsDpwXU+GWZBwMHA8yYMW
Pw75YkSZIkSZIk9VAvK5jnACAAu3Hq8E3DiPfX6dmQ9k5tXAXT6cmy8sfl3NnAKsFPEY5UkSZIkSZIkza
deJpjPA9aKiNUiYjKwHfCzwTtFxAwgi9z5ZuLwvsHi4TkhQ4YLkmnGvNDwCMxLkyGL/Fh2SJgOsRmHqRqtQuXbV6NMxEBGMXEp5M7G4Kl2hwL4E7E
gOmAh8PzNnRcS+wMzMHEg2bw8ckZnZ+vGnAgdFxMNU0vzLmWmCWZIkSZIkSZIk9ZGeJZgBMvNY4NhB2/Ya9Hifefzc2cC6vYxNkiRJkiRJkiTNn162yJAkSZIkSZIkSdI4ZoJZkiRJkiRJktQVE8ySJEmSJEmSpK6YYJYkSZIkSZIkdcUEsyRJkiRJkiSpKyaYJUmSJEmSJEldMcEsSZIkSZIkSepKZGavYxgREXEzcG2v45g
GxQSzJEmSJEmSJEnzSQbpOgkRuFYqwcQJyB1OTmYIGCxpMMCmBqEwHnEJVzdzJGiUvkJrHcqDPNB6xTOSbIz1XuXEHZNYuQhyy5CoNUtMGEJKgaWBhJYuY5DoUkC0XrgdIRg3mmMXMDpr2mAJebDgHSNfAoqIwRwxbnL3WkzYSR
H2OcNEXFJRMyKiMa23eb7TlkbfJcBxvJXJoIIPObPnTeomYUubdvTpK+eVjgTaolA1FEdU0tD6Zv4a+Ra0hBIscWJoTMn0Hs3PPTrnV4P9VqSxDdI8jDQRgV2gz3iAe
C8zLwjIpZ3KC+pGjdufEc0jntLi6Yn32DzvPg0cHkPNI2LzZeEXyC4LlyE0lNE6CbkRFTH0WDSJe9kVTzsBB2tfiRpaMg

Zm+8uAEzLz9ua5E4CtehirJEmSJEmSJGk+9TLBPB24vvV4TrOtbW1g7Yg4IyLOjoit5uNniYhdImJmRM
ycO3fuCIYuSZIkSZIkSfpveplgjnlsy0GPJwFrAZsB2wPfi4ilh/
izZObBmTkjM2csu+yyTzBcSZIkSZIkSdL86GWCeQ6wcuvxSsCN89jn15n5QGZeDVxGJZyH8rOSJEmSJE
mSpA71MsF8HrBWRKwWEZOB7YCjB+3zK2BzgIiYRrXMmA0cB2wZEVMiYgqwZbNNkiRJkiRJktQnJvXqF2
fmgxGxK5UYngh8+wMzZ8lUYngh8PzNnRcS+wMzMPJr/
JJIvAR4CPpyZtwFExH5Ukhpg38y8vVexSpIkSZIkSZLmX2Q+qrXxqDRjxoycOXNm12FIkiRJkiRJ0lgw
r3XyHqWXLTIkSZIkSZIkSWOYCWZJkiRJkiRJ0rCYYJYkSZIkSZIkDYsJZkmSJEmSJEnSsJhgliRJkiRJ
kiQNiwlmSZIkSZIkSdKwmGCWJEmSJEmSJA2LCWZJkiRJkiRJ0rCYYJYkSZIkSZIkDYsJZkmSJEmSJEnS
sJhgliRJkiRJkiQNiwlmSZIkSZIkSdKwmGCWJEmSJEmSJA2LCWZJkiRJkiRJ0rCYYJYkSZIkSZIkDYsJ
ZkmSJEmSJEnSsJhgliRJkiRJkiQNiwlmSZIkSZIkSdKwmGCWJEmSJEmSJA2LCWZJkiRJkiRJ0rCYYJYk
SZIkSZIkDYsJZkmSJEmSJEnSsJhgliRJkiRJkiQNiwlmSZIkSZIkSdKwmGCWJEmSJEmSJA2LCWZJkiRJ
kiRJ0rD0NMEcEVtFxGURcWVE7DWP598SEXMj4oLm6x2t5x5qbT+6l3FKkiRJkiRJkubfpF794oiYCBwI
bAHMAc6LiKMz85JBu/40M3edx6+4NzM37FV8kiRJkiRJkqQnppcVzBsDV2bm7My8Sh8Hz8y8Ouh8HzgC2KaH/
z1JkiRJkiRp3LrtttvYc889uf3227sOReNILLxPM04HrW4/
nNNsGe21EXBgRP4+IlVvfs2Vk6Fa9r6ZY6yjG5hhZmwkLsFkMjxUIO1ew/
7ULJBsJUmyrgKN+XcI5Fk6F2rhzF9Pk1pK53bqOfqbmX+zmBhfEfAr92cqmcgEt//
QETs0uwzc+7cuSMYuiRJkiRJkjS6HHbYYcyaNYvDDjvsUc9dMPMCPvzmPfnEG/
ZmzzftwaknntpBhBqLetYiA4h5bMtBj48BDs/Mf0XEu4EfAS9unlM2+MiNWBkyLiosy86hG/
LPNg4GCAGTNmDP7dkiRJkiRJ0rhw2223ccIJJ5CZH/88eywww4ss8wyAFx5xZX8dJ/
D2OfpezMhJpCZHHjgQUxeeDLPfcFzO45co10vK5jnAO2K5JWAG9s7s6Zs7ZOtmfmv5uXc2cAq
wUQ9jlSRJkiRJkkatww47jIcffhiAhx9+q9++BFVzD/55o/
ZY70PMCEqFRgRvHe9XfjVIUd1EqvGll4mmM8D1oqI1SJiMrAdcbHR/88c3HUAAjwO/cLj0B88Dv3B49AfPA79wePQHzwO/
cHj0B88Dv3B49AfPA79wePQH8bNcbAHsyRJkiRJkiRpWKxgliRJkiRJkiQNiwlmSZIkSZIkSdKwmGCWJ
EmSJEmSJA2LCWZJkiRJIyYiFu06BkmaXxGxsJ9fkjQ8JpgljRpRJkXEhPa2LmPSv4+Lx0HCz6R+FhFLR
sTiXccx1jXvgU9HxJJdx6JHGrh+ioiNI2Ja1/
GMBxHxpIiY1HUcenwRMbH5dgfgtV3GMh5FxJTWMZA0SplgHkO8qV1wImKCr/eCFRETsjyYmQ8PbM/
M7DKu8SYiFomIlSNi6YFtzXHxOPQRP5+6034vOPjSH1o3rXsD3222eVxGWOs13RTYNjP/
ERGTfa37ysDn0w8AE8w91CqG+BywepexaEgG3hvbAQ8C+Pm1YETE8sD3gIf/277qTpP/mGGF/
4I3mj6HTDCPUs0Jb52IWDciFgMTbb0WEYtGxOYRsUxmPjzweo+mN/xoM/
DaRsTqwL4RcXlEfCsilouIxSJi7Yh4WsdhjnmtiqfXAWcBPwc+HxFLRcQzI2KviNjSyoPuRcQi4PmgK0
0Fzrub8/PkwYMvni+6kZkPNd8uBBBzXfO+xGHkDr+k04PSIWDIz729fL/
ke6FZmZlNZfglwPdQAjOfvkZeZDzf3aNsCV4IFKv2sVbxyDbBMs+1+7/d6p
/
WavgBYqvl8mui5or+0Bss2Bz5N8/6IiOdGxO4RsXJnwY0TzXtj7a7jGAqn64wiETExMx+KiDcBH6RGV6
8Abo2Iq4BrgXMyc26XcY4lERHNG3ot4KPAS4HlI+LbwP7A9sDkiPhVZl7eZaxjVFAVBYdQF3xfB14NvA
t4OvBM4MaI+FhmntZVkGNdc5M0gXr9d6MqDHanKgGnAg8AbwHeA5zcUZjjjVuvc8Gxg14h4DXA5cALwW+
CCzPxHp0GOcQPnCqpK7f3AO4HLImI2cC51DK4z8d+5fwHbR8RpmTm762DGoIG/
72cCbwLWiYhDgT8BszLzX51FpoGZYA9TFeZrAJ8A9m4NwLQ/y/QEDDonnA08BzirPQPP17r/
NAUtzwB2johnAacAfwb+2n6faMQM3OdNAu6KiGdk5l86jkmPNoG693sbcGlm3hARO1LV/
psCr4+IbTPzliagNgCMj4tXAZVQeZBPgvMz8c6dBDhKe10aP1h/
YHGBf4CbqwmU9YAqwPPCxzDyjwzDHlIGL8Yj4OvAUKnGwAvBV4DpgI+rE+HdqOqgfrCMsIpYBrsrMKc3
jjamL9R2AOVTC8yHgnSbRRl7rc+rflwFcy86nN9hnA6dR74F5q0GtzTzBcCZwJHUDdLrm38
XB96Rmd/vMMwxrXUMPgk8G7idOkc/hboAvAq4FDgsM8/
uLtLxq0kcXAgsBtxADb4cQw3M39plbGBdwK/
BJz9Xdiog3AHtS9w9zgdOA3wOnZOZNJj6S748nUc
m1BH6VmYd2GN6YFRHfuM/G5E/
A74XmYe5fljZLXuLz4HrJaZO0TELTEllTLt3Wp69m3ZObFnQbaYoJ5lImIlmZ7CqAjdqcAqAdSdDMvPw0GnG80M/
a0E8xXA9pk5s9l+B+EkXEU8CfkZ9sB7ZZbxjSetkthPw/
sx8drN9beNkzJzePF4nM38fTfdmZuY3O4x/zDIJ/0h5Gx+bLm8mRLgjC29vMc+Bx0IUMGIOOl8bm/
Nnj2f8/Zdc1ZmeBKF8FWaVPEAAgZ8l2iJoj8VBPLhTbz3NzmmjKXH5KLOEdbOKO0t2W7hUJ+OTFyfe1jlSmpFmCDQxLJ3OdZKAj/
Axq3gNrATOAdYApmfmBbqNS07ZhIrAS8FxgY6oaaiPqHHJ6h+GNKRHxU4I5ZGrsmF2qAzE5JWIkZfkYtQyCuQNlNNNGlQ2AQ/

tOZOaur2PT4ohZm3Ig6djOA32Xm8V5HjbymeGglaoB+DWAyVbxyG7B7Zj7QYXji32tZfJK6ZrqXKp7YK
jP/FRFzgU0z8woTzCOrdR9+IlWk8v2I+A2V+/h8RPyYKpT433557W2RMUoMjF5QI6rnRMR7M/
NbA89n5nVURa1GUPOGXpKqzLwjIhbLzHuAZYGDm33+3iT4b+8w1LFoYNrUi4AHI2Irqgrw/
dSUtQHPAW5e4NGNE62L6OnAWhFxCHAS8F7goNauW1MJTi1ArYuJNamK/
lcAP23vk5lntr73pqhHmsGvB6g+vwN9sK+OiA9Q75m3Ap8CdoyIU2wXsOBFxNKZeTvwo+aLiHgqsGSng
Y0BrUHhJwEvpKr/plMzXw5r9nlSlzGqZOY9EbEOcENzs3o4sAQ1GHBut9GNLZl5DnAOQHM/
sTGVQHsaVXmmPtEMiG0IbEENBtyQmfsB50XEYTQ95r2OGnmZeS5wbkT8kppl9Ewqqb94Zj7QL4mz8SZq
XZf7s9aeeqhpeXUANfPlHU1yeRvg7sy8AlwDZqS1Pm+OB14UEc+j3iM/
abZvAgzMrBjInXTKCuZRJiK+AexKTbX6OfAb4I+ZOafTwMaw5o38W2ra+TXUa78ndfFxLzV16lZgaT9U
R15TKbsNNU3tn1SVzYnUh+ksaiX0szPzM50FOQ5ExCZUFccGwNrUxd91VG/
NY6k+2Ttkpj2YO9D05PoGlSg7FjiVOjdc0mlg40hT3fFVakBy94Ep0BGxG/CezFw3qk/
2kZm5WoehjjvNsXk1tZbC6sAF1HnlHpMFI6OVYP4ONfD7G+p69ZVUT/
inA2dk5n0dhjlutaqgngHsBDwV2BL4XGZ+MmoBawslRkDrvbAElax8OrA+cGhm/
rbZZ3Jm3t9lnCqt98ZLqL7kc6nK2Y0z81nNdPS5mXl+p4GOMa33yVOAPajk2MrAXpn5126jE0BEfBP4V
mZeGhHL5aBWoM2MsF2Af2Xm11oFkXqCmgH5ewfOExGxELAf9T45PDMviIhnAsdm5pM7DPVRTDCPQhGxM
PAyanrnllTi7R7g+SYTRlYzmr0QsBk1jXAjqgfz0lSPqEupZNuLBrct0ciKiBWo6cwbUDevS1IXga8AN
vRvv3eaz5yHMvPBZjT7yVSSZn3qJnUdYP3MXKHDMMe1iFiNpsqc+pxagzo3LEW1mDm1w/
DGjYjYEPgKVQV1P3WTegPwo8w8PCIOANbOzK07DHPcaCUOtqEuzN9DvU+
+lpnTI2IX4EmZ+eVOAx0jmkTBeZm5XNOG4SZgGtWH+XDgdSYxu9Hq4/
h7anD4c9TA8G+bxMC+VAsAe8Q/Qa3Xen/
quvVUYFvqPPCViHgtNb35pk4DFfCI43UUMLOZdv41YKnMfHtE7AMskpl7WUk78iLij8BsqmDlS9Tsl4e
B51N9r+/pMLxxLSI+Sl0v3d8cp1upz7NzqfPIjc0gge+LERYRHwHOzcxTmhmS/
8zMG1rPLwFsT31OHdBPrXtskTHKRMSkZlrt0c0XEbEideFyVZexjUXNh+X91LSE4wEi4snUBeOLqMqcD
akFPDRCWqPaW1EJ/J80F+K/
aL6IiHWBFwAXmVzujdbJ6q3A6hFxFnAl1YrhlMw8uZnyuQo1vVYdycyrI+L6rN6ZP4iIZanBmGdSA2Gu
WL8AZOYFwEsjYi3qs2t54MTMvDYiVqVmvHyjuwjHnWj+3Q74WWae0STSTmm2L0Gdw/
UEtD5bNgYGzscvA65rE8YLywAAIABJREFUpjevAKxucrk7TQItgGdl5lbw7/
Yw72t22Qb4XVfxjSWtCr53UoPvN0fEm6nFFKEGum6mBmDUsdbxmgIMVM5uSfWbhWr5M9B6bAL/
6aGtYWoN/
m4GLJ+ZL4yI5YGPZ+Y1EbEeNePo8E4DHecy84sRMaF5+FUq97EjsDtwC3B+RJwLuIj4yPsn/
7me2hVYPiL+SuX7rgRmZS2wOBH6q3WPCeZRoPUhvDzwxoj4ILUa92nUwnIXAN9qLhw1gpqbojlUi4xfA
L/OzJuBXzZfNH3sbuwsyDGolQRbE9gH+HJE3Eq1xvgFcEwzfcopVD3UOlktQiUrnw/
cR41aXxQRl1InuautMFjwWgMxSwFbAVs0n0eXUVNxT4qI07JZHMXk8sgbNBV66+brXOCkganQLbcA+zf
/
agFoJQ6ub21+FfDx5vuXU60cNEyDqmZmAnc3yYEX0fS5pqpsXDiueysDFzfTau+gTguXRcSiwFOsXh45
EbEGlUC+NSKWA5bI6scMNcvIdgt9YNDn1w+AHSLiTGC5zDwqIqZR9yLHwCPOKXpiBq5H1wb+0ny/
HfDn5vt1qd6/aduF7rRf+8z8FfCrZvvy1MDLa4FtMvMQi1hGVmYeCP+eTX8K1bd/Darl0sPA3IgYuK/
4Z0dhzpMtMkaB1tSdn9H0XaFGWV9DVdK+PTN/
0WWMY1XTGmBb4MVUNc4qwN9oegBn5okdhjcuNB+sq1B97F4OPIu6STqt+fpKZv69uwjHj+aCYhPq5miT
ZvM9VNL53R6HBat1btiXutC7kJpi+GxqiuH+mXmCF32908wqejAivkydj2+kbozWo6YSXgJ8ITNP6DDM
cS9qhfrvU9NvvwE8A3gK8B1qFXQXSR4hzefRztR5+jDqPbE28OXMPKPL2Maz1mDYblTLt9uBJTPzzRHx
VWDVzHxNt1GObu1kZdMv85vUgO8twHaZuXVE7Ah8yLZ6/
SdqwfaDqVktS1PnijWAazJzD6+lRsag98nSwO+p8/
NbgG9k5hFNzuPCzNzPBHN3WvcZX6TuL36dtbDf4tQaFtna1/fHCJpXy4vmvLIuNej1XGBKZr6zi/
gejwnmUaKZnnAHsEK7UjAi3k8lFt6ZmXd2Fd94ERGrUxcer6UqcgA+m5mf6i6q8aOpslmLqtbcv9m8Ql
NVrh5qXzg07UlWpY7B06kK5jU6DG9ci4i/AZs00wonUL2XP05dhLzd90fvtJI2l1ODLCcNbKcGxd4N/
DIzfzyQjO4y3vEiIqZSq5r/
q7Xt9fxnkb9fUJ9dX3KAfviiFq2clpnHDtr+dKqafw1gIrB3ZjrTqw9ExHTgs9T6FdOogfqZwA8y8+Iu
YxtrIuL/AV+gPnOuB04GlqP6yv5fl7EJImI/
4Jv56IXLtgJmUOuNnJuZP262m0DrgebcvDOVMDuLai12BjUoOcfXvXsRMZcajP9TRLwHeBfVQvS9mTnT
YzTyWsn93aiWSkdn5j1RCysukZnXRcTimfnPfnv9TTCPEk1i81Bgx8y8prV9KtWDdsWuYhsPmqTNysBi
VEXzClTLgBcBb83MHz3Oj2s+DEpkrgIsS/V13Ibqe7YeVUX+O+CmzDykq1jHslZrnnWpv/
f1gbcDd1MXf3cAP6Ta9ZyXmZd3Fet4FtVr+Q/ATk27pPZz1wHPzMxbOwlunGgGvj4P/Dgz//
zf9lfvRS2u9YNm+v86VKXN9VELz21MLX55Rmbe1mmgo1xEbEdV0Hw7It5GFTwcCpxqxVn3ImID4KDM3L
Tp07hoZt7dPDeZGgB4MnCax+uJac7F783MzwzavgjwOqqK/0nAwZk5q4MQ1dK8Hz6XmXs1j0+kKjR/
kZnndhrcGBYRb6IW1v1WRCySmfc121eh1tVZFPgX8NPMvL/
DUMe9eGSf7IMyc52o9UWOBfYAXgIslpnv6DLOsS4i5lDFQsdFxNupfMjSVL/
y0+ZV6dw1ezCPHnOAi4GjIuJTVN9TqAqRq+CRfXL0xDXTP55LJZQXB55KXYifQJ38vkpVMtsWYGQFkBH
xPepm9clUj8xfUknN3wEPD1Ty99uo3VjROlkdS00lP4Kq/rs7M88CX/
s+cStwFPC5iHhHZt7UfHa9Brg3M2/1OPVG66LumcAOwMub9gBnZOa13UY3fjUDwg8BVzSbDgDuiIiLgY
uar9uo6hs9MVOpNhgAc4FlgP8FlohajOYk4OTMPMfPoU5cDezdfL81dQ9xOrVo9W+aQclLuwpujJlG3R
sQEc8Dvka9N36TmT/pMjDN07Op+
+qB+70TqXu+w5tzyMVU39PfZOZlXQU5Bt0NDBSk7B4RWwPHUYUSR2XmvZ1Fpkdo3QdOB65uEs1vAM7Jz
F9HxMPAp2De7Rz0xEX18l+oSS4vAexLLT66BvDRiPhT9uEaSFYw97GIWCEzb2o9XgT4DDVt53ZqeuflV

```
IuGc0wwj4zWiN1eVFXaFVRv0wOAP2WzYJZ6JyImUQv4TaKSZ7dR0zivBO7wAmTBaCpyfkxV3UynqjuOB
87MzL883s9qwYmItalej1tQ03DnUAs+/
LhpzeC5oYci4mnUdPOnUf3iF6LO0VcBh2TmhR2GN+5ExAuADZoKqQlUVdT/
o2YdLU3NuriKOscc6E3R8DStFi7MzKlNS5jtqQHJpalWVs+jqsWfDzzfc8aCFxG/
APbLzAuaY7QBVU37Kurz6nYqwbxPZp7cXaSjXzON+eTMvLCZ+bUL1et9Lep8fB51HfszWxp2LyJ+Dvw5
Mz8fES+nzgsXUNe661DJ5i2Av2X1znaAbARExHeAD2fmPyLiFdQA/
dpU271J1KDYFVQvZmcYdaw5b0yk8iEbU/
mQ7zWfc0cBl2Xm3raA642IeCHV0mof6jPpBZm5VUQ8leqHvWY/
fjaZYO5TzVSRD2bmh6Iaes8ALs3MO6NW516PukG6amC6m0ZWRLyVumG6g6rMSSqhfzX12s9u93fUyGmS
AhtSF+czqMqQxakLwDnANdRJ7Q9dxTheRMRSVBX5GsBm1OJ+q1EL+11KJTHtYbqANCPY38/
MNzSP2y1lplPJnCdTfX+v7y7SsS0iXktV2+TAMWhaZaxC3SytRyU2D8jMU6zuWHAi4qcAmfnGiNiTOlc
c0zw3sDDKS4B/Zea7uot0dIuIl1IDW8+m/tb3zMyXNM8FMBlYAlg2M//
aWaDjWEQ8ACyXmXdExHHA6zPzrua5CdQC1rsA/
5eZv+4w1FEvIu4A1sjM2yPi3cDR1HXSSlRB0HOBVwKfyczvdxepAKIWkbslM98XEecDu2Tmec1zASxMF
VdkZt7iOfyJayr7D8nMdZuq8TdRLZWWB1ak7i3WpZLOO5lg7l5znlgceJha4P28rEX+NqGKHj+UmbP6M
ck5VkTEx4E3U0Ve32iKSr8JTM3MHfoxuW+CuU9FxPrAhpl5WPOBvC9VbXMN8GdqVeLb+7EsfixpkjkbU
h+qa1O9aCdRbTHuBT6frj4/
opp+mVe0L+QiYnmq8mZDauXUVYELMvOjnQQ5jjUXG2tQF4BbU8fhK91GNX5ExJOpRV0/
GxEzgD8CP6Vax5yc9lvuuYh4JlWBs31zPHahzstXAXMy8+7mBnUqdZ72pnQBioh3UQnk/
wF+AnwlMw+0kn9kRcTSwBepqqZlqOui91OLvnpd1LFmcPggYAJ1fvhKZi7bbVRjU3ONejLwCarC79Rsr
Y0T1e93EWAKMNfilO41109fotbWmQF8nWrTcK6Jzd6IiFcBXwE+RA1uPWNgULK1zxSqr//
sDkLUIM1A8vFUwv8nzbag1qVaMTPP7jK+sS4i1svMSyNiJeC2zLy3mbX6ReBbmXlCPw5+mWAeBZpq5ld
SI3urUCNJdwP/oBYj+H2H4Y05zajcdsCR1EjdA63nVuQ/
ic51gXelixCMqIi4Hti6mdL5JmoK7kWD9lmH+vyyKmoBiFqxdhOqmuMKYNZAmxJHrbvRJPqD6of2Gupi
/
UnAJcDp1BQ2F5wbYa0WSlOaqsDNgC8D91DToGdTvRuvBC7JzBu7i3Z8iohpVGXNS6n+8cdRAzHnU8fnD
qo/uefuJyhq0cT1qeTa6dRn0tLALcDM5uv4fquuGS+iFvn7EHVNO5lqAXA8tYDWBY/3s5o/
UYtcvoNKIi8PfByYBZzlZ01/
iognAR9svg6n1tqZDNxIVQuenM2aIxoZTSuZjanz8zXU+i7XUi0oXbuij7Sud18H7ER9nh1gIcuC0fRf
PhE4EPhh1po6k6hq/6WBi/stsTzABHOfeqzRiObGaXWqd9oWVP/A003yjJyI2BL4FlWt/
BDVN+0Y4LjMdDGUBSRqhfNZVLXsLVTvumOAP2TmDV3GNh60LizWp6awLUotjrU4lUi7lqpePrzDMMed9
rkhIt4PfHsgeRMRT6EWJX0LcGRmfsGqzZEXEctk5u2Dtq1LtSfZhBoIXgn4Umb+0PNzNyLiJdQiW9+ke
jCvSPWcvQg4LTO/
12F4o1rUmiBLNVPHA9gTOISqalqDSjqvS12rbm27nm5FxBHU4MoN1LT0jahFLq8F3p+Zp3YY3pjRzGg5
mnptr6L6Lz9MDcxfTJ2X53QXoQAiYnXgxsy8LyKeQ83AOJF6X6xGtbh6DjVIvEc/VgiOVk1rjHWoxdt/
Rt1TLEu9T/7WfH1z8DWWutPcj28JfAR4ANg/M0/
oNqqxrdV6b3squX8h8MXR8r4wwdynWn9YKwJ7USNG1zbPPZ3qvfxPT3q91bQneQN1c7oGcDNVqXMKcHh
m/
qO76MaeiJg6eGpac2J7BXUcNqP6y87KzA0WfITjx0BiMiK+Sl387Uq9B9aiBrnWoy7Q9+owzHGnmWo7l
aqYvY7qweWJfAFp2ibdRSVqTgB+QQ0+PtjaZyK1wNnVmXm9CeYFJyJ2pwYhL2we/
3uApWnp8GLgtcDd9l8evoj4IDAtMz/RXKeuCPxlYMZXc95eFlg+M//
UYajjUkQsDLwX+J9moHjVzLym9fziVBXhm4AfZOYZ3UQ6NkTEmpl5ZfP9jsCpwL+oQZY1m3+fD3zaxEz
3IuLPwMsz8+aoRbRuptbVGfj8WoQqMrovM2/yHP7ENVP878/MW5rHb8zMnzZVmgP3FqtRg5Q72Uam/
zTnlY8CmwMnUWvwXOP7o7ci4tnUrLyNqBmTBzU5wL593U0w96lWcmcPqvpjs6ZCandqQZVTM3P3fv7jG
q2a6Qc5uOqvueDYGng1Nd3wuZl5TgchjlkRsT/wMuBM4FzgzMy8YtA+K1L9yX/
XQYjjTkQcQE1dO2zQ9unABCvTFozWoOMLgbdTN69bUueDAG5tEgkLA0dn5ss6DHdMi4hVgU2pROUrqdf
/cuBY4GeZeW5nwY1zEfF5qvrppog4iKokPJ2qWLZdyQhpkgWLZeblzTliZ+As6rx9NrVmyN/
aLca04ETEhsCbMvPDTbXmXlQF1FVUNe11tm0YGU0hyqcy82VNy4VnU9XKc1uDW0tSCcurfU90LyLWzcy
/NsfrUqp//KXUZ9c51Pn8Fmd/
jZyI+DY1sPvhprXYotR5+e7WPktQi5Laf7lD0SwaFxHrUbORXggsCcyhkpyvAOZSrUQ/
m5k3dxbsONK8b95PtXv738y8s9uIHpsJ5j7VSjD/
kRqp+L+IOJS6kT2Kqkw4JDOP6DTQMS4iFqJWf77V3lC9FxEvAJ5BVceuSfWU/
Tt1sX46cHpmzm32dXClx5o+vwdQybQ9qBvUe9J+mp2J6hG/
PbANtcr5qVQLmQupxMGrgE0z87m2x+iNVvuYFwJvBJagXvstqSo1qNYl7+sqxvGq1Rt7ItW2YS1qUdjF
qRuiP1NJ0GM9fzxxrYGvTaiemi+hqtD+TlUF7pqZl3cZ43jTOiYLZ+a/
mhYAuwMTqf6yd1GzX26g+gOf32G4o1rrtV4rM6+I6lX6E+qa9WKqWOJPwDWDZ+epWxGxUGY+0JwrNqVm
qr6IpnKZ6h2/W5cxjhWt98lA4vIb1KzIe6lWiL8Dfguc3zzv/V0faBV9/
RqYRhW2HE+tATaVuv59FrCN5/mR0fpcmk69vmsCt1J9/
Tel1hWZSF3LfjIzT+4s2MdhgrnPRcRh1Ejq+dTqxG/LzIua6T37ZObRtskYWa3k/
ouoRP5SVDP1W6kFUs6lFgy6yxPgyGmSmTSJm8WB6VRyYH1q4Y2nUDdH9wKv6+eRu7GiqXz6BfX3/0/
qc+hM6n1wrRWB3YmIj1A3sMtT1QVPASZQSbQfZ+avTTD3RivBfBXwnsw8vtke1Hn6ucDXM/
P4gRuqLuMdr5rjMZV6b6xK9QN+BrBQZm7dYWijXus9sNDgisxmFsULgNcBe2fmHZ0EqUeIiBWoZMD6wN
pU24ZvZOaRnQY2RrQSA1Oov/9XUm1IlqDWc/
l4Zv68yxhVWp9fy2fm3wY9tyRVoblEZn7P66iRMa+kcdQaL2+gCiPWo4om1k8XcO9EM9jydmqR6usHzy
```

B+jJ85DfhJZh7U6/jGk4h4BzVr/moquf9Has2Ehah81POoWZR7ZB8uRGqCuc9FxLOAfalViQ/
NzO9ELeR0ETWN5L5OAxyDWhce51DJ5COAg4G/UFNDlgY+kJk/
6zDMMSciXk59YJ5NTeOck02P64hYhurLtT6wUmZ+ubNAx6GIWIxaVPTl1A3q2sD3MnOPTgMbhx7rZieq
5+kzgJvSRYR6rnm9/wS8Y3CrpIg4Gdg5M6/rJLhxrDVA/BJqUOy89vslaqHkqZl5WWdBjgGt1/
ntVNXyB6iq5VWBZWwf1h+agfs3Aqdk5k3NtoFqwvWAm3OULBo0GkTEctn0mG1tmw68Hjgpm97w6g8RcS
7wFeDn1GDkc6h7jws6DWwMan3uLE8VRlw4uDguIjbLzFM6CVADCf8jqATmrdS9+MXUArE3U22v7mn2Hc
iVfI0aqLy6o7DHjIjYj3qt/zSUc0VEHE+9j/bseXDzyQTzKND0nH0wa7XuxYE3Ay/
IzDdZvdwbTQXC7Myc0jy+g6p+
egWwIbBXZt7VYYhjTkS8C9iBagPzT2rU7hKqL9pV1IntPv/
mF6yIWKLdI63ZtiqwiFUGC1brAn0R6kZoJ6rS4NeZeUm30Y0vTaXH3lRP/
j2onrMTqPYAx2bmUh2GN+5FxCXUQPCJEfE0qkXAksBuA4k2DV8rwXwq8PPM/GZE7AzsQvWg/
RbwsYGbUS1YrZv//wd8iRoYnky1Vnol1fv0wC5jHCtar/
WzgIOoFlZXU1ObnwqcMbhKVt1pXUetSyX9V2xmXfyIKnKZCOySmb/
pNNAxptUe4wtUReZu1DXTNlTh1mFNeyvv8TrQel9MporpXkidy1eg1rK4nrofvwa4YGCQPiIW8zz/
xEXEVODr1Hl6UaqV1eXUa/7nzLxqHj+zG3B2Pw7om2DuQ62LlU9RfzgD028nUR/
Gy1MJ5xvtUzSyWh+wL6UW7Xhh1AIe38vM9Zophr/LzI06DnXMiojVgM2ATagL9IWpkdS/
Uie4w6246b2o/uNbUEn/
lal+T2cBx2Xm37uMbbxqJXX2pQa7zgJ2pNr43E7NbPlsZv6hwzDHjWZmxaeoFbXnUINj06nZRgc4tXbB
ap2/16CunZaNiEWBM4DTqBulPwFf9gZ2ZETEzcA6mXln0zJmV+qm6FAqSTOr0wDHqda54lBqVstHm5vR
11ED9isDX8rM4zoNdAxoJc4OoGZHvCUitqAWVnwOMJNqb+jCZX2gdY/
9AWCLzNw6It5Ctbt6TkS8j1rH4k3dRjq2tF732dQMr9Mi4ivAM6nesj+levb/
o9NAx7HHaGMyjWr79jyqrdK6wBcy80fmoEZWU92/
CvUar9F8P6V5ei5V3Xx6Zp7W7L9Iv3YymNR1AHq05gN4MvAeqv8pEbEj8EngRuAt2fQ+9Y09slqv51+B
/
4la0XYh4OqI2Bx4DfUm1wgbuEhvptlcDfyg2f5sauGsGVQi50fdRTn2tZJirwL2oxZ3uIn6PNoK+G5En
JmZL+8wzHGplazcBXhjZp4atdDc3sB2VL/
HSfDYrTQ0cpqBrg82lVAvAB6gFl0cWBDWJGY3nk31igfYGXgoM3eLiFcBn87ML3YX2tjR3HieBezWzK6
7MTOPbZ57OjW7Qh1offYvCxzbfL8d8LXMPDIifkvNttATN/A5vykw0L7tw9Rg/
Esi4ijqGtYEcx9oDS7+Dbg3Ij5HVfj/
oNm+GtUz2+uoEdTkNpaiqjMvagbodwJmZOZ1EXEhVclsgrkjAzmQiIhm05KZeStwTPNFRKwFuK5CDzQz
Xf4GnNsUla5EtR1bh/pc2pRaaPG0Jrnfl8llsIK577RG+F4JfD4zn95cqB8OfJ5K8FybmZ/
oNNAxKCJWb1cYRGvxmog4iHrtzwf+NzNP7CjMMatVcfNmqqfjdzPzjEH7OBWnx1rH4XfU9MGvRMS3qQv
uX1DJzG9m5jGdBjpONRd3f8jMVSLiScDVmblMU/m/
J7B7Zt7fbZRjX3McngXcQ00ZvNLPpv7QtPA5jFrc70zgp5n58yaRsHpmbt9heGNCq1r8BcAHqeTZLzLz
7KYacI/
M3KDTIEVE7EC1h1kYOIdqEXNPRNwCvCgzL+00wDGkqcZcHLiPGux9fTPT9BrgzQNVZ+ofEbE31f7waGp
26j8iYhbw0cz8je0aRlaTVN6PGoyfAkzKave5BnBuZk7tNMBxrnVen0b1jX8bsAxwPHAK8JvM/
GeHIY55TXI/Bn/uNG18VgPuzMy/9Xv1uBXM/Wfgj2U14NKmMuoDwMzM/
L9mRGNn+E8yuqM4x5SmDcangJc1SZsZwF8j4k4qgbBrs212Zt7cXaRjV6tK4B/UasLHNFVRFwA/
AU7IzL/6d99breOwKrXgIlR15rsz8/SI2IPqDaVuTAVObS4A16Wmo0NdrG+bme/
r9wuP0SoeuYDc3sBaVL/4icCVUX1/T8/MX3cZ53iXmdf8f/bOOlrO6nr/
nycJDiG4a0Jwd6c4FKct2kLxIuULLcVaXAot7p7iFHcI7i4BAiRBAwQJJLgGnt8f+wx5ub9AWzIz596Z
81krK/e+M8naa955j+zz7GdL+jVRhfFUSnrODWxA3LfCWFLbhKYy58HAF7Y/
ltSX8J09NnOIBcD2xZLeJlSDDxGKzd8Aw0tyue4cCRxOWCX9JSWXFwUmKsnlzoWicfUEto+UNJXt4en6
+oSd0i3wA7VzoQ7YHiHpQuJQ8h7g3ymh9nvicy+q8bx0B0YR92dN4DDgN0T1yybAJZIusv3bfCG2PBMC
m0o6kHAtuI+wqHwSeKOW4O/
se7yiYO6kSOoFHEd40D4HnGb7maQqvC9NimUQrgOVE7s5bA+RtDFwEfAMMJBQQD0GDAG+rqmaC41F4Z2
5EGFL8qd0ecWyUG886bPfhZjYniJOro9Pfw8BeqeyqUITqYxVvRitADmBWJBMDjxoe4+a3UzOWFuRSoL
5FmCAw9f0TGBS4mBsM6IB7EnlIKy5VJ6NyYk+FS9Uyj27E/7LKxONhMqzMZYkscOJwLm2H0/
qmt62n1c0Sf7S9hd5o2xPKpWQKwNDbb9UeW18wk9Ttu/MFmSLUBl3piP6IHxLJJQ/
kjQJsC0pkZk10ALwgzl8O2KeONnhH78gsb97QdLkLn1eGkISzQ1Kz0ztXkxPWPD1TweWZe2Uicrc8RJh
x3q/opHvycCHRE+e02w/WnJQ9aXyPOxCzBsHA+sSNjIjiSqkC2zvljHM/
5qSYO7ESJqFUBHelx74VYgT8k1tv1JUavWn8oBPRtg0rEmUuk1ILBz3s31lzhjbDUWzuf8D3rd93n96f
+HnIWlq4CPbX1Wu9UyqtO2A7YkNlGyvmSvOdkfSjLbfrPy+CqEweAS42fbbZYHeWCS9BSxn+1VJQ4CNb
D8r6VTgFNsDyz1oLpW5+yBgHmALolx9eWAB4Hrbz2QMsSWobEDXBw63PZ+iMc0eRJJgELCqSyPY7Eh6G
jjU9pVJWb4DcUD8L3di78auRCXBfCZwI3Bd+n1lQhz0PpFwLr6ynYDK+PU80cj9CknbEmuo1YB9XTz66
0rlGZmJOJT8SxJzTUvYMNxl+7mS0+gcpIOxAcQad1iqgJkrHZo9Sqx33/zp/
6Xwv1JZwz5IHNyfLelsQuB4A3AGcKHtS7tCcr9b7gAKY0bSVESZ4VTpD8DzwF5OPsFlIK4/6eGW7ZG2r
7C9ne0FgNWBsyhNaxqCpG7p77Uk7S5pkaSiJSnGnyVZwxQaxgHA9ACS5kmLwVqy+TLgAqLB4uZ5wmtfU
gkhkmYGjpH0V0mbSVqMOIDc0fa5tt+GUtbZSNLcfB/QIy3Ev2N086bNgDeh3IMM1NZDGxMq5W+A/
Ynqly2AvdIhWmHsqDX/2YjRzeN2IZL4KxKWVr/
LEFeBH8wVcwPTpuTyxMBpxMHLQYSvZqEOpMTZNESC8ob0+9+BU4lm4YuX5HLnISWXJyDWuleny4cRz8d
KwFpJUVuoH7Vc06ZAz5RcXpJINm8F/
EPSNCWn0WnoARwDTJ4OAV4FVpS0HDBHSS43hkrCeEKg1n9qFcLm7S1ijftch/

d2WooHcydCP2xytgPxUK9HGOIfQ3hv3p8xxJalcsI6KbBmOtEeSXRNvS4l9f+ZNcgWppKMmZ/
YpK4CfCbpZaJhyhqEPUOhAaQS8sG2X02XjiWUNwMV/
prPERumkjjLg4jFxQLAIsCqhMLgY+CjpFQbAtxTNrMNZwTRcNfE2PQYcIOkZ4ly9I+Kern5pMRBD6An0
WF7HOIwbGXbgyQ9B0wNvJczzq5OZWPzDvB+OuRaGTja9lMpOTMoW4CF2lyxAjFHQCSUJ7S9iqTVgb+S5
vPCz6cyzv8SeCbt39YlDl9+CaxN9G95JGOYhf+fKYE7ib3eXMDLtq9R9LWYz/
awvOG1HLXE8RqEUAXi4HcksBZhv7A5cFxRMefH9kjglIra/3IiD/
UpcC4Un+xGoeg71Q+YUdIrxCHljJJeJyrqN84NK59WAAAgAElEQVQY3v9ESTB3IioP6+HATsCtxJfriX
R9D+AkQiFSqC81Y/u9gHWAy4EZgUOAsyR9AfzT9hH5QmwLzgNuAmYHFiTuQR/CB/
ukjHG1NGnsOTGpn7oRao5FgGWI5+EL4HXCl/yEXHEW2AY4mpgbpibuz5+JxPMQYGVJhxX/
wMaRnpXvrRYk/YPwSpuUmLdhtMqz0FwmAK4Fzicsre5IyeWpgFlsP5c1utbiPELB/H/E9//
mdFC5KKN7JhSaTOVg6wFgKUnnEv78tfXTLwnxSmEsqXzWHwIvJ7uqzYBLk0pzQmCSbAEWxojtNyTdSyi
YLwV2Ti9tTVIIlgRa/
ag8J3cAq6b5eCrgINvDk33PJek9tQOyQkaSTeg3RFL5ZOA1Yt54Mb2lCCgagKOB3/
GpqtvAdUTC+W3CNuPrriJgKR7MnQxJswO32e4tqSfwmu3J02vvAnOX5EH9qSiY7wH+YfuGdL07MAthtP
627X/njLMdkDQHMCr5mwro4dJYsaGkz1ljmrSSVcZCRKL5c9t7NDu+wveNF0cCU6RFSO363IS64Bqi/
PkKwiu+0y9AuiLJZmEX4l6MtP2vzCEVKkiaFdiSWJDfnDwE9wVWs71yzthaEUkz2H5L0TzuD8CGtlfIH
VcBJP2e6CFyLiFUmZ7YsO5v++af+reF/55kldSP8PA9EOhne6Skh4EzXHqHdFo0us/
IvMQ66mrbF5QEc/2peDBPT/
gw3yNpIUJNPn3xhc9PyjvtCCxOCIs+ICowHgXesv11xvDaAnVoMippAWAcYEgaq7qEyr8kmDsZknoTSo
N9CNXBQbZXkrQM0T2yd1f5cnVF0oLctvvljqVdqJThLEWooaYj7GC+IUztz3Ol8VyhMVTuwymEMq1/
Oi2dCfjG9juZQ2xrkurjHOCq6vgkaUbgIdszSVqUUG8uUp6Z+lF5NhYmNqHfEAqcaW33STYBn9gu1gCd
DEWz5HWB52zfnTmclkFSH2BEh41Qd2Dqmhd8oXMhaVnCZ/
bIcgBZf9JBfXfboyStRyj517f9YebQChWSFeL8wAtEY+tR6fq4xP6vCFqaQEpm/
hpYwPbuXUWZ2YpULFp3JSrxbiCSy7MTa91ehA3QnhnDbGnS+mkbYEOiQnUwcR/6234/
vafL5P+KRUYnIg2uL0u6FbiS6IB+k6QFgX2J5AJECXs5Wa0TleTBJETi4HhJKwK3AQ/
afi1rgK1PbbA8mRhMzwHeBRYm/OtGEk3mCg0kPQMTEl5o/0jJ5QMIH/
hPJf2f7WLPk4lUSngtcJKkTYB7CT/gVYGn0tumBsYpyeW6U7O82BZ43/
bWkrYiGjtBeKPNCOzUlRaArUQqKVyASKD1BN4gEghP2j45Y2gth6RdgO2A+SW9RNi2XQs8Zrs0Qs5EpR
JvEuC3wMxEafPrxGb1EdsP/MR/UfgZpCqinsBA4HMA29dJutP2p1mDKwA/
SKAtRghZVgGmAQZIGkDsPe61XTz6G0A6fFkfmIxImL1l+2PgnNQ7AYo1Rk6qPtlH2L4Yvq/
Ym5WoYn0zXSsHAXWk8nmuS9iyHkt81isQ1rjHSXrJ9rJdaW9RFMydDEmbE/
6/6xCnGCsTybYjiEH5k7KBrS+SeiTFwanAL4CHidKQWRjtn3ZOKYVuHJKmIzqlTlu5Ng7hibYBsElZqD
eOyiHLRsA+tpdIivILiXKpdYDJbG+dM852ppI8mJvweVwQmJPwQj0zleSeDXxcVAb1pfLZ3wqcavtaSf
cTVUVnSLoYeNH2IaW0trlUEgdrEP0rvmD0AeUrxKHxI7YPzhhml6fyDMwF3EWonB4HhhIHwJumt/
ZKiYNCk6nM4/2A3oRNzOpEY8sRRPPFY20P+PH/pfDfUBl3dgC2IA63xiM8fG8H7gPuLIe9nYPK/
boG+NT2lpLOBGYg+icsQ+zzti977PqSlJnnAPOmS4sSTcSvIYQSF5eEZecgKZh7AieXebw5VObtfwLDb
R+VqilqTatnJ9ZVN3Wl/
UVRMHcCKgv3FYGT0snR1ZJusf2FpHGqJTtl4qs7tYd1GWBzj+6EPj2hSluc2EQVGseUwKuS1rB9K4Dtb
yTdQtjElORyA6ks7iYFhkvamFAu32j7jmST8btsARZIc8Qstl8ADuj4erICuJE4ICvUkcqcexGwj6Qhw
DzAxen6MkTzRSjNT3LxR+C6lOQ/
C3iMaMa4DWEbUxg7apVzGwH3pUOWLYEHbG8u6bX0c9mUZiJtUicg7lFf2+9Ieo9Qm/
+FaNxb7k8dqGzyDyF6Hpyr6JNzJ7ADYXO4EpFAK2Smcr+WIg4DIARcGwGfEJV756brpdFcHagoM9ciPv
fViD3GTYRX+dnAGrYvzBdloUY6PD6EOCjrLukuovplJPBFV0lsdjUq++
+PgcklTWD7C+BroiLmnVQBQFe6ByXB3LmYCPi+iVz6gtUSbaUkoUGkxM0ERHnbFOnaMGAY8Lik/
kBRITSIdMDyrKTrgKMkzUyUqy3IaC+oQnO4gDgt3YVQp52arv+eKIEuNJHKyXYfouR5BUlTECXpDxMNY
YcA2H6dWAwWGkCyYLic2CgdTswJWyWvzVtr9jHlALi5VBbc8wB/TT8vA/
ze9qOS5qMcEI8Vtc1NojehDIcoM787/
TwesDRxyFVoMhXV5UrAoJRcXoZIDNwr6StgM9uvZg20BaiIgpYDPkzJ5TkAbO8jaRDwle2SXO4EVNTLM
xHVLRNL+pKwE3smvWd3ov8RZa9dN2rzxjrE4e8bknYEHrV9XrofD0HX8pZtRdIcP5ioRFqSuGfbEs/
LEKJK6fpsAbY4ydZqCWL+nkLSQ0TF0Uu23+mKz0ZJMHcOaqeliwAbSvoO+BfhwTLc9jdlwmsMlUltPmB
uoJ+k44FniUF1qO3Pc8bY6lQGzrOJ5+DPwOnAi8SkVtRnTSA9C6PS9//
ftp9N11ckFB7FB7v51BboBxJ+gUcTvuRzE8mckyUdaXv/
cgjZGColaQcQ3bQPBbYCviTKa68gjVFlk5QHRbOga4CvJI1PzCO1Q+E1CXVz4WeSvtO1RP4pxPce4GVg
EUmrERvSvTKE1/
ak5EB3YFS6dG+6NhvwaLo2P7GBLYwF6XOtqfnnAWqe4ysAT6Sf3wP2Y3SFSyEjlUPId4HdiNzHeMCLqS
x9JDDS9sdlHVUf0nNSWwt1JywxINau/dPPCxOWVrdRVOPZqKxbTdyb/sChCv/
llQhv4OnTe7uMRUMX4zvC7vB2wu96HUL5/5mkJ22fkDO4n0PxYO4kpMH4UKAPYaj+MdGc43miWc0Nxcu
rcUian9iIzgzMREx2HxCJtats35MxvJanY2ImqQVnSqrMQgOpqGR/
QXijnW7701qZjqTxgNlsv5g51LZF0nCi0/bbkl4mSp6nB5YD/m779bIxaiySniO84AdWro1X5uX8JI/
HmYlc6GuSjiSejcHAQrYXzRpgF0ZSb6Kq5XHbIzu8Nj2h+puZOBDeqoxBeUl7iVmJJOe0ROVR7cDlbNu
n54uutZA0JWG38AiRYN6fqEJdl3he9s0YXgGQtBlRwTKgo9WepA2BPxH+5BfZvqwk0OqPpMmJBtSDiEP
I1YEHgJ2BZW0PLofznQtJE9n+LP08le3huWNqFxT9p+YG+hJCosGOXi9dao9XEsydEEkzEpujpQjVwSj
ba+SNqn2Q1Is4QVqaKBU5yvZDeaNqPSplawsT/oArEOVSD6a/
XwE+KQryxlK5DzcC99s+UtIixGZpPWB72/2yBtnGpATPTYRlzITAc7ZraoLngcVrC8FCY5A0MbExGuTU

```
XTtd7wZ0d6VHQqE5SDqWOIC/
reNBpKIR5t8IxdpptgdnCLElSCXNmwPvpD8DCaXmQNtfpvVST9vFhiQDyTfzTKKS4lbbgzq8vgRR9vwU
cEXNeq/
wv5O+678i7Kle6JiIlHQw4TM7GDiwCCTykqwPLyIUy18SieZnCfu3Qalibxrb72YMs+VI1jzzA08Cr9p
+v/LahMCxwGTEfuOkPFG2NxWbn5mISoyviXzH7MCnRP+pvuntI2yvnCfS9kKj/
ZeRNBEwnu0RmcP6WZQEc2Yk9UiT3A6EwuBa2x92eM+sSZVTTlYbRFIwLwdMR2yibrH9iqQewHdd6dSoq
1BJbD5OJJUfJGxiViCa/o0D/J/
tqzOG2TYkleyCtodJeoxQ4nwJrArsbPutrAG2KalMbS2ibG1yIplwGDFW7W+7d1c72e4qVBbh6wDXAR8
CJxAVRU/89L8uNIqkWO4H/IJQ8r8K3EHco4erG9rC2CGpLzAvoYrtA0wFjE9sQl8hDoNfsj2oqNCaT/
L+3YNIBsxOzNlPALcCt9t+J2N4LUXyXD6UmAc+Jcadp4CnbL+W3jO17feyBVn4nrR/
W4iYI+YFZmH03uJjouriCSLZXLzJ64Sk/Ym5eQRRCfw6cejycvr7a8L/
+stsQbY5lcrV8wjLt4+Aewirhq+IvfhIwubnedu3l31GfajsK6Yk5uw+RPXwKMK6ZzFiHif9PJPtT7IE
OxaUBHMnQdJpwI5EKdtbRDLhItt3ZQ2shakkONcH/kBYY9wN9AImAI61fX/
GENsCSU8BK1QHUEUjs3WIrvQv/eg/LtSFVOp8FTEG9QGOsT1reu0DwiKjdJ/
PhKRxaipZSQcBGwLDiUTn8bWDypwxtjKSpiUW3EsTnnQzEpukYcDfyjyRj1R+uz6wMbA8MXcPJHwdr7X
9QMbwWoo0L89GJDP7ENYYUxJrpi1tv5kxvLYlHbh0B+YiKl2WIcaraYC3gReA450amhV+HknBPB/
hQT4vMQ9MRljqvU8km18EnugoFCrkQ1LP5K88LTF+zUGMXzOm3x9MfSzKAVkdSLZ6cxHzxJLE5zw+4bf
8HjEeDQPu7GhbUmgukrYh8h9PAKd6dNPLG4AbbZ+WM75WpJLcP5iotLubsI+Zl5g/
ZiTm7yMJlf8TXXFsKgnmTkA6xTiI8PvtTwzMmxGK2q+IU6WjSrK5vlQSzA8R3nTnpBPv2Qh7gBmBX3f0
HSyMPZUBdjbCA+1+25fmjqudkfQnYqHxJJG4PF/
SGsAptvvkja69SWVskxKJs26Ex+NLhLKgKAqajKRZCGXB2sBZth8u6o7mUFF/TE/
M0c9XN6lJ0bk+8DuicdOKmULt0lQ+5/mBXwNvVzebyet32vRnPtsXZAq17UmWML1sP5Tuy/
hE4nNWYqO6LrGHuDNflK2DpEmAzwhRSh8ikTYboZTtTVQWlc+6EyDpD4RP9mHVKry01+tFWDmMsD2gzO
H1Ix1Gyvb7kiYjDmYWJvIb0xFj1MbFAjE/klYhGonPAhxDVIcNBNa3/
WTG0FoaSb8BjicOXvaxfXm63g943faBGcMba0qCOSOVBOeuwEa2V+7gi3Mw0TxifmJQ/
q3tt3PG3IpIGgRsYfvxDtefAnay/
UieyFofSZsTk9lXwOXADURZZ1HLNpG0KRWhfBppe2BK1BxJJHAOyBpgm1GZB6YD/
ggsSyjUJgWuAY4svsvNIW1E5yR8aGchDnyvsT1cpclf06k8GycCEwMH2R6qaIzybUkQ1IfKIXA/
otH0CSlZ0INopvitpD7EPmJI1mDbkMpzsAAhUHne9l9/5L3fV8AUfj5JAbsz0MP2fh1emxDoCSxBUWZ2
CiTNANxCVFcMSNdEjFnfSdqIEFN8nTPOVqEyZ6wPrAzcZfuaMbxvCmBe2/
c2PcjCD6jMI+MSVkvrEr0r5rY9T97oWp/
0LPyN6Ll2oe2TJT1DHFJenze6saNb7gDanNpGaHzg01TGYwDbbxBeRfMQDz3A75sfYmuTFhsXAPtJmrR
yvTeh/ig+m43ldaKseU+iEccBwABJLye/u0Jz2IFQOj0BDE7PxavA4cDJOQNrU7qnv/
9MlDyfDexGqAwWBo5MZdGFBlH5fH8LXEio1IYDuwD9Ja1ZksvNp6Je3gDY3am5nO1vasllSTspmscWfi
aVRP3awBk1X2vbozy6F0gfYIe0SSo0F6W/
fw+8VUsuK+iWfp5H0uHEQUzhZ1L7PIEtCHuF49L17pV5YkJ
gYdvXleRyXtL6FcI26Y2aOhli/khJ0HEJZfMmueJsYfYEBgA3Qjw/
6WASSasB85Tkcn4qyeWJ0yHLCcB5xJ5jKkkblH1G41BYG34AHEU0ct9bUn9gsq6eXIaSYM5KxU/
lUmIBeLCkpST1lbQ6UZb4dFIefEj4MxfqSLoHVxOlbY9JulnSNcCpwAXF17RxKPzsjgOetH0WkeRcG9g
SOJdIcBYajKTZgf2Ay4C7iHuyOZFQe9GlYU0Oagmc5YHDbZ+fStWuIZoMLUs0wyw0jtp8uwehGP+17T0
Ilf+twJ6SZs4WXRtSSRysRzSW++RHNkDjE41rCmOBpHmAr2y/Wfnsq9xLKNXGaW5khQq/
JNawNQWhK4cDrxM2MsvkCq7F2BS4MlWvyPa3lcOWSYF1ysFWp2J5oObB//
34laqHvyYamy2frpV8yFiSEvcTEGKV82pVE7a/q+ylvwI2lTRVrjgLQSUHdYCkXra/
tH0OUYVxDnAsIXIs1Jk0V48CsP227UMIa9wJgRkk7ZismLosZUDtBDgao+xNeHidD5wI/
B24HbgolfmsRNgHFOqM7YG2lwL+D7iPaIpyXvq9UGcqG9W+hK/
jN5LGTZPbMEdTpiOrfmmFxmH7FduzEAvt04Gp0t+PAGMsuS00lqQq6AFcBPwild+SNrQPE0mDkVA2Ro0
ibZZENN94vnL9c9v7EF6bU8EPxrRCc+gLPJd+7lb7/
CvJ5o+I7tyFsWM84EVJi6cxaZyk2qx932cFprH9Tr4Q25M0PtUaZ31Uu1Z7PSXRPiO8T4ud0lhQ+VynI
xr5VZMztWTBy8S4NE3zIyxUqdybp4E5JE2QLH16pDmi9voaQK3xZZnD68PSwOA0X4zpM30GWIW0fi3kQ
dJEkiZNB2I7uNKU1PYIonr1caA07m0AlWq7syXNma7dTzwb+xGVkovki3Ds6ZE7gML3Ho9vEuU8UwCLE
n5qr6cBehZCTftsxjBbkqRA25Ao5+lv+6bMIbUD3QiF5vzA+JJWsn139Q3FR7P52B5G+GH3SxUU2xCK5
kIeViWV4gLTSLqS2BgtRYxVT0N5VhqBRjf7mRuYHDhP0i5EY8VvicZmE9l+An6YbCg0jsrnfCtwqKTJ/
MMmvLXXf0V05i6MHU8DHxOH7Vu44uOb1qbbMlohWGg+3xDf870kbVYdh1JCbWoi6XlPpvhaBkWjsoeJp
OSpad9m4LvKHLw48FCmEAv/
PzcC1wJrAldXK1Il9SUsMnZIl8o6qj58CHwlaTXbt6V5ojvJt5+oePnG9iiVpopNp2aLQczd2xKHk0Mk
TU5UIn2YrN+mA77osL4q1IGk8p8HeB/4te3taq/Z/
krSUUQT8TfS+9UV9xglwZwJjTbDXxnYmmgOsRKwmu2b02KmtqF6MP0p1AGNbq64EnAY8RzsQ3gOPQ/
cQZitF//
lBlApKVyU8JO9RdIdhAfRLUkJUmgCyYduVaIpzZe167b7S9qXUG8WmkxaUNwiaS5iQ7spsD0wLrERO1T
RYGuoS4OautNBsTY+0W3+X4SSuSexEN8bSgOtTDxEPAvXpsX4U8BHtj+TtBWhXt45Z4CtQFKh/
QO4UtIw4BKgP7EpPRgYRayhChlI69jriU70+0k62/
a7ioaXcxJzxl1dcXPamUjz8UhJDwI7SbrR9uuV18clvPqH2v4oW6CFjgwAziLGr6eJhPMjRGJ5C+Bcj/
bwL8/IWJKekyclDQb2kPSq7ZeIeQJJsxH2e1elf9KNkthvKpXv+ZXACMJyb3wix/
Q68KCkxwgF7UfwvVdwsQutH8sQ1ns9gI8krUAczLzi8O8XMIXtV6Drjk3qonG3DJIGEt6n1xGJzaWIkr
```

```
etgNNtf9BVTy86K5UE8yXAJ0TzrFHEidLqhAfwYNul+UMDSSemPYlE83Lp71nSywuUhXrjkbQIoQb8FB
hK2PLcQyRoTrLdpT2gWg1JixHNzX5FlKePS5yAX5kzrlZH0njAioT6ZnFgLuJ5eQp4FriuWPo0h9p6SN
KshMJ/VuBlYt20CJH8P8D2pblibDUUDZD3IObpxYgN0DXAUbaf/6l/W2g8qbriEMK/
8XnieVgBuA041PbgjOG1DGnNeg6wPnHQchvRjH1b4hDypDIXdz4kLUo0w1ySmLufJhLPlySLvrLHriOS
5iU+31kIS4yXiQbJmxBVAH+1Pax87vmRtCtwBpH/
WJcQOk4L3AmcYntQUZrXlyQOqiWZJyD2Ed2IPcVbhF1lT9ur1fJV2YIdC0qCOQOVDdKKwKm255XUG3jA
9rSSZiRUB3NkDrWlkXQycJnt+8bwWpd9qLsKqXSqZy2RrOhEPwMwR1mkN4fk4TgdkaRZLv1ZgrACONr2
5fmia1/SszE3cBCj1QWX2n6t8p5xiY3uY9XrhbFD0XxmX6Lnwd3w/9uQpPesmv6sA6zv8MYuNBiFX//
XkpYkSgwXIXzrJgAeBR6x/XjOGFuF5Fc6E6F0moDwZB5JqM5s+/
OM4RU6kKrylgbmINTmj5aD+vqR1qifE1VFGxCHjdMTAqGTyrjT+ZC0oO0BHa6ND3ydqohLkrNBSNqaEA
71JqzGrgJOSBYMhYz81Pc+VcBMWOaOxiJpA2AIIbRbnNjzTQm8C1xs+8GunIsqCeaMpC/
XrrZXlbQbsIbtdSRtBOxve9Gu/OXqjFSS+5MQyvGpgf2JTemHP/
2vC2NLxRpmMeDXxCnefMAv02BaTko7AZImAj4vC+/mUqmu2JjwPR1IKDOXJco6D7L99zIv1J/
K3LAbUcmyve13JE1JlJpvSZTcHu3kf13Ih6TPgdndocGcUtPLMo+MHcmm7VDiWehDqGvuBs60/
WB6T5mvM5MOI+ciDornIPz5X80bVWtQmRNmAf5AzMNDgTOJhuA9bX9YbJI6J5JmIhTmQ4hKo9sIO7ihW
QNrUSr7u+WAt2y/mvxmR7mDf3/
ZW+Sjss9YmrD2WZ84rL8VuML2o+l95T41gP+Q3J8S+KAVPvfSfT4TaVF4LTBc0hHA74AzJU1IlPKU8s4
GUHloZyBOiT4H/
gycIekYSdtLmidbgC1OZTN6GjAxsCNR0vxlWgzurWhMU2gwio7aS0vaW9JhkraRtIykSW1/1goTXBdmV
6K6Yifbe9hegkhwriCpTy25nOaRQn2ofZabEJYXtcTl34DfEP6NswInS5pcUjdFo6dCk5C0mKRbJe1FN
KN5ZwxvO4XSX2SsSAq/
vxNq2AOJqpaDCTXz7amstiTxM5LU5RBesmcR92kPYqxC0sKSJs4UXqtQ2yPvAyxIND2egngWpqqJUmo2
C3lCLHSk8mzMDFxNNCr9jDgkeELSI5JOTOvdkgepH7U9w7GERQ+EInN/SbulZHOX9ZRtIWqf/
1lEVdL2RI+RpYD+kr6TtGW5T/
UnHcJY0ryS+kl6T9J1kvZLFUjjEtViXZ6iYM5MOkE6lFCnjUP4p91AqESK/
3IdkTQnYaL+Tfp9AkL1MR+hAJk5/X2t7ZOyBdqiVNQgiwL/
tt1b0nTAANtTS+pFWAEs4krDuUJ96aCS3R/
4gPBgXgB4DfiWsOs5OF+U7Y2k+4D9avY9FWXIU8BfnLpzl7mh/
kgaRFQTvaZoSvMosI3t69NmtHZv7in3oLlImh/
YC1iPKCt8hehdcZntOyWtRswtk2UMs8tSmaM3Jg7eNxyDQnx3wkJmiaIEzEflXr0B/
M72XZKGALvbvknSRcBptu/PHGqXpfIZvw2santguv4osKft+8sc0PmorHEfB/
5JWDP0IJKdBwB9CVXzbMARtm8v97E+KCw+B9ieIq2XbiUSmgsSh5bHl885PwqLvUdsL9zh+sREonmg7b
dLlVJ9UWqYKOlOYv16E/Fs/IKwWxof+Jvt8zKGWReKyiMTksa3/aXth4BVFabfUxFJ/
wdr7ysDcX1QeApdTvg1ImltYBDR9fkV4LqU4JwLeCNboO3BtEChHM94IX08+LE978kl5vDbsD5to+Xd
DHR1OE1QilbLADychFwkaJ501PASEk9iaYpD0GZGxpBWlzfA+wn6XxgG+DplFyubVr7Ai9CuQfNxvazw
O8kHUMkCD4kxqvLFZYOzxOK88LPoxtxwLgZcHuyiKl972u2POcTyuaV0s+FDKTE53TAdym5PC7hc3pXe
svKwN7ZAmwB0mc8K9CtllxO9Catkcoc0PlI45UItflTtr8GvibsTbaTdBNhc7IVUTX53I9UwxT+SyoJ+
hWIvTWE/cLUthdU9Ew4wfZx2YIsVO/T7MBrktayfXPtddufEs3ea7+X5HIdsT0q/TgecLDtN4gDsAPT/
mMtYh3b5S3ISoK5SVROwucBNgZmSmqDc21/YPsl4CVJE0ma0fabeSNuLVIJ2/
rp5GgmYmP0HvGZP0R0tn3BpVFTw6gsxO8HtpL0B2IwPTuVmm9FlLMVGohHe/
f2Jk5PIUqg17X9gqSFgDL+NJnKHDEN4Q8/DVG69g5h6dMHOM72p1194dFZSZ/
tycDRwGHEpvTg9Nq3klYFPrX9blE85cP2nyq/XgogaXagF6MPLAv/O7UxZWlgB/
j+e9+NGKLGsz0yJTOLSjw/XwMPJOX+N8BLtr9IlZFfl31EXVgemErSpYQaE8Jf9tOMMRX+M+MC1wD/
kLSz7TcljUdUrS5v+xHgkVQBUJrOjSWVtdADhJXbOUSC/
5R0fW3gbRitMG9+lIUKCxDK2bMkXUkIWZ4EXkwHMoU6U9njTQs8AaxJ2JQA3yf3L6/83qX3eMUio8lIu
pdYxL9BdD4/
jDgJX4jwUpsE+JPt27IF2eKkTdJXkpYhHvCViASOCIP7v+SMrx1INhkXE01pBgKvE6fex5VNUeNRdEPf
BziDOGh5iPBufA0YBkxn+/
NsAbYhFaXgzcBRxEJ9LaIL9xvEPDEgHZaV5GadUTRy+uxiB+gAACAASURBVMT2CEkzk1Rqtkem1+cF9g
Tesb1/2SQ1F/
2wQewewAG2X04Hxj07KAwLP5Okin2LSNxfB9xo+5MO7xkGrO3S7DI7kv5IHAZMRJTcngmsC7xs+8CcsX
V1KgmBFYFfAesQVUQApxJ2SffbfitXjIUfR1Jv4CRgHsLq6kNiPfWA7T9KWgs43fYsP/
HfFP4H0mc6E1GR+i9iHTsF0XPqRNuXlbVTftJ6dk5indubsI/pTihr/
2b7iYzhtTSSNgIuIarFriNscfvbfi9rYHWmJJibSEqqXWN7pvT7VsARwEdEd/qHCBXC+bY/
yxZoiyPpGuAfth+oXJuEUCqMst0/W3AtRmWBPhXwS6Lk/
DVgr6QCnIcox73f9q0/8V8V6kBScMj2l6n08zvbQxWNRlcg7s0stpfPF2X7kso67wI2tv1B7njaBUVTs
0sJ64sXgGeIctovagctSdm/EGEd8GZJ8jeXinfdaUTJ+o6SVgD+CGwE9Ad+3TEZWvjfSM/
CuunPSsCMxLxwM3AhUd3yhO3SjLeTIGl5ojJyciK5cwZwVVGi1Z/
0fKxJCIKWJSzfNrJ9TdbACmMkramWJ8azHoSq+WHCDugo4Cvbe+WLsOtT2ectQOQvFurw+vKEZc/Rtr/
IEmThR5E0OWGZ0ZdQNh9te0TeqFqXZMc6KzA/scZakNEJ/h1t3/Sj/
7gLURLMTaCivDkUmN72tun6FoSCeb6SUG4sqaTzV4RK9h5gSiev38rkuDZwV5kA60flu38yMDexSV2IU
BKMBHYnSnOOt13sMRpE5Tu+KzGh7WP7JUmrEGW23YGdCF/Tc2y/
li3YNqTynMxNWDI8ZvsfHd7To+LfVagjCn/rXQlFx2xEU5phRLL5BcIz/
sVSFp2PyjMykGiwdauku4HbgLMJ5eYp5YC4viQV4JpEgmYJwobkbtsrZw2sUNuoLk4k/
```

QV8aHtY3qhaj2QR0404kP+uw2uTAZ/
bLjYLnYSUVF6Y2HN8ROz7XqoeCEuakGio9XWZ18eOyty8GiGO2Kl6PXN4hTEgaQnCLuYt4E0nD/
IinGg+aR6fgziEucr2kFZ4dkqCuQl02Bj1IBpv3Aj0I5Qgx6YFDISVUbkpdSapx48iPv+FiXvwHpE8GE
KUKrxte9psQbYwkt4FlksD5yzE5340oSbYlihh+10pNWwc6ZT6HuC3tdJmSV8AnxCKzceIRP+7+aJsby
RtCpwITEqUdN4I3GT7mfR6Wfw1mFTNsjTRSXtBQqH2MXEodp3tSzKG19ZI6k4kkx8BRhEHlCsmW5O3gH
VsP5Uzxq5Omp8XAW72GBruSlqcSGQOaXpwhaqV0qpEE62FgfmAv9veT9Istl/PG2VrkRKW3/
vMShonWVUtCmxje5esARaAHzwbawBHMrqh+DeEzdhg4EnbN2YMs+Wo5DiOIKz2jrZ9Zu64Cj+k8nzsBm
xOWIP2Ig4oBxDK/
muLwKhxpFzf3MDMQE9ifHqt1ewxoDT5awqVU4hdCIuA04CpCcXB+JKeAAbafj9TiO3A84Rv4z7AhETyY
HLidHskURoyOFt0LYykPsRh1hAA269LMnBUUovvJul1wvupUGcqScn1gA8qyeXFgTds95X0O6Kh3NtEg
rOQh+uIJpjzAcsBvwC2VTSF+I0r3Z4L9SMlELoROYRPCLuF/um1aYiE8zrEnF2UOZlIm6NzCAuAV4E/
p+TywsA4Jbk8digaJR5IWFZ9KWkiwhZjCDEuPWT7sZwxFr7nYMKuZytJlwO1kuZDJV1dKsLGHknj2/6y
g/JVlbdsCUzf/MgK/4H/Iw6DD0oK86XTn9WJvMeNZQ6vH5XP8RPCWmx/Sb8n9tQPAg/
bHpArvkLg0b7X+wF/tH25pJeA84FNicOB54DXipilvlR8xzcD/
gRMQAhXvgFeVzQbvd327RnDrCtFwZwJSVMSperbAcsQybWBwBKlDLq+SFqKaI71haS+wLtECfQShEJtV
uAl4E7bz2YLtMWo2DLsBhxEeDAPJRJn+9V8uiTNQNyfKbMF28JU1AWXEo1/9k/
XVwEWs31U+n1LonHT5hnDbTsUvtgbEYuOgYSK/Kmk1pyEUOEsCtxq+/
2y8GssyUdwX2Ke+NL2PplDKoyBihpnOuAvALb3yBxWl6QyRxxArId2sv21pM2AYwl/
8iWB94EtXHyus5JUUG/anj79/g6wku0XJT0H7GD7waxBdnFSIvkYYs36IpEsG1rdn0kaBBxs+
+I8URbGhKSdgFc9hr4ukia0/
XlZR9WfZDXWnVBnLkp4zPYFJrW9XM7Y2p3Kfnwe4BbbM0uaGnjB9hSSViJyUf908e6vO5U11hDgINsXS
XqK6L02N7Hu2sb2Xa0yNhUFc5OobIYWBb5JJc9XpD8k781lHU1syslqnUjJyweBz9LDfDlwbzpNvT39K
Yq0BlAZIK8lSnHOSr/3Ik7sFiC6nm9N2AEUGkDle/02UVZeG4/uAO6onKyuT/
hhF5pAZRGxJbAjoRBcGLhA0mq23yZsGT4kNrjAD56rQp2oLP5+ARwAPE003VgS2CcdxoyyfU/
OONsdSTMCawB9gHEkHW77bUl/BibKG11LsDGx+altMNcC+tneN1ky7EgII67PFF8hmBR4QNEo/
ClibHoxeTnOUJLLdWFaYCqiyeVKhNrsLUmvEBZvw4hx6LpcARZGU9ljz0gc1n8p6ZWOVj5OTXvLOqohf
EP0sfjA9rkAkuYirACKxVtGKp/
7LIzeby9OVHcDfEbYVB5R7lP9SfuLXsDkti9Kl2e0vXOqkjyWEBi1zNhUEszN5wTgMoUfc+1LNL7tF4A
XSqKzvjg8fbtJWoQoTdgVOF7SCCKhcz1wh4tnXcOwPZSwJtknDbALEP5P1xJJzyUJe4ZCY7kBuFDSFVW
lflqUT0Z02j4oV3BtiIg5YGsiiXMqgKTbgVWJRHOZD5pDrex5K+Ap23tK+hOjbXuWIRbm95TFd3OpKG+
mJuzFJicOhncA9k5z+3i2H8oZZ1cmbX66EYnLJyovnUN4M2L7dkn7EUmEQkZsj5TUj1g3rQc8K2kZoiL
y2pyxdXUqc24v4BDi+z470Zx6PkJttgqRgB7i0iCuU1Ap/
+9GePTPDQyS9CZwK3A1odws66k6UpmfNyL21wZmkDQS+Iftq2rvLeumTsHdwAeKnjyvA90lnUEcpN2V3
tOdJEYq1JWZgZsk9SDU/
TXf5QkIgWlL+TB3+89vKdSDlMTpTixQzrT9re1aR+KJJJ0vacYy+dWXtGnC9pO297I9JzAu4YX9KZFQe
1XSyfmibB9sf2j7XkeX4dmBXwGbAP/OG1lb8ABwC/CgpBMlrShpEknrEyXQt9semDfE9qEy1i/EaHU/
hNr/1fRzmaObQGVz2ovYnEKMS9eknxdjtIq83JPmUvu8Nwd62F6WUJi/
me7bzIRirTB29CKSy2vWLiTF/seSuqVN0WLAfZnia3tq/r+StgGeJXzipyYOv/
5I2Dj8JVuArcVdwCCi6nHD9PMBwJ+Bi4iGWAdki67wPZLWV/jFY3uo7S1tL0qUnR8CTEEILI5L79eP/
V+F/42UXO4JHEXsI/Ym5urLCC/mpXLGV/ghtr+w/ajtEbafI/pZ9CFsQo9Kbyt5qMbwCnAmMWe/
QeSeTgX+SlKSp3VWS1A8mJuIpHmJDuhr2P64cr0X0UWyV7bg2gBJmwNP236+w/
WpgZ62X8oTWaHQPCTtTCTP5iQmupeIxfffW+0EtbOT/
NCeIzasA4hFxoW2J88aWJsiaU2iymh34tBrckLNMQhYzfaQomBuLhWF1BnA+7b3l3RB+nkPSYcBs9reM
nOoXZbKZ7wP0SBrO8IbcETtuy5pT2Az24tnDLXtSfuFwbanrlybicj1vJkvstZB0jjAeUSflieAL4GVi
cPfAYRK/
F6iGWZR+mVE0cPinNr4n6osXgCe7FiZKmli25+WyrD6ULEW24houLtM5bXxiQOYKWzvmC3Iwvekg5Xew
ArAi8VKKS8pJ3UwcAdwrO3BrTQ2lQRzE5E0CXF68TnRwfOztFjcEtjE9vIVP9RCHZE0KbEoHJfwGhpIL
BDvsz08Z2yFQrNJz0Ntg9qNKPVsiUmtK5HK1NYDZgPmIZQEswEXEonn+4qqvHmkzeq+wLLE/
biOaFZzg+1DcsbW7khaHdiTqHp5jEj4vynpYeAY25dnDbCLU1HOHEd4Mz5KqPYnA35BlD6fYvuaMf8Ph
UZS8ZhdANgN2M32l2N6T54IWw9JaxPNqR8GLiF8+X9DeJXPXw6C85Pm7LlsD5A0PXAyMVZ1B4YTY9hTw
DO2388XaesiaQNC2b99svusXd+aOJRco4xN+ZDUw9Hf6w9EvulbQjwxCXFg9iBwme1Xf+K/
KYwlil5rcxP5p1dst7TdWEkwN5iOpxGSliAUUpMRHsCzE143R9m+owzCjSEtQuYlfNPmIh7yxYEJiaZ/
22UMr1AotBmSZgWGp4PGHowudZ6HsFKaPf18gu1i4dNEJK0MLAWMAN6wfWPmkAqApIsJr/
gZgCMJe4xuwLYdk22F/56kNluPONQaCWxKJNZ6AUOIBmenOJpTFzKSlOR/
A24GjgCeL4fD9aWSkBkPWJ2w1HsPOCT1dSl0QpJNxlSESrMvsYaaFpgeeMz2PhnDa2kknUPsq08h7GXW
AHYCTrPdr+Q28lFRmg8imsndSIjt5ibWuesDJ9s+s5UUtJ2BysHwRoSN1VyEZc8nRGL/
FuBa229kDLMhlARzA5E0M7C67bNTacJEqTxnImLxvjxxunql7XdyxtpOpM3UFMBviS7ph9vunzeqQqHQ
LqQx6FLCeuH59PerRPftUen1aYlE8wDbbxRrhsYiaUrCGmBOwq/
8mqJ4yksl0bM3MNT2JZJWITZFMwHDgCNKmfrPo2KN8RtCFbut7cHptQWIJP43wO4dy80LzUXSorafkLQ
JsXdYgEgSfErsI54l7JU+yxhmyyBpnJrCTNKcREPFXxINq29qdfVZV6GaEJN0P7Cr7afT7xMSh/
ZzAcNsP1ISaI0h2fRsS/gvzwo8DpwLXGz784yhFfheZHctsL7tryrXxyWa+35k+
+tc8bUqlQTzw8Cttg9M11
clvP23Jix+/thqY1NJMDeQ5Of4C9t7p/LOHYDrgcdL2XNzkTQ/8C6RwPm2cv06IsH8yI/
+40KhUKgjqSnKrkQyczaiqcYwItE8kGjW9IbtkdmCbAMqyo7ZiRLoD4ChRAKnD3C87b1zxtjOVBKgjxE

K2n6Kxr3jVDdJhZ9HZfNzLfCQ7b+n638i1DaPAhMBbxOHL5RDruZR+f4vBhxke52aWIU4gJwdmINIoM0
DbGz7w3wRd30kTUHYYEwLLENUsbxDKJmnBLa0fWu+CAtVkmDrV4RqdrDt8Suv1Z6flYEHypzRGFKScmL
i+RhOVLzMYHto1sAK1TXuHMBpxHh2GGHR8Gne6NoHSScTa9gXxvBabR3WUiKikmBuEB2/
KIoGf38lShJ6ER0kHwaeBO6xPSxLoG2ApN5AP6JZxyCiqdn7hPqjPzB9UX0UCoUcJG/
+pQlV5oLAdMQCfQRwve1LMobXslTn6NTcbEnbG1ZeX5VowHGA7TsyhVkAJO1FWImdWebq+iPpVWCD5GP
ai/AsPQY4B1iY8GX+i+17MobZdlSSA8cTzbJ+27HUXFJ3oiJvctsvZgu2i1NJRu5BfPf7E+XL3YBnCN/
38YnmoqXUv5MgaVHgYkKFOSFwOnAfcLvtL9Lz8Y3tbhnDbDkqY9PqwKHABMDLRI+pfrZvyxpg4Qekypc
/EL7LLwFvElWTrwOPuDR4rzuVZ2Q24CrgLWCXdqkG6/
Gf31L4OaSFSnUh2Nv2ZgCS+gBrEl2JdyRKSoa12ulFLioP9SLAF0Qy+Tyi3PzXRPOHzwnP04vKhrVQKD
STpELrRkwVnxCb2f7ptWmIhPM6gNK1liqd6iRsIukTosnJZETT1+
+xfbukbYFVgDvKPchD6lvxN+J5GVfS3cQB/
Yjiuzz2SJqAsISZn3gWdiRsF85MJbMPKppnFRu3fKwAnArQMbmZfn8v/
Sn8TCp7ryuJJMzGhI3e4+n6KOAFYAJJn5W9WqfhGWLsuoTw5u8JHAVMKOk94CPgGihNMOtMN6Ly7kJgb
6LqbnJgSeAKSTvbvihjfG2PpEnS/gLbl0nqT1S6LE5UvaxAVGrsR5k/
6k5lv9CLsEKcD7hB0hvEuPUIcIftjzOF2FCKgrmBSBrP9lfJm+gB2zP/yPvKxrWOVBLMbxFqwCeI5EF/
4iGfgvBvHAwMKZ99oVDIRfI63Zew8PnSpRFNw0ke17cR3eaHE8qC9QjP2TvStS+AOwkLpZvLPN180kHM
OMDawBJEmfpURIJ5EOE5e1e+CFuDpG66iBiDRhC+1pek1/
oS69epMobYtqTy88+JBNqzhPrsVWLM+qAkzBpHUvMvSQhTViEslN4C9rT9cs7YCj8kecwK+Ipo/
tqbqArrBlxh+8Uyh489kiaqibLS83GH7UU7vGcz4nBmvdIfIQ+SlgXOs903VUnOafvxDu+ZGVgUuK3YZ
dSXJHB8pvb9T37w0xAuBvMR9ohLAMfZPrcVBaYlwdwAKqVW6wPLEob3M9peRqlpTXrf3MBetrfJGG5LI
mkcQrW8BJFg/pJYIE5HlH/
eRJS83dRqD3WhUOjcVA7BfgEcADxN+NctmRaEqwCjSkl640jJywWBFYFFiK7zIkoGv0vXrifm6DJHdBI
kTUs0590cON/2BZlDaglSZd3ihPf7/enarIS12zi2t8oXXftR2UesR5T9n0AoziYlOtC/
QjT3GwK8PiZvx0L9SPPFdMCqRAPYllSddUVST4vtgOtsv5Q7nlZG0iFEQ9jrgOeAhYgmZXdW3rMK8C/
bM5akfvOpzB0z2x4qaSPgCkKl/Bixru1v+7WccbYqqQr1dNsbpp4h65BsSWrzhqTJiEOwl2x/
WBLMhf8JSVsDywGbEIvBe4GRxKLwKaKJykS2f1dKdxqDpLWJzs8PEwqQKYkGHhsB89meMmN4hUKhDak0
dehHlPrvqWistaTt30j6GzCL7e1aceGRmzF9ppImJqxJliXUBYsBV9rep9yDfCgaMP6FUHEOAW4uG6Pm
IGlhQjV+m+0nc8fTTlTmiH8Br9o+KF2fhWg+tyzR4G9q4v78JVuwhUIGKs/
INkTzxZUVTRq3BvYh9nx7uTT3qxuKZnFLEcr+BYB5iX5GFxNWJFOl64/bvrTkNvJS/
fwlrUnko1Yj7tO3wB9s/ytjiC1HqpBcyPbDScl8MZFgHkpU0b9A5ASHtfLYVBLMTUDSiYSPV+/
0ZxJiQJ4c+LPt+8opX32pKcVT2dTqwDbE6d0htt/KG12hUCiApGuAS5I/2qPA8bYvlnQtcJ/
tf5YFeuOobFC/96qrvDYTMLHtF8r83Fwq92U1YC/
CruRzwtpqFuBE4JhyTxpLUm1SDlfyIWkQsK3t+8c0F6QNrGw/
kSfCQiEPlXniBuB+23+XdBRRjXQH0efo2FpFRqF+pL31lISqf1lgeUJQ14uwKVnY9iv5Iiz8FMneZHPg
UduPl31G/amMT3MRBzJLEWNTD8LT/
2Lb5+SMsZGUBHODqJQojAPMUFPcJB+WGYiN0nMunTsbhqRxbH+Tfp4T2J5QM+9DWGN8kzO+QqHQ3iRFw
QnA7sC/iUPH7oS/
7Gq2hxT1bONI5WvrEd215wDuBu4hSm1HZgytrakszK8GnrV9QOW1bYkqpB3cJt24C+2LpDlsD6n83j39
6HLAUiiApFuBfxJWY/cCf7R9m6SHgBNtX1IOiRuPpMkJy4yFbR+TO552ppKDmgRYifCRN3A/
UfHyWsbw2ppUGbYW8IjtlmOgXhLMDaKyQdqV8Ew70fYnqdxzDqJpSjFVbwCpROo3RHfUZYimNe8QSuYp
iVKqW/
NFWCgUCt+rQPYlFCDzEL52iwI32D4kZ2ytTMUDey1CDXs64b98NNHEaQbgedvzZQyz7ZH0AHCA7TvS77
X79jDwD9tX5o2wUGg8SUk+yZh8f1t1c1oo/
LdI2gA4l+ifcKPtv6brI4G+tofnjK+VqMzBsxOq5T7AZ8AAYIDtYVkDLAA/
yEHtC2xF+C5DKGmnIZLNF9g+PFeMrU4SmK4J7ErYY9xCNMX8MGtgTaIkmBtEZRB+hijR6SdpF0ItNR1w
lu2TijqtflRO7PYAjgH6Ew90N+AZwtx+fOD9UgpSKBQ6C5JWJsqnRhBNtm7MHFJLU1l8XwIMtn2gpMOJ
A8hDCVX5JbavKKWD+ZC0PbA/sDPwmO3hkiYiFuvzl81soZWpjFNrArsQm9WXgduJhMHjtj/
IGWOh0BmQNDcwZbKcnJDw7V/L9pJln10/KrmN24mKu+eIcv8pgG8IX9/
DbT+TMcy2p3Kf7iVyUNckNXNPYHpC1PK67avLGre+VObtrQgB0bXA7ISSfFJi/
XqB7YPzRdl4euQOoFVJD/a4wIy2+6XL+wM7EYPxIZKuL2UK9aOygLiS8LneGPgt4X8N8bm/
AEwg6bOy4CgUCrmQNCXR+XxO4AHgTNvv542qPagspnsDl6afVwOOsP1msp79rMN7C83nYkIhtT2wgaTZ
iAP6fiW5XGgDasrkc4HTiD3EYkSjpkuAXpJ+ZfuqTPEVCtmRNC0wATBumiNeAy4Hagf1IhSbhbGgkjjr
CSxqe7KU55iJmKf7APMR/
Y4KGalUtZwHjJOufQJ8Arwl6WniMKCscetPbaxZGTjX9tG1F1Jvl80J4eP3/
cKaH2LjKQrmBlA5OZoZOBt4iGjqt5jt1VKHybdtT5Y10DYgGdkvSfgPrQJ8QJRA72n75ZyxFQqF9qJDe
eElxHg0lNGlhsfb3jtnjO2CpB5E6eDQ5Nd4E1Hx8m+i0/
Pitl8u6qfmUHk2+hEKqCHpenei8msB4F3gDaB/6aFQaAdSMud62yuO4bV5gTdtf9T8yAqF/
EjaGTgEeBYYCXwMnGz78Z/8h4X/CUmrE8njuwlx4g7AX2yP6PC+iWx/
1vwICzUq1dwTARsCxwIXEbmop20PzhpgmyDpV0Af23/
PHUsOSoK5QVRO+tYGdiM2RafbflLSfsCatlcopQnNI/
nYTQesClwzJj+7QqFQaATVRKWkfYAlbW9YeX1V4GAqnrOFxiNpPNtfSVoP2I9QdQy3vUHm0NqO5En+BT
CB7a/Std/aviBvZIVCc6kcuCwBHAD8y/blueMqFHJTeTaWI0RcvyEqjmYjDiO3AFay/
WzGMFsKSbsDfyTUmcOIvfRThKXYO4Qy9qtyGJ+fmipW0tHAusBgYDgwGXH/
RgFX274sY5gtTVL2r0AoyO8iKiruaqfeayXBXGcknUGoY8d4giepD+Gl9pDtf5cEc6FQKLQ+kjYlFuED

```
iEPHd2wf1+E9lwCv2t6vNG9qLB2VyUkpuCSxWb3b9kXlHjSHiuJmPeAg24uk632BW23PVpTkhXYk+Tge
Qti+3QXcD9xDKNHK2FRoOyoCrsOAmWxv1eH1I4Fva83+CvVD0ozA2sASwDJEdfbLhDr2GaK6qG2SaJ2R
ynrqOWBb24+kivo5CFu4RYCrbPcva9z6Uhmb9iT2efcA4xFWiL2IqtULbJ+YMcymUDyY64ikKYhy288k
TQbcRPj/
PgA8bPs12y+lrp6joHjfFAqFQquTbJF2IdQDwwmbnt0lfQ7cka59QXjZnZ8rznagtvjumKxMFS23STqJ
uCeF5tGNUI5vQGxUa6xLbFqr7ykU2ol7gI2AvsDCwIpE2fOUSd3/
RM7gCoVmU9k3DwLmlTRxh6TmdISq9vuET7NjbDVqqljbbwJnpj9IWgRYixiXdgLmzhdlAaIfVVLQPkQc
TGJ7KGHHd0eqFPsuXS/J5fpS9V8+0Pb5kiYlmmDOnK5/Aq0/NhUFc52RNIHtLyRNA2xNfKH6Eg/
5CGKzdLPte/JFWSgUCoVmkix6FiQW4osQ84KA14nF3iLA9cBeRalZXyqKjtmJBd6jRJnnx7a/
rrxvXOBzohv9h3mibV8kDUg/Xkok+U8Ezrd9aqsvxguFH6O2r0g/
9yL8yOcDzqtdLxTaDUkTA7cDUxPN3R8GZid6K2xhe0CpfKkPFWXmRsQa9hLgkQ5VYLX3FFVsBtIeo1u6
B0sBZxC5p4OAF4k17/Ca/VihcUg6HbjN9pUdro8PjGrVxn5VioK5zlQWe+/
ZPkrSJMD0RGnC3MCyRBOCe8ogXCgUCq1PZZPzdPpT2xwtTcwJcxIKzVEpEVo2RXWk8llODexFqAzeBB6
R9DAwxPaLwBqEdcmHZX7OwqaEh+aqwPbE2uk9Sd2AAZKGElVi5dkotCwVj9lpgF8By0vqTXhpnm37rv/
X3r3HWzqX/x9/
vedgGHI2COOcs3JMUiTkTKiITqKi47dUEkWlEpX8lFJEIpFQqYwOGMohhHGcGMI4jMNgxBjj/
fvj81ncdukw9lr3zF7v5+Mxj732vdbsxzVrz1rrvq/P9bkuSZfk/
Sn6VT1HmgZsJGlvSnurQyg7wfa1fS284LM/XprOe81clK3/ewJPSLqZUhjxe9s3Qapi29C4Zugswk+j/
F5WpAxk/AcwCbhL0m+y86V7ajuSVwF7SNoAGE9paXWP7afaja53UsHcJZL2owz2uwm43/
a0urq0GPCk7ceTRIiI6B+NCo+X2X58wH3LAPPZvinJze6qVQRrAjsBu1KS+5cDGwF/
sL1fZ0toi2H2NUkjKb+P3SjJg6WBO2xv0mpgEV3W+Jz4PuX//Z8p1xIbUdpkfLEmmXMNEX1L0sqU/
qYLUq6rkzTrMklnUha6rqIknbcA9gfupSQwj7V9ensR9i9JH6Y8/
x5wfGlKv+xXU6rPD7T9u3x+DJ7mcylpWWA7SsvDscC8lMT/
E8Av3SfDepNg7gJJYyjbO5ejvBFfAdxAaY9xB3BPtnpGRPSXWom5I7AfZVfLhZQem7+w/
UiLofW12iNtKTq6ZAAAIABJREFUQ2AP4DjbVybJ33uNys2FbT884L4xwOq2L2wnuojekjQVGGP76Vqgs
hBwMLAssE8+M6LfNNpd7U1JbC5Puc6eDJxm+6wkzgZX4zl/DeU5Xr5znFI0dwDwKCWJtgfwAdt/
ftEfGIOutn87xvb2kpYDTgFOpCQ0Hxzw2JzbDjJJ7wFOb7askjScskD8Csr13vrAubbP7YffQRLMg6hR
dbATzw+neQLYhJJUmEZJJlwBfDsfgBERQ18jcbYNpa/
sdyn9l79GGfi3FHCj7TVbDDOiNY2L2D2Bt1K2GN4GnAOcbfuu5uNaDDWi62pLjLOAvWxPGFAhNRlYJoU
q0Y8kvRy4Enin7d9LWo8yCHNvYGfbl7ca4BDT+GzembLAtSdwa+P9qDPQbNPae3aE7X1aDLmvSVoCOBx
4PaVFxmRgHOVc6leUzjE5hxokkhalJJe3qLsjv0PJ9V0N3Gb7H/
Vx8wIz3Jj7MpQNazuAIepQYLztY2yfYPs9wNaUhPNU4LPAyS3GFxERvaP69Z2UCpCvU6rRvk/
pw3w25XOjs+od0VfqBewY4AfAb4H3U7bh7gPcImm6pDG5MIqhrlYG3g78BviKpGXr62NxSZ8B7kxyOfp
N3QEG5Xr6lppcHmb7KtufBY4GPtRehENTZy4IcB5lhsgXgE0kjZH0JuDTwF/
qwycD87cTaX9rXDvsRWmjtDJlJ/0hlGF/5wDfbPw+YxDYftD2FvXb0ZQ+5Z8AzgXOk/RVSW+mDA/
vi+QyJME8qBonfHcDzwy470pgPuAISh+11euqa0REDGGNz4YVKavaAFsCv7F9d/
3+iQGPjegLjYudV1Mmbx9Xv37S9trASsCuth9oLciIHnFF2eY8Apgk6QHg15RWPoe1GV9EGxpbyicBz0
pad8A28+GUnsxZqB8EkkZKWgSee0+aARwJPE1JNl8OfJ7SH/6o+py/HfhFSyH3tca1wyE8fz3xd9s/
tP0WYCSlwBGeL3qJl0jSMEkj6rfbAPvX89ZtKUn9VYETKAsx9Etyf8R/
fkjMghOA02sCeRxwM7AK5cTwYdtPSloeuK/FGCMiokfqCcj3KNOcAR4ExtbtbG8APlUflxYA0Vca/
98vBjaWtIbtGxr3T6ZURkUMSZJWAo6yvXP9fqTtG4Ft6hbcdYHFKT01p7YYakTbLqOcPx0taRylqGtZy
qL9N9oMbIh5B7Aa8ElJi1H6wd9QjyNpDWCY7evr9ysDpwE/bynevtVoY/IySuu9lwNTajKzs9gy0/
Y0eMFiTbxEtf1h5xz2BMoOiwtt30zJ/
30LnpsjAiW5P+Sv8dKDeZA1XuRbAm8DFqVMfX4cONT2z+qWkhNsL91mrBER0VuSRtmeLmlH4CDKYJQpn
cRCRL+S9BbgS8A8wB+APwOXNJPNEUNRXWjc3vYPJO1A2V57FqVNxrjGTpeIvidpJPBhysLLPMBY4IvAr
5I8GxySxgIL2b5W0mGU9iPjKa2rLgWupeQ2Ztb5UymOaEljzsuBwJcpv6P32r6u5dD6Ru2/
fARwhu1LG8f78nWRBHMX1BWkJ4A1KH1v7qA09p5SVwF3B6bZ/mF7UUZERK8MPMmQND+lJcBbKavdp/
bDZOGIpsai/OqUwU3fAO6ltMVYEVgYeBbY2o0J3RFDUU2czQR2o1RCbQ0sQRl4eTmlh+ZV7UUY0Xu1/
cKywCLAfY2hryMBavuG6ILGZ/
RmwKb1zwrAw5TP6gNs39RiiFHV86hdKLsiNwGeouwMuwD4QWfgXAyeRnJ/
W+B4Sv7vw8CNlNfIk0kwx0tSBxBsR5my+gpK9c04SrXy4wMeO8L2M//
8UyIiYqj4T6vXkm6m7G45PQnm6DeNk/N9gQ1t71uTBgtSWgKsACxg+5RWA43oon/3OVFbZGwP7A/
8v7wWol80Ph8OAD4IPElJan6Esgi5HXA9cEpaxwyu//CeNB+wGWUh7CDbk/
u1UnN2JWkBysyv7ShtTd5o++r8nrqjtox5B7AesDIwhVJJfh1lZ8XfWwyv55JgHgQDVi+
+AhxDqTTYmjIB/
XZghwxviogY+hoVHysAmwNXUHrIPtacIixpLkpP5kVzcRT9qPFa2Zdywfo527cNeEwWXqIv1GTyV4FvA
rdQZuXY9vRWA4toSd3tdTulInMmZV7FSsAzlB3CmwCftZ3ev4Oo8dk8L/AW4F2U2VG/
sf2jdqOLjkYO6vXAeykJ/3vajquf1eGYW1Cu/3YC3mH7gn5K7g9rO4AhojMRck/
gZ7ZPsD3B9lHAayjP89taiy4iInqmcQIxBvgkcAZwOvB5STtIWrXe/
ybKds+pdQdMRF9pvFY2pUzgPlrSAZJ2krSapNFJLsdQV1sAQGmZtHztO74EcBTwpKTTJC3UWoARPVYHl
EGpwLzF9s22JwI/BDaoxw+o3+/
XTpRDWuec9EjKc30S8DpK6yok7SNptXZCi4bO7+ndwKPN5LKkuSVtWYtdYpB13qMkrSTpxM7zbPsh2z+
1/X7bSwC/q8f7IrkMSTAPikZl8r3A6AH3PULpH/
gyeK6NRkREDHG2L7O9CvAq4EDKZ8ERwDmSTqYkD35ZH57PhuhL9bzoe5Qt0BMp2zrfCxwCfKHF0CJ6bU
```

fKcD8o7fbmAbYE5qNUakb0i8450a7ArY3jGwLn2n7a9qOUz4z5IdfYg6mR29gFOMz2ycADlEF/
UCqal4cXLAZE73V+T2sDp0FZsJQ03PZTlIr/Lerx/
J4GV+f9Zi9gEdu3d+6QNFLSppLW7qfEcseItgMYYk4DzpI0A7gIeAxYEliN0jsNoO/
+k0VE9LN6kveX+ueQ2httQ8qJ4Yn1YanSjL5UK5Q7F60/kTSasiizOfD0i/
7FiCGikcx5AphL0taUhPL7bP9J0uGUJHNEX2i8JmYAG0gaB1xN6XP6lcZDd6AMMYPndxTHIJC0OKUNyS
2SRgFjbf+u3r0apR1oX1Vmzm5qG5MRlCHJ7wIuG9CSdQ3gQ60EN/R1/t9vDJwMzy1yDbc9Q9I7gHuA6/
qpPQakB/Ogk7Q5ZXrkaMqH4srA522f3mpgERERERbMZSUtRWsmMBoYD1wDHZW5F9BtJ61B2tjwF/
M72N+uCy73AsunVH/
2mthQbS7meXhNYh1I5+ADwJ8oulw1sX9NvSZxuqokyUeZKLUWZJbKz7Q0l7Qx8yfaamZEwe5C0LqVdzI
XAqZQi0vdRFgU2bzG0IU/SQZSiiI/Yvq9x/EbgY7bH9dvrJAnmQdL8UKvTVTcE7qf0iXrA9q/
bjC8iIiJidiJpbuA8yo6vG+vh1SkDnD5o+4G2YovotlqUcpftiZJG2Z4uaSVgHtvXS1oQ+ADwattvbjf
aiN5qDDBbktJq8iHg5ZQ+wKsArwSWs71xi2EOaXXw6OeBN1KGVT8OjAK+Y/
tXtRVDFoNb1Gh9sQVwECX3NAU4EzjJ9o39luDsJUkrA8cDf6Z0MJgX2JpyLvu6flz0SoL5JfpPq6WSbq
VUMP8kK6sRERHR7xqJg12Bz9hev3HfOsDRwIm172PEkCTpp5RKwOsl7Utpm3QlJTnwuO0nJC0NjLJ9W5
uxRrRF0i+Ai2x/
vXFsJLAsMMz2rbnGHhy13cKewC3AJNv31yTzJsB6lB1G1+b9aPYh6bXArbanNI6NBqYn+d8b9XdwIKUv
+WRKa4yv257Qj+9NSTDPgs5/lDotcnPKtpHJwGO2n248bgRlm9ui2dYWERER8YLzqE8D69l+az3eSTx/
CHhjqjZjKJM0wvYz9fYplMrM6cDdwE2Uqv7bgAn9doEaIeknwNmUBceNbP+9Hu98TnwUuMD2jf/u58R/
T9LawLcpCbJHgDuBa4EbgHuSsJw9NM6hVgF+QclHTQHWB94CXGf7h23GOJQ1nv+tgDtt31KPj6K0xu7r
+SFJML8EkjaiNPU25WTwcuAyYKLtmyXtQOkjuHS2JkRERES/kzS83nyWst35F8CvgO/
avrcOwTwXONv2t1oKM6LnJL0MeA2lWvCVwBhK8cqbWg0sosfqa+FYYDNgGeAuyjC/
Myn9yWdKehZYwfYdbcU51NTey68C1qK8B42lDBi9F3iwfv2j7WtaCzLotCaRdDCwvu2d63DYQykLA8OB
w21f1GacQ5mk+Sl5vx2B2ymvl49RFmW+Zntai+G1KgnmQVB7C
K4J7ATsShk+cDmwEfAH2/
s1qxQiIiIi+l29mN0d+AgwF2XQnyiDaj5r+8H2oovorkaSYAdKQmCc7X807n85sKLt8a0FGdEiSTsC+1
AGmL2Xcm29MDAJuMb2bv24Bb2X6vb/91KqY9ekDDM7Ns97expV/
GdQck3flfRz4Hrbn5f0Q0pl7aEpchxcjc/ttwOfsL1ebe32HcrOo8WBb/fz/
LUkmLugVt9sCOxBqWC+Mi/
uiIiI6GeS9gN2pkw5v8D2vfX43MDGwKL1oef0+xbDGPoaSYJbgANtn904tiiwkO2JbccZ0QZJI23P+Bf
HV6a0k7nB9l25xh5cdWFrXcpgxZ0pQ/1GAstRFn+/bPuePO/tk/RuYD/KYOQpwMdt3y5pAmW+xS/
zexpcjc/oo4GpNYn/XWC07XdK+gqwpO139+tznwRzRERERHSdpC0oi+
+rUbY93wOMp2x9vtj2Uy2GF9FzdZvtJNuL1O9FqeIfCfwS2Nv23S2GGNEaSasCWwOLANcDl9ie3G5UQ0
ujn+wbgM9R2mAsCCwAnAM8Boy3feOLJf2jNyQtZPuRZuJS0psprUwuqL+jjYGzgLH5XQ2uAc/
7ZsBxlNfHHZShfldIOh84z/
YxnWrn1gJuSRLMERERERdETdQjKvMBbgS9Q+tXNR6mWuga4mVIh9UhrQUb0iKRXUIZq7d+sVpY0ljKoac
HWgotoQSPhuQYlwXkXcCuwEuWz4iHgOkobpb6rDhxsjef7R8BewBPA/sCpA5/
ftMVol6SvAjfbPknSusDjAz435gK2AJawfWK/
VtB2Q6M1RjPJvBXwBuA029dLeg3wc2AD23f36+slCeaIiIiI6JmaODgV2Bt4gLIFdwvgKMrE+jfZfqK9
CCN6o1Ys/4AyWOsgyrCg0cCewEq2d2kxvIieayRyvggsW7edLwUsQanUfBUwzPYh/
ZrA6QZJmwDzAztQWmMsTul1/
WvgTNsXtxheAJJeB0yoVczHUpKbdwI3AJdQdoJlcb4LJB0DPGP745KWASY3q5MljaC0yF3H9rfbinN2k
ARzRERERHRdI3HwSeA1A5Nnkj5KOTc9up0II3qv9iA/HFiH0kfzNcA44Ij0YI5+06io/
RwwxfZxA+6fDxhu+9F2Ihx6JK1CaVe1RKM6c2VgK2AnYG1gDDAmw3fbVwckrwQsT/
ndrElZfJkLeArY3faU9iIceiQtSXnfuVvSqZR2bxOA3wBn2L6q1QBnI0kwR0RERETPSNoFOBj4tO0LGs
e/
QRmU8oHWgovoMUnz235M0orAa4FLgXvSkzz6Va3sPw7YjtJK6RLgftsPtxrYEFWrL88Ebrf9iYGtFWo7
ho/afleqxtsx8HmXtATwiO3pkuYBXkGZb7G87a+0FWe/kPQqYFvKe9Q6wAxKO5/tbd/
RYmitS4I5IiIiInpK0teANwKXANKqntgI2Ava1fWmbsUV0U2MK/
dLAjpQL1HUpPWVPlDSX7afbjTKiPZIWB44EFgMWAu4Dbql/
Jtoe32J4Q1LtH3sC8Hbbf63HFgX2A94JnG/7Q+nr2y5JHwE6u7+eASZSWpj8od4/
yvb0tuIb6uqOowVs3984Nopy/roDcHC/
Lw4nwRwRERERPVUrbrah9JpdBvgLZVDKJa0GFtFljVYxP6Qk0I4AvgyMt32QpE8Af7R9dauBRswG6iDM
N1Bax7wK+J3tA1JJO3gai17fowz5+xTwHuD/
gMcpVeTjbM9Igrk9tW3M7sD5wCPASGAtSquMo2yf0mJ4Q56kvYHtgTUow6ovA06ivDayKFwlwRwRERER
XdXoqzkKeDkwHHh0YJ/AJA2iX0h6EFjO9jRJfwP2sn2ZpIuaAwzoVaRH9RtJclN6y81EG+nWqM+ejVA/
ek0TnS9P4TF4e2BK4GFgSOJ3SW3YscKTt41sMMypJq1F+Nx9qVvBLWoEyMPmDwNq272opxCGp8Tr5P+B
dwHWU18poyi681YFv2T62xTBnK0kwR0RERERXNSqkDgf2AaBtwJ3ANcCdwN/
tv1Qe1FG9IakpYCfAe+gvBYm2F603jcVWMr2Ey2GGNEKSaOBQ4B3AzdQqjNfSWmV8bjtqe1FN/
RI2ohShWngXkrCbBLwKdvjJc3d71v+29Q4dzqUkkDepS7UP2N7Zn2MgF8C5w0cihkvnaSFgD8DH+7MDa
nP+eKUYX9HApumvVsxou0AIiIiImJoqqxdIC1K23q5KqWB+NbAxsCtlQv227UUY0VP3UhIC+wO3AX+C5/
pr3pDkcvSbTusYYDfgjbaXlPQm4Lha5b89sBmQIbCDyPZlwKq1t+zqlGrmdwLHSroNmCDph7YntRlnsB
1wFECzx3Kn57KkCcCK9Viq+wdBY0fdjsBU2xd0nu96/
D7gm5LWAF5PGdDb95JgjoiIiIheeDlwqu3b6ve3AqqfUSpC1bN/
TXmgRvVMXXMYDxwIfA6ZKOgOYSanejOg3nW3VbwR+XW9vC1xYb48FFoEXJKNjkNQq5avrnyPqgvAGwL6

U531SWlj1XiNRvCKwpqTplArz22xPaySbNwe+1UaMQ9gwymfyLsCp8HxyX9JwYGR93UygDPlLcp8kmCM
iIiKiiySNsP0MsAkwVtIetn/
Sub9esF7XWoARPSJpCeBwSiXm9cDPgXOBycDDwG9tP9ZagBEtqInLTlLmd8BW9fZmPF+xvA1wWo9D61u
1FckF9U/nWJLLLagtGn4MrEAZdPkEcJ+kO4BbgJuBVagLM/2e4BwsjUWsDYDhkh4G/
mL7lnpf5/5NgavqbfU4zNlOejBHRERERNdJOp6y1XARSkL5F8BZtie0GlhEFzWGBK0FfB5YGjgFWI6yJ
X0Dnu/
v+PfWAo2YDUh6OWXRZSLwFuBtwErA9sAuth9sMbyI1kiaH1iZ0l5sDUoPYAOLAmNsr5EK2sFV+13vS0n
uL0P5/
J5GOYe9FDgP+Buwje0JqfJPgjkiIiIiekTSSGBtSrJg23p7FLCI7UfajC2iGxpDmk4AHgeOsn134/4VK
AnnG2y/Lxeo0U8kvRq4y/bkxrH5gIOAdSiV/
QsBH0wf4OhHnV1gkl4F3Gf7vnp8KUqieX3gFttnpX1Md0haj9LmbSalWnx1YFlgSWA52y9rMbzZShLME
REREdETnQEpje/nBdaxfUmLYUV0naT7KFVO19T+jcOAmTX5vAfwHmC/Ro/
yiCFP0reBH9TXxV6UBM5FtifXis2nbD/dbpQR7ZN0FfA+4K+dJHISyt3VSO6fAIyz/
VNJcwELUxLOq1Leo36e30WRHswRERER0RWN9gArADsDi0uaSekZeDUw0fYlqdqMoUzSSpTCnmvgud6Oz
QvRXwGfHXAsoh+c0HldAK8DlgJ2l/QAcC0wQdLNnarNiH7S2AGzOfAy21fVBcqOdSRtY/
uLbcU4lNX5IQDzAp1B1M/
U96P7gKslDauPzec3ZeU8IiIiImLQNZLG3wV2r7ffC7wDOBE4R9IqSS7HUCSpM/
BnG2CEpA0lrS5pAUnDG/
cvBixu+45WAo1oQW2Z9NdGwuxw4EjKsLKpwGuBTwDfGZBUi+gXnXOjVwJXQklkNl4PKwHbQUlG9z68oU
/SYpTZIe+tVcov6HGdntcvlArmiIiIiBh0jerlDYDVbC8jaVHg/
cBhwFHAHUBaAsSQ1Fg4OZdSmXkCMBK4EbiMUv00njLI7IpWgoxoz4GUPsu/
knQdZUfLRcBFdbjWMpQes6NrUi07XaJvDBjY9wtKgvMDwGm2H6vHdwPOr7eHAUl2Dr6lKIn+rYHrJF1B
Gcz7pwyp/mfpwRwRERERg66xtfMzwHq2d5P0PuCttreQ9GZgK9v7tRxqRM9IWhDYEHgLsAUwGXgNsI/
tE9uMLaKXJG0N7EEZ9joauB+4CfgjpYXSJNsz2oswYvZQq5M/
Skly3kFpp7QJZbHyQNt3ZAGmOySNpiSZF6EseK0JrEAZ9Pct28e2GN5sJwnmiIiIiBh0jQrmPYBFbB8r
6XBgYdv7SToFeNj2R1sONaIVtUXGksDGwG9tT2s5pIhWSPoiZdFlKuU1sSClqv8vwJdtP95ieBE9JWlb
YITtX3QGzdXjWwBvqA97jDIg86G24uxHkuYBlqAkm6+1fVeS+89LgjkiIiIiukrS4sAUygn5jynDUsYA
H7H9pzZji4iI3pM0yvZ0STtRqjN3t/
1AvW8H4DvAXbY3bjPOiF6T9FFK8vJCSfsD61Eq+y8F7kgyszdqn/
gtgC2Bu4A7gVvr12n5Pfyz9GCOiIiIiEEl6WWUqsztKNueT6m9BK+vF07vAL6T5HJERN/
qtL94HXBbI7k8zPYvJa0IPFiPpUIw+skPgCfr7WcovZV3orSUeVTSREqrjJ+lun/
w1WF+M4F9gT2BhyjtSRaktLW6Czgd+GlrQc6mMmkyIiIiIgZFY4r5p4AjKUOadgE+KGkRSZsBrwem8/
w2z4iI6DONAWbnAZtLep+kMY3j2wHL1tvqeYARLbH9RJ1hMS/
l9XEg8DXgNOBm4OXA7qRgtFs6i1n7Ad+3vSMwETiaMrR3/
c5jGue9QVpkRERERMQgkzQZ2Aa4Bdgc+AYwL+XC6G5Kgvl421e3FmRERLSuJmg+RvmsuJ9SsflqStXge
2z/PRXM0Y8kfQLY0vbWA44vQ5lncW07kQ19tT3GJGBl209KegBYy/
b9kk4EPmf77rw3vVASzBERERExaCStDFxqe0zj2ExgM+CWzjboiIiIDkmbA5sAw4FpwFm2b283qojeqe
1hnpV0DGUxfmfgJNvHD3jcW4FHbF/
QRpz9oCbxPwN8E3gKOBvYCJgHeMD2PC2GN9tKSX1ERERERvGSNKo6tgeGSNqL0qdsYmGB7fKsBRkTEbEf
S0sCKlMFZF9XepxF9p9Ee5gHKovxGwFKStgJ+D/
zW9iTgMOAIeD4p3UK4Q5rtuyR9AZhJ2U1xK3AvcA3wK3hBr+aoUsEcERERERYNG0lhgf0r/
TAELUBIH76e0xrgPeCJbCiMi+pukg4F9gKuBsZTPiuspiZzTbT/
aYngRrZG0FnAS8GlgB2ADymtkFKV1w6a2n3zRHxCDRtJwYDHKe9UTlN0Vf09y/
58lwRwRERERXSFpQWBt4O3Am4AHKRO4P277tjZji4iI3uvsdpG0OvBbyrCyuYHfULaj7w/8EdjD9j/
aizRi9iFpPsrg5CWAe23fnP6/3SFJwLrAFyjnrJNsf7ndqOYMaZERERERREV1heypwMXBxPWFfEtgCmNJ
qYBER0ZZhlG3nOwF/tf0nSbsBl9s+UNLTwNNJLkc/
kzQvsAIwhtID+A7KHIub6v1JLg+yRsuLXYCPUBa9NgbWB74saQdgalq+vbgkmCMiIiKi6+qF0GTgR23H
EhER7Wj0LJ0fuKTe3hq4ot4eCcwF6S8b/akuyH+J0pLhz5T2DDcBkyTdB5xs+7EWQxzq3gOcb/
tzVljsXSwJHASsA9wMeB9ZJc7o7G4tdw4LJ6+y2U9yoolcx/7XVcc5JUMEdERERERERREV3Wq/mr/
5TWAn9S7zgX2rt+fAPwKIBWC0WdUv+4EjLP9qKR1gXNsf1LSP4BHIdX9XXYKcJykAyg9r/
8oaVFKy5LzIO9NLyYJ5oiIiIiIiIjIjolX2BhWz/
WNIo23cD60laGJiW3rLRjxoVtE9RqvoBNgF+X2+vBqT/b/edSUkmHwA8BBwFvAE4yvZD6X/
94pJgjoiIiIiIiIhu6yRlHgLuA9A7A9/
bk77YfbCCpidlH7L3+XMhQZYBywuaS7gU2BL9bjSXB2ie0Zko6gtMl4HaUH89m2L6n357l/
EcpzExERERERERHdJkUcBKlDcDxwPnArzwJ1kgabnumpK2BRW3/
uB5fltKyYTlKBe0xLYY55ElaHngtMDdwA3C97WntRjXnSII5IiIiIiIiIrqUjD1gcGEupxHwcmALcYPwA3Gj/
D1gcGEupxHwcmALcYPvkFsOLaEWjP/
lZwE22D5Y0ornoktYM3dF47tcFfgw8DVwHLAw8AzwA3Gj76BbDnCOkRUZERERERERREDLpOUkzS5sCbge
Nsf1XSWGBFYAywEvAqYGr9OxlgFn2l8f/
9KuBZSXPZfnrAY5Jc7o7OcMW9gYttf0DSypT3pxWBNYH5IUn+/yQJ5oiIiIiIiIiIgYdIgIgYdI1kzGHAj4D76/
dnAAtRBphdD3wUeLTz13oZY0SbGu0xFge2B9YGRksaD9wJ3G/70X/
7Q2KWNYYrPgTcVo9NBCYCSFoAGNlOdHOWJJgjIiIiIiIiioiskvRJY0vb36/
fzAqsCbwW2Bg4Chts+FlKpGf2hUw3bSHDOoCzCrAhsAmxGWZC5S9J422e3E+nQ1dhhMRewAvA2SQ8Afw

Gm2X6qmdzPe9O/
lwRzRERERERERAyqRquLnYAJjbvWAn5vexwwTtI5wAHAsS2EGdGKmtg8DLga+LXth4HvQnntABsCGwNb
UiqZ0z5mkDUSxotT2vU8BXybUr18uaQrgJtt39ZSiHOUDPmLiIiIiIiIiEHVqA78GjDT9mca7QDmAmbU
+w8CVre9V+f+lkOP6DpJw4GzKEnkhYGbgNOBC2xfMfCx9XWTHsCDRNImwGUAA4YprgtsS6ki3wg4xfaH
89z/Z0kwR0RERERERERXSNoUOBXY3vZf67HnKjElXQN83faPk2COfiRpPeBgSrX/05Q+5JdREs6/t/
23FsMbkiT9CdjJ9hRJBwJX2P7DgMeMBBazPTnV4//
ZsLYDiIiIiIiIiIiDOOrPAAAHjUlEQVQh6zLgIuBHkvaStJDtZyUtU6ubnwTOgRcM3IoY8moCE+ANwN+
BHYCxwHaUlrbHASfXx6qNGIei+lzuUpPL81Ge/
+Mk3SlpnKRPSVrL9gzbkwGSXP7PUsEcEREREREREV0jaQngUMrW89HAs8AUYCrwcduXZwt69JtG64
E/
aPr9x3yrAB4Dv2b45FbTdI2l+YFFgNeC1wPrAusA1trdsM7Y5SRLMEREREREREdF1klYA1gQWoQzU+p3
tKe1GFdGeWk17EjADONj2fY37bgd2H9iTObpH0ty2n6oDGEfY/mxa9/
x3kmCOiIiIiIiIiIhogaS1gW8BVvKXAHMBmwFb2l6lxdCEx8O
L+TBHNERERERERERERQLJA2jJJTfD6wO3AfcBfzI9oVJCA6ORvJ4BWBz4ApgMvCY7acbjxMnyDHcmxtB2WGxqO2p7
UQ750mCOSIiIiIiIiIiomWShgOLNVtlxOCStBFleKBERERERERET0kaSSwC/
Ap4GZgEvBX4AbgSdt3tBfd0CdpbkpP+J2AXYYFgQ2APSgXzlaxXzlalg/
u8lwRwRERERERERERERERdEDjQTzecCltr8s6Qjj4phbQcERERERERERHRD2zPrDfx
OfX2zsC421PBP5OaZUBpU9wxGwvCeaIiIiIiIiIigekLQ/0ty2n6oDGEfY/mxa9/
D7LKwHLA68FXgOsDjwKXG/
7bS2GFzFL0oM5IiIiIiIiIiIiKiB2pbjFuBWyVdCIwCGdgf2Ay6D51tptBZkxP8oCeaIiIiIiIiIgekLQ
ycJ/
tx21PB6YDx0laC7ilZMKOkhYBgLWx+H7gHmAicBswVLMaVLBHBERERERERERERERER0UWS9gG2pLTEWBiYSalgvql
OiIiIiIiIiIijojUeBBYGFgLWA+YB1gZtsXWA+YB1gZtsXwGQ5HLMaVLBHBERERERERERER0UWS9gG2pLTEWBiYSalgvql
+vc721PRfjjnRsLYDiIiIiIiIiIiIGKokHQV8GHiE0hrjOuAJYG3gHtsX254Kzw0BjJijpEVGRERERER
EREREF0haG9gV2MH2hHpMwCrAB4CfSFrX9iRJywB3p0VGzGnSIiMiIiIiIiIiImIQdVpdSDoUWMv2rpL
mBqY3E8iSzqV0GHga+LXtE9qJOGLWpUVGRERERERERERETE4OokkRcDrq+3n0su12QzwI3A+sClwG/
qfephnBEvWSqYIyIiIiIiIiiukDSVsAJwE62r67HHtueWADN0RERERERERERERERHd
cCIwHTpa0l6SFbM+UtKykywi5udNajTDiJUoFc0RERERERERERERxCCTJNuWtDhwKLAdMJrSPK8Cjwads
Xdx7bXrQRsy4J5oiIiIiIiIiIiC6TtAKwBrAIMB34ne0p7UYREREREREREREEREREREEsyQ9mCMiIiI
iIiIiIiJiliTBHBERERERERERERERERRGzJAnmiIiIiIiIiIiIiIiJglSTBHhERERERERERERERExCxJg
gaTlJrn8Obhw/sXP8f/x5E/
6bvyPp0Przd5uVuCMiIiIiuiEJ5oiIiIiIiIiWfceFfMCb2k7mIiIiIXkuCOSIiIiIiIiJi1twOr
ncA1CTzp+VdKekxyX9UdIa9b/9xxJn5E0qf698yWt0Mt/VERERETE/
yIJ5oiIiIiWXMTcDmwd/1zDC13vce4EvAdcQ2AcyWNBD4PbAucSUlIv6LzAyAyW9C/hy/
blfBdYGzujBvyUiIiIiYpaMaDuAiIiIiIg52InAMcAoYGvg6/X4XX4x4tvXrx21PlPRq4O2Aoh2w/LPAh2w/
LemdwNL18dvXr2+rfwCWkLRwV/8VERERGzKAnmiIiIhZdzrwTeBu4IJ/cf+LDe9rHte/uHte/
uL0n8EC9PQz4x0uIMSIiIiKia9IiIyIiIiJiFtl++jNIe4/22n23cdV79+g1JHwZ2BG4DbgX+CAwHjpV0
OLBU4+/9sn59F7AMsClwiO2nuveviIiIiIiIiYdalgjoiIIh4CWz/
9F8cPomSON4X2By4ktISY4akLwKrUlpgnAA1+tkSUsA7weO1RG/
6ufHxEREREExW5D9Yrv2IiIiIiIiIiIiJeXFpkRERERERERERERERMQsSYI5IiIiIiIiIiIiImKWJMEcERERE
RERERERERERMUuSYI6IiIiIiIiIiIiIWZIEc0RERERERERERERETBLEmCOSIiIiIiIiIiIiJilvwBkWJMEcERERE
RERERERERbPk/wNA4S/nr1j8gAAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<Figure size 1440x504 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAABZgAAAHwCAYAAAArRQrgAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzs3XmcjfX/xvHrM/
tmMJtt7HvJloioKEwJqYTRogUtfq3KUto0qPWpQsadBXSVkn2fd1rM2GMcYsZ879+4MmYyxjmn
PuWV7Px8PDnPfc576vQRrX+ZzPbSzLEgAAAAAAAAAAAF8vD7gAAAAAAAAAAAAAAAAgOKJghkAAAAAAAAAAAUCAUzA
AAAAAAAAACAAqFgBgAAAAAAAAAAAAAUCAUzAAAAAAAAAAAAAKBAKJgBAAAAAAAAAAAAAAVCwQwAAAAAAAAAAAAKBAKZgAAAA
AAAABAgVAwAwAAAAAAAAAAAKxMvuAAVg2R0AAAAAAAAAAAAAEo4k5+DWMEMAAAAAAAAAAAAACgQCmYAAAAAAAAAAAAAQI

FQMAMAAAAAAAACoSCGQAAAAAAABQIBTMAAAAAAAAIACoWAGAAAAAAAAABQIBTMAAAAAAAAAoEAomA
EAAAAAAAAABULBDAAAAAAAAAAoEJcWzMaYKGPMFmPMdmPM0LN8/
i1jzOpTP7YaY464Mg8AAAAAAAAAoPAYy7Jcc2JjPCVtldRR0j5JKyX1sSxr4zmO/
z9JzSzLuvcCp3ZNYAAAAAAAUKwkJSVp9OjRGj58uEJCQiRJf61eo0kffyOHw6hG9XA9Muh+BQYG2pwUA
Iolk5+DXLmCuaWk7ZZl7bQsK1PSV5K6n+f4PpKmXeikUVFRhRQPAAAAAAAUZ1OnTtWGDRs0depUSdKCn
3/V2HE/ql6Lp9W47TPK8uuse/oPVlZWls1JAaDkcmXBXEXS3tMe7zs1y8MYU11STUkLz/
H5AcaYWGNM7KZNmwo9KAAAAAAAKF6SkpK0YMECWZal+fPnKzk5WVO+nK021z0uLy8fSVJoeHXVvKSPvv
5mhs1pAaDkcmXBfLYl1Ofa3qK3pOmWZWWf7ZOWZX1oWVYLy7JahIeHF1pAAAAAAABQPE2dOlVOp1OS5H
Q6NXXqVGVbfnmOq1qzhWJXrXN3PAAoNVxZMO+TVPW0x5GSDpzj2N7Kx/YYAAAAAAAAkvTLL7/
I4XBIkhwOhxYuXChPk5HnuIN716nxpfXdHQ8ASg1XFswrJdU1xtQ0xvjoZIk868yDjDH1JZWXtNSFWQA
AAAAAQAnSvn17eXl5SZK8vLzUoUMH9ejeXn/
+8XHOyubUY4navOoTRUffZmdUACjRvFx1YsuyHMaYQZLmSfKU9LFlWRuMMS9JirUs65+yuY+kryzLOtf
2GQAAAAAAALlER0drwYIFkiQPDw9FR0crJCRE5cr+pqlfvyTL8lJEWIAmfTBafn55t84AABQOU9x63RY
tWlixsbF2xwAAAAAADYbN26c5s6dqy5dumjQoEF2xwGAkuZs99jLw2UrmAEAAABXsSxLy1Ys14p1q3V
V85Zq0by53ZEAADaIjo7Wnj17FB0dbXcUACi1WMEMAACAYiUjI0O9HxmoPVUCpNqVpK37VTfJqS/
fnpCzFycAAACA/yxfK5hdeZM/
AAAAoNCNHveW4trWkl+7y+RXOUx+1zbR1qYV9M5H79sdDQAAACh1KJgBAABgG6fTqR07digxMTHfz/
lz9zb5RobnmvnXrqw/Nq8t7HgAAAALoD3EAIAAKBQRUVF5eu4pGNHddArS/
5N6siRdFTW1v2qUzbigttcbEqJV+VbLpcx/75jz3I6tWr5ynxfOyYmJl/HAQAAADg/
9mAGAACA2yUnJ6vz4IHyuqN9TlGcmXxUjZcd1Mevv5Pr2KioqFyF8LTvp+v1Lb/
J78qGObP0X9doZNse6tKxs3u+AAAAAKDky9cezKxgBgAAgNtNmvaFsq9rLO/
TViH7hJTVpqOr5XQ65eFx7p3c+txym/aM36850xYrvby//
A+n6a4rrqFcBgAAAGzACmYAAABo8PChir+IfZD/q/Xbt8rj4S7yCQnONT/
43ne6qny1XAVzfNw+RVSLzHMOp9Op9PR0+fn5nbeQdqWIsDCNHTXGlmsDAAAALsYKZgAAAORPfGKiyvf
u4LbrNfn7Mv2+4DeF97o+Z5adkSk/Lx+FRl+f69jybkt18eK/
Wmh3BAAAAMBWFMwAAABwu+AK4YosH669U2IUcOUlciQdVcaKLbqmz812RwMAAABwESiYAQAAYIvGHa9W
vZRUxa3dqIDgUFUZdI+MTVtdAAAAACgYCmYAAADYxi84SPXatrQ7BgAAAIACYokIAAAAAAAAKBAKJgB
AAAAAAAAAAVCwQwAAAAAAAAAKBAKZgAAAAAAABAgVAwAwAAAAAAAAAKhIIZAAAAAAAAAFAgFMwAAAAA
AAAAgAKhYAYAAAAAAAAAFAgFMwAAAAAAAACgQCiYAQAAAAAAAAAF4mV3AAAAAJRuCTv2KG7jVpWPCFeN
Fo3l4ckaCAAAAKC44Lt3AAAA2MKyLC369Bv9uX2z0tvU027fLM19b7JOHD1mdzQAAAAA+UTBDAAAAFvs
WrlGWQ0jFRp1pXzCyqlMs3oK7d9Vy7+fa3c0AAAAAPlEwQwAAABb7NmwRcGtLsk18wr01wll25QIAAAA
wMViD2YAAAAoPm6f4l+b4tZrpiUekv+JDHkG+OWap+
+L1xY3ZwEAAABQMBTMAAAAUES1SJXv3cGt16xw4JD++H6RIu7onDNLXb9TNZo3Uv0u17s1S0Ed/
mqh3REAAAAAW1EwAwAAwBblKlfUJfXra9PEH6SQMrKOpSmsXIia3nKD3dEAAAAA5BMFMwAAAGxTs0UT1
bi8sTKPn5C3v688PD3tjgQAAADgIlAwAwAAwFbGGPkGBdgdAwAAAEABeNgdAAAAAAAAAABQPFEwAwAAA
AAAAAAKhIIZAAAAAAAAAFAgFMwAAAAAAAAgAKhYAYAAAAAAAAAFIiX3QEAAAAAAChqxn/
8uWYvXaMM46kIH6dGPfGgateqZXcsAACKHApmAAAAAABO885Hn2pafKD8bh4mSTqYlak7n3lR8z58XWX
KlLE5HQAARQtbZAAAAAAAcJrZy9fJr2mHnMce3j7K7nCfPpgy1cZUAAAUTaxgBgAAAACUeFFRUfk+drs
JU40zZn4Va2jCc4/o59kzCjXX6WJiYlx2bgAAXIWCGQAAAABQ4l1Medut/
+M66nTKePz7pt+0tYv09qgX1O2GzpJOFtYUwgAAsEUGAAAAAAC5jHjwbqV/87IyjyTIsiwdX/
+7asb9rps6d7Q7GgAARQ4FMwAAAAAAp7mieVPNfH2Yrtk9W7UWvqWn6knTJrwhDw/
+CQ0AwJnYIgMAAAAgDNUrFhRo595yu4YAAAUeS59+dUYE2WM2WKM2W6MGXqOY243xmw0xmwwxnBLXgA
AAAAAAAAoJly2gtkY4ylpvKSOkvZJWmmMmWVZ1sbTjqkraZikqyzLOmyMiXBVHgAAAAAAAABA4XLlFhk
tJW23LGunJBljvpLUXdLG047pL2m8ZVmHJcmyrHgX5gEAAEAx58jM1KZflynpwCEFh5ZXo+vayifA3+5
YAAAAQKnlyi0yqkjae9rjfadmp6snqZ4x5g9jzDJjTNTZTmSMGWCMiTXGxCYkJLgoLgAAAIqyzBPpipn
4mZLqhCnwns46fnlNxXz0pdKSj9odDUAJs2RFrG594El1emCYbntosGL/
WmN3JAAAiixXFszmLDPrjMdekupKulZSH0mTjDHl8jzJsj60LKuFZVktwsPDCz0oAAAAir41Mb+oXO/
rFFj75JoFvyrhCu9/k1bOmW9zMgAlyao1a/XYRz8osctQZXYdrISop/XQuC+1acsWu6OVSn///
bcWL14sFpsBQNHlyi0y9kmqetrjSEkHznLMMsuysiTtMsZs0cnCeaULcwEAAKAYSjmaouCKoblmXoH+O
ubMtikRgJLojclfyu+mx2U8Tq7HMp6e8uv6qF77cKI+eeNlm9OVHpZlaciwl3UwyVvlwy/
T4c8+UrVKRi+/NFTGnG09GwDALq4smFdKqmuMqSlpv6TekqLPOOYHnVy5/
KkxJkwnt8zY6cJMAAAAOIuIsDDFf7XQ7hhnFR+3TxHVIuWMi1d2RqY8fX1yPmc5nXLsTdBhm7JHhIXZc
l2gtBg8/FklJCa79ZqxuxNUqbN3rpmHj5+Wrtmouwc8lGt+5uOiJDwsRGNHFd9C/
MNJn8kEXa0rG7c8NemgnZsXaeq06eob3dPWbACA3FxWMFuW5TDGDJI0T5KnpI8ty9pgjHlJUqxlWbNOf
a6TMWajpGxJT1mWleSqTAAAADi7saPG2B3hnKKiojTlw0natGWz7njzBXn0ujpn9VpmTKwmvjBa119zr
b0hAbhEQmKyKvV62q3XLDvlPTmOp8grMDhnlnU0SeWr11OlXg/kzCq5NdXFO/
j1a3ZH+E9WxG5W82t65ZrVanCNfl38EgUzABQxrlzBLMuy5kqae8bsudM+tiQ9ceoHAAAASonFS/
7Qh999pazsLHVt20HRt/a84FueG9ZvoLfveVSvf/
qBjphsBVkeGnhzL8plAIWq5U29FPP5SIXf9oR8QyspPWG/
Er97Uzfe84jd0WwzbPgIJRbiSvK9cTsueEzCYaPm1+Sdx8bGKioqqtCynE3VarUL7VxhYSEaPWpkoZ0P
AIoilxbMAAAAwJnGfTJJn2xbJt8bmkoeHhq7ZpkWP7NC748ae8Hntruytdpd2doNKQGUVkEh4epyzyNa
Ne8bHU09puDgcrrp/
ifkFxR84SeXUImJyere273bbfw872vt3rZcNeq2yplt3fiLOt1wh9pd282tWf6LmV89a3cEAHA5CmYAA
AC4TWZmpr5eulB+fdrlzPyb1tXKn1Zq+/btqlOnjo3pAOAk/

+ByuqrnvXbHKNWu63S7pk55S7u3L1GFKo10cN9aBfhm6/bo/7M7GgDgDBTMAAAAcJu9e/
fqRESQfM+YZzeoot9WLKNgBgBIkowx6nv3EzqcHK+DB3aqdcteKluOG6sCQFFEwQwAAIBCdb69MR0Ohx
KDslXluma55mkbd+ujpQv1zZQvLvqcBRETE1Oo5wMAuEb5kAiVD4mwOwYA4DwomF3Esiz98vP/
tHX9BrVs11bNW1wuSUpKStLo0aM1fPhwhYSE2JwSAACg8F2ovH36lRe1cNMe+TSsLknK/
DtZlxzz1A9//
OGOeACAYig9PU2zZnyso0ePyXI6dGmj5rrq6i52xwIASPKwO0BJlJqaqod7RSv127nqlJChde99rCEDH
1J2dramTp2qDRs2aOrUqXbHBAAAsMWrw59TX6/qKvftCgV/
u1ztt2fpq3fetzsWAKCIsixLkyaOVL3GvdXp5hfV+ZZXdCTVTz/
NOfu7XgAA7sUKZhd49+VRGlLvclUsU1aS1LtseS3du0ufvv+hFixYIMuyNH/
+fEVHR7OKGQAAlDrGGD0+4EE9rgftjgIAKAbWr12qmg06qnxoZM6sUfMumjv9WWVnO+TpSbUBAHbib2E
XOL7/
oCo2qplr1rpqTX0483s5fTwlSU6nU1OnTtWgQYPsiAgAAAAAQL7sjduh917rY9v14xNSdOs9U3LNTqQd
1fHUY3rrlZ7y8/
OzKRkAQKJgdgmHTJ5ZpsOhQwnx8qkQfvIYh0MLFy6kYAYAAAAAFGlVq9VW994v23b9rZtWacPWJWreqq
ecTqcWzB4rSYqs0VxHksuqTu06ur7z7bblO5+ZXz1rdwQAcDkKZhdo2uFq/
bxig66vUS9n9vH6lbqi43XasHGjHA6HvLy81KFDBxtTAgAAAABQ9NVr2Fy/
LJypvRG1tXtHrBo27qhqNZvlfH7pr59o6+a/VK9Bs/
OcBQDgKtzkzwV697tbe2pV1HOxv+i91Us1PPYXVevWSU8PGSIPj5O/5B4eHoqOjrY5KQAAAAAARd/
9D4xQ2pE12rVtaa5yWZJatr1DS36fZ1MyAAArmF3AGKOHnnpSDodDKSkpKleuXE6x3LFjR82dO1edOnX
iBn8AAAAAgCIvLCykyGz14OuTd0tK4+Gp/
XHri0zG04WF8e9+ACUfBbMLeXl55SmRo6OjtWfPHlYvAwAAAACKhdGjRtodIcezz43R4aR9Kh8amTNbs
3KGnh3+iLrc2NnGZABQelEwu1loaKjGjh1rdwwAAAAAIq8gwcPauQr7yol1ch4ONS8SU2tWT1BPkGNF
BLRUAfjlmjNyh/00duL7Y4KAKUWBTMAAAAAAChyMjIy9PCjL6rdDa/IxzdAkrRr6+
+6rJHU8bortGXLDrW+u7fuv/
9nGZN36wwAgHtwkz8AAAAAAFDkfDt9puo3vTOnXJakmvXaauWqnbrsssvUs2cPRUZGnucMAAB3YAUzAA
AAAABwm6ioqHwdd/DvFN358Iw88/
iEY+rcubM8PP5dM5ffc+ZHTExMoZ0LAEoDCmYAAAAAAOA2+S1wlyxZps+/
+1mNLr8lZ2ZZlipVCNCXn813VTwAwEViiwwAAAAAAFDktG7dSh6Z67Rtw8+yLEtpxw/
rt5hXNOC+W+2OBgA4DSuYAQAAAABAkWOM0bh3RmnOnBgtWDhKZcsE6o1RD6hq1ap2RwMAnIaCGQAAAAA
AFEkeHh7q1u1Gdet2o91RAADnwBYZAAAAAAAAAIACoWAGAAAAAAAAABQIW2QAAAAAAAAAQCFJS0vTl5O
nKG7TDlWsGak7B96j4OBgu2O5DCuYAQAAAAAAAKAQpKSk6MnogWq5PUjPVOqm6w5W1NC+Dys+Pt7uaC5
DwQwAAAAAAAAAhWDSWxM0uG4P1Q+rLkmqUb6ynm8crYmj37Y5metQMAMAAAAAAAABAITgc97eqBIfnmpX
1C1JW0nGbErkeezADAAAAAAAAKNWioqIK5Txp+5P1ZJUu8vP2zZllO7O15K8V//kaMTEx/
zWeS1AwAwAAAAAAACjVCqu83bxps94c9o6GNe8jY4wsy9LE9bM1+oM31apN60K5RlFDwQwAAAAAAAAAh
aBBwwbqOqSfug54XFde0kwZ3pZuGtizxJbLEgUzAAAAAAAAABSaK1q3kleVYL389Xi7o7gFN/
kDAAAAAAAAABQIBTMAAAAAAAAAoEDYIgMAAAAgDPs37xGG5f+IsuS6jZrpZrNSu7emQAA/
BcUzAAAAABQTPz2xxJN/m6GHJalm6+9Wrd16ypjjN2xSpy/5s/Q3hNSWM+hMh6e2rhktvZ/
+7Ha9rz3vM87kXJEO/
78Xb7+garVoq08vbzdlBgAAPuwRQYAAAAAFAPvTPpYT8xeoJ0db1Fc1O16dVOcHnnuRbtjlThZGenaHR
eniM53ycPLW8bDQ6Ftuyspy0PHDyec83nrfv1RMd99qUOVmmind7h+GD9aCbu3uzE5AAD2oGAGAAAAgC
IuIyND3y6PVUCHG2U8vWSMUUDzVlqemqG4uDi745Uoh/fvkm/Nxnnm/
pe20cEt68/6nLSjydq+a48q9RmqoBqXqOylrVX5/
jFa8uPXro4LoJSZP2eenr5zsJ6JHqYXH39BycnJdkcC2CIDAAAAAIq6Xbt2Kb1iVQWcMXfUa6Tfl69Qd
LVqtuRytb/
jdurv1x9w6zUzMzN1POQShVzVLdc8bWusDq2fq9TFeUvjA0lHFHzvm7lmxtNTDv9yWvHKvfLx8XFpZgC
lw/dffqcjsxP0Qq3hMsYo+cRhDb97qN6ZPk6+vr52x0MpRsEMAAAAAC4UFRX1n8/
hcDiUFBSqgHbX55qnbVqr9/78Q1MmTyrwuWNiYv5rPJdp1LyFEhLdvzovfvM2Hd+9XoE1GkmS0uP3yrn
uF9W+9N+VzX/H7VSFarVOft7sV0rqkbwnyjyhCtVq2VIwh4eFuP2aAFxryQ9/aGS9ETmPQ/
zL6+6KffXdl9MVfW9fG5OhtKNgRqnidDo1edxb2rnqDxlZqtSwuR4cPExeXvynAAAAANcorAL3sRde0p
Jtm+RXt6EkKSP+kC7JOKbv//i9UM5fFI0d9bIt183OztZLb47TsplzZElqXrm8Rv1vtvz8/
HKOiYqK0vNDB+vIkSOqX7++OvYfLKvBFTIeJ3eizEpJ1uUR/po2/
j1bvgYAxUd+X4isnFRBqpd71ji8kZ4cO1RTvvncBclOKsovRKJoMJZl2Z3horRo0cKKjY21OwaKqVeef
lQdnFvVvHIZSdKWhOOadjRCoyd8YnMyAAAA4PycTqden/
C+JnwzXQ0vvVSNKlfQy08NzlV6wj2SkpLUrPMtqnhND2UHhihgT6xuufIy/
bR8jY6WrS6TdUI1zDF9OHqEgoOD7Y4LoIR48vbH9XLd53LNfj+wRCe6ONTttu6STpbVFMJFQwn5vTD5O
Yhlmyg1kpKSZPauU/
MrwnJm9cMDVXnfbu3atUs1a9a0MR0AAABwfh4eHhoy6CH9MmeWYiaOsztOqfbgiNGqNuQzeQWcXLiiVl
309bej9NNbI5Weni5/
f3+VK1fO3pAASpyb7uuqNyeM08MNBsjXy1dbk7dpxrHZGtdjvN3RUMpRMKNY+S/71x09elQP1/
XIM29UzqjX7T0VEhp2lmddnBLwyhQAAACA8zhx4oTOnzl+0+5fIqzdU99Mf0H/V//
e2xKBqCka9+5g8IrRejND8bLmZ6tyEur6c233panp6fd0VDK5atgNsZcJekFSTUk/fOn1rIsq/
YFnhcl6Z1Tz5lkWdaYMz7fT9LrkvafGr1nWVbB706BEu+/FLjHjh3T2P4355mvSLI0a/
YcVaxY8b9EAwAAAFCEFcbNFqWT+zMfCqyh6mfMjYenPvjgQ/
343dcFPjcLVgBcSKPGjdRofCO7YwC55HcF8zRJkZIyJDny8wRjjKek8ZI6StonaaUxZpZlWRvPOPRry7

IG5TMHUGBlypRRhcuv03eb/qce9UNkjDRrw0GpVmvKZQAAAKCEK8zy9paBjysh/bg8/QL/
HS6brl9jZiss7L+/MxIAgOIkvwWzkfSsZVmjLuLcLSVttyxrpyQZY76S1F3SmQVziZKSkiI/Pz/
5+PjYHQVn8dBTw/W/
ec01csZUybL067oT+mPya3bHAgAAAFCMTHhxiO56+kUlRTaXMzBUAdt+16PdrqFcBgCUShezgvlGY8xy
SYf/GVqWteo8z6kiae9pj/
dJanWW4241xlwtaaukxy3L2nvmAcaYAZIGSFK1atXyGdm9Vq1YqSlvvK0K8lJqdpb8qkdq6JhX5O3tbX
c0nOG6zlG6rvPJt8dFRUXJmHzdEBMAAAAAJEkVK1bUvM8maO3atUo+fFhXPvGK/
P397Y4FAIAt8lswD5ZkSZp/
xvx8u4ifrbWzzng8W9I0y7IyjDEPSPpMUoc8T7KsDyV9KEktWrQ48xy2S0lJ0ZSXx2hMm845ZeWOpAS9
9sxzeua10TanAwAAAAAUNmOMmjRpYncMAABsl9+CeYrylsMXsk9S1dMeR0o6cPoBlmUlnfbwI0mvXuQ1
ioTpn3+p+
+s1y7UStnZouFJWrJNlWayQBQAAAAAAAFAi5atgtiyrnyQZYwJOPU7Lx9NWSqprjKkpab+k3pKiTz/
AGFPJsqyDpx52k7Qpf7EvbMTQYUpOTCys02nHvjw7d+RIOXBIN/
Tol2eefOhvt23BUDuy6oUPyqeQsDCNHMPKa7hHUlKSRo8ereHDhyskJMTuOAAAAAAAALgI+SqYjTFVJH
0pqd2px4sk3WlZ1v5zPceyLIcxZpCkeTq5lcbHlmVtMMa8JCnWsqxZkh4xxnST5JCULKnff/
liTpecmKiXo3oU1unOa9v+vfr2tz80sGW7nFlWtkMBQUH6qv8DbslQmJ6NmWF3BJQiU6dO1fKlv6l/
7wWqWsFPDhOkqzr2Vp87+9sdDaekpaVp5g/
fKD39hLrf3IsXAgAAKIKys7O1bds2hYaGKjw83O44AACgFMnvFhnjJbW
VtPTU46sljZN0y/meZFnWXElzz5g9d9rHwyQNy2/
YoqpularNchX7y37Rd0bNNHBY0f1xfpYDbz1drujAUVaUlKSZnz/
veqU+1tP317p1PS45v05UbPLlFXXm/
lvyG5Llvymie89pVbNTsjHV3r6icnqfOOj6nn7nXZHAwAAp0yfPUfjZszW8So15HkkWXWMQ5NeHcVN5w
CglPsr9i99N2m6lGmpauPq6vdgP/n6+todq9RxOp2a/d0PWrnwD/
mXCVTvgXerdu3adscqVB75PO4aSc9YltXWsqy2kp6RdK3LUhVD/
bv20HXXX685CXHa5WvpxYEPq2alynbHAoq0qVOnKj0lTo/cXCHXvFNTb/
08+3ObUuEflmVp4vhn1ecWS7Vr+atqFX/d2tVTP80Zp9TUVLvjAQAASfv379ers2Nk9b5fAe2ul2/
X27XjyuvlxEuv2B0NAGCjmJk/6edXYvRUuUf0TKWndMWmxhp895NyOp12RytVLMvS0Acel9/8/
XqmUnf1926jz54cq18XLLQ7WqHKb8GcJqmeMcbLGOMtqZ6kE66LVTxVr1BRd0d1UdfW7eTr7W13HKDI+
+WXX+RlsuXnk/uvImOMPJVpUyr8Y9u2bYqseCzPPvKNGhzXwoXzbUoFAABON2HKF/
K8vluumU9YhNbFJ53jGQCA0mDBF/M16JIH5OPpI0lqGNZAnX07aMHcBTYnK12W/
bFUTdMjdE21pjLGKNgvUMOa99HsSV/
ZHa1Q5XeLjG8kPSrprlOPPSS945JEAEqN9u3ba+qUbdoUl6aG1QJy5sfSHPIvV83GZMVHVFSUy86dkZG
hyIpHJEXkmh8+6tCoV8ZowoQPL3iOmJgYF6UDAMB1Bg8frvjEol3Q3jVgoCRp9ZatCnq8VZ7PH4iPzzn
GnSLCQjV21Ci3XxcA8C+n0yn/
DL888zYVrtQHv3+qzjd1tiFV6bTk50XqG9ks18wYo6Asb2VlZcm7hCxQzW/BPETSMUk3SrIk/
SiJ7xoA/CfR0dGaN2+eXv12te7r5FCbS8to674MzV4brNcnjrY7XrFQ2AVuVFRUrnMOuL+7Uo/
v09oNx7R128ltMQ4c8tOvi36Tn1/eb1gAACgJ4hOTFNarn90xzinstI/
1jO3DAAAgAElEQVSb743T0gVzVKF7r5yZ43iqgoKCbPka4r/+1O3XBAB3GzFkhJKL8AuR/9d/
kBIPxEuX5Z7/eegvLT+wXA/f95A9wSSFhIVq5KsjC+18I4Y+U6R/Lxb/8bua1yyr1lVz/
2ZsP7Bbjz4wKM87ht0lJCxUI8cU3nZa+SqYLcvKlPTcqR8AUChCQ0PVuXNn/
fijQ2uPV9XeTV6qf2kzvf/03dx4oIh49fVP1ef2a3VFM6lf36qSpP0HM/To/0Xrg4+
+tzkdAAAIqVpNldev1f6pHyuweStlJRxS5tpYXXf3fXZHA4ASKzkxSS9eM8LuGOc103uWvto8Xb3q3yp
jjOKPJ+ibnd/p/fvek5dnftebFr7nFxVeuSyd/
L146foBhXrOwpTpyNKID15To4jaKuN78p3bv+5epRYNm+iujrfZluu5ny/8juSLcd4/UcaYFEl3S/
rsLJ+2LMsqW6hpAJQ60dHR2rNnj4YPH66QkBC74+AMAQEBiowMVJtW/
65WrlLJV7vjdio2doVatGhpYzoAACBJzW64SQ2OHtW+DesUVCFcFf/
vCdtWRAEAiobubbpp4Zpf9cyKF+RtvOUX5KdhdwyxtVwujXy8vPVk3wf02pxpMlmWHFa26tSsrTuvvdX
uaIXqQn+qkiRlSUrWya0x/
mHOeIxT4o8cltOyVLE8RRmQH6GhoRo7dqzdMdxm2LDBSkyMtzvGefXvf1fOx6mpx1XWP15SaK5jalSzN
HzYo6peo67bcoWFRWj06NLzZwUAgIvhX7as6rZpa3cMAEAR0qHJterQ5Fq7Y5R6EeVCNfSOQXbHcKnzF
syWZdU89eEcN2Qp1g4mJWrC9G9UMyhYxnhoe0qyenS4Xn9t2aQsh0Nd2rRT9QoV7Y4JwGaJifG6rVuw3
THOI3c2hyNQU77I+3ri5i0n1KNrQ9Wq6b6vZfqsol3MA0BJMDPmJ30ya6bSjVHVoDJ65cnBioiIuPATA
QAAUGrla128Maa3JA9J30v6UFIlSc9alrXchdmKlXHfTNPo9jfJ39tHkpSV7dAd0ybpra695Ofto8/
mz1NIZBX1bH+9zUkBIP+8vDxVtWqkfll8SNdcVVYeHkbbd6Yp7oC3OnYKu/
AJAADFxhffTdfbq1bIt8eNMsZo3fE03fr4o5r30WQFBATYHQ8AAABFlEc+jxspqY6kvpL6SLpS0nhXhS
pudhzcr6ZhlXLKZUny9vRSn6YtdSDlqEICAvV4m+sVt3OXUo4ftzEp/
pGRkaG3X3lB3ok7NaTfbZr+5RS7IwFFVsfrLlNoWH19/nWqpnyVot37QtTvzmvsjgUAKGSfz4uRX/
t2OXv3egUGKK1DW3305ec2JwMAAEBRlt+dvSMl7ZZ0laSPJcVKesdFmQrFjn171WfSu265VkpKiu6r1j
DPPNDHV2lZGTmPr69ZX7e/
95rKh4bmORbuNWTAHRoQmaYHo5tKkuYv+VwTDuzTQ08NtzkZUDQ1bVJNTZtUszsGAOAiRUVF5fvYzZ5S
1TNmftWq6p2hL+in72YUbrBTYmJiXHJeAAAAuE9+C+ajvpJqivpZZ28yd8JF2UqFFc0babkxES3XMuy
LM1du1F9m7bKdbfm/23bqJc698h5vPNwompUidTR9BOqHXnmt+9FR0hYyX7b+8rly9TCO0m1Q//
9OjvVLq8Xly9URsaT8vX1tTEdAABA4bmYAvfGAffrqGXl+n42ffNWvThkiO68vZeioqIohAEAAJBHfgv
m9yU9J+mgTu7D/Jqkda4KVRhGjhnt1ust/2OJhrz2pnpE1pGXh6c+jP1dN9W5RD5eJ3+Jj2Wk60/

```
HcX0143vdcMMNGj/
pI7fms8uIoU8rOTHB7hi5xO3cpjGtyuSZl8s6ogH97lRQYKANqfIKCQvXyDGv2R0DAACUEoPvuEtPT50
i75ui5OnnqxN79yk8dq36TH7M7mgAAADFTrYzW5Pnfq2k+AQZGfkG+WtAtzsU6Odvd7RCl6+C2bKsF4w
xb0tKtSzLYYz5P0kO10YrXlpd1UZNvm6un2Ni5HA49MZzj2ncyFFatWKhjCQrtJxenDgu14qQ0iA5MUH
Pd2pgd4xcNsWV1ZK/
lqtOeO6SOT4tW6NuaypvL0+bkuX24vzNdkcAAAClSIerr9YXFSrojU8m62h6ulrUradHPvhQXl75XZMC
AACAf7z97WTdFnmVLqlfQ5J06FiSxnz+nkb2f8reYC5w3u8WjTHv6uSey/
eeNvvnQ0vSoy5LVgz5+fnppptvznn80rtvybIsSSp1xXJR1rBaJf2wyGhl3GFdUa28nE5L01bvV62atY
pMuQwAAGCHBvXr6yPeQQUAAPCfHElNkV+GdElEjZxZxTKhuiKsntbv3qJGNerbF84FLrQcYZCk30/9fC
YK5nygWC6anu7bVd8vitUPC+MkY9S2WSO1aVTH7lgAAAAAABRJWVlZmvn9HG1av1Wtr75C113fns4Dtt
uxb7f6fjrc7hh5pKamqm9E2zzz+iFV9cT0t1QuLMSGVK5zoYK5vaSNp34GiqV1O/
frt9UbVD44SF3bNFeZAD95enioZ/
uWdkcDAAAAAKDIO3z4sB7tP0xX1+iltpXu1dpZv2v6F49p3KSx8vb2tjseSrHakTX00vUD7I6RR6YjS2
98Oj7PfNHe1Zp830hVLB9uQ6p/
Pffzh4V6vvMWzJZlLZIkY8xuSV6WZe049bi22IMZxcB70+epmvcJPdqkkg4eS9ebn0/
X7Td0UMNqle2OBgAAAABAsfDGqHG644oRKht4ctXllfWjFLK/oj7/dJru7X/XOZ/
ndDr17dffa8Xiv+Tj66W+9/
ZUo8sauSs2YBsfL281rN9AE2Nn6J4mN8rb00szNi+Ss4y37eWyK3jk87ifJfU77XG/
UzOgyNqw+4AivU7ozsurKtDXS3XCgjT2hgaavuB3u6MBAGwUFxenwS8P1cChD2vJsiV2xwEAACjyjiVk
5JTL/6hXpanW/7nlvM978uFhOhJbTj0bPqMbqj2mT1/9UbNm/
OjKqECRcXO7KLVo00qvrf1GL8V+oXJ1q+iB7nfaHcsl8ntL6CqSdp/
2eM+pGVBkLV69SQMvrZBr5uFhVMbTqWynU54e+X19BQDcw7IsLVr8q/
bs2aHr2kcpMjLS7kglzsyY2Xpp5usyN4XIw99Ly2OeUadfWuu1YaPsjgYAAOBWUVFR+T42LV6KbpF7lp
GVrj+WLj7neY6lpOraWn3V9NKrJEm+3n7qeeUjGvZKL41//9187d8cExOT74xAUXRJ9bq6pHpdu2O4XH
4L5p2SBhtj9ksykp48NQPOa8e+g7rr44O2XDsl/
pBurFJR5fx9cs13Jh1Xv09+5WYEAIqU5ORkDXzsdoXWTVBwhKX5r09Ug4qdNGLYaLujlRiWZenN6e/
J+66KOTO/DuH6efZS7d69WzVq1LAvHAAAgJtdTHn73Tc/
aPGimWrXsHvObGbs+5r8xYScLS+ioqJynfPt18erXtYNec7VvGEbPTf+IQUHB/
+H9ACKkvwWzK9K+kzSP+9jMJJK5ppuFKrakZX0fKcGtlz7eHqmXv3kG71xY7C8PE+uVl574KguqV1dD9
zcwZZMF+PF+ZvtjgCgEKWlpWnO3JlyOrN10403KygoKNfnR7z8mJp2Pyz/
QD9JUsXq0rpFP2nZ8m66slVrOyKXOAcPHlRqWJbOvA2Ns5mf5vxvrgbd95AtuQAAAPLrmWEjlJSYbMu1
43bt18LVM1TWP0zJaYcUEOKh997dl+uYB/o/
nPPxgf0HZWpWVWhwx1zHbNuzUYMfHyIPN76rODQsRK+MHum266HkS8s4oRmLY7T/
74MqX7a8el7bReWCSu+LJvkqmC3L+twYs0fSTadGsy3LWuy6WMB/
F+jno7u6ddLQeYsUaLKV6ZTKh4Wrf7eiXy4DKFkW/fY/vf3RcFW/
Ik3GSN8MelcDokcoqlPXnGOSju1U7cDc/1tu2MZbX3//
CQXzRTjfWz2zs7OVEHRIlW8MyzVP35OqT2d+ojnfznJ1PN7miRLLsizNmTdP3y2Yr/
Jlyuixe+5V9erV7Y4FACVOUmKyHu5mT1GakHxIX836VE6H5OtfTa2aX6W2Lc797+vs7GyNem+4GlZrrr
KBoZKk2K2/6LIGTdWraz83pT5p/KwRbr0eSrbU9DSN/PgtPdz0ZjVs2UlxR/
7Wa1PG6/9636dKIRF2x7NFflcwS9IRSQckzZJUxRhT1bKsva6JBRSOOlUiNOLenrIsiy0xgHw6fjxDPj
5e8vb2tDtKsXG+UtOyLCWcWK++w6rImABJUtW60ojRD+rNseNyVm4kZ+5RS1XO9dysTKf+9/
PCi9of71xKS7F5oa/z4RGPaumebfKpfnIFeXZaliptCtD/lizKs4rmzLd5Aji3/k8/
pVUhZeTf+Rplp6Vp8Ssv6eU+fRXVnhf2AaAkSDuRqvenvKUHbxylAN8gWZalmD+namHmXHVoc+NZn+Pp
6alH7xuuL3+YqMwTWcq2HKpbu4Fuv+luN6cHCte0n2fqqSt6q3q5k1vvVStXQa9cfb/
GxkzXU9EP2pzOHvkqmI0xvSV9LslD0lpJwyQdl3Sz66IBhYdyGbiwbTv+1sKFfymkvKUTJyQv7zK6/
bbW8vKiaL6Q85WQq1at0sSZ/
fL8PXT59WUV3e5ZXX311ZKkF15+Won7Figs8t9949cusDRz+gLVrFnTNcFLoXeef0PPvP6clv+xSlnKV
m3/
yhr8+HN6aMSjSkxLUvWQqhr24FMKCwu78MkASJJiV63SX36eCriiuSTJKyhInj27640vP1fna9vzfRgA
lABzf52hnlcNUoDvyRfpjTG6oUVfTfhp2DkLZkkKDiqrB+940l0xAbc4fPiwqtevmGsW5BsgZTltSmS/
/K5gflHSQknXn3r8o06WzMinmd98q99/mCNfp6XM/
Ye0fu1aNWrc2O5YACDp5KrlhQtX6v67InKKgAMH0zX9++XqfXsbm9O5xpBhg5WYGO/y6xw9clSmwglJ/
rnmqSlZeuOtV/
XZ55MkSU6nU5sXZCog7IiCwzx0aGe2yvrW0Mujnnd5xosRFhahV0ePtTtGgXl5eenVYaNyHm/
etkV3jOov3R4qr0Bv7Tq8SUsf66nZ70y3MSVQvMxe+D95Nbks18wYo6N+vkpLS1NgYKBNyQCg5Nm9d4e
eGh/t9usmHTimG/r3zzM/
knLYljzAP0LCQvXczx+69Zq7EnfoeOYJBfr8+288p9Op9Qm7cmXZsW+3akfWcGu2/
AoJCy3U8+W3YK4s6WP9WzBn6cx/
KeOc5nz3vY789KtebtJWkpTd5CqNGDFST058R5UrV77AswHA9RYv2aIbO5XNtcqsciU/
paUllNgtZhIT43X1be64CUOwPpu4QVkZTnn7ntyCwZHl1L5NDvV7qEauX9tr1V4n0jKVmpKu0JvLyMOj
6P26/
zbd9aW8O42cOFoefcPl4XNypb53eT9l3CaNef91m5OhtBo8fJjiExPtjnFOdw3IWy7s2bdP6R2vkXe5s
rnmibt2a+Cjj7jtJk4RYWEaO2q0W64FAHapUbW2LXswL/
h9ltbvXq7Lal6ZM7MsS2WDy+qpB95ye56LxR7MJdfIMa+4/Zrbt2/
```

X60+8qucuv0MeHh6yLEvvb5itYW+8qKuuaZdzXFRUlMZPet/
t+eyQ34J5naS7Tn18p6QoSWtckqgE+u2H2RrZ6N8VgJ4eHnqySRt9PuF9DXn5JRuTAXC3vXvj9db4olf
QJcQfUuvLy+SZH0/
L0lvjt5fIgtmdut3eVnOmLFGZMKeMkVLijW66te1Zf139A3zkH+BzlrPAFZIdR+Xh451r5h3ipz3J+87
xDMC14hMTFdLrFrtjnFXIOeZlHQ7NHT9RAXVqydP/5BqU1I2bVbFmDYX16O62fPFff+
+2awFAadO+9Y0aM+FZBfqVUa1Kl+p4+jFNW/SWOl/
b9cJPBkqYOnXqqNdzD+q5dyfJL9Mo3TNbXe7vmatcLm3yWzA/
KWm2JCPpbknJkga7KlRR9F9usFT+cKrUKPdbzEMDgxTz5RT9Erviv0aTVHpu3gQUd1WrRui2bu5YNXtx
4vaG6Lc/VqpL53/rA4fDKT/fAP3ffXVtTPav6bNS7I5QYOVDg3TngE5KOXJCkqXgcgF2RyoyBg9/
SvGJCbZdf8e2bSp/
cwMZr39XWDqOZWrjqm2K9K2guwb0sy3bhUSEhWvsKFZaw36eXl7qcNcdWv7V90o3ksnOVsVKldT05m52
RwMAFBIvTy89/cCLmrNwun7Z9I28vX3Uo2tPVa1Uw+5ogC2atWiuZlMm2B2jyLhgwWyM8dDJG/
pdJqmZTpbMSyzLOuzibEXKfylwh/R/QJkOh3y8/v3lXnNor+4Z9LDuGnB/YcQDgP+kWtUQ/bU6VDN/
TFSbVkFKPuzQ/35LVY/ube2OVqIEl2N3qTPFJyaoXHR1267ffIePVnyxQuF31JHx8pAzI1tJU7ar/
cOdFBSWd1V/URI/dY/dEYAcgeXLq8O9d9sdAwDgQj7evrqlc1+7YwAogi5YMFuW5TTG/
CbpCcuyPnZDphJn4NCnNPzxp/XopS1VtVyIYg/s0bS/
d+nNV5+1OxoA5OjetYUOHUpR7OqdKhscoAH315aPt6fdsQCXiqhdSa1MK62dvErZ3pa8nV66ukeHIl8u
AwAAAEBRkd8tMr6UdJMx5hvLslJdGagkqlW7tl789CN98cGHSti7Xpdecbnein5e3t7eF34yALhRxYrB
6hLV1O4YgFuF16qo62rdaHcMAAAAACVcWlqafH195elZshZz5bdgvluSv6Sjxpjjp2aWZVllz/
McnCYkJESPDBtqdwxISkxJ1bzla2WM1LllE4UGB9odCQAAoEg7vHefVv40Tw5vL8nhUJWqVdW4c0duAg
sAAJAPSxYt1rfjP1OoM0DHnOmq2KSWHntuSIn5Xiq/
BXOiJMuVQQB3WLByvTZsWK+7mlWW07L06Xez1LxJE7Vvfond0UqNNav/0vfTJsnDy0vR/
Qapbt2icQM5AABwdhmpqVo8a44qDbxH5tRqm6RVa7QmZoGa3tDJ5nQAAABF24EDBzT7zc816vK7cgrlP
w9t1YTX3tbDQx63OV3h8LjwIZKkepLelPTnqR9jT82AYiM9M0uxa9bq+evrqmZooGqHBemljvW09M81y
nQ47I5XKkx4Z7RmTbxHPeqsVJeqS/TxqF76+ku2dgcAoChbv/
BXhdzSNadclqTg5k10YN9eG1MBAAAUD1M/
+FQPN+yaa7Xy5RXrac+fW2xMVbjyu4L5I0l3SPrnV6KHpJY6uXUGUCys3rFPHWqUyzNvV72M1u06oMvr
VrMhVemRnJysHX/O0P3X+UmSfCXdda2nxs+drB4975CPj4+9AQEAKALi4/Yq/vV37I6Ry/
6EeFXqknelcsaxVG0uYlkBAAAKKioqyiXnTN2XqAd7tsnzuQO79170NWNiYgorWqHKb8HcTdIMSU/
r5Krn107NgPMKCQvXi/M32x1DkpR8+LDaeWfkme85kq4/Vx/
SnF1pNqQ6t5CwcLsjFKqlS37X5dWOSyqTa14vIlWbNm1SkyZN7AkGAEARElGtqkJ63WJ3jFx8Y1dpZ+x
fFKtvy8pyZlZ2tgKBANRhwv43Jckv++nu7IwAAgGLMVeXtooW/
aM5ny9S9zlU5sxNZGarSuLa+mPSuS67pbvktmH+RtNSyrJ2SZIxZIvZkRj6MHPOa3RFyGdT7JqVmOBTk
e/KPfkp6lvb4VtGXX35nczLXe2boYCUlxhfKuXbvu/
jzHDuWok71M9XsjC2XN+5K0YzHHpOvr2+hZPtHjciIQjtXaFiEXhkzttDOBwBAcVLj8mba8dFkHfXwUH
DzJspMPqyk6TPVOor9lwEAAC7k6vbX6qU5P+vY5gXqXPUK7Tp6QNP2/65nJoy2O1qhyW/
BHCJptDHmn1XLrSX9boyZJcmyLKu7S9IBhez5cZ9ozNBHFHDskCxJizbt1fe/
Lrc7llskJcZrYJeQQjpbwc7zxkdzdfhYlsqX8ZYk7U/KUJYJ0puPFO3Vyx/8WDjFPHC6v/
cf0ZJF6+XIylJktYpqeXV9eXrm99YIAOA+xhhdd/+92hW7SnGff63AssHqGN1b/
mWD7Y5WqqQdOax1/5uvtGPHVPOypqre/PISc+d5AABKMmOMnn/zFa1ZvUbf/
zhfkU2q651bJ5eorULzWzBfferndqfNrj31MyuZUWyEh4fr9cnTlJFxcquMFd27KzQ01OZUpcfAvtfps
xlLZGUdlSXJL7CcBvS56oLPA0qazev2ae3atWrbI0S+/r6K2/
K3pk7arzsGXEdZUAQk7YnXjpVbFRAcoHrtGsnHv+R84wcUlPHwUK2WLVSrZQu7o5RKf2/fpmU/
z1f4LX0VVD5E21f+oR2TP1CH+wby/
w0AAIqJJk2bqEnTor3ArqDyWzDXdGkKwM0KezsG5E9QgK8e6Nve7hiA7WKXbVDUfWE5pUC1+oFKS0nRx
r/idGnz6janK92Wf7tYh/
2Pq0znikpMztDOybPUuks7hdesIGe2U+kpJ+QX7C8PVpsDcKNVP89T5fsekfE4+XdPuVbtlOxwaP/
6dYq8rLHN6QCUdJZlKTMrQz7evryoBeCs8lUwW5a1x9VBAHc4dOiQPhr7ijKP/
C3Lr4xOHD9udyQApYxlWfL0zc7zzXmdpkFaOn0fBbONkuMSlOybqtBuNSRJ3mV95f/
wJfpz4jJVrF5J+w8elEeYt5wJWapRs5ouvb6ZvYEBlBoOH7+ccvkf5Vq21c5vP6VgBuBSv8cu1NKVvyn
IJ0THM1NUu1Zt9ejcx+5YAIqY/K5gBo
q9xMREvTywt0a0LKdylX2UnnVEB1cnaMO6tbqUb8xRCmVnO7VsxQ5t33FAAf6+an9tI4WFBtkdy23274
3XtLfcv7/20WPJOuGZIik81/
zgrhOK2+ShaW9td3smnLRj5VYFd6yUa2Y8jdK9HYoPSlH4gHo5833z9qrM6l2q1pQ3eQElUXzcHsW//
qLdMXJkZObdlTD9wF6d2LBaG4tQTgAly7Y9m7V5wzY9fOOrObPfN8zVgsWz1LFdt1zHZmZlaOlfi5SRk
a4rm12j4KCy7o4LwEYUzCg1Pnl3rIY0D1a5U3tp+nl76t3ujfTSu6/p1Y+
+sDld6ZCQnKoVa3erXJkAtWpaQ168xdw2TqelyZ8sVKvLPXRHzyAdS3Xou1mLdHW7Fqpbp4Ld8dyiStU
IXX2be29QlX4iU998sVk1awZr/bLDanRl+ZPztGz9Of+4HhwRJW8fT7dmuli/TU8p1PPFxx1S/
JhDhXrOgkpNSpBp5iHvcrm3UUpPPK6K9zfINSvXMVKrhy5Vegw3AQVKoohq1RXWq5/
dMXKcmPODjq5bpeDLmkuSnA6Hkn+crhufHy0f/
wCb0+WW+PWndkcAUEgWLJqtPlc9nWvW9tIb9f68Z3IVzFt2bdD0WV+qQ+OeKusTqI+/
fF9NLmus9m1ucHfkEsuyLC3ZtEz7E/
apRf0WqlWRRQ4oWlxaMBtjoiS9I8lT0iTLssac47jbJH0r6QrLsmJdmQmlV8rf+xRR3y/
XzMvTQz6Zx2xKVLrMmP+Xjh/

er+uaBirhqENjP1inu269VpUr8Mq2HVaviVPTy4wuu7SMJCm4jLf6RYfr4y/
WqG6dTjanK7nWrtylJtcGKbJOoP6fvfsOs6q6+jj+XTP0Xgak2rCAXUSx62vFKPbeE43G2EuiJmrsxp6
YqLEndo0ag9iNDdSIDRFsiKAgKCBFpZf1/rH2wAE04jB3zuXe3+d55mHmzJ1hPffMaWuvvfa7AybR//
bR1KtnjB89iwOP3Lnok8uFsE7PdRk/cULeYQDQplMVA+8bROPfrY9VRguTGaO+hclzl+i5bBVG/
WYNab9ihzxCBaB9Vbsff5GIlISeu+7BO088xpdvvIpX1qP+vDlsu88BRZdcFpHCa1vVhuv7nVsn/
9cnX4ymYf1GS2yfOPXLBTG4O58MGcPFh95HRWrls0aX9bnioRMYPPZFGjQojsWS21a1yTuEGpsybQqX3
3Mlu3Tdma1bbcF/
Xn6Bx+r156S9TlBPbCkaBUswm1klcD2wIzAGeMPM+rn7+4u9rjlwEvB6oWKR0tGnT58a/
+x34z7j2M5r0K7Zwsq0ufPmM+CdYcv0e7OeeuqpWvk9pWb0uMnM/
nYsR+9SBcAqHWGD1Zpy7aOvcfIvvv+9Hzl6Io899xYNK+cwZ14F3Vbpyi7brluXYZe09z/
4nAP2ar7INjOjYYN5OUVU96qq2vPyQ3VbfTrqs+lssEfcBK6/
VRvW3ypudJ+9ewJDX5zL8EZRHfzF6PF07tq+TmNbWlVVVtRvXZdeWau/
b1m9N+w9zrruPMZXTObLUePYq/duNNtrY54c+xYNOzVd8LqZn37Lr/Y9itN/dWqO0YpIuTAzeu66
UERK3iWXXVRn/9edd9zBeWau/
ILlcre8mP6f5xhPYZ7+96izeUnVL/9u4YJNzad24FQDd2qxKv+H9GTjsFbZaZ8ucoxMJhaxg3gT4xN0/
BTCz+4E9gPcXe91FwGXAKXUbnpSAPn361Pj/DwGP3rX/AvsWj/+4xN8dRyzc3AAaRKUOHVUm4Hcf7ss+/XxVjaYNCpLF9QvsplY9+u78rTt2u5Kk2z+gf9
fHM4u/detBVBg3oVNK0/hzlz51G/3QzGjT2Cb/
dvT2VFJONeHvIlTl77F59hhe/
n9HUrw+z5zShLZ5U5qu04bpg3KjdpYAACAASURBVCfRY2IXTr7hxLzDFREREamxHyu0mj9/
PpPGTmOjPjOAXxnSjirjAZ/
qC5HK1XbvtwgXvXqoEsjM1+PAXpnX2mGwAz/
mf1ggtnMjgGOXHN6Oc1lOSQ1/
lxqsvZc6U8cxV1cy0ifZr5/
IJDsDqu7Hqm2rBbVz/s7ttv5HpuAdvb1ehutcmnaNoYzSteAbxzyfA+fH2lfzR5fRm9v1ZRaGdXn+ZXNHcBm1e7KQEgrtC4ZPzY4x6kxgczmyBr2PPTkxy2S1yTOHN
5+fCXI3mHDP5TQqRlHAxEoyKu9pGUXkdh9Au8J0mJAKV3F
fn/Sbn5YqIbn3BP2DGV8pdeUz+rAA9yE1S9g9aRY
-dWLJmsGWjNsDWsr6ysxWhTV5jQO37OiIF42VKEl3E
GdbdmQAEjXXtR1JmsYcIZQdaKyq9jFTGekQR8Qv+SPngXqgvpfh/+WTIwCsS+1Dkn3ZxmaKNSsrbRw
XEQmyQPEsDUR/qAGanFhK2wKHYHvyvgA3/AaTECgDlt72Wmb+N1kaXtFFzEkmOAGBBKUIbnlGY6Cu
WCsvjzA2zvjRcTg49WHTqhNdVwqTZBR5pj2B1QNd85dZFgF2rJE1TJnEs7zkbvlw0WeTfsZIKbrr
t/T6L+zJvhL2Zc7v9Cdo/vIbkrx7cGfLbHsdbUf2/ApdsNHU7k9Ngrns2j+Jb5Dl2xr8yadWErzqk
3bDUdTL8zvOBVKmX1AX5OvS82NDxZmMT9MCUsNiECpXz9iUw4HUHpfbTWGoqB+HTfmyDlGCFbOGu
49QNpE45EJcTgU8HP2x/WbDX8bsN0HPmbSRTp1CSI0ZYXe/XOYOnxcrQZ3KxRP2WpR9mx8dU4sZwF
PzoqV9bO4C+XVZkZWHYHKTo2tTGDmdzOhHmUEvGhN+ZeoFUbdEJWkxPS3fuYD8NA4yh+pQoZ2ebSQ
7SJgPr0wCqybunbbJtH8zbmFSL1BAYsjJLTx5oJsBoMf8N0TbnaeQ7QYt+QhBGgQ0/9yn9dZqQ27ttf
DZKyJJHcQ8lmYI7QH6oy2iwNMxlpV/sbt6pXJCGqDBIbpK2mIXiC+6YoVMsLd0LKC5zEYjVzAH9Lt
NpUAvXYWjM6hNCIB4fkIJvvFtcvtqlHtJtpyF8Zt2d6p81XWIp0/ZNbHTbTZZtK4v+aD1AMP/T7c
JrjXIVqHd5G4XU/jCJF/Dmy3YMqP3xi7LA2Y5vF+4xW0QXG8V9UkW8LL2zL6fY9WXH0P3oZ6Y2RD
rDR2CZbMs/Rm71Ur9P84qpQ/+2Vr9D2jXG2dUo55YwovlEppFoE1uXVVvUEn1hMY0Tr2FTZoSmRCk
Z48YfqHVX0amFyfrhp2z/FM6bmBDyrdlnxP0oeEpRr3N/+Xn3zPpgbbZbT1QSJJGOvY4Cu6sQ3hr
E3jG/Fc2zPuc/Z3trDD7xbf3CBf+9hSt5NtoyjOxpAhs6bT2Pz7bD9bfzcsGWCnzuJdfXj1Pu2Tc
EJ/ssk+BzkerjYWVc5UYSnuzoSJ0Pq+W+vvv5Q55bwSJeZI3wdSVfoBqcJOOAqCNVsdycJJcd1v
rX6HKGNZyM2XeEbBoKlwD8qUZ9TWvPgtCTHb3s4ZOpNVBo6LTXt1C/33F5nArN8b2HrT/A/7vZk
Jgc6uAcYoHeXPkV1K7VmKiwTsL7Y1yPGvCkl5tUP9XZ5tIvzM61NqHVD/WWtoHEJJRVoBNPeL05Z
tEenRjhVmvMjgoVL5S0MlOIHwGvk2pGw/wRV2JHi8r1SZs7uuK20R84c5+6vx3mFuRV/W7r++FJ

49mDwJG/UJb4/6hIqKiqX+nUujpgs4i4iIiIASzCIiIiIiIgX1UxK4B554Cp9Pn0a9Jk0XbJv34tM8+
+gjdO3atRDhiYiIiCwTJZhFREREREESKxF//
cA6HnP5bJq60BnNbtqbJh0M4aqtNlVwWERGRoqUEs4iIiIiILMLd+fLDj5g7axYd11qLeg3q5x1S2aiq
quKpv9/G4MGD+fKrr9jymINp3rx53mGJiIiI/CAlmEVEREREZIEpY8cy4KF/0ajn+lQ0bsTgO/
7OOr16scpGG+YdWtkwMzbcUO+3iIiILB+UYBYREREREkQVeffQxVjjuKCrqxaNCy54bMPSWf9BlrR7Ub9
wo5+hEREREpNhU5B2AiIiIiIgUh2lfT8I6rrAguVyt2dabM/
Ktt3OKSkREREESKmSqYRUREREESKRPuqKsY/8Ehu//
+MGTOYX8+W2O5z5jDu8aep+uLLHKJaOu2rqvIOQURERKQsKcEsIiIiIlIkrrr0srxDYLejj+LrmbOobN
QQiAX/
mr41hGYtW3LnzbfkHJ2IiIiIFBu1yBAREREREkQVuvvAiWj3Sn5nPvcj0Aa9RedeDXP7r46msrMw7NBER
EREpQqpgFhEREREGRBTp16sSTt/+dESNGMH36dNZee20qKiq4NO/
AREREREKQoKcEsIiIiIiIiJL6NatW94hiIiIiIiMhyQC0yREREREREREREREREKRGlGAWEREREREREREREEpQd9M/
4Z3Rw1h6rSpeYciJUwtMkREREREREREREREqIu3PrE7cx4+vprNNmbV6ePIDKFpUct/
uxmFne4UmJUYJZREREREREREREESkhDz15tN0r1iTPr13BKAvu/
Ly5wP516uPsvcWe+UcnZQatcgQEREREREREREREpIYM/
epedV9lhkW1br7glH336cU4RSSlTBbOIiIiIiIiIiIiMhSalPVlj+8dFHeYXyvEV98SrfOq/
LFpDHf2wpj1JTPco29TVXb3P5vKRwlmEVEREREREREREREREJbSRZcXZ3IZoE+fPlx/
2w3cc9vdvPbKf9ms46YLvvfO+Hfpc9jP+OVJv8wxQilFSjCLiIiIiIiIiIiIiUkIN/
cQgXDr2Adz4cQs9m6/PutKF8vcIULjjhwrxDkxKkBLOIiIiIiiDB//nw+/
vhjmjZtSteuXfMOR0RERJaBmfGHa89nzJgxDB0ylL3W3o+VVlop77CkRCnBLCIiIiIiJS5l5+7VXOvelGv
u3SicqZM+kw9TvuuOxyqqqq8g5NRERElkGXLl3o0qVL3mFIiVOCWUREREREEESkjE2fPp3f3f3ng9lYcdQJO0G
NCE6dM55txzeOTGv+UcnYiIiIgUu4q8AxARERERkfw80r8/s3pvtMhK8/nUnnMEUtsnzhlMnvvvTcNGjT4Sb/
WaNOGLehVMmTIlx8hEREEZHmgCmYREREREkRLVp0+fH33N2K8nUnnMEUtsnzhlMnvvvTcNGjT4Sb/
vp3jqqadq9feJiIiISN1TgllEREREpEQtTQJ3ypQp7HzGKbDKygu2zZ8zhzUbNuaJ558vXAiIiIiUhK
UYBYREREREKWOtWrXitN325Np7HmLGej2omDGTsM+4rqLLsk7NBEREREFZDijBLCIiIiJS5g7Yc0/67rQ
Tz7/0Ii2at2DLs86jokLLtYiIiIj1OCWUREREREEaNKkCbvt8rOwO8wxARERGR5YzKEkREREREREREREREREREES
kRpRgFhEREREREREREREEZEaUYJZREREREREREREREREREGpECWYREREREREREREREREREREqREElmEVEREREREREREREREREESkRpR
gFhEREREREREREREEZEaUYJZREREREREREREREREREGpECWYREREREREREREREREREREqZGCJpjNrI+Wd/z/
V+Z2XtmNtjMBBprZWoWMR0RERERERERERERERERqT8ESZGZWCPQnke919XXffALgCuZKZQ8YiIi
IiIiIiIiIhI7SpkBfMmwu/qm/7BuB/bIvDEtOwDzsbkMFQ0PPnke919XXffALgCuKZQ8YiIi
d2dgdObrMUDvxV9kZscDpwENgO2+7xeZ2XF2THAMQArrrhirQcqIiIiIiIiIiIiIj9dISuY7Xu2LVGh7O7X
u3s34EzgnO/2dm5sZtKmCplgHgN0zXdBRj7P15+/
P7BnAeMRERERERERERERERERERkVpUyATzDGqT6ZV9gZqtnvtwVGF7AeERERERERERERERERERSkFhWs
B7O7zzWzE4CngUBrgdncfZmE4CngUrgdncfZmYXAm+6ez/
gBDPbAZgDaOKFQ8IiIiIiIiIiIiIiIiIiIlK7CrnIH+7+BPDEYtvOrfb5SQPWSTvE8iHb7UBHPPgRgmzT//H+//
fxEREREREREREREEpnEK2yBAREREREREREREGREqYEs4iIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIiIlIjSjCL
iIiIiIiiISI0owSwiIiIiIiiIiIiIiNaIEs4iIiIiIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIiIlIjSjCLiIiIiI
iIiIiiSI0owSwiIiIiIiIiIiIiNaIEs4iIiIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIlIjSjCLiIiIiIiIiIiI
iIiISI0owSwiIiIiIiIiIiNaIEs4iIiIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIlIjSjCLiIiIiIiIiIiIiI
SI0owSwiIiIiIiIiIiIiNaIEs4iIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIlIjSjCLiIiIiIiIiIiIISI0o
wSwiIiIiIiIiIiNaIEs4iIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIlIjSjCLiIiIiIiIiIiIISI0owSwi
IiIiIiIiIiIiNaIEs4iIiIiIiIiIiIiIjUiBLMIiIiIiIiIiIiIiIlIjSjCLiIiIiIiIiIiISI0owSwiIiIi
IiIiIrIcmz17NleecwVNvmzMWfv9lst/90dmzZqVd1hSJszd847hJ+nVq5q5e/
+eabeYchIiIiIiIiIiJSFM765Zkc0fAgVm65EgCfTf2cO2bew+W3XpFzZLKcs6V5kSqYRURERERERERE
llOjR4+m/
aS2C5LLACu1XJGOk9vz2Wef5RiZlIt6eQcgIiIiIiJSrvr06bNMPz/Mz958mTOXPnUJbav2xmBl9t9
f1q3br1Mv/+pp55app+X0qcEs4iIiIiIiIiISE6WRNYH77fcfcsWhl7ELOy+y/
a0Zg3nmmWdo2bLv1M1+kR+jFhkiIiIiIiIiIiLLqebNm3M9N1m56W6+N7mTNvDnPmzeZu+sVi5u5ULn
VjuFvnr06ePqzRfREREREREREREREREREREkoUGvDeKxu/
thZux68G703rx33iJ8m+pFvlb7hLMwHIXsIiIiIiIiIiIiIiIiMhyZzQkSzGqRISIiIiIiIiIiIiIiIiIiIiIiIi1ogSgSzGqRISIiIiIiIiIiIiIiIiIiIiIiIi1ogSgSzGqRISIiIiIiIiIiIiIiIiIiIiIiIi1ogSzIiI
iIiIiINRIvbwDqAFbqheZPQVUFTiWmqoCJuYdhADaF8VC+6F4aF8UB+2H4qD9D9UDy0L4qD9kPx0L4oDtoPxU
P7ojhoPxQH7YfiUcz7YqK796mtX2buXlu/
S5aSmb3p7r3yjkO0L4qD9kPx0L4oDtoPxUH7oXhoXxQH7YfioX1RHLQfiof2RXHQfigO2g/
Fo5z2hVpkiIiIiEiNKMEsIiIiIiIiIiIiIiIjUiBHM+bs47AFlA+6I4aD8UB+2L4qD9DUUBy0H4q
H9kVx0H4oHtoXxUH7oXhoXxQH7YfioP1QPMpmX6gHs4iIiIiIiIiIiIiIjUiCqYRUREREREREREREREKRGlGA
WEREREREREREREREREREREEakQJZhEREREREREREREEEWSZm1jjvGEREREQNqbOwRvWOEr69kSMVGgvMjjhjhjw/
PzCqy2/
KMSRbsF+0HkUTHQ3Ezs+Zm1jTvOEpdOg7+YGbN845FllR9L2Vmm5hZ3d7xlAMza2/kMOQ36cmVWmTw
8G9skzlnJkZq0z+0BElmNKMC+H9DBYBFbnZaxMC+H9DBbd8ysQu933TOzCg9z3X1+9XZ39zzjKjdm1sjPupqZq+ptab9oPx
QZnafykz0eNABTPDIPq2cDt6Rt2eje7s6e7fmlkDvdPrddFp/
o8dQegBHMBZQMZ0qOjWTxX7S/YTU9nAlixBcsmZn9FaUYvs/KSXW1MyU/KSfS8Zn9n5m1cff51e/
+ViezkVKMBe5MBe5dHFb08y6m1kTUJt0MyssZn9n5m1cff51e/

```
38nRgL4+q318zWxW40Mw+NrMbzKy9mTUxszXMbJ2cwyx5mSqnfYHXgIeAS82shZn1NLOzzGwnVRoUBzN
rBLou5CVV3fwqXacbLD4Ao+tGftx9Xvq0PvB0+lz7o/
ZVv6dVwEAza+7us7P3TjoO8ufunqrL3wdGQwzC6Fpe+9x9fnpm2xP4BFSwUuwyxSyjgDZp22w9AxZO5j
3dCmiRzlGVumYUl8yA2f8BfyAdH2a2mZmdamZdcwuujKTjY42841gamrZThMys0t3nmdkhwCnESOpwYK
KZjQA+A1539wl5xllKzMzSgbs6cCawA9DBzG4E/ggcBDQws0fd/
eM8Yy1hRlQQ3Ebc4P0J2As4FlgP6AmMNbPfufuAvIIsdenBqIJ4/08mKgpOJSoA2wJzgCOB44AXcgqzr
GWuERsDJ5jZ3sDHwLPA48Bgd/821yBLXPU1g6hOOxH4JfCRmX0KDCL2wedK/BeFWcBBZjbA3T/
NO5gSVP033hM4BFjTzO4E3gKGufus3CITYMGssPlElXk34Bzg7MwgTPacJstgsWvDf4HewGvZ2Xh6r4t
TKnBZHzjCzDYCXgTeBj7MHitSa6qf++oB35jZ+u7+bs4xyZIqiGfBXwAfuPsXZnYoUe2/
ObCfme3p7uPzDLJUZXJU6wIPmtlewEdEbmRT4A13fzvXIBdjur4Vn8wf0hjgQmAccaPSA2gNdAB+5+6v
5BhmSam++TazPwErEQmDjsA1wOfAhsRFcCoxBVQn0QIwszbACHdvnb7ehLhBPxgYQyQ85wG/
VAKt9mXOPbsAV7n72ml7L2AgcRzMIAa+ugL766a77mUSzIOBV4EHiYei/dK/
TYGj3f32HMMsaZl9cC6wMTCJu
FavRNzwjQA+AO519//
mF2l5SwmDIUAT4AtiAOYxYpB+Yp6xlZJUbbYf0BlYh6hwqgS+ASYC5+qanT8z2x84g3iemAAMAJ4CXnT
3cUp8LrvMfdShwJVEkdBdxMDjO+7+Wa4Byg8ysy7ANkAn4hhpSSTXHHjU3e/
MMbySZWa3EH2v3yNmG32UPj7TdSN/mXPaMOBkd3/
OzN4GbnT3W8zsCeBWd39E15Dal3neuARYxd0PNrOdiPZv3Yl72yPdfWiugWYowVykzKwt8Ky791xs+4r
ARUl73+USXAnKJJiHAwe5+5tp+3vAg+5+kZm1BP5JnEQfzDPeUpO5eB0OnOjuG6ftawAvuHvn9HUX4FV
3XzHHcEtW5ji4FZjj7sel7UcD+7n7zunrHYAr3X3DHMMtaymh8yqw4+LXAjPbHBjj7p9nKtekAMxsHLC
+u49P+2Rl4qavPZFw3g44yd2f/uHfIoWWBi/7EknQzYFWwG3u/
stcAytB6ThYHegFrAm0dveT8o1KAFLbhkqgC7AZsAlRBbUhcS0ZmGN4JcXMegOrEcfAqsTAb/WA/B/
cfVhescmPs1iccUNi//
UCnnD3Z3RPVftSMVEXYrC+G9CAKGb5GjjV3efkGJ6wYE2Lc4l7pxlEIUUfd59lZhOAzd19uBLMtS/
zbP4cUbRyu5n1J/Ihl5rZXUTRxF+L5f1Xi4wiUz1KQYyevm5mv3b3G6q/
7+6fExW1UovSgducqMqcbGZN3H060A64Ob1makrwT8ox1FJVPU1qG2CumfUhKgBPJKaoVesNfFXn0ZWJ
zE1zZ2B1M7sNeB74NXBT5qV9ieSm1LHMzcNqRFX/rsAD2de4+6uZz/UgVCBpAGwO0eO3ug/
2SDM7iThufg6cBxxqZi+qVUA+zKyVu08C/pE+MLO1gea5BlYCMoPDLYGtiaq/
zsQMmHvTa1rmGaMs5O7TzWxN4Iv0kHof0IwYEBiUb3Slxd1fB14HSM8XmxDJs3WIijMpImlgbANgR2JA
4At3vwh4w8zuJfWa1z1V7XP3QcAgM/
sXMduoJ5HUb+ruc4olaVZuLNZ4me2xHtW81PrqamL2y9EpubwH8J27DwetB1MImXPOM8A2ZrYFcZzcnb
ZvClTPrqjOp+RKFcxFysyuA04gplY9BPQHXnb3MbkGVsLSAfs4Md18FPHen0HcaMwgpklNBFrpBFoYqV
J2D2Ja2jSisuY54sQ5jFj9/L/ufkFuQZYBM9uUqNpYF1iDuNn7nOip+STRJ/
tgd1cP5pykHlzXEUmyJ4GXiGvE+7kGVkZSRcc1xMDkqdVTn83sZOA4d+9u0Sf7QXdfJcdQy1LaP3sR6y
qsCgwmri/
TlSSoHZkE89+IAeD+xL3rbkRf+PWAV9x9Zo5hlrVM9dP6wOHA2sBOwCXufq7FgtYqnKgFmeOhGZGoXA9
YC7jT3R9Pr2ng7rPzjFMWyhwf2xO9yScQlbObuPtGaSr6BHd/J9dAS0zmWFkJOJ1IjHUFznL3D/
ONTgDM7C/ADe7+gZm198Xag6aZYccAs9z92kyRpNSCNDg/o/
p6YWb1gYuIY+U+dx9sZj2BJ919hRxDXYISzEXMzBoCOxNTOncikm7TgS2VRKhdaeS6PrAtMW1wQ6IHcy
uiH9QHRKJtm8XblkjtM7OOxDTmdYmH1ubETd+uwAb6+y+cdN6Z5+5z0+j1CkRyZi3iwXRNYC1375hjmG
XPzFYhVZoT56tuxDWiBdFm5qUcwysbZrYBcBVR+TSbeDD9AviHu99nZlcDa7h73xzDLCuZhMEexM34cc
Sxcq27dzazY4CW7n5lroGWiJQgeMPd26cWDOOAKqIP833Avkpg5ifTv/
EpYqD4EmKQ+PGUFLiQmP6vXvHLKPNe/
5G4h30J2JO4HlxlZvsQ05rH5RqoLJDZZ48Ab6Yp59cCLdz9KDM7H2jk7mepkrb2mdnLwKdEAcsVxCyY+
cCWRN/
r6TmGV9bM7Ezivml22k8TiXPaIOJaMjYNEui4KAAz+y0wyN1fTDMmp7n7F5nvNwMOIs5VVxdT+x61yCh
SZlYvTaftlz4ws07EjcqIPGMrRenEOJuYfvAMgJmtQNwgbkNU42xALNghtSgzit2HSOLfnW6+H04fmFl
3YCvgPSWXCyNzYfo5sKqZvQZ8QrRheNHdX0jTPFckptRKjtx9pJmN9uiZeYeZtSMGZHoSA2Jaqb4OuPt
gYAczW504f3UAnnP3z8xsZWLmy3X5RViWLP17IPBPd38lJdFeTNubEddzWQaZ88smQPV1eWfg8zStuSO
wqpLL+UrJMwM2cvc+sKBFzPHpJXsAT+QVXynJVO/9khiI/
8rMDiMWU4QY7PqKGISRIpDZZ62B6srZnYh+sxCtf6rbkFWwsI+21FBmEHhboIO7b21mHYDfu/
soM+tBzDy6L9dAy5y7X25mFenLa4h8yKHAqcB44B0zGwRoQfHCmMbCe6sTgA5m9iGRB/
wEGOaxyGIlFFf7HiWYi0jmhNsBOMDMTiFW4B5ALCw3GLgh3ShKLUoPQmOIFhkPA/
9296+Af6UPUt+6sbkFWaIyCbDVgPOBK81sItEa42HgsTRdSlOmCihzYWpEJCq3BGYSo9TvmdkHxAVtpC
oK8pEZjGkB9AF2TOelj4gpuM+b2QBPC6IouVz7FpsC3Td9DAKer54CnTEe+GP6V+pIJmEwOrN5d+D36f
NdiFYOUkOLVcq8CXyXkgLbkPpcE5U1WjSuOHQFhqbptJOJy8NHZtYYWEnVy7XHzLoRCeSJZtYeaObRjx
litpFaLRSJxc5jdwAHm9mrQHt3f8TMqohnk8dgkWuLLJvqe9M1gHfT5wcCb6fPuxO9f11tF/KTfe/d/
VHg0bS9AzHwsg+wh7vfpoKW2ufu18OCWfYvEv37uxGtl+YDE8ys+jljWk5hfi+1yCgimWk6/
yT1VyFGVPcmKmmPcveH84yxVKW2AHsC2xEVOCsCX5L6/7r7czmGVzbSSXRFonfdLsBGxIPRgPRxlbtPz
S/C8pFuIDYlXxCfrAAAIABJREFUHog2TZunE0nnX2k/
1L3MNeJC4uZuCDGtcGNiWuEf3f1Z3egVTppdNNfMriSuy2OJh6EexPTB94HL3P3ZHMMUFqxMfzsx7fY6
YH1gJeBvxOrnWjC5lqRz0hHE9fpe4rhYA7jS3V/
JM7ZylxkUO5loATcJaO7uh5nZNcDK7r53vlEu37KJytQn8y/
EwO944EB372tmhwKnqc1ecbJYxP1mYnZLK+Ka0Q0Y5e6n676qdix2rLQCniKu00cC17n7/
```

SkPMsTdL1KCOT+ZZ47LiWeNf3ss7NeUWMvCM6/V8VHLvq/lRbq+dCcGvjYDWrv7L/
OI739RgrnIpKkIk4GO2SpBMzuRSCj80t2n5BVfuTCzVYmbjH2IKhyAi939vPyiKi+psmZ1olLzj2lzx1
RZLgWUvVFI7UlWJvbBekQFc7ccwyt7ZvYlsGmaSlhB9F7+PXHTcZSOkcLJJGs+JgZanq/eTgyM/
Qr4l7vfVZ2MzjPecmJmbYnVzGdltu3HwkX+HibOYVdosL7mLBaurHL3Jxfbvh5R0d8NqATOdnfN+ioSZ
tYZuJhYy6KKGLR/E7jD3YfmGVupMbOfAZcR553RwAtAe6Kn7D15xibBzC4C/
uJLLlzWB+hFrD8yyN3vStuVQCuAdI0+gkiWvUa0GXuFGJwco/c9f2Y2gRiUf8vMjgOOJdqK/
trd39Q+KoxMgv9korVSP3efbrG4YjN3/9zMmrr7tGLbB0owF5mU2LwTONTdR2W2tyX6z3bKK7ZykJI1X
YEmREVzR6JdwDbAz939H//jx+UnWiyRuSLQjujnuAfR56wHUUn+BDDO3W/
LK9ZSlmnP0534m18LOAr4jrjZmwz8nWjZ84a7f5xXrOXOotfyf4DDU9uk7Pc+B3q6+8RcgisTafDrUuA
ud3/7x14vdcNiYa070tT/NYkKm9EWC89tQiyA+Yq7f51roMs5MzuQqJq50cx+QRQ/
3Am8pEqz4mBm6wI3ufvmqT9jY3f/Ln2vATEIsAIwQPts2aRr8q/d/
YLFtjcC9iUq+VsCN7v7sBxClMWkY+ISdz8rff0cUaH5sLsPyjW4EmZmhxAL7N5gZo3cfWbaviKxzk5jY
BbwgLvPzjHUsmeL9sm+yd3XtFhr5EngdGB7oIm7H51nnOXAzMYQxUNPm9lRRI6kFdGzfMD3VTrnTT2Yi
88YYCjwiJmdR/
Q8hagKGQGL9sSRZZememxGJJSbAmsTN97PEhe6a4hKZrUEqH0GuJndSjykrkD0xvwXkdR8AphfXc1fbC
N0pSJzYXqSmEJ+P1H19527vwZ674vIROAR4BIzO9rdx6Vz2N7ADHefqH1VGJmbuJ7AwcAuqTXAK+7+Wb
7Rlbc0ODwPGJ42XQ1MNrOhwHvp42ui6kaWTVuiDQbABKAN8FegmcUCNM8DL7j76zoX5WYkcHb6vC/
xTDGQWMS6fxqc/CCv4EpMFfGsgJltAVxLHB/93f3uPAOTH7Qx8axd/Qz4HPEceF+6lgwlep72d/
eP8gqyBH0HVBeonGpmfYGniaKJR9x9Rm6RySIyz4WdgZEp0bw/
8Lq7/9vM5gPnwfe3cpDaYdHTv35KLjcDLiQWIO0GnGlmb3kRroukCuYiYGYd3X1c5utGwAXEFJ1JxJTO
j4kWDa8rwVw7MqNzZxHVaMOJnqZXA295WihLCsvM6hEL+NUjEmdfE1M3PwEm64ajbqQqnLuISpvORDXH
M8Cr7v7u//pZqVtmtgbR43FHYvrtGGKBh7tSawZdIwrIzNYhppmvQ/
SMr09cq0cAt7n7kBzDK0tmthWwbqqMqiCqoX5GzEBqRcy+GEFca67Xw1DNpDYLQ9y9bWoLcxAxMNmKaG
m1BVEtviWwpa4d+TCzh4GL3H1w2k/
rEtW0uxPnrUlEgvl8d38hv0iXf2n68gvuPiTNAjuG6Pe+OnFdfoO4p/
2nWhwWBzN7CHjb3S81s12I68Ng4t53TSLZvCPwpUf/bA2U1QIz+xvwG3f/
1sx2JQbr1yDa8NUjBsaGE72YNdMoZ+naUUnkSDYhciS3pnPdI8BH7n622sEVjpltTbS2Op84L23l7n3M
bG2iJ/ZqxXh+UoI5Z2layCnufppF4+5ewAfuPsViRe4exEPRiOrpbVK7zOznxEPSZKIax4mE/
kjivf8029NRaldKBmxA3JD3IqpBmhI3fGOAUcRF7D95xVguzKwFUUXeDdiWWNxvFWJhvw+IBKZ6l9ahN
GJ9u7vvn77OtpXpTCRyViD6/
o7OL9LSZmb7EBU2Xr0PUquMFYkHpB5EUvNqd39RFR11y8weAHD3A8zsDOKa8Vj6XvViKNsDs9z92PwiX
b6Z2Q7E4NbGxN/7Ge6+ffqeAQ2AZkA7d/8wt0DLnJnNAdq7+2QzexrYz92/Sd+rIBa0Pga4x93/
nWOoyz0zmwx0c/dJZvYroB9xz9SFKBDaDNgNuMDdb88vUqlmsYjceHc/
3szeAY5x9zfS9wxoSBRbuLuP1/V82aXq/tvcvXuqGj+EaK3UAehEPGt0J5LOhyvBnL90rWgKzCcWfH/
DY5G/TYlCyNPcfVgxJjhLiZn9HjiMKPy6LhWb/
gVo6+4HF2OCXwnmnJnZWsAG7n5vOvleSFTYjALeJlYhnlSM5e+lJCVxNiBOoGsQfWjrEW0xZgCXulacr
3WpT+bw7I2bmXUgqm02IFZJXRkY7O5n5hJkGUs3F92IG76+xH64Kt+oyouZrUAs7nqxmfUCXgYeINrHv
ODqt1xwZtaTqLo5KO2PY4jr8whgjLt/lx5K2xLXaz2I1jEzO5ZIIP8ZuBu4yt2vVzV/
7TKzVsDlRDVTG+Ie6URi8VfdIxWBNFB8E1BBXCeucvd2+UZVmtL96gvAOUR130ueWSvHotdvI6A1MEHF
KsUh3UtdQay30wv4E9GmYZASm4VhZrsDVwGnEQNc61cPTmZe05ro7/9pDiHKYtKA8jNEwv/
utM2Itao6uft/84yvHJhZD3f/wMy6AF+7+4w0i/
Vy4AZ3f7YYB8CUYC4iqZp5N2IUb0Vi1Og74Fti4YGncgyv5KQRuAOBB4lRuTmZ73ViYZKzO3Csa8GBWm
dmo4G+aRrnIcTU2/
cWe82axLlK1VB1wGJ12k2J6o3hwLDqNiUapc5PSvYb0QNtb+IGvSXwPjCQmLamBedqWaaVUutUDbgtcC
UwnZj+/CnRr/ET4H13H5tftOXLzKqIipodiD7yTxODMe8Q+2gy0aNc1/
FlZLFo4lpEYm0gcV5qBYwH3kwfzxRbRU05sVjk7zTiHrcBMf3/GWLxrMH/
62flp7FY6PJoIoncAfg9MAx4Teeb4mVmLYFT0sd9xPo7DYCxRKXgC57WIJHakdrJbEJcp0cR6718RrSl
1DoWRSRz77svcDhxTrtaRS11J/Vffg64Hvi7xxo79YiK/
1bA0GJLLFdTgjlnPzTqkB6WViV6pe1I9AwcqARP7TGznYAbiGrleUSftMeAp91di5/
UIYtVzYcR1bLjiX51jwH/
cfcv8oytHGRuJNYipqw1JhbFakok0T4jqpfvyzHMspS9RpjZicCN1YkbM1uJWJz0SOBBd79MFZu1z8za
uPukxbZ1J9qTbEoMCHcBrnD3v+s6nR8z255YYOsvRA/mTkS/
2feAAe5+a47hLdcs1gdpkaaMG3AGcBtRzdSNSDp3J+5b+6plT/
7M7H5igOULYkr6hsRCl58BJ7r7SzmGVzLSzJZ+xHs7gui/PJ8YpB9KXJ/
H5BehVDOzVYGx7j7TzHoTMzGeI46NVYh2V72JAePTi7E6cHmVWmOsSSzm/k/
iGaMdcax8mT7+svj9luQnPZ/
vBPwWmAP80d2fzTeq0pdpxXcQkeAfAly+vBwbSjDnLPMH1Ak4ixgd+ix9bz2i9/
I0XeAKK7Un2Z94IO0GfEVU57wI3Ofu3+YXXWkys7aLT0VLF7JdiX2xLdFbdpi7r1v3EZaP6qSkmV1D3O
ydQBwHqxMDXT2IG/
KzcgyzLKUptm2JitnPiZ5bunDXkdQ+6RsiQfMs8DAxCDk385pKYnGzke4+WgnmumVmpxKDkUPS1wsGWV
JLh+2AfYDv1H+55szsFKDK3c9J96ydgHerZ3+l63c7oIO7v5VjqGXLzBoCvwb+nAaNV3b3UZnvNyUqCA
8B7nD3V/KJtDSY2Wru/kn6/
FDgJWAWMdCyWvp3S+APSsoUBzN7G9jF3b+yWEDrK2KtnerzWCOi8Gimu4/T9XzZpen9s919fPr6AHd/
IFVoVj9rrEIMVh6uVjLFJ11bzgT+D3ieWJNnlI6PwjOzjYkZehsSMyhvSrnBon3vlWDOWSaxczpR8bFt

```
qow6lVhE5SV3P7WY/4iWV2magS9e7ZduLvoCexHTCzdz99dzCLGkmdkfgZ2BV4FBwKvuPnyx13QiepQ/
kUOIZcfMriamqt272PbOQIUq0upOZvBxa+Ao4qF1J+K6YMDElEBoCPRz951zDLekmdnKwOZEknI34v3/
GHgS+Ke7D8otOMHMLiWqnsaZ2U1EFeFAomJZLUtqSUoSNHH3j9O14gjgNeL6/V9i/
ZAvs+3GpG6Z2QbAIe7+m1SpeRZR+TSCqKb9XG0bakcqTDnP3XdO7RY2JqqVJ2QGuJoTycqROi6Kg5l1d
/cP0z77gOgj/
wFxDnuduLaP10yw2mNmNxIDvL9JbcYaE9fn7zKvaUYsTKr+yzmytGCcmfUgZiVtDTQHxhAJzl2BCUR70
Yvd/
avcgi0z6dg5kWj99ld3n5JvRD9MCeacZRLMLxMjEveY2Z3EA+wjRCXCbe5+f66Bljgzq0+s9jxRfaDqh
pltBaxPVMeuRvSTnUrcoA8EBrr7hPRaDbAUWOrxezWRSDudeCid7uqjmSuLXvEHAXsQK5u/
RLSRGUIkDHYHNnf3zdQeozAyLWS2Bg4AmhHv/
U5EdRpE65Lj84qxnGX6Y1cSbRtWJxaHbUo8CL1NJEGf1HVk2WUGvzYlemluT1SfTSWqAU9w94/
zjLEcZfZLQ3eflab/
nwpUEr1lvyFmwXxB9Ad+J8dwl2uZ93p1dx9u0af0buL+dShROPEWMGrxmXqSPzOr7+5z0jVjc2L26jak
ymWih/zJecZYKjLHSnXi8jpiluQMojXiE8DjwDvp+3reKwKZIrB/
A1VEkcszxLpgbYl74Y2APXS9rz2Zc1Nn4j1eDZhI9PffnFhjpJK4rz3X3V/ILdj/
QQnmImFm9xKjpu8QqxH/
wt3fS1N5znf3fmqTUbsyyf1tiER+C6Jp+kRiQZRBxCJB3+hiV7tSMpOUtGkKdCaSAmsRC22sRDwQzQD2
LeZRulKRqp0eJo6BacS56FXiWPhMlYD5MrPfEg+uHYiKgpWACiKBdpe7/1sJ5sLIJJhHAMe5+zNpuxHX
682AP7n7M9UPUXnGW87SPmlLHB8rE/2A1wfqu3vfHENb7mWOg/
qLV2OmmRRbAfsCZ7v75FyClCWYWUciEbAWsAbRtuE6d38w18BKRCYh0Jo4BnYj2pA0I9Z3+b27P5RnjL
JQ5jzWwd2/
XOx7zYkKzWbufqvuqWrH9yWNLdZ82Z8okuhBFFCs5VrQPRdpsOUoYsHq0YvPKP6BnxkA3O3uNxU6vnJj
ZkcTs+lHEgn+l4m1E+oTeaotiFmVp3sRLkaqBHORMLONgAuJVYjvdPe/
WSzg9B4xZWRmrgGWoMxNxutEMvl+4GbgXWIaSCvgJHf/Z45hliQz24U4Of6XmLo5xlOfazNrQ/
ThWgvo4u5X5hZoGTKzJsTCorsQD6VrALe6+
+m5BlamfugBx6Lf6frAONfiQQWX3u+3gKMXb5lkZi8AR7j757kEV+Yyg8XbE4Njb2SPGYtFk9u6+0e5B
VkCMu/zUUTV8klE1fLKQBu1EiseaRD/AOBFdx+XtlVXEvYAvvLlZLGg5YGZtffUXzazrTOwH/C8p/
7wUjzMbBBwFfAQMSjZm3gWGZxrYCUoc+7pQBRJDFm8YM7MtnX3F3MJUKoT/
vcTycuJxLP5UGKR2K+I9lfT02ur8yfXEoOVI3MKu6SY2UXE+/3W0lwzzOwZ4lg6o+DB/
URKMBeR1G92rscK3U2Bw4Ct3P0QVS8XRqo4+NTdW6evJxMVT7sCGwBnufs3OYZYkszsWOBgohXMNGKE7
n2iD9oI4kI2U3/3dcvMmmV7oqVtKwONVFVQ9zI35Y2Ih5/DieqCf7v7+/lGV15SdcfZRG/
+04l+sxVEa4An3b1FjuEJYGbvE4PCz5nZOkR7gObAydVJNqm5TIL5JeAhd/
+LmR0BHEP0n70B+F31Q6jUvcyD/
8+AK4hB4gZEi6XdiL6n1+cZY6nIvNcbATcRraxGElOa1wZeWbxCVvKVuafqTiT+O6XZF/
8gil4qgWPcvX+ugZaYTHuMy4hqzJOJ+6c9iGKue1ObKz3z5SBzXDQgCuy2Jq7pHYk1LUYTz+ejgMHVg/
Vm1kTX+9phZm2BPxHX68ZES6uPiff9bXcf8T0/
czLw32Ic3FeCOUeZm5PziD+Q6mm39YgTbwci4TxWPYlqV+ZkugOxSMfWFgt23OruPdKUwifcfcOcQy1p
ZrYKsC2wKXFT3pAYOf2QuKDdpyqbwrPoQb4jkfTvSvR2eg142t2n5hlbOcskdC4kBr1eAw4l2vlMIma4
XOzu/
8kxzLKRZlecR6yiPYYYIOtMzDq6WtNp617mWt6NuI9qZ2aNgVeAAcQD0lvAlXpwrR1m9hWwprtPSW1jT
iAehO4kkjPDcg2wjGWuGXcSs1vOTA+h+xKD912BK9z96VwDLQGZpNnVxAyJI81sR2Jhxd7Am0S7Qy1aV
iQyz90nATu6e18zO5JofdXbzI4n1rQ4JN9IS0vmff+UmO01wMyuAnoSfWUfIHr3f5troGXsB9qYVBEt4
LYgWit1By5z938oL1X7UoX/
isT73C193jp9ewJR3TzQ3Qek1zcq1g4H9fIOoJylk20D4Dii9ylmdihwLjAWONJT31MdxLUr
835+CPzZYvXa+sBIM/s/YG/iYJYCqL4xT9NqRgJ3pO0bE4tm9SKSOP/IL8rSl0mI7Q5cRCzmMI44J/
UBbjGzV919lxzDLFuZZOUxwAHu/
pLFQnNnAwcSfR7rwQ+30pDakwa7TknVT1sBc4hFF6sXhlUCMz8bEz3jAY4A5rn7yWa2O/
AHd788v9BKR3rgfA04Oc20G+vuT6bvrUfMsJCcZK4B7YAn0+cHAte6+4Nm9jgx60KWXfX5fnOgupXbb4
iB+e3N7BHiflYJ5iKRGWT8EphhZpcQVf53pO2rEH2zdU9Vi1K+owVRmfleGqw/
HOjl7p+b2RCiklkJ5pxU50XMzNKm5u4+EXgsfWBmqwNaX6FA0oyXL4FBqdi0C9GCbE3i3LQ5sdjigJTg
L8rkMqiCOTeZ0bzdgEvdfb10c34fcCmR3PnM3c/JNdASZGarZisKLLNgjZndRLz37wB/
dffncgqzpGWqbA4jejne4u6vLPYaTb0psMx+eIKYLniVmd1I3GA/TCQy/+Luj+UaaBlLN3T/
cfcVzawlMNLd26Tq/zOAU919dr5Rlr60HzYCphPTBD/
R+al4pFY+9xKL+70KPODuD6UEwqruflCO4ZWETLX4VsApROLsYXf/
b6oCPN3d1801SAHAzA4mWsQ0BF4n2sRMN7PxwDbu/
kGuAZaQVInZFJhJDPrul2aejgIOq642k+JiZmcTLRH7ETNWvzWzYcCZ7t5f7RpqV0oqX0QMzLcG6nm0A
O0GDHL3trkGWOYy1/cqonf8L4A2wDPAi0B/d5+WY4hlISX4bfFzT2rlswowxd2/
LPYKclUw56f6j2IV4INUEXUS8Ka735NGLo6AhcnonOIsKakNxnnAzilZ0wv40MymEImDE9K2T939q/
wiLW2ZqoBvidWDH0vVUIOBu4Fn3f1D/e0XVmY/
rEwsuAhRmfkrdx9oZqcTfaAkP22Bl9JNX3diKjrEDfqe7n58sd9oLK9s0cXjzgZWJ3rGVwKfWPT8Heju
/
84zTgF3H2Vm+xGzMd5JSc8ewJ7EvpNlVP3wmaY3fwzMcPdvzGwNoufsNTmHKIm732tm44iKwdeIas39g
QlKLte6y4BLiJZJv03J5Y2ApkouFx+Lhawbu/
tlZtbO3Sek7XsQbZWegkWqnaUWuPskM7ubGJx8CXgwJdN+TrzvqhrPVyUwl9g/
```

fYCLgf2JGTAHAPeZ2T3uflh+IZaFJsCBZvYHopvBAKJt5dvA6Ookf7E/86mCOWdm1gq4lug/
OxS40d2HpIrCAekCqBNuLciMzq3u7sPNbB/
gHmAIMIyoenoDGA7Mrq5qlsKz6Jm5AdGa5PS0eRvdnBdeeu+PJy5i7xAj1X9K/
w4HuqVpUlLHMuesViys+vgzcQPSBnjV3U+tbjmTZ6ylKJNgfgp416Of6c1AS2Jw7CBiIdi/
aDCs7mWOjzbEmhUfZKZ5VhL9l7cjFhDS8bGMUuHDdcDt7v5mqqjp5u7vWyyYPNPdZ+QbZfnKzIzcDvjc
3T/JfK8R0UvT3P353IIsEZlzT0diPYR5REJ5qpk1B44iJTFzDVQWyFzPjyauF3/16CO/
PvHM94GZtXGt+1IQqZDuo3TcVO+LTkRLvmfSwKXuo3KSuX58QrRoHWixoO9fgSnEGj03uvsg5aVqX+aY
OJ64flwA9CVayUwmZiPd5e4n5hjmUlOCuQiY2UpEBeGAdHBvT4yIH+jun6o6rfZlDuTWRIuGPsTUtibE
jeLv3P3hPGMsRxaLzZ0CTHT3O37s9VIzZtYemOruszLbWqRqtKOBXxIPTebuffKKU8DMurj7mMzX2xNV
Ba8DT7r7ON2UF5aZfQFs6e4jzWw4sLe7v2dmNwDXu/
sw7YO6l7mOnw+sBRxCTFXfClgPeMzdh+QYYknIPHjuAVzi7utYLEZzKpEc+AjYwbUgbFEws8HARe7+cK
ouP4YYLP6HF3HPxuVJJsF8M/A40C99vR1RLDSRSDirp2yRyJzH3icWd3/
IzI4i7qd2BM529eqvVZnjpCsxOPnbVODVgWjD8IK7D1WeozikwbF3ifvdsWkWTPc0cDaIuPcd879/
i9RE5n72VWIQ/1Yzu5UofOwP3ATc7e73Lw8J/oq8Ayh3ZtaOmFrYLn0AvA/
8xlOfYJ10a186iM3dJ7v7Q+5+tLuvB+wE3IIWqikYM6tI/
+5iZiebWc9URUuqGn+P1B5GCuY8oBOAma2Vbv6qk80PAHcRCywenE945S1NG8TMVgSuNrNzzOwgM+tFD
EQe6+63u/s40FTOQkrX6AFAvXTzPZ+FizYdBIwB7YOcVN8b7UNUKc8Bfk/
MgjkE+E0aTJNlU73oz94sXDjueCKJvw3R2urwHOKSJHPN6AF0SMnlZsCNxODL+URPTakFKWm2ApGc7J+
+/iNwA7F4+MZKLheXlFxuTNz7/
ittvpg4RrYFdkkVtVJ7qvNMBwItUnK5N5FsPgK40sxWUJ6jaNQDrgbapEGAkcA2ZrYlsLqSy4WTSRg3A
arXpNqeaPn2BXG/O3Sx1xYt9WDOgS26wNkxxAG8O9H8/
mqi5+bAHEMsWZnR1JZAnzR6PZlYIbVfSupflWuQJS6TiFmXeDjdHphmZiOIRVJ2JtozSAGkqeMfu/
vItOkaotpmmEVfzaHEQ5KSZvkx4mZiPaAnsANRVfANMDVVqA0HXtJDbMFNIhbedeL89AbQ38zeI6ahT1
X1cj5SwqAe0IJYVbs+MSi2nbt/ZGZDgfbA+DzjXN5lHma+BCamga7tgCvc/
Z2UlPkotwAFFl4ztiauFRAJ5Sbuvr2Z7QScQ7q2S81lzve7AkPS81xfYgBmV+BnxHour+cYpny/
KuB54vmvOzDC3R+1WONiHXcfm294Jac6cbwzUbgCMQA8GdiFaL9wMHCtqpjz5+6T+X/
2zjvaqup6289LsRds2FBUsPfeezd2TbEkGnuN0cTYYo8aTey9E40lMSrYxd5FRUXFAjZQsQv2Avp+f8x
1YHs/9JfIOXdxz1nPGHfcc/fZMOY4++y91prrne+Esytq/
2uI3NRnwCVQfLIbiaIXVR+gh6RXic3KHpKGEZX2W2UM73+iJJgzULkxjwP2AG4nvkQD0/
H9gTMJVUihvtRM7A8ENgauAXoAxwAXSvoS+Lvt4/
OF2DJcCtwCzAMsTlyH3oQX9pkZ42pq0vPnjKR46kSoN5YCViLuiS+BYYQ3+em54iwAkSA4iRgjuhPX6I
9E4nkosJakvxTPwMaR7pexNguS/kZ4o01LjN8wTuFZaH8mB/oBlxH2Vnel5PJMQE/bz2WNrrm4lFAw/
564B25NG5ZLM653QiEDlQ2uh4AVJF1C+PTX5lI/
I8QshQmk8lmPAl5JtlXbAFcnheYUwNTZAiz8ILbfkHQ/
oWC+GtgrvbUjSR1YEmj1o3Kv3AWsk8blmYCjbL+fLHyuSufUNskKGUnWoaOJpPJZwOvE2PFiOqWIKRqE
o4HfaanS28ANRML5bcI245uOImgpHsyZkDQPcIftXpKmAV63PX16711gwZI0qD8VBfN9wN9s35SOdwZ6
Eobqb9v+d844WwVJ8wJjkrepgC4uzRUbSvqcNb4BKlllLEEkmr+wvX97x1cIUinnSGCGNOmoHV+QUBT0
Jcqe/0N4xk/0E46OSLJY2Ju4FiNt/
yNzSIU2SJoL2J6YhN+avAMPAda1vVbO2JoRSbPbfkvROG5PYAvbq+WOqxBI+i3RU+QSQrgyG7FQPcz2r
T/2bwv/PckyqQ/
h33sk0Mf2SEmPAue79BGZqNG4viMLE3Oq621fXhLM9afiwTwb4cN8n6QlCDX5bMUbPj8pF7U7sCwhNPq
QqMJ4DHjL9jcZw2sZ1KbRqKTFgK7A0PS86hBK/
5JgzoSkXoSy4GBCZXCU7TUkrUR0iezVUb5EHZE0AbftPrljaSUqZTcrECqoWQlLmNGEgf2lrjSeKzSGy
nU4m1Ck9U87o3PzHrLoAAAgAElEQVQAo22/
kznElicpPS4Grqs+pyT1AB6xPYekpQnl5lLlvqkflftjSWLhOZpQ3cxiu3eyCPjUdrEFmAhRNE7eBHjO
9r2Zw2kaJPUGPmqz+OkMdK/
5wRcmPiStTHjMnlA2IutP2rTvbHuMpE0JNf9mtkdlDq3QhmSPuCjwAtHoekw6PgmxJiwCl3YgJTN/
Dixme7+OospsRiq2rfsQVXk3EcnleYh5bzfCCuiAjGE2PWkutROwBVGxOoS4Fv1tf5DO6TB5wWKRkYH0
IH1F0u3AtUTX81skLQ4cQiQVIMrXyy5qnagkDaYmEganSVoduAN42PbrWQNsDWoPxrOIB+fFwLvAkoRn
3UiiyVyhgaT7YArC++xvKbl8BOEF/5mk39suFj0ZSeWD/YAzJf0SuJ/
wA14HeCqd1h3oWpLLdadmebEz8IHtHSXtQDR0gvBC6wHs0ZEmfM1GKiNcjEieTQO8QSQOnrR9VsbQmg5
JewO7AItKepmwcOsHPG67NEXOSKUyb2rg18CcRFnzMGKROsD2Qz/
yXxR+AqmaaBpgMPAFgO0bJN1t+7OswRXGUkmgLUMIW9YGZgYGSRpErEXut128+htA2oDZDJiOSJa9Zfs
T4OLUQwGKNUZOqj7Zx9u+EsZW781FVLW+mY6VjYA6U/lMNyHsWk8hPu/
VCMvcUyW9bHvljrTWKArmTEjalvD/3ZjYrViLSLQdTzyAPy0L1/
oiqUtSGJwDrAk8SpSB9GScX9rFpQS6sUialeiKOkvlWFfCA21z4Jdlct44KhstWwIH214uKcr/
SZRHbQxMZ3vHnHG2OpWkwYKEv+PiwPyED+oFqRT3IuCToiyoL5XP/
nbgHNv9JD1IVBedL+lK4EXbx5Ry2vankjBYn+hl8SXjNipfJTaQB9g+OmOYHZ7KfbAAcA+hbnoCGE5sB
P8qndotJQwKGaiM6X2AXoRVzHpEc8uPiAaMp9ge9MP/S+G/
ofLs2Q3YjtjgmpTw770TeAC4u2z6TjxUrllf4DPb20u6AJid6KWwErH227Wsu+tLUmVeDCycDi1NNBXv
S4gmriwJy4mDpGCeBjirjOftR2X8/jvwvu0TU0VFrYH1PMQc65aOtN4oCuZ2pDJZXx04M+0SXS/
pNttfSupaLc8pg1zdqd2UKwHbelz389kINdqyxMKp0FhmBF6TtL7t2wFsj5Z0G2EVU5LLDaQymZsWeF/
SVoRy+WbbdyWbjN9kC7AAxPNfUk/
bLwBHtH0/2QDcTGyUFepIZey9AjhY0lBgIeDKdHwlovkilIYnOfkdcENK9F8IPE40ZNyJsI4pTBi1Kro

tgQfSRsv2wEO2t5X0enpdFqMZSYvTyYnrNJ/tdyS9RyjO/
0Q08S3XqA5UFvfHEL0PLlH0zbkb2I2wPVyDSJ4VJgIq12wFYkMAQtS1JfApUcl3STpeGs3VgYoqc0Pic
1+XWHPcQviVXwSsb/uf+aIs1EibyMcQm2WdJd1DVMCMBL7sKEnNjkhlTf4JML2kyW1/
CXxDVMa8k6oA6EjXoSSY8zAlMLaJXPoi1ZJspfygQaSEzeREOdsM6dgIYATwhKT+QFEdNJC0yfKspBuA
EyXNSZSnLc4476dC+3A5sTO6N6FKOycd/y1R+lxoZyo72b2JUufVJM1AlKM/SjSGHQpgexgxASw0gGS/
cA2xODqOGBt2SB6bt9csZMpGcPtTmWQvBPw5vV4J+K3txyQtQtksniBqC5pEL0IZDlFefm96PSmwIrHR
VchARXG5BvBSSi6vRCQF7pf0NbCN7deyBtoEVERCqwCjUnJ5XgDbB0t6CfjadkkuTyRU1MtzEFUuU0n6
irAWeyadsx/RE4my/
q4btfFjY2IT+A1JuwOP2b40XY9HoGP5yjYjaawfQlQkLU9cs52J+2UoUa10Y7YAW4Bkb7UcMY7PIOkRo
vLoZdvvdMT7oySY25fazuhSwBaSvgP+QXitvG97dBncGkNlAFsEWBDoI+k04FniATrc9hc5Y2wFKg/
Ji4h74Y/
AecCLxCBWVGftQLofxqR74N+2n03HVycUHcUHOw+1SfmRhEfgSYQ3+YJEIucsSSfYPqxsRjaGSgnaEUQ
H7WOBHYCviJLa/5CeU2VhlA9Fk6C+wNeSJiPGk9oG8QaEurnwE0nf61oi/2ziuw/
wCrCUpHWJheiBGcIrMK65HDAmHbo/HZsbeCwdW5RYuBYmgPS51hT9CwE13/
HVgIHp9XvAoYyrdClkprIZ+S6wL5H3mBR4MZWkjwRG2v6kzKnqQ7pXavOizoQlBsQ8tn96vSRhbXUHRT
Wejcoc1sS16Q8cq/BfXoPwBZ4tndth7Bk6IN8R9od3Ep7XGxPq/88lPWn79JzB/
RSKB3M7kx68xwK9CfP0T4hmHM8TDWpuKt5djUPSosTic05gDmJg+5BIql1n+76M4bUEbZMySSk4R1JkF
hpIRSG7JuGFdp7tz2olOZImBea2/
WLmUFsaSe8T3bXflvQKUeo8G7AK8Ffbw8piqLFIeo7wgx9cOTZpGZ8nDpK345xELvR1SScQ98cQYAnbS
2cNsAMjqRdR3fKE7ZFt3puNUPvNSWwM71CeQ/
lJa4u5iCTnLEQVUm3T5SLb5+WLrrmQNCNhtTCASDAfRlSlbkLcM4dkDK+QkLQNUckyqK31nqQtgD8QHu
VX2P5XSaDVH0nTE82oXyI2I9cDHgL2Ala2PaRs1E9cSJrS9ufp9Uy2388dUyuh6Em1IDAfISwa4uj90q
HWfCXBnBFJPYgF0QqEymCM7fXzRtU6SOpG7BStSJSFnGj7kbxRNSeVMrUlCU/
A1YjyqIfT71eBT4uKvLFUrsPNwIO2T5C0FLFA2hTY1XafrEG2OCm5cwthGzMF8JztmoLgeWDZ2uSv0Bg
kTUUshl5y6qidjncCOrvSK6HQfkg6hdiMv6PthqSiGebhhFLtXNtDMoTYFKRS5m2Bd9LPYEKlOdj2V2n
uNI3tYkOSieSZeQFRUXG77ZfavL8cUfL8FPCfmhVf4X8nfd+3JmyqXmibhJR0NOExOwQ4sogl8pPsEK8
gFMtfEYnmZwk7uJdSBd/Mtt/NGGbTkex5FgWeBF6z/
UHlvSmAU4DpiPXHmXmibG0qVj9zENUY3xA5kHmAz4ieVPOl0z+yvVaeSFsPjfNfRtKUwKS2P8oc1k+iJ
JjbCUld0oC2G6Eo6Gd7VJtz5kpKnLKL2iCSgnkVYFZi4XSb7VcldQG+60i7Qx2JSmLzCSKp/
DBhFbMa0fSvK/B729dnDLNlSArZxW2PkPQ4ob75ClgH2Mv2W1kDbGFSadqGRKna9EQS4S/
EM+sw27062k52R6Ey8d4YuAEYBZxOVBYN/PF/XWgkSbHcB1iTUPO/
BtxFXKdHqwvZwoQhaT5gYUIR2xuYCZiMWHy+SmwKv2z7paI+y0Py/
t2fSATMQ4zfA4HbgTttv5MxvKYieS4fS4wHnxHPnqeAp2y/
ns7pbvu9bEEWvkda0y1BjBULAz0Zt9b4hKi+GEgkm4s/eZ2QdBgxRn9EVAcPIzZeXkm/vyH8r7/
KFmSLU6lkvZSwf/sYuI+wafiaWJuPJKx+nrd9Z1lz1I/
KOmNGYuzuTVQUjyHse5YhxnPS6zlsf5ol2AmgJJjbGUnnArsTpWtvEUmEK2zfkzWwJqaS3NwM2JOwxrg
X6AZMDpxi+8GMIbYMkp4CVqs+LBVNzDYmutG//IP/
uFAXUonzdcRzqDdwsu250nsfEhYZpeN8RiR1ralkJR0FbAG8TyQ6T6ttWOaMsZmRNAsxyV6R8KHrQSyM
RgCHl/
EiL6nsdjNgK2BVYhwfTPg59rP9UMbwmoo0Ps9NJDJ7E9YYMxLzp+1tv5kxvJYmbbp0BhYgKl5WIp5bMw
NvAy8Apzk1Myv8NJKCeRHCh3xhYjyYjrDY+4BINr8IDGwrHCrkRdI0yV95FuI5Ni/xHOuR/
n449bQoG2V1INnsLUCMF8sTn/NkhN/ye8QzaQRwd1vbkkL7ImknIicyEDjH45pe3gTcbPvcnPE1K5UE/
9FE1d29hIXMwsQ40oMYx08glP4DO+LzqSSY25G0W3EU4ffbn3gIb0Moar8mdpBOLMnm+lJJMD9CeNFdn
Ha35yasAXoAP2/rNVioD5WH6dyE59mDtq/
OHVcrI+kPxMTiSSJpeZmk9YGzbffOG10hla5NSyTNOhHeji8TaoKiImhnJPUklAQbARfafrQoOtqPiuJ
jNmK8fr66OE1qzs2A3xANm1bPFGqHpvI5Lwr8HHi7ushMPr+zpJ9FbF+eKdQCY21hutl+JF2byYjE51z
EAnUTYk1xd74omwdJUwOfEyKV3kQSbW5CJduLqDAqn/VEgqQ9Ca/
sv1Sr8tL6rxth5fCR7UFlPK8faVNStj+QNB2xObMkkfOYlXhObVUsEfMjaW2iqXhP4GSiSmwwsJntJzO
G1vRI+gVwGrH5crDta9LxPsAw20dmDG+CKQnmdqCS4NwH2NL2Wm08cI4mmkUsSjyAf2377ZwxNyOSXgK
2s/
1Em+NPAXvYHpAnstZA0rbE4PU1cA1wE1HKWdSy7UhaiIpQO420PTglaE4gEjdHZA2wBamMB7MCvwNWJp
Rp0wJ9gROK73L7kBaf8xMetD2Jjd+
+tt9XafKXhcr9cQYwFXCU7eGKZijflsRAfahsBvchmk6fnpIEXYhmit9K6k2sHYZmDbZFqdwLixGCled
t//
kHzh1bCVP46ST1615AF9uHtnlvCmAaYDmKKnOiQdLswG1ElcWgdEzEs+s7SVsS4opvcsbZLFTGjs2AtY
B7bPcdz3kzAAvbvr/dgyx8j8pYMglht7QJ0cNiQdsL5Y2uNUj3w+FEL7Z/
2j5L0jPEZuWNeaObMDrlDqBFqC1+JgM+SyU7BrD9BuFLtBBxgwP8tv1DbG7SxOJy4FBJ01aO9yLUHsVf
s/EMI8qZDyAabxwBDJL0SvK4K7QPuxHqpoHAkHRvvAYcB5yVM7AWpnP6/
Uei1PkiYF9CWbAkcEIqhy40iMrn+2vgn4Q67X1gb6C/
pA1KcjkPFfXy5sB+Ts3lbI+uJZcl7aFoIlv4iVQS9RsB59d8rW2P8bi+IL2B3dLCqND+KP3+LfBWLbms
oFN6vZCk44jNmMJPpPZ5AtsR1gqnpuOdK+PFFMCStm8oyeX8pPkshH3SGzV1MsQ4kpKgkxDK5l/
mirOJOQAYBNwMcQ+lDUokrQssVJLL+akkl6dKmyynA5cS64+ZJG1e1hyNRWF1+CFwItHY/SBJ/
YHpOnpyGUqCuV2o+KZcTUz4jpa0gqT5JK1HlCI+nZQGowh/
5kIdSdfgeqKU7XFJt0rqC5wDXF78TBuLwsPuVOBJ2xcSSc6NgO2BS4gEZ6HBSJoHOBT4F3APcU22JZJp

L7o0qclFLXmzKnCc7ctSeVpfornQykRDzELjqI27+xOK8Z/
b3p9Q+t8OHCBpzmzRtSiVhMGmRGO5T39g4TMZ0bCmMAFIWgj42vablc+
+yv2EQq1r+0ZWaMPPiDltTT3oygbBMMJKZqVcwTUZvwKuTVUssv1tZcNlWmDjsrk10bEqUPPiH/
scSxXF3xCNzVZNx0ouZAJJifvJCfHKpbXKCdvfVdbXXwO/kjRTrjgLQSUvdYSkbra/
sn0xUYlxMXAKIXwsNIA0Zo8BsP227WMIy9wpgNkl7Z4smTos5aHajjiaoRxEeHZdBpwB/
BW4E7gilfSsQVgHFOqM7cG2VwB+DzxANEG5NP1daACVBep8hJ/
jaEmTpMFshKMZ0wlVf7RC47D9qu2exMT6PGCm9HsAMN4y20LjSUqCLsAVwJqp7Ja0kH2USBaMhLIYahR
pgSSi2cbzleNf2D6Y8NicCb73XCu0H/MBz6XXnWrXoJJs/
pjoyF2YMCYFXpS0bHoudU2Kzdp3fi5gZtvv5AuxdUnPqVrTrI9rx2rvpwTa54TvabFVmgAqn+usRCO/
amKmliR4hXg2zdz+ERbaUrk+TwPzSpo8Wft0SWNF7f31gVrzyzKe14cVgSFp3BjfZ/
oMsDZpLlvIg6QpJU2
bNsV2c6Uxqe2PiGrWJ4DSwLdBVCrvLpI0fzr2IHF/
HEpUTi6VL8IJp0vuAFqJlEB4kyjdmQFYmvBPG5Yexj0JNe2zGcNsSpLybAuidKe/
7Vsyh9QqdCLUmYsCk0law/a91ROKf2b7Y3sE4YfdJ1VR7EQomgv5WIdUggvMLOlaYjG0AvHMehrK/
dIINK7Bz4LA9MClkvYmGit+SzQ1m9L2QPh+kqHQWCqf9e3AsZKm8/
cb8tbe35roxl2YMJ4GPiE23rdzxcM3zVN3ZpwysJCH0cR3/
UBJ21SfRymZ1p1Iet6XKb6mQdGk7FEiIXlOWscZ+K4yFi8LPJIpxML4uRnoB2wAXF+tUpU0H2GRsVs6V
OZU9WEU8LWkdW3fkcaLziT/
fqLyZbTtMSpNFdudmi0GMYbvTGxQDpU0PVGRNCrZwM0KfNlmnlWoE0npvxDwAfBz27vU3rP9taQTiabi
b6Tz1RHXHCXB3GA0zvh+LWBHohnEGsC6tm9Nk5faIurh9FOoAxrXXHEN4C/E9/1gwl/
oeeAuwlS9+C83iEoZ4dKEl+xtku4i/IZuS+qPQjuQfOfWIRrRfFU7bru/
pEMI5WYhA2kCcZukBYiF7K+AXYFJiMXPsYrmWsNdmtLUnTZKtcmIDvP/IJTM0xCT74OgNM7KyCPE/
dAvTcCfAj62/bmkHQj18l45A2wGkvrsb8C1kkYAVwH9icXo0cAYYj5VyESa195IdKA/
VNJFtt9VNL2cnxg77umIi9KJiTQuj5T0MLCHpJttD6u8Pwnh2T/
c9sfZAi2Mj0HAhcRz7Gki4TyASCxvB1zicV7+5T6ZQNK98qSkIcD+kl6z/
TIxXiBpbsKO77r0TzpREvvtSuV7fi3wEWG/
NxmRdxoGPCzpcUI9+zGM9QkuFqL1ZSXCiq8L8LGk1YjNmVcdPv4CZrD9KnTc55M6aNwdDkmDCd/
TG4jE5gpEidsOwHm2P+youxQTK5UE81XAp0TTrDHEztF6hP/
vENul0UODSTuk0xCJ5lXS757p7cXK5LzxSFqKUAF+BgwnrHnuIxIzZ9ru0H5PzYikZYjGZlsTpemTEDv
e1+aMq9mRNCmwOqG4WRZYgLhnngKeBW4otj7tR21uJGkuQuU/F/
AKMYdaitgAOML21blibDYUzZD3J8brZYhFT1/gRNvP/9i/
LbQPqcriGMK38XninlgNuAM41vaQjOE1DWn+ejGwGbHZcgfRnH1nYjPyzDImT5xIWppoiLk8MY4/
TSSer0qWfWXdXUckLUx8vj0JS4xXiGbJvyQqAf5se0T53PMjaR/gfCInsgkhfpwFuBs42/
ZLRWlef5JYqJZknpxYV3Qi1hhvERaW09het5bHyhbsBFASzA2ksihaHTjH9sKSegEP2Z5FUg9CZTBv5l
CbGklnAf+y/
cB43uuwN29HIpVKTVNLJCs60M8OzFsm5u1D8m2clUjOrJJ+liNsAE6yfU2+6FqbdH8sCBzFOEXB1bZfr
5wzCbHAfbx6vDBhKBrOHEL0PrgX/n8bknTOOulnY2Azhzd2oR1Q+PZ/
I2l5oqxwKcKrbnLgMWCA7SdyxtgsJJ/SOQiF0+SEJ/NIQm1m219kDK8wHlKV3orAvITi/
LGyaV8/0nz1C6K6aHNi03E2QjB0Znn2TJxIWtz2oDbHJgO+SZXFJcnZICTtSAiJehG2Y9cBpycLhkJGf
ux7n6pgpijjR+ORtDkwlBDfLUusAWcE3gWutP1wR85RlQRzO5C+RPvYXkfSvsD6tjeWtCVwmO2lO/
KXaGKktyfmlCOdwcOIxaio378XxfqQcUeZhng58SO3SLAz9KDs+yMTgRImhL4oky0259KlcVWhOfpYE
KVuTJRynmU7b+W8aH+VMaIfYmKll1tvyNpRqLEfHuizPYkJ//
rQl4kfQHM4zYN5pQaX5bxZMJIlm3HEvdDb0JRcy9wge2H0zll3J4ISJuSCxCbxvMSPv2v5Y2qOaiMDT2
BPYnxeDhwAdEgfBrbo4pd0sSLpDkIlflQouroDsIebnjWwJqUynpvFeAt268lr9kxbuPjX9Ya+aisOVY
k7H02Izbtbwf+Y/uxdF65Tg3i/
0jwzwh82AyffelG32DSJLAf8L6k44HfABdImoIo2yklnQ2gcnPOTuwGfQH8EThf0smSdpW0ULYAW4DKI
vRcYCpgd6KU+as0+TtI0Yym0GAUHbRXlHSQpL9I2knSSpKmtf15MwxmHZx9iCqLPWzvb3s5IsG5mqTet
eRyGk8K9aH2Wf6SsLyoJS0PB35BeDbOBZwlaXpJnRQNngrtiKRlJN0u6UCiCc074zntbEpPkQkiKfv+S
ihhjySqW44m1Mx3pnLaksTPTFKYQ/jIXkhcq/
2JZxaSlpQ0VabwmoXa2vhgYHGiAfIMxP0wU02kUrNYyBNiYXxU7o85geuJhqWfExsFAyUNkHRGmv+WHE
j9qK0hTiFseiDUmIdJ2jclmzusn2wTUfv8LySqk3Yl+o2sAPSX9J2k7ct1agxpI8aSFpbUR9J7km6QdG
iqRJqEqBzr8BQFczuRdouOJVRpXQm/
tJsIZUjxX64jkuYnzNJHp78nJ1QeixCKjznT7362z8wWaBNTUYAsDfzbdi9JswKDbHeX1I2wAVjKlYZz
hfrSRiF7GPAh4cG8GPA68C1h2XN0vigLkh4ADq3Z+FTUIE8Bf3LqyF3GiPoj6SWiquh1RSOax4CdbN+Y
FqC1a3NfuQbtj6RFgQOBTYlSwleJPhb/
sn23pHWJMWa6jGF2WCpj9VbEJvwW41GI70fYyCxXFIB5qVyvN4Df2L5H0lBgP9u3SLoCONf2g5lD7bBU
PuO3gXVsD07HHwMOsP1gGQsmTipz3ieAvxPWDF2IZOcRwHyEqnlu4Hjbd5ZrWR8Utp+DbM+Q5k63EwnN
xYnNy9PK55wfhd3eANtLtjk+FZFoHmz77VKtVH+UmiZKupuYy95C3B9rErZLkwGH2740Y5h1oSg+Goyk
yWx/ZfsRYB2FufdMRHL/4dp55aFbHxT+QdcQHo1I2gh4iejy/
CpwQ0puLgC8kS3Q1mEWoFa2uSnwQnq9LHEPlORy+7AvcJnt0yRdSTRxeJ1QyZby//
xcAVyhaNr0FDBS0jREo5RHoIwRjSBNqO8DDpV0GbAT8HRKLtcWqvMBL0K5Bjmw/
SzwG0knE4mBUcRz6xqFpcPzhOq88NPoRGw0bgPcmWxiat/
9mjXPZYSyeY30upCJlPicFfguJZcnITxO70mnrAUclC3AJiB9xnMBnWrJ5UQv0nypjAUTJ+m5JUJx/
pTtb4BvCIuTXSTdQlid7EBUUT73A1Uxhf+SSoJ+NWK9DWG/

0N324oreCafbPjVbkIXqdZoHeF3ShrZvrb1v+zOi+Xvt75JcrjO2x6SXkwJH236D2AQ7Mq1HNiTmtB3e
jqwkmOtMZed7IWArYI6kLrjE9oe2XwZeljSlpB6238wbcXORStY2SztEcxCLofeIz/
wRoovtCy4NmhpKZfL9ILCDpD2JB+dFqcx8B6J8rdBAPM63txexUwpR+ryJ7RckLQGUZ1AGKmPFzIRP/
MxEudo7hLVPb+BU25919InGxEr6bM8CTgL+QixEj07vfStpHeAz2+8WlVNebP+h8ufVAJLmAboxbuOy8
L9Te66sCOwGY7/7nYjH1KS2R6ZEZlGJTxx8AzyU1PujgZdtf5kqJb8p64q6sCowk6SrCSUmhLfsZxljK
vx3TAL0Bf4maS/bb0qalKhkXdX2AGBAqgIoTecmkMq86CHC1u1iIsF/
djq+EfA2jFOYt3+UhQqLEarZCyVdS4hangReTBsyhQZQWfPNAgwENiCsSoCxCf5rKn936DVfschoEJLu
JybubxDdzv9C7HwvQXinTQ38wfYd2YJsctLC6GtJKxE38hpE4kaEmf2fcsbXKiSbjCuJRjSDgWHELvep
ZSHUeBQd0A8Gzic2Wx4h/
BpfB0YAs9r+IluALUpFJXgrcCIxOd+Q6Lz9BjFeDEqbZiW5WWcUDZw+tf2RpDlJ6jTbI9P7CwMHAO/
YPqwsjNoffb9R7P7AEbZfSZvH07RRFxZ+IkkR+xaRuL8BuNn2p23OGQFs5NLwcqJA0u+IDYEpiVLbC4B
NgFdsH5kzto5OJRGwOrA1sDFRTQRwDmGb9KDtt3LFWPhxJPUCzgQWImyvRhFzq4ds/
07ShsB5tnv+yH9T+B9In+kcRIXqP4g57QxEH6ozbP+rzKPyk+a28xNz3l6EfUxnQlV7uO2BGcNreiRtC
VxFVI7dQNjl9rf9XtbA6kxJMDeAlFDra3uO9PcOwPHAx0RX+kcI1cFltj/
PFmiTI6kv8DfbD1WOTU0oE8bY7p8tuCakMimfCfgZUW7+OnBgUgAuRJThPmj79h/
5rwp1ICk2ZPurVO75ne3himajqxHXpqftVfNF2dqkUs57gK1sf5g7nlZB0dDsasL64gXgGaKE9svaZkt
S9y9B2Aa8WZL87U/Fr+5colx9d0mrAb8DtgT6Az9vmwwt/G+k+2GT9LMG0IMYH24F/
klUuQy0XZryTkRIWpWolJyeSOycD1xXVGj1J90jGxACoZUJ+7ctbffNGljhB0nzq1WJ51oXQtX8KGELd
CLwte0D80XY8ams+xYjchpLtHl/
VcK25yTbX2YJsvCDSJqesMyYj1A2n2T7o7xRNTfJpnUuYFFivrU445L8u9u+5Qf/
cQeiJJjrSEVtcywwm+2d0/HtCAXzIiWh3FhSGefWhEL2PmBGJ5/
fykC4EXBPGezqS+X7fxawILE4XYJQDowE9iNKcU6zXewxGkTle74PMXgdbPtlSWsTpbWdgT0IP9OLbb+
eLdgWpXKvLEhYMjxu+29tzulS8esq1BGFv/U+hIpjbqIRzQgi2fwC4Rv/
YimHzkvlPtdi8c8AACAASURBVBlMNNe6XdK9wB3ARYRq8+yyWVxfkvpvAyIxsxxhQ3Kv7bWyBlYAxi5Q
lyUS/
wJG2R6RN6rmI9nEdCI2579r8950wBe2i8XCRERKKi9JrEE+JtaCL1c3hyVNQTTT+qaM8RNGZYxelxBK7
FE9njm8wniQtBxhF/
MW8KaTB3kRUeQhjefzEhsx19ke2gz3T0kw15E2i6EuRKONm4E+hPrjlDRhgbAtKh9+nUnq8ROJz39J4h
q8RyQNhhIlCW/bniVbkE2OpHeBVdJDsifx2Z9EqAd2JkrWflPKCxtH2pW+D/
h1raRZ0pfAp4Ra83Ei0f9uvigLkn4FnAFMS5Rx3gzcYvuZ9H6Z8DWYVNWyItE9e3FCmfYJsTF2g+2rMo
bX8kjqTCSTBwBjiI3K1ZO1yVvAxrafyhljRyeN00sBt3o8jXclLUskMYe2e3AF4HuWSusQDbSWBBYB/
mr7UEk9bQ/LG2VzkZKVYz1mJXVNllVLAzvZ3jtrgIWxVO6P9YETGNdgfDRhOTYEeNL2zRnDbDoqeY/
jCeu9k2xfkDuuwvep3B/
7AtsSdqHdiE3KQYSyv18RHDWWlANcEJgTmIZ4Rr3ebPYYUJr81ZXKbsPehD3AuUB3QmEwmaSBwGDbH2Q
KsRV4nvBqPBiYgkgaTE/
sZI8kykCGZIuuyZHUm9i4Ggpge5gkAycmxfi+koYRXk+FOlNJSG4KfFhJLi8LvGF7Pkm/
IZrJvU0kNwv5uIFohLkIsAqwJrCzognEL1zp8FyoHylx0InIHXxKWC30T+/
NTCScNybG7qLGyUhaFF1MlP+/
BvwxJZeXBLqW5PKEoWiUeCRhXfWVpCkJW4yhxLPpEduP54yx8D2OJmx7dpB0DVArZz5W0vWlOmzCkTSZ
7a/aqF5VOWV7YLb2j6zwX/
B7YmP4qKQyXzH9rEfkPG4u43n9qHyOnxI2Y4dJ+i2xzn4YeNT2oFzxFQKP870+FPid7WskvQxcBvyK2B
x4Dni9CFvqT8V7fBvgD8DkhJBlNDBM0XD0Ttt3ZgyzrhQFc4ORNCNRpr4LsBKRWBsMLFfKn+uLpBWIpl
hfSpoPeJcofV6OUKbNBbwM3G372WyBNiEVW4Z9gaMID+bhRNLs0Jovl6TZiWs0Y7Zgm5iKmuBqotnPYe
n42sAytk9Mf29PNGzaNmO4LYnCG3tLYpIxmFCSP5WUmlMTypulgdttf1Ame40leQceQowXX9k+OHNIhR
+gosKZFfgTgO39M4fVIamMFUcQc6M9bH8jaRvgFMKjfHngA2A7F5/r7CT105u2Z0t/
vwOsYftFSc8Bu9l+OGuQHZyUSD6ZmL++SCTKhlfXa5JeAo62fWWeKAs/hKQ9gNc8nj4vkqaw/
UWZU9WfZDvWmVBmLk34y84HTGt7lZyxtTqV9flCwG2255TUHXjB9gyS1iDyU3938e9vCJX51lDgKNtXS
HqK6Mm2IDEH28n2Pc3yfCoK5jpTWQAtDYxOpc7/
ST8kz82VHY1ryi5qnUiJy4eBz9NNew1wf9o5vTP9FCVag6g8DPsRpTcXpr+7EbtzixGdznckrAAKDaDy
3X6bKCevPZPuAu6q7KJuRvhhF9qJyqRhe2B3Qh24JHC5pHVtv03YMowiFrbA9+6tQp2oTPbWBI4Aniaa
bCwPHJw2ZMbYvi9nnAWQ1ANYH+gNdJV0nO23Jf0RmDJvdE3BVsSCp7aw3BDoY/
uQZMewOyGSuDFTfIVxTAs8pGgc/
hTxjHoxeTjOXpLLdWEWYCai0eUahMrsLUmvEnZvI4hn0Q25Aix8n8q6uwexcf+VpFfbWvo4NfAtc6qGM
JroafGh7UsAJC1A2AAUu7eMVD73noxbfy9LVHwDfE7YVh5frlNjSOuNbsD0tq9Ih3vY3itVTZ5CCI6a5
vlUEsyN43TgXwo/
5tqXZTLbLwAvlERnfXH4+XaStBRRgrAPcJqkj4hEzo3AXS4edQ3F9nDCnuTg9DBdjPB76kckPZcn7BkK
jeUm4J+S/
lNV66dJ+HREZ+2jcgXXoogYC3YkEjjnAEi6E1iHSDSXcaF9qJU77wA8ZfsASX9gnHXPSsRk/L4y4W5/
Koqb7oTV2PTEJvFuwEFpnJ/
U9iM54+zIpAVPJyJpObDy1sWEJyO275R0KJE8KGTG9khJfYg51KbAs5JWIiok++WMraNTGXu7AccQ3/
l5iEbVixAqs7WJBPRQl+ZwEw2V8v9OhFf/
gsBLkt4EbgeuJ5SbZW5VRyrj9JbEmtvA7JJGAn+zfV3t3DKHmii4F/
hQ0aNnGNBZ0vnEZto96ZzOJHFSoe7MCdwiqQuh8K/5Lk9OCE+byoe50/99SuF/
ISVwOhMTkgtsf2u71oF4SkmXSepRBrr6khZK2H7S9oG25wcmIbywPyOSaa9JOitflK2F7VG273d0FZ4H

2Br4JfDvvJG1BA8BtwEPSzpD0uqSppa0GVH6fKftwXlDbC0qz/
wlGKfwh1D8v5ZelzG5HagsSLsRC1KIZ1Pf9HoZxqnIyzVpf2qf+bZAF9srEyrzN9O1m5NQqhUmjG5Ecn
mD2oGk2v9EUqe0EFoGeCBTfAXG+f9K2gl4lvCL705sgv2OsHH4U7YAm4t7gJeIKsgt0usjgD8CVxDNsI
7IFl3he0jaTOEbj+3htre3vTRRcn4MMAMhuDg1na8f+r8K/xspuTwNcCKxrjiIGLP/
RXgxr5AzvsL3sf2l7cdsf2T7OaKvRW/
COvTEdFrJTTWOV4ELiLH7DSIndQ7wZ5KaPM25moLiwdwAJC1MdD1f3/
YnlePdiG6R3bIF1wJI2hZ42vbzbY53B6ax/XKeyAqF9kXSXkTibH5iUHuZmGz/
tdl2SzsCyQPtOWKhOoiYVPzT9vRZA2tRJG1AVBvtR2x8TU8oOF4C1rU9tCiY25+KMup84APbh0m6PL3e
X9JfgLlsb5851A5L5TM+mGiMtQvhB/hR7fsu6QBgG9vLZgy1wNj1wxDb3SvH5iDyPG/
mi6x5kNQVuJTo2zIQ+ApYi9gEHkSoxO8nGmIWlV9mFP0sLq6NA6na4gXgybbVqpKmsv1ZqRKrDxWbsS2
JxrsrVd6bjNiEmcH27tmCLIwlbaz0AlYDXix2SvlJuaqjgbuAU2wPaabnU0kwNwBJUxO7FF8Q3To/
T5PD7YFf2l614oVaqCOSpiUmgZMQvkKDiQnhA7bfzxlboZCDdE/
UFqWdiPLOphjAOhqpNG1TYG5gIUI9MDfwTyLx/
EBRlrcfaYF6CLAycT1uIBrU3GT7mJyxFUDSesABRPXL40TS/01JjwIn274ma4AdnIpa5lTCk/ExQrk/
HbAmUfJ8tu2+4/8fCo2m4i+7GLAvsK/
tr8Z3Tp4Imw9JGxGNqh8FriL8+X9B+JUvWjaEJw7S+L2A7UGSZgPOIp5ZnYH3iWfZU8Aztj/
IF2nzImlzQt2/a7IArR3fkdicXL88n/IhqYuj59eeRA7qW0JIMTWxafYw8C/br/
3If1OoA4oebAsSealXbTe19VhJMNeJtrsOkpYjlFHTER7A8xC+Nifavqs8cBtDmnAsTPikLUDczMsCUx
BN/
3bJGF6hUGhRJM0FvJ82HLswrsR5IcJSaZ70+nTbxcqnHZG0FrAC8BHwhu2bM4dUSEi6kvCMnx04gbDH6
ATs3DbRVvjvSSqzTYmNrZHAr4ikWjdgKNHc7GxHo+pCZpKa/
HDgVuB44PmyUVxfKsmYSYH1CIu994BjUp+XwkRKssmYiVBpzkfMp2YBZgMet31wxvCaGkkXE2vtswmLm
fWBPYBzbfcp+Y58VJTmLxGN5G4mBHgLEnPezYCzbF/QTOrZiYXKBvGWhJ3VAoRtz6dEcv82oJ/
tNzKG2RBKgrkOSJoTWM/2RakMYcpUijMlMWFfldhJvdb2OzljbSXSAmoG4NdEZ/
TjbPfPG1WhUGg10rPoasJ64fn0+zWi4/aY9P4sRKJ5kO03ijVDY5E0I2ELMD/
hWd63qJzyU0nyHAQMt32VpLWJxdAcwAjg+FKi/
tOoWGP8glDE7mx7SHpvMSKJPxrYr22ZeaH9kbS07YGSfkmsJRYjEgSfEeuKZwmbpc8zhtk0SOpaU5ZJm
p9oqPgzonn1Lc2uOutIVBNikh4E9rH9dPp7CmIDfwFghO0BJYHWGJJVz86E//JcwBPAJcCVtr/
IGFqBscK7fsBmtr+uHJ+EaPL7se1vcsXXzFQSzI8Ct9s+Mh1fh/
D435Gw+fldsz2fSoK5DiQfxzVtH5RKOncDbgSeKOXO7YukRYF3icTNt5XjNxAJ5gE/
+I8LhUKhAaRGKPsQycy5iUYaI4hE82CiSdMbtkdmC7IFqKg55iFKnz8EhhOJm97AabYPyhljq1NJgD5O
KGj7KJr4dq0ujgo/jcqCpx/wiO2/puN/
IBQ2jwFTAm8TGzCUja72pXIPLAMcZXvjmniF2IicB5iXSJ4tBGxle1S+iDs+kmYgbDBmAVYiqlneIZTM
MwLb2749X4SFtiQR19aEanaI7ckq79XuobWAh8rY0RhSknIq4h55n6h8md328KyBFarz3XmBc4ln2l8I
e4bP8kbXWkg6i5jPvjCe92pzsqYSFZUE8wTS9guhaPD3Z6L8oBvRKfJR4EngPtsjsgTaAkjqBfQhmnO8
RDQ0+4BQe/QHZisqj0KhkJPk0b8iochcHJiVmJR/
BNxo+6qM4TUt1bE6NTZb3vYWlffXIRpuHGH7rkxhFhKSDiRsxS4o43b9kfQasHnyL+1GeJWeDFwMLEn4
Mv/J9n0Zw2xJKomB04hGWb9uW2YuqTNRoTe97RezBdvBqSQi9ye+//2JsuVOwDOE9/tkRIPRUuY/
ESFpaeBKQoU5BXAe8ABwp+0v0z0y2nanjGE2HZXn03rAscDkwCtE36k+tu/
IGmDhe6Tqlz0J3+WXgTeJCsphwACXhu8NoXKfzA1cB7wF7N0qlWFd/
u9TCj9GmphUJ369bG8DIKk3sAHRhXh3onxkRLPtUuSicvMuBXxJJJMvJcrMf040eviC8Dq9oixSC4VCD
pL6rBMxZHxKLGL7p/dmJhLOGwNKx5qqVGoi4ZeSPiUam0xHNH8di+07Je0MrA3cVa5BPlIPi8
OJe2YSSfcSm/UfFd/lCUfS5IQtzKLE/
bA7YblwQSqVfVjRNKtYuuVlNeAcgLbJzfT3e+mn8BOprMWuJRIwWxG2ek+k42OAF4DJJX1e1m4TFc8Qz
7CrCI/+aYATgSkkvQd8DPSF0gizznQiqvD+CRxEVOBNDywP/EfSXravyBhfyyNp6rTWwPa/
JPUnql2WJSpfViOqNQ6ljCENobJ+6EZYIy4C3CTpDeLZNQC4y/YnmUJsKEXBXAckTWr76+RD9JDtOX/
gvLJgrSOVBPNbhApwIJE06E/czDMQno1DgKHlsy8UCrlJPqeHEFY+X7k0n2k4yeP6DqLD/PuEkmBTwm/
2rnTsS+Buwkrp1jJe5yFtxnQFNgKWI0rUZyISzC8RfrP35IuwOUiqpiuI59BHhK/
1Vem9+Yi57EwZQ2xpUun5F0Ty7FlCefYa8ez6sCTLGkdS9C9PCFXWJqyU3gIOsP1KztgK/z/
JY1bA10QT2F5EhVgn4D+2Xyzj+YQjacqaUCvdI3fZXrrNOdsQGzSblj4JeZC0MnCp7flSxeT8tp9oc86
cwNLAHcUuo/
4k4eMztXsgecLPTLgbLELYJS4HnGr7kmYUnpYE8wRQKa3aDFiZMLfvYXslpUY16bwFgQNt75Qx3KZEUl
dCtbwckWD+ipgQzkqUfN5ClLjd0mw3b6FQ6BhUNsPWBI4AniY865ZPk8C1gTGlHL1xpMTl4sDqwFJEp3
kRZYLfpWM3EmN1GSsmIiTNQjTq3Ra4zPblmUNqClKV3bKE//
uD6dhchM1bV9s75IuuNamsKzYlSv5PJ9Rm0xKd518lmvsNBYaNz9OxUD/
SuDErsA7RCLYp1WYdldTfYhfgBtsv546nmZF0DNEY9gbgOWAJokHZ3ZVz1gb+YbtHSeq3P5XxY07bwyV
tCfyHUCk/Tsxx+9t+PWeczUyqSj3P9hapf8jGJGuS2vghaTpiI+xl26NKgrkwXiTtCKwC/JKY/
N0PjCQmgU8RjVOmtP2bUqbTGCRtRHR6fpRQfMxINOzYEljE9owZwysUCi1MpYlDH6LM/
wBFU63lbf9C0uFAT9u7NONEIzfj+0wlTUVYk6xMqAmWAa61fXC5BnlRNGH8E6HgHArcWhZE7YOkJQnV+
B22n8wdT6tRGSv+Abxm+6h0vCfRfG5losFfd+Ia/
SlbsIVCJir3yU5EA8a1FI0adwQOJtaBB7o096sbimZxKxDq/
sWAhYkeR1cSViQzpeNP2L665DvyUv38JW1A5KjWJa7Tt8Cetv+RMcSmJFVMLmH70aRkvpJIMA8nqutfI
HKFI5r5+VQSzHVE0hmEb1ev9DM18fCdHvij7QfKjl59qSnFU4nUesBOxE7dMbbfyhtdoVAojENSX+Cq5
In2GHCa7Ssl9QMesP33MilvHJVF6Vh/

```
usp7cwBT2X6hjNPtT+XarAscSFiWfEHYXPUEzgBOLtelsSTFJmWDJS+SXgJ2tv3g+MaEtHCV7YF5IiwU
8lEZL24CHrT9V0knEpVJdxG9j06pVWYU6kdab89IKPtXBlYlRHbdCJuSJW2/mi/Cwo+R7E22BR6z/
URZczSGyjNqAWJTZgXi+dSF8Pa/0vbFOWNsJCXBPIFUyhG6ArPXVDbJb2V2YnH0nEuXzoYhqavt0en1/
MCuhJr5YMIaY3TO+AqFQgHGqghOB/YD/
k1sPnYmvGXXtT20qGcbRypX25ToqD0vcC9wH1FeOzJjaC1PZTJ+PfCs7SMq7+1MVCTt5hbpwF1obSTNa
3to5e/O6aXLJkuhEEi6Hfg7YTt2P/A723dIegQ4w/
ZVZcO48UianrDMWNL2ybnjaWUqeampgTUIL3kDDxJVL69nDK/lSVViGwIDbDdtQ/
GSYJ5AKouifQiPtDNsf5pKPOclGqUUA/
UGkMqhfkF0Ql2JaFTzDqFknpEom7o9X4SFQqEwjqT8OIRQfSxEeNktDdxk+5icsTUzFQ/sDQkl7HmE//
JJRPOm2YHnbS+SMcwCIOkh4Ajbd6W/a9fuUeBvtq/NG2Gh0D4kNfnU4/
P9bdZFaaHwvyBpc+ASopfCzbb/nI6PBOaz/
X7O+JqJylg8D6Fa7g18DgwCBtkekTXAAvC9vNQhwA6E7zKEinZmItl8ue3jcsXYCiTh6QbAPoQ9xm1EY
8xRWQNrJ0qCeQKpPHCfIcpx+kjam1BJzQpcaPvMokqrH5Xduf2Bk4H+xI3bCXiGMLKfDPiglH0UCoWJD
UlrEeVSHxENtm7OHFJTU5lwXwUMsX2kpOOIjchjCVX5Vbb/U8oF8yJpV+AwYC/
gcdvvS5qSmKAvWhaxhWan8rzaANibWKS+AtxJJAuesP1hzhgLhYkFSQsCMyYbyikI//
4NbS9f1t71o5LvuJOovnuOKPWfARhN+PoeZ/uZjGG2PJXrdD+Rl+qb1MzTALMRApdhtq8v8936Uxm/
dyAERf2AeQg1+bTEXPZy20fni7LxdMkdQEcn3cSTAD1s90mHDwP2IB68x0i6sZQk1I/
KZOFawud6K+DXhP81xOf+AjC5pM/L5KJQKORG0oxEt/P5gYeAC2x/kDeq1qAyge4FXJ1erwscb/
vNZDv7eZtzC3m4klBG7QpsLmluYrO+T0kuF1qEmjL5EuBcYk2xDNGk6Sqgm6StbV+XKb5CYaJA0izA5M
Akaax4HbgGqG3ai1BsFiaAStJsGmBp29Ol3MccxHjdG1iE6IFUyEilsuVSoGs69inwKfCWpKeJzYAy32
0MtefNWsAltk+qvZF6vWxLCCLH9hFr/
xAbT1EwTwCVXaI5gYuAR4imfsvYXjd1knzb9nRZA20Bkmn98oTX0NrAh0Tp8wG2X8kZW6FQaE3alBReR
TyXhjOuvPA02wfljLFVkNSFKBccnjwabyEqX/
5NdHZe1vYrRfHUflTujz6E8mloOt6ZqAJbDHgXeAPoX/
opFFqFlMi50fbq43lvYeBN2x+3f2SFwsSBpL2AY4BngZHAJ8BZtp/
40X9Y+J+QtB6RPL6XECbuBvzJ9kdtzpvS9uftH2GhRqXCe0pgC+AU4AoiP/
W07SFZA2whJG0N9Lb919yx5KAkmCeQyq7eRsC+xELoPNtPSjoU2MD2aqUMof1IvnWzAusAfcfnX1coFA
qNpJqolHQwsLztLSrvrwMcTcVvttB4JE1q+2tJmwKHEkqO921vnjm0liT5kn8JTG7763Ts17YvzxtZod
D+VDZdlgOOAP5h+5rccRUKEwOV+2MVQtj1C6L6aG5iU3I7YA3bz2YMs6mQtB/
wO0KZOYJYXz9F2Iu9Qyhjvy4b8/mpKWIlnQRsAgwB3gemI67fGOB62//
KGGbTk9T9qxEq8nuIqop7WqknW0kw/
0QknU+oY8e7WyepN+Gd9ojtf5cEc6FQKLQOkn5FTLwHEZuP79g+tc05VwGv2T60NG1qLG2VyUkhuDyxQ
L3X9hXlGrQfFaXNpsBRtpdKx+cDbrc9d1GTF1qV5N94DGEDdw/
wIHAfoUIrz6hCS1IRdf0FmMP2Dm3ePwH4ttbsr1A/
JPUANgKWA1YiKrZfIdSxzxBVRi2TQJsYqcyrngN2tj0gVdnPS1jELQVcZ7t/me/
Wn8rz6QBi3XcfMClhjdiNqGK93PYZGcNsF4oH809A0gxEme3nkqYDbiH8fx8CHrX9uu2XUwfPMVB8bgq
FQqFVSPZIexOKgfcJu579JH0B3JWOfUn4112WK85WoDbhbpuoTJUtd0g6k7gmhfalE6Ee35xYoNbYhFi
sVs8pFFqN+4AtgfmAJYHViZLnGZPCf2DO4AqFHFTW0i8BC0uaqk1Sc1ZCVTs22dPeMTYbNVWs7TeBC9I
PkpYCNiSeTXsAC+aLsgDRoyqpZx8hNiexPZyw5rsrVYx9l46X5HL9qfovH2n7MknTEo0w50zHP4Xmfz4
VBfNPRNLktr+UNDOwI/
HFmY+4oT8iFki32r4vX5SFQqFQyEGy6lmcmHwvRYwPAoYRE7ylgBuBA4tKs75UVBzzEBO6x4jSzk9sf1
M5bxLgC6ID/ag80bY2kgall1cTif4zgMtsn9PsE/BC4ceorTPS626EJ/
kiwKW144VCKyJpKuBOoDvR8P1RYB6iz8J2tgeVCpj6UFFlbknMZ68CBrSpCKudU1SxGUjrjU7pGqwAnE
/ko44CXiTmv+/
XbMgKjUXSecAdtq9tc3wyYEyzNvarUhTMP5HK5O492ydKmhqYjShDWBBYmWg4cF954BYKhULrUFnYPJ1
+aguiFYmxYX5CnTkmJULLQqiOVD7L7sCBhKrgTWCApEeBobZfBNYnrEtGlXE6G78ivDPXAXYl5lHvSeo
EDJI0nKgYK/
dHoamp+MvODGwNrCqpF+GjeZHteyQ9WJ5ThVYmzZc+A1aQtBNhdXU4URW2q+1B8L15QGHCqD1vJiHK/
rcDPpf0IiGSuMv2C1BUsTmorB9qm/GfEdelF9GQ8QvgNeANSbeW6pfGkixJlgC2kbQs8ABhbfWW7a/
yRtd+FAXzBCJpT6Kx3wvAu7Y/SztJMwFf2v60JA8KhUKh9aioOqa2/
Wmb9+YAprL9QkluNpakGlgE2AzYikjuDwBWAO62vWetDDRjmC2PpK7ENdmaSBr0AF63vUrWwAqFdqAyX
lxIfPcfIdYWKxA2GcemJHNZUxRaGknzEt6m3Yi1dkmaNRhJ1xCbXQOJpPM6wF7A20QC8yzbV+eLsHWRt
C/x+bvN8R6EX/byhPr8YNt3ljGkvlQ/T0k9gZ8RFohzAlMSyf/
PgRvdIk17S4J5ApDUnSjpnIt46D4GDCbsMV4H3irlnYVCodCaJBXmpsCeRHXLvYS35g22R2YMraVJnmj
LAdsA59p+vCT581BRbU5v+6M273UHFrJ9b57oCoX2R9IooLvtb5JgZTrgz0BPYJcydhRakYr11U5EYnN
uYu09ArjS9rUlcVZfKp/5isRnPHftOCGk+yPwMZFA2wbYw/YjP/gfFupOsoI7w/
bGkuYCLgcuIZKZH7Q5t8xzG4Ck3wJXV62rJHUmNornI9Z/ywD9bPdrhetQEsw/
gYrKYDPGNaT5HFiFSCZ8RiQRHgPOLoNdoVAotA6VpNmGhKfseYT/8klEw7/
ZgedtL5IxzEIhK5XF63bAL4iywleAvsD1tt+onpcx1EKhXUiWGNcC29t+ro0yagQwRxGuFFoVSbMBjwO
/sX2XpKWJZpg7AZvbHpA1wCajMkZvTmxybQcMqTyTas3MVk++s11s75Ix5JZG0izAccBqhEXGCKA/
Mae6iXCOKXOpOiJpRiK5vE6qljyHyAE+Cbxi+4t03pTAaFf6wDQznXIH0ME5CnjA9hm2L7b9W2ADIuE8
CjgM+EfG+AqFQqHQ/ij9/
g2h+jiZUKFdSPgwX0+MH7Vd7kKh5UgL1+7ARcBtwO5E+e0uwEuSvpbUvSyICq1AUgW+CtwKnCCpZ7pHZ
pZ0CDCsJJcLrUiqBoNYY7+UksudbA+0fRhwGrBPvgibk1qPEOBmop/
```

```
IMcAqkrpLWh84CHginT4CmCZPpK1NZR2xPWGlNC9RXX840eyvL3Bq5XoW6oTtD2yvk/6cgvAq/
wPQD7hZ0l8lbUE0E2+J5DKUBPNPojLBexMY0+a9x4GpgBMJ37SF0g5roVAoFFqAyhjRi9jFBlgXuNX2m
+nvz9ucWyi0DJVFzvJEt+1z0+8DbS8G9Aa2sv1etiALhXbECaLEuQvwmqT3gFsIS5+jc8ZXKOSiUk7+G
vCdpKXalJh3JjyZy6Z9HZDUVdIMMPa5NBr4G/ANkWweABxJeMT/
PX3m2wI3ZAq5pamsIw5n3NpiuO1Lbf8c6EqIHmGcAKZQByR1ktQl/
bkhsFeaw25EJPYXAC4mYXlPoAAAIABJREFUNmNolQR/l//
7lMKPcDFwdUog9wdeBOYnJoIf2f5S0tzAOxljLBQKhUI7kyYc5xMdnAE+AOZMJWxrAn9K55Xy/
0LLUfnO3w+sJGlh24Mr748gFFGFQtMiqTfwd9ubp7+72n4e2DCV3i4FzEz4aY7KGGqhMDHwKDGXOk1Sf
0Lo1ZPYwD8lZ2BNxq+BBYEDJc1EeMIPTseRtDDQyfb/
a+8+oyyrqr2NP7Ob2CCSJEhSgkhUQBARL4iASFRMmCMq5mtE1GsOKMYXxesVRBHEHFEBEwKKqKDQxJag
ZFokS2z+74e1CzatKDRVtbvrPL8xatSpvU/
VmH1OnzTXXHOe3v28DnAE8O2B4h1ZvTYmD6C14XswMLtLZI4ttsxJcgPcbbFG46Brhzj2fvZg2i6LXyY
5m5YX/BTcOVMEWoJ/yn/
mswfzPOo9oHcAngksT5vyfD3w7iTf7LaPHJxk1SFjlSQNo6oWTXJLVe0O7EcbhjJ7LKEgjbKqejrwfmB
x4OfAb4AT+slmaarqFhx3TfKFqtqNtq32W7Q2Gcf0drxIoi3CAK+hLb4sDqwOvA/
4ocmz8VFVqwPLJPlTVb2H1n7keFoLqxOBP9HyHXO6mVQWSgykN/
NlX+CDtPvoJUlOGzi0kdL1X94f+HqSE3vHR/
KxYYL5fuhWi24ENqD1uLmQ1sB7drfitxdwQ5IvDhelJGmyzf2moqqWorUDeAZtdfvwUZgkLM2tt0C/
Pm1g08eBy2htMdYClgXuAHZKbyq3NFV1SbM5wNNoFVA7ASvRhl7+ltY/8w/DRSgNo2u/sAawHHB5b/
jrwgBd+wZNgN5r9bbANt3XmsDfaa/Zb0py1oAhqtO9n9qTtkNya+Bm2g6xY4EvjA2b0/
jqJfh3Bj5Pywu+BjiT9ji5yQSz7pVu2MAutImqD6NV3BxDq1a+fq7rLpTk9n/+K5KkqeY/
rVZX1dm0XS5HmmDWKOq9Id8b2CLJ3l2yYGlaO4A1gQcmOWzQQKUJ9u9eL7oWGbsCrwT+n48HjZLe68Sb
gFcBN9GSmq+lLUbuApwOHGb7mPH1H56XlgS2pS2G7Zfk0lGt0pxfVdUDaXPAdqG1NXlCklO8nyZO1zbm
ecBmwDrAbFo1+Wm03RV/
HTC8SWeC+T6Ya5XiQ8CnaZUFO9Gmnp8P7ObQJkkaHb0qjzWB7YCTaf1jr+tPDa6qRWg9mZf3A5FGVe/
xsjftg+r/JDlvruu4+KKR0SWTPwx8AjiHNiMnSW4ZNDBpQN3Or/
NpFZlzaLMr1gZup+0a3hp4exJ7/46j3mv0EsDTgRfQ5kn9OMmXh41OY3p5qf8CXkJL+F8ydFyjrhuQuT
3t8+AewPOSHDtKCf5pQwewgBmb/
Pgc4JtJDk4yM8kBwGNot+czB4tOkjTpem8YVgDeDHwdOBJ4V1XtVlUP784/
kbbF85puJ4w0cnqPl21oU7c/
WVVvqqo9qmq9qpphclmjoNv+D6110kO73uMrAQcAN1XVEVW1zGABSgPoBpRBq8A8J8nZSWYBXwQ2746/
qft5n2GinNLG3p9+lHZbHwo8jtbCiqp6aVWtN0xo6hm7n14IXNtPLlfVYlW1Q1f4ogkw9jxVVWtX1SFj
t3WSq5J8LcnLk6wE/LQ7PhLJZTDBfJ/0KpMvA2bMde5qWs/AB8CdbTQkSSMiyUlJ1gUeCexLe03YH/
huVX2JljT4QXd1XyM0srr3SP9L2/
o8i7ad8yXAO4H3DhiaNITdacP9oLXfWxzYAViSVqUpjZKx90dPBc7tHd8C+F6SW5NcS3vtWAr83D2eev
mOPYH3JPkScCVt0B+0iuaHwt0WAzT5xu6njYEjoC1aVtX0JDfTKv637457P42/seec5wLLJTl/
7ERVLVxV21TVxqOUWB6z0NABLKCOAL5VVbcBxwHXASsD69F6pQGM3H8mSRJ0b+x+3329s+uHtgXtzeAh
3dWs0NTI6iqUxz6sfrWqZtAWZrYDbr3HX5SmkF4i50ZgkaraiZZQflmSX1fVB2hJZmlk9B4XtwGbV9Ux
wCm0Hqcf6l11N9oQM7hrl7HGQVWtSGtDck5VLQqsnuSn3en1aC1CR6oqc37TtTFZiDYs+QXASXO1ad0A
ePUgwY2Gsf/
7WwFfgjsXuqYnua2qngdcApw2Su0xwB7M86yqtqNNiZxBewFcB3hXkiMHDUySJGk+VlWr0NrJzACmA6c
CBznDQqOoqjah7XC5Gfhpkk90iy6XAWvYs1+jqGsvtjrtM/aGwCa0qsErgV/
TdrtsnuTUUUvgTKQuSVa0WVOr0OaKPDnJFlX1ZOD9STZ0VsL8oao2pbWL+SVwOK2A9GW0RYHtBgxtJFT
VfrQCidcmubx3/Ezg9UmOGbXHignm+6j/
AtZNUt0CuILWE+rKJD8aMj5JkqT5VVUtBhxF2/11Znd4fdrgplcluXKo2KTJ0BWpXJRkVlUtmuSWqlob
WDzJ6VW1NPAK4NFJnjJstNLk6w0wW5nWfvIq4MG0PsDrAo8AHpJkqwHDnNK64aPvAp5AG1x9PbAo8Nkk
P+xaMbgoPKBe64vtgf1o+ajZwDeAQ5OcOWrJzclWVesAnwd+Q+tssASwE+197eNGceHLBPO99J9WRqvq
XFoF81ddRZUkSbpLL2HwVOBtSR7VO7cJ8EngkK7fozRlVdXXaFWAp1fV3rT2Sb+jJQauT3JjVa0KLJrk
vCFjlYZUVd8Hjkvysd6xhYE1gGlJzvVz9/jo2i08BzgHuCDJFV2SeWtgM9pOoz/5nDT/
qKrHAucmmd07NgO4xeT/5Onuh31pvckvpbXG+FiSmaP4/GSC+d8Y+w/
RTYXcjrZF5FLguiS39q63EG1b2/
JuY5MkSbq73nuqtwKbJXlGd3ws8fxq4AlWbGqqq6qFktzeXT6MVpV5C3AxcBatsv88YOaofTCVAKrqq8
B3aAuPWyb5a3d87PXidcCxSc78d39H915VbQx8hpYcuxr4C/An4AzgEhOW84fee6l1ge/
TclSzgUcBTwdOS/
LFIWOc6nr3wY7AX5Kc0x1flNYee6RniZhgvheqakta8+7Q3vz9FjgJmJXk7KrajdY7cFW3IUiSJN2lqq
Z3F+
+gbXP+PvBD4HNJLusGYX4P+E6STw0UpjSIqnoA8BhapeAjgBVoxSxPHDQwaQDd4+FAYFtgNeAi2jC/
b9B6lM+pqjuANZNcOFScU03Xe/mRwEa056HVaUNGLwP+1n3/
RZJTBwtSjLUmqap3AI9K8uRuQOy7aQsD04EPJDluyDinuqpaipYP3B04n/
aYeT1tYeYjSW4YMLxBmWC+D7q+gRsCewBPpQ0a+C2wJfDzJPv0qxIkSZJ0l+5D7F7Aa4FFaIP+ijag5u
1J/jZcdNLE6yUIdqMlA45J8o/
e+QcDayU5frAgpYFV1e7AS2kDzF5C+7y9LHABcGqSp43i9vPJ1G39fwmtOnZD2iCzA73dh9Or4v86Lf/
0uar6NnB6kndV1RdpVbXvtvBx/PVev58NvDHJZl2bt8/SdiCtCHxmlOeymWC+H7qKmy2AZ9EqmH/
nA1mSJKmpqn2AJ9Ommx+b5LLu+GLAVsDy3VW/
```

O+rbCjUaegmCc4B9k3ynd2x5YJkks4aOUxpKVS2c5LZ/
cXwdWkuZM5Jc5Ofu8dUtbm1KG6z4ZNpQv4WBh9AWgT+Y5BJv9+FV1QuBfWgDkmcDb0hyflXNpM25+IH3
0/jrvVZ/ErimS+R/
DpiR5PlV9SFg5SQvHNXb3wSzJEmSJkRVbU9biF+Ptt35EuB42pbnXyW5ecDwpEF022svSLJc93PRKvkX
Bn4AvDjJxQOGKA2qqh4O7AQsB5wOnJDk0mGjmlp6vWQfD/
wPrQ3G0sADge8C1wHHJznznpL+mhxVtUySq/
tJy6p6Cq2VybHdfbQV8C1gde+r8TfXbb8tcBDtMXIhbajfyVV1NHBUkk+PVTsPFvBATDBLkiRpwnSDT5
YAngG8l9ajbklaldSpwNm0yqirBwtSmkRV9TDaQK1X9quVq2p12pCmpQcLThpIL+G5AS3BeRFwLrA27T
XjKuA0WjulkasMHG+92/
vLwHOBG4FXAofPffvaFmNYVfVh4Owkh1bVpsD1c712LAJsD6yU5JBRrZ6dKL3WGP0k847A44EjkpxeVY
8Bvg1snuTiU
X3MmGCWJEnShOoSBocDLwaupG293R44gDap/
olJbhwuQmnydBXLX6AN1dqPNiRoBvAcYO0kew4YnjSIXhLnfcAa3ZbzVYCVaJWajwSmJXnnqCZvJkJVb
Q0sBexGa42xIq3X9Y+AbyT51YDhCaiqxwEzuyrmA2mJzb8AZwAn0HaEuUg/Qarq08DtSd5QVasBl/
ark6tqIVrr3E2SfGaoOOcHJpglSZI0IXoJgzcDj5k7cVZVr6O9H/3kMBFKw+j6kH8A2ITWQ/MxwDHA/
vZg1ijqVdT+DzA7yUFznV8SmJ7k2mEinHqqal1a26qVepWZ6wA7AnsAGwMrACs4hHd43aDktYGH0u6bD
WmLL4sANwN7JZk9XIRTU1WtTHvuubiqDqe1fpsJ/Bj4epI/DBrgfMQEsyRJkiZUVe0JvAN4a5Jje8c/
ThuO8orBgpMGUFVLJbmuqtYCHgucCFxiX3KNsq66/
yBgF1pLpROAK5L8fdDApqiu8vIbwPlJ3jh3a4WuHcPrkrzAqvFhzH27V9VKwNVJbqmqxYGH0eZcPDTJh
4aKc5RU1SOBnWnPU5sAt9Fa+uya5MIBQxucCWZJkiRNuKr6CPAE4CRaxdSOwJbA3klOHDI2aaL1ps+vC
uxO+2C6Ka2f7CFVtUiSW4eNUhpWVa0IfBR4ELAMcDlwTvc1K8nxA4Y3JXW9Yw8Gnp3kj92x5YF9gOcDR
yd5tX19h1VVrwXGdoHdDsyitTD5eXd+0SS3DBXfKOh2Hj0wyRW9Y4vS3svuBrxj1BeJTTBLkiRpwnWVN
k+i9ZldDfg9bTjKCYMGJk2CXruYL9KSZ/sDHwSOT7JfVb0R+EWSUwYNVJpPdMMwH09rH/
NI4KdJ3mQl7fjpLXz9L23I31uAFwH/DVxPqyI/
JsltJpiH07WN2Qs4GrgaWBjYiNYq44Akhw0Y3kioqhcDuwIb0AZXnwQcSnt8uDjcMcEsSZKkcdfrp7ko
8GBgOnDt3P0BTRZolFTV34CHJLmhqv4MPDfJSVV1HPCesWo0aRRV1SK03rJL0gb6jVVnLkmrHLzEROf9
03ttfiiwA/ArYGXgSFpf2dWBjyb5/IBhqlNV69Hum1f3K/
irak3a4ORXARsnuWigEKes3mPlv4EXAKfRHi8zaDvy1gc+leTAAcOcr5hgliRJ0rjrVUZ9AHgpcANwLn
Ah8CfgYuA3Sa4aLkpp8lTVKsA3gefRHg8zkyzfnbsGWCXJjQOGKA2mqmYA7wReCJxBq858BK1VxvVJrh
kuuqmnqrakVWAGuIyWLLsAeEuS46tqsVHf7j+k3nuod9MSyHt2C/
a3J5nTXaeAHwBHzT0UU+OjqpYBfgO8ZmyGSHe7r0gb9vdRYBtbvTULDR2AJEmSpp7ug9HStC23D6dVMD
8a2Ap4Km0y/c7DRShNustoyYBXAucBv4Y7e2ueYXJZo2isfQzwNOAJSVauqicCB3WV/
rsC2wIOgx1HSU4CHt71lV2fVs38fODAqjoPmFlVX0xywZBxil2AAwD6PZbHei5X1Uxgre6Y1f3jpLe7b
nfgmiTHjt3m3fHLgU9U1QbAf9EG9Y48E8ySJEmaKA8GDk9yXvfzucBhXfXHRkkuGS40aXJ1iy7HAwcCr
weuqaqvA3NolZvSKBrbUv0E4Efd5Z2BX3aXVweWg7slozVOuirlU7qv/
buF4c2BvWm3+wW2spp8vUTxWsCGVXULrcL8vCQ39JLN2wGfGiLGKW4a7bV5T+BwuCvBX1XTgYW7x85M2
pA/E/
yYYJYkSdI4q6qFktwObA2sXlXPSvLVsfPdB9XTBgtQmkRVtRLwAVoV5unAt4HvAZcCfwd+kuS6wQKUBt
IlLscSMj8Fduwub8tdFctPAo6Y5NBGVteK5Njua+yYyeUBdO0ZvgKsSRt0eSNweVVdCJwDnA2sS7cwM+
rJzfHUW8jaHJheVX8Hfp/knO7c2PltgD90l2uSw5zv2INZkiRJE6KqPk/
bXrgcLaH8feBbSWYOGpg0wXrDgTYC3gWsChwGPIS2HX1z7urr+NfBApXmE1X1YNrCyyzg6cAzgbWBXYE
9k/xtwPCkwVTVUsA6tDZjG9D6/wZYHlghyQZWz46/ruf13rQE/2q01/
EbaO9nTwSOAv4MPCnJTCv9TTBLkiRpAlXVwsDGtCTBzt3lRYHlklw9ZGzSROkNaDoYuB44IMnFvfNr0h
LOZyR5mR9MNWqq6tHARUku7R1bEtgP2IRW3b8M8Cr7AGsUje0Gq6pHApcnubw7vgot0fwo4Jwk37J9zM
Spqs1oLd/
m0CrG1wfWAFYGHpLkAQOGN18xwSxJkqQJMzYUpffzEsAmSU4YMCxpUlTV5bTqplO7vo3TgDld8vlZwIu
AfXp9yqWRUFWfAb7QPTaeS0veHJfk0q5i8+Yktw4bpTS8qvoD8DLgj2NJZBPKE6+X4D8YOCbJ16pqEWB
ZWsL54bTnqW97fzT2YJYkSdK46bUGWBN4MrBiVc2h9Qo8BZiV5AQrNjXVVdXatIKeU+HOno79D6A/
BN4+1zFpVBw89tgAHgesAuxVVVcCfwJmVtXZY1Wb0ijp7YLZDnhAkj90i5RjNqmqJyV531AxTnXdLBGA
JYCxodS3d89JlwOnVNW07rq+jtNW0CVJkqRx0Usafw7Yq7v8EuB5wCHAd6tqQVJkqQZWPSLJ9VT0F2DHJPgOHKk2qqloa2AJ4OrA9c
CnwGOClSQ4ZMjZpslXVTsCzaENfZwBXAGcBv6C1UrogyW3DRSjNH7rq5NfREpwX0loqbU1bsNw3yYUuw
EycqppBSzIvR1v02hBYkzbo71NJDhwwvPmOCWZJkiSNi14F87OA5ZIcWFUfAJZNsk9VHQb8PcnRgBg5VG
kzXImNlYCvgJ0luGDgkaTBV9T7awss1tMfF0rTK/t8DH0xy/YDhSZOqqnYGFkry/
bEhc93x7YHHd1e7jjYg86qh4hxVVbU4sBIt2fynJBeZ4L+LCWZJkiSNu6paEZhNexP+FdqAlBWA1yb59
ZCxSZKGU1WLJrmlqvagVWfuleTK7txuwGeBi5JsNWSc0mSrqtfREpe/
rKpXApvRKvtPBC40kTl5ul7x2wM7ABcBfwHO7b7f4H3xz+zBLEmSpPutqh5Aq8jchbbd+bCuh+Dp3Qem
5wGfNbksSSNvrP3F44DzesnlaUl+UFVrAX/rjlkdqFHyBeCm7vLttN7Ke9BaylxbVbNorTK+aXX/
xOiG+c0B9gaeA1xFa1GyNK291UXAkcDXBgtyPuW0SUmSJM2z3vTytwAfpQ1n2hN4VVUtV1XbAv8F3MJd
2zslSSOqN8DsKGC7qnpZVa3QO74LsEZ3uSY9QGkgSW7sZlksQXt87At8BDgCOtBt4MLAXFotOpLEFrX2A

/0uyOzAL+CRteO+jxq7Tew8sbJEhSZKkcVBVlwJPAs4BtgM+DixB+0B0MS3B/
PkkpwwWpCRpvtElZ15Pe824glax+WhaxeCLkvzVCmaNoqp6I7BDkp3mOr4aba7Fn4aJbDR07TEuANZJc
lNVXQlslOSKqjoE+J8kF/v8dHcmmCVJknS/
VNU6wIlJVugdmwNsC5wztv1ZkqS5VdV2wNbAdOAG4FtJzh82KmnydO1h7qiqT9MW5Z8MHJrk83Nd7xnA
1UmOHSLOUdEl8t8GfAK4GfgOsCWwOHBlksUHDG+
+ZVm9JEmS5kmvcmMnYHpVbUnrTbcVMDPJ8YMGKEmab1XVqsBatKFZx3V9T6WR02sPcyVtcX5LYJWq2hH
4GfCTJBcA7wH2h7uS0gOEO+Uluaiq3gvMoe2oOBe4DDgV+CHcrVezOlYwS5Ik6X6pqtWBV9L6ZhbwQFr
C4OW01hiXAze6jVCSBFBV7wBeCpwCrE57zTidlsQ5Msm1A4YnDaaqNgIOBd4K7AZsTnuMLEpr27BNkpv
u8Q9oXFXVdOBBtOerG2k7LP5qgv+fmWCWJEnSuKmqpYGNgWcDTwT+Rpu6/
YYk5w0ZmyRpOGO7XqpqfeAntGFliwE/pm1FfyWcC+BZSf4xxKKTS/
KOqlqQNUF4JuCzJ2fb+nThVVcCmwHtp718vSPLBYaNaMNgiQ5IkSeMmyTXAr4Ar4BfdW/
SVwa2B2YPGpgkaWjTaFvO9wD+L0oc0w/
r8X0uZZnNWdN7k8AXotL/YEXktb+NoKeBTwwaraDbjG9m/
3zASzJEmSJkT3AehS4S4tMtDxqGu9v2R2YpK6ZnaizbZPYFfvfP2XP0cbjgxwDLBdVV0MbAO8r/
IbWmuEs4IKwuhaTkSZ4mTnSRNlzEbtwEAzFDjV0S9KmgKqvJpFLHSwOaRDrZJwPLjP8yPwzoRN5D/
qfw8ES9hUiZ7kU9gfnLZwyRRNsfrAMsxsXBtrE+XTY8mB0E7LL4zm8CGs/vQE0uNGGlkrZ18+gaVtDIxAWPIsnvkZa3yKvf3xQuahlcr4bYOtMPWEY86LwnMl4sPwiyFlcNbr/aFfFDt9WbH8R5p/
ynN76CCO0kZnLZiIJLRlL5HT8DkCLY4dmqS/MUmJHYK1o6+FlpvGuCQgzexyc+EB/CL6mKQlF0MIP0zcU/
ZAhkkTKHfbUojpMaeyOGjdoDTJEpAcZd+sHPmgvOoTgqKz26+OuKb6rBYFDjKwBPhYFdkxR1/7Xa6dvBvvzjYxYc8xVcXb38mRxB63UdcKrCT1n6i7JmJjITQWmhSbTaCzTS29xAlSw+JkaTCjmHi1C2EB8uNHhqN39vR8Xc83ylsslDsufS8IzbVnzZ0ibdE6pGpzjmxfbl/
eOBJaSqcuHeUAIUWsqQ1EJjVy0jFbNSmn1TT3lZgwdxVkt1qFYBx2zbzQoBzGSptUsq5xFNzV/
b8eWK9lVrCrpTvZxBLvDZT3Mrv1lYw2r2qEQBLzXzjJBGZPuPXm6Z1qN5g/7Pz2g3NsbKkkRcNMiJ7HRHH
b/PRMydMoJqAOABvAm/X8jXoZB9gUPf3DVoXpZ/L5dfJPH2QpBDkMSwqqKoe7NP2+YeUjwrOVppK6/
Y7nrnb56wJt4xoh9ar+nWjBEhfyhHCqU07W/RbvnWDBIfv+wUtfGb/9kdcfN1XxSzL6s3qZCU/
Dqf0oyh4apL7B8Q9WaLdCJlzwmt7T+XmWEiTJ/
cXXGUR2mphRWP7PQWcK1ULk6Onp+1R3JuPP6+wYd1W6XOtZ4WT/5L0rpW/
WdYoFugzFBcXcaeYhm2yB7VLmgkHi8KyfkLDDgPr4uRDEgiJsrbSmu3wZ4hlQrHY6ZCe6v8h5uS+v2smlo
7vR+Y/dJ6HI3yHv+7xhUWfwYd+KkWNaRbcw+vT1jZzBEWkSkSZBsWL4UgVYJtctvVCnqs8WExxE14t3Ag/
...

```
97Me/M7VfXu7u8/
bV7iliRJkiaaCWZJkiTpvnlRNUsATx86GEmSJGlIJpglSZKke+98YE1gW+CZwMLAJQBd0vntVfWXqrq+
qn5RVRt055auqqOq6uqqOrT7vTtV1duq6oLu946uqjUn8x8lSZIkzSsTzJIkSdK9dxbwW+DF3dd3gWu6
cy8C3g+cBrwd2Bz4XlUtDLwL2Bn4Bi0h/
bCxP1hVLwA+2P3dDwMbA1+fhH+LJEmSdL8tNHQAkiRJ0gLmEODTwKLATsDHuuM7d9/
fkGRWVT0aeDYtmbwtcAfw6iS3VtXzgVW76+/afX9m9wWwUlUtO6H/
CkmSJGkcmGCWJEmS7psjgU8AFwPH/ovz9zS8r3+8/
sXl5wBXdpenAf+4HzFKkiRJk8IWGZIkSdJ9kOQ6WnuMlye5o3fqqO77x6vqNcDuwHnAucAvgOnAgVX1A
WCV3u/9oPv+AmA1YBvgnUlunrh/hSRJkjQ+rGCWJEmS7qMkX/
sXhw+lJY73BrYDfkdriXFbVb0PeDitBcZ3gFnAOt3f+lJVrQS8HDiIVhn9r/
6+JEmSNN+p5J528EmSJEmSJEmSdM9skSFJkiRJkiRJmicmmCVJkiRJkiRJ88QEsyRJkiRJkiRpnphgli
RJkiRJkiTNExPMkiRJkiRJkqR5YoJZkiRJkiRJkqRPTDBLkiRJkiRJkubJ/
wfyGi7jUPwVgAAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<Figure size 1440x504 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAABZgAAAHwCAYAAAArRQrgAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzs3Xd4FOXexvF7dtMAAAAASUVORK5CYII=\n",
      "text/plain": [
       "<Figure size 1440x504 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "# Plot the data\n",
    "plt.figure(figsize=(20,7))\n",
    "plt.plot(data['Close'])\n",
    "plt.xlabel('Date', fontsize=18)\n",
    "plt.ylabel('Close Price USD ($)', fontsize=18)\n",
    "plt.show()"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.3"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```

X4eNlTTFtu3Gkl6QNMZTeQAAMGHekpmq38Yv6+FZJcv4qkLTOH03e5bhZACQ3dr5CxV2U1/5BAVJkiyH
QxF9e2vTHysMJwMAACi8Zk6eqkopJfVwm5sVEhAsSXI4HHqw2Q36efr3htPlD0/
eOtVS0nbbtndKkmVZMyT1kbTxjGPqS3rk9MeLJX3nwTwAAHiEy+U669y2bSUH7JRUPtu8fE1fvfDiKH0
8cbLnw50WFRXltbUA5F3s3mjFvvGOkbX3Hz2iitf1zDazLEtJx49rs6FMAAAA5zPqyacVH3ckX661I2Z
3vlzn307GxKljRCNdFlY5x3uHdsSc8+vJs6lZqVq+ZAoND9OLr76cL9eSPFswV5QUfcbrGEmt/
nXMGkk3KHMbjb6SSlqWFWbbdrbfGZZlDZE0RJKqVKniscAAAOTF+crb/
ndfLSkh2yxmU5pee+VtXdP92rOe43K5KISBYiqySmWF3ny9kbWTfpqrhJh9CqxUMWtmu90KCglR3SF3G
8l0MeJnfmM6AgAA8LL4uCN6ocsQ0zHO67tff9bejdu17uAONSpXM2tu27bKlCqt/
7vjUa9nenbBxHy9nse2yJBknWVm/
+v1cEkdLMv6W1IHSfskpec4ybYn2rbd3Lbt5hEREfmfFAAAD7m5z736c26qMtLdkqRDe5J1bHsVde/
aw3AyAMiuYedOOvbNbKUcjpMkZSQl6+CU6WpSRB4+AwAAYELPVp20N/
WIxv3+lfYcOyhJSkpL0XOLP1av9l0Np8sfnryDOUbSmfd+V5K0/8wDbNveL+l6SbIsK1jSDbZtH/
dgJgAAvKpv75tUuWI1fTr1PaWmn1LDOq304viH5XB48nu8AHDxfAMD1G3w3VodNU/
HTibI12GpbdcuKnPGHc0AAAC4OL4+vnph8HB9Nm+WRiz8QL6WjyLCwjXo2n6qVbG66Xj5wpMF80pJtSz
Lqq7MO5P7Sxp45gGWZYVLirdt2y1ppKRJHswDAIARzZu1VPNmn5uOAQAX5F8iSK1u6Gs6BgAAQJES5B+
oob1vk3rfZjqKR3isYLZtO92yrAckzZPklDTJtu0NlmW9IGmVbduzJXWUNMayLFvSMknDPJXHG5KTk/
V/
L7ykE3tiZEsq37Cuhj3xuJxOp+loAAAAwCXZsWKltq9ZJ9vXRz7pGWrew6XSFcpf+EQAAIAi7GRSoj6a
PVVpp1KUYbtVqWJF3dbtBlnW2XYPLpo8eQezbNueI2nOv2bPnvHxLEmzPJnBm566b5iGlq+lKo3bSJI2
xh7QS8NH6Lm3x17w3L1792rCq2/IcTJRyQ6p560D1bFrF09HBgAAAC5o+
+8rtOP4UYUNHiRJcqemadlHk9X1tlsUGFLKcDoAAIou27Y19eOpWrt4jXxsp4IqB+uhUQ+pZMmSpqNBm
b8+Y6a8p6da3qKywaGSpJX7NumD76ZoWN/
bDafzHjaAzCebNm1S3QwfVSkdljWrH1levvsP68iRI+c9Nz4+Xq8/8LAer1BXzzRsrRfrXan1k2dowZy
5no4NAAAAXND2tesU1u3qrNcOP1+F3Xy91i1cbDAVAABF3wevj1PoryU1uvpTGlVjhG5NuVEj7nhctm2
bjgZJf25bp04VmmSVy5LUomI9pRxL1KmUJIPJvMujdzCbNOrJkYqPi/PaejExMXqs5uU55hWd/rp/
8D0KDwvLNh82+J6sj3dv3qp32/
dUoK+fJMmyLA1u1FL9n39R337zjWeDn0VoeLhefHWM19cFAAAo7iLDwxU7M//+/he7NzpfrpOsjBwz//
Aw7Vu5Spt37M7zdSOrVL7wQbm9Vnh4vl0LAICCIC0tTdHL9+iuBrdkzcoGR6prUGctXbRUHa/
uaC5cAbEjZrdumfyUsfWPx8ZrQruHcsxL+QTq1klPKTAw0EAq7yuyBfPK1X97db2UlBR9v3mNWlWpkW2
+cNcWxQf76XjSqWzzHTH//GU/5cQJVQopk+19y7JUwnJmO85bTKwJAAAAaewrBfOb/
Nfff58OZWTIOuPZIknbd2rE/
cN03x13GkwGAEDh53K5zjpPTU1Va7tFjnnD0vV0+6NDVKp8iKejSZKioqK8sk5e1KxUTS90GWJs/
V0Ho7Vk6X9VO7xKtvmh5GP6csgb8nEWzOr12QUT8/V6BfP/ZT6oWamyXnJ59wnY47+dpa/X/
6Xr6l8ut+3W56tXqHWTpurfpdt5z5s2P0obDu1Xg7IVsmZut1uRoaF6/+57PR07h2eivvX6mgAAACi4n
rv/Ad3zxquy+lwj31KllLxrjyKW/6m7PvrYdDQAAAq9cxW4brdbT/QbnmP+39jf9MGn43X5FTl/
kt7lchXoQrioqV6usr6zEjV323K5LrtSKemp+mT1j2rcoFGBLZc9gT2Y89HQvv1UskYVjf59gV5asVi1
mza+YLksSX3bd9L4v3/
T9rhYSdLx5FMategH9cvFuQAAAICnNWnYUN+NeV1t1m1TtTmLdJv89P3Ej+Tn52c6GgAARZbD4dBVN7T
RhE2fKDUjVbZta/
n+37U1ZOdZy2WY8fCNg5VWLlDPrpis19Z8qRZtrlSvq7qYjuVVxadK95K2jZqobaMmF3VOoL+/
nh98r2YunK/
Pt6yWn7+fBl3fT5UjIj2UEgAAALg4FSpU0FvPPW86BgAAxcr1t9ygv+r8pbGT3pM7NUNNOl+uV24pmFt
qFVeWZenqK9ro6ivamI5iDAVzARHkH6A7e/
QyHQMAAAAAAAAFyBXNr9AVza8wHQM4J7bIAAAAAAAAADkCQUzAAAAAAAAACBPKJgBAAAAAAAAHlCwQ
wAAAAAAAAAyBMKZgAAAAAAAABAnlAwAwAAAAAAAADyhIIZAAAAAAAAAJAnFMwAAAAAAAAgDyhYAYAAA
AAAAAA5AkFMwAAAAAAAAAgT3xMBwAAAAAAAEDRdPjwYb02+v+UfMyWW+lq0Lym7n9oiCzLMh0NQD6hYA
YAAAAAAEC+S09P12P3jdIdrV9QiYCSkqSN0Sv0xsvv6IlnHjacDkB+YYsMAAAAAAAA5Ls5P0apTdUbss
plSapfuaV2rT+g9PR0g8kA5CfuYAYAAAAAAChmXC6Xx9eIP3RSz/
X7PMf81LEMuVwu+ficv5aKioryVDQA+YiCGQAAAAAAoJjJ7/LW5XLluOaa1Wv03fsL5br81mxz/
zJuzZ82n32YgSKCLTIAAAAAAACQ75pc3kRJJaO1fMtcud1uJSQd1xe/
vKYbb+9JuQwUIdzBDAAAAAAAAI8Y89ZozZs7X99EjVGJ4EA99uodqlatmulYAPIRBXMB43a7tWj1n1qz
dYtqVqqsHq2ukp+vr+lYAAAUe8nJyfps5udav32jerTvLleX7tx5AwBFlG3bmr9oiX5c9F/Vvaya7ux/
owIDA03HAgoly7Lk6tFNrh7dTEcBPMbtdmvp2j/099Z1ql6hqnq06iR/Xz/
TsbyGLTIKkNS0ND09cZyC405oRNN2qp3ho2cmvK+jJ0+ajgYAQLF24MABdbm7h8ZnzNIfHaM1cs1Y3Xj
/
AGVkZJiOBgDIZ263WwOHDdfTvxzQ6haDNTmpurre+ZCio6NNRwMAFEBp6Wka9ckb8j+YrBGNblL9jAg9
O/
ENxZ04ajqa11AwFyBfLl6gIY2vVOeadeXn46PmlarphQ49Nemn701HAwCgyPpy9tdy3ddHbe7vqh73Xa
eFyxbnOOaJN55WyqDSCrgsRA5fhwJahmp7o3h9/
tVUA4kBAJ701Xeztb1qe5Vo3l0OX38FVa0v66Zn9eQb75/
zHLfbrdFvvqurBz+m9oOfUP8HHldMTIwXUwMATPnmlyjdXqebutZsKT8fX11RoY5ebj9Yk3+aaTqa11A
wFyAxBw6oQdkK2WahQSWUdirJUCIAAIq2eYt/
1qvL39exgf7K6F9GRwb46omZz2vDpo3ZjtuXckjOwOw7i/k3CNH8FYu8GRcA4AVzf1mhoAZtss2c/
oE6cJ4vy55+7W3N9amvjL5PydF3hPZ1fkgDh49Wamqqh9MCAEzbFb1HTcvXzjYLCQhWRkqaoUTeV2T3Y

A4ND9czUd/m2/V2xHj+x6GS9h1UypWd5e/zz57Ltm1r9b69GvDxu+c9t2alyvmWIzQ8PN+uBQDA+Qx/
6nHFxh02tv7y7SsV+lz9rNeWZcl5Xbj6De2vqgEVNGjIHZKkPbt2KUINs52bkZiuNX+vyTrG2yLDIzT2
lTeMrA0AhZHL5crVcTsPn1BEy7vkW7JM9vnm9We9hm3b2uwOUfWH78iaOf0DdaJZX7Vs20HlQkNynTEq
KirXxwIApNDwMD27YKLRDDsP79Kp1GQF+QVkm6/dv814tnMJDQ/
L1+sV2YL5xVfHmI5w0davW6dxL76hR5u1y5p9tWWtHn7hOV3Tp7fBZAAAeMb6v9YZXT/VSs/
xoD6Hv1NJ6cmSpNi9ByVJJU766diSfSrdsaKkzDLh0MebVd0dmXWMt5laFwC8afhTz+hwXHy+XKtslRq
5Oi44NEF/
TX1Fle99PetzxLEV81QhyKmyVapmHXdo706VrVJD6elp8kvI+QDAgIo1lVKitMpWqZ7rjLcPuT/
Xx15IRHioxr7yUr5dDwAKohdffdl0BG3evFnvj3xfT1xxc9bsux2/KsE/Q+M+/
tBgMu8psgVzYdSwUSMdunOgnpoyVYGpGUr2sdTimu6UywCAIiuySjmVHlj1wgd6yKEpJ5V+MlU+Jf95w
nPy/gRVbFZNtfu2yprVlrR23p+K+XCLbD9LzmSpRedWqtTQXPZj0/YYWxsAvOVwXLzK3/
yE19cN2LRaf00epQz/IFmpyapUpZqaP5H9p0rLn/
HxXx+MkW3b2b5peWLVz2ox4D8qW7Oel1Jnd2Dm60bWBYDipm7dump///
V6+pMvFJhiKdnHrabd2yj4YO5/
gqWwo2AuYK6+xqWrr8ndj24BAIBL07x3ay34eK5K9a2swGqllLj5qJLmHFTr+3rlOLZx92ZqrGYGUgIA
vK1SvctVqd7luT7+ig7dteKLlxTRe6h8g8vo6B9zFRi3W2Vr5vx8AgAoejp27ayOXTtnm30+a7qhNN5H
wQwAAIqtwNIl5Lq3jzYtXqOji6JVtkpZ1RnWR05fp+loAIBCpHKDK1SmXCWtXvi5Ek8lqkHTK1Wl0wOm
YwEA4BUUzAAAoFjzDfBV42uam44BACjkgsMi1famwaZjAADgdQ7TAQAAAAAAAAAAhRN3MAMA4EVfTP9E
cxZOU7qVIF+7tO665VFd3am76VgAAADAJTkRn6j7Bg6X0+2vDN9k9b/rOnXs1M50LABeQMEMAICXfP/
D11qy4X017+cnSbLto5owY6TKl6uk+vUaGE4HAAAA5M3sb39Sx9r91efKuyRJtm1r2oQ3VL5CpOrUqWM
4HQBP8+gWGZZluSzL2mJZ1nbLsp48y/
tVLMtabFnW35ZlrbUsq4cn8wAAYNJ3cz5TvTa+Wa8ty1KzHk5N/PQtg6kAAACASzN/9tKsclnK/
HvuDS0f0GcTphtMBcBbPHYHs2VZTknjJHWVFCNppWVZs23b3njGYc9I+tK27fGWZdWXNEdSNU9lAgDgf
0aMHK64uFivrrkteoMaWeHZZr7+Dv2x8r+6+55B2eb/
fl1QhIdH6rUxY03HAAAAKJaeHjlKR+LiTcfIYf+OY1Kz7LMAvyCt/
P1v3XfPMDOhzhAWHqqXx7xoOgZQZHlyi4yWkrbbtr1TkizLmiGpj6QzC2ZbUqnTH4dI2u/
BPAAAZImLi1X7fqUufGA+OvxFsJIS0xVY4p9Pvwf3nFLdy8upvevMLN7NdTGWzfJuKQ8AAIB/
HImL17DeBa8o/WDKG0pMOqESgf/8PXb7/
nVqe0VXXd99oMFkmcbNHmU6AlCkeXKLjIqSos94HXN6dqbnJd1qWVaMMu9efvBsF7Isa4hlWassy1p1+
PBhT2QFAMDjuvVqrnmfxmv/zlNyu23tWJuglT8lq01n9l8GAABA4dW/z50aH/
W0tu1bK7fbrT+3LdWPf36iXp37mY4GwAs8WTBbZ5nZ/
3o9QNJk27YrSeoh6XPLsnJksm17om3bzW3bbh4REeGBqAAAeF7JkEDddm93ndgdoWXTU+STVEWD7usqX
z+n6WgAAABAnoWGhGvE/
S9ob9JqTfnlJWWUPKoRQ1+Sr6+f6WgAvMCTW2TESKp8xutKyrkFxt2SXJJk2/
Zyy7ICJIVL4udvAQBFkp+fj9pcXf+izrFtW8vmr9O+mANyOG3J7adu17ZQaERJD6UEAAAALo6/
X4B6XX2j6RgADPDkHcwrJdWyLKu6ZVl+kvpLmv2vY/
ZKulqSLMuqJylAEntgAABwhkVzViuw7BF1vzNUXQeFqeOAEvpu5lKlpqabjgYAQLFju92mIwAAUKB47A
5m27bTLct6QNI8SU5Jk2zb3mBZ1guSVtm2PVvSY5I+sizrEWVun3GHbdv/
3kYDAIBi7eDBg7q8R3jWa19/h5p1L6k/
f92m1p3qGUwGAEDxYNu2Vv30pfbti5btGyBnSoKatu+uyg2amo4GAIBxntwiQ7Ztz1Hmw/
vOnD17xscbJbXxZAYAAM5mX3Sspr9d8Hdksm1byf6pOeYRlfw1/dM92r3a10AqAACKl7/
nfaOjkXVUrstgSZmfn1dOfUUhEWVVKrKC4XQAAJjl0YIZAICCqmLlSLXvV8p0jFz5fOJO2bYty/
rn+blbViWo1y2NVbuBd7+oXTbrhFfXAwCgIIjZu1uRHe/Iem1ZliL63K/
VCyepff8h5oIBAFAAUDADAFDAtW7bSD9//rfa9Q1VYLBTW/
8+qUPbnOp4Z3nT0QAA8KhDe3fq0Bv3mY6hUxk5vyntE1xa0X8v0+rovwwkAgCg4KBgBgCggLusfgWFRp
TSb/
PWK+lUkmrVq66b7qie7Y5mAACKorJVaqj8zU+YjqGDH41VRkqSnP6BWbPja5bq8r53qfaVnQ0mO7cDM1
83HQEAUExQMAMAUAiERgTr2huvNB0DAIBi6arrbtPPk55R6W6DFFjxMh3/
a6GsbStV657hpqMBAGCcw3QAAAAAAAXbkSNHNHz4cMXHx2eb27atgwcP6sQJ9mdH0VYqoqyuGzZSZQ9t
UMbc8WoYUVKuIY/LcvAlNQAA3MEMAAAA4LymTZumDRs2aNq0aqNq0aXrggQckSSv+/
FMj339XJ0JD5EhOUU2nnya+MkbBwcGG0wKe4ePnr0ade5mOAQBAgcO3WwEAAACc05EjRzR//
nzZtq2ff/5Z8fHxcOnXqlP7z9ptKHnC9/K/pKt+
+12p7uxYaOuoZ03EBQLZtKzExUW6323QUACgWuIMZAAAAwDlNmzYtq6r6Rxu92aNm2agkLLKLVtSwWdsT2
AX5ky2paUoKSkJAUGBp7
rcgDgUd/
PnqsZs+bL4RuqjNTjat2ylh56cIjpWABQpHEHMwAAAIBzWrx4dX4sdLT0yVJ6enpWrRokY6dOCFHiRI5jnX
7+io1NdXbEQFAkrRu3Xp9+f1qtXW9rKuufktztrnlBuw6W1WdTZpiOBgBFGgUzAAAAgHPq1KmTevfurWdG
DRx3qvihww/
DRx8dHnTt31oA+18lavirbcXZGhsKTUhUSEmIiJgBo0uSv1KLdfdlmtRp00+JlfxlKBADFA1tkAAAAAD
ingQMHav78+ZIkh8OhgQ0VIOat9Lns2YrvfnlshMSFLTyb7357GjDaQEUBi6XyyPXPRSXqsHNns
wx37xlx0WtGRUVlZ+xAKDIo2AGAAAAcE5hYWHq2rWu3ZZo27duik2agkLLKLVtSwWdsT2
AX5ky2paUoKSkJAUGBp7/
r176du4chdesol7DHpWvr6/htAAKg/
wucF0ul6KiojTl8xlav2OFqtZsmfVeYkK82rZurLGvP5evawIA/
kHBDAAAAOC8Bg4cqD179mjgwIHZ5hEREOy6HZDqQAgu1sG3qihw0Yo4fg+Va3VXof2bdDerd/
q4w9fNR0NAIo09mAGAAAAcF5hYWEaO3Zs1OTRj/

```
hm7pW13uY1+rQ7N0zZw6TqVLlzYdrdBKS0vVr38u1n9XLlBKarLpODiL7du365sZX2vTxk2mo6AY4w5m
AAAAAABQJFiWpTZtWqtNm9amoxR6G7au1ndRX6l9/
T5yOnz0fxNfVZcO3dWsEf9uCwK3263nHnxWkYfC1KJ0My37ZqE+LTlJbrfbdDQUQxTMAAqE9PR0HTt2T
KGhoXI4+OEKAAAAoDhzu92Kj49X6dKl5eNDdeFtbrdbs+fN0sO935RlWZKkJjWu0ns/
PqHG9ZrL14c9902b9slUdUlur5Z1WkiSmpRtpE1HNuuXQ78YTobiiD+lARg3/
t1XtXHFjyoTmKwjSUFq2/02DbjtHtOxAAAAABgwdfrX+nHucvkHlVfKqcNq3rSqHntkqOlYxcr2PZvUo
HLrrHJZyrw7vNVl3bRm80o1b3iVwXSQpI2/btANFa/
NNqsXVldlbLaEgfdRMAMwataMKXLGzNT9Xf2V+UdSqr759QMtr1lXra9qZzoeAAAAAC9asWKV5i/
bp7auF7Nm2zcu0JTelj18AAAgAElEQVTPZ2rQbTcbTFYw7I7eocfHDbzwgZcoMTFRrSr0zTE/
kXRc074fp5lLi3eJOWrEKMXHHTGaYe/
6PbIr2Nm+CSBJ6XaGht19v6FUFxYaHqYXX3vxwgeiUKFgBmDUrwtnaUhb/
2yz3i39NGX6hxTMAAAAQDHz+dTvdEXrJ7LNLqvfRUsWP0/
BLKla5Zoa1ts75dzL7z2l1LQU+flmfr2WnpGuddG/aMLIHwvdtobjZo/
K1+vFxx3R6A75e82Ltaj0En2/7QddV7v3P7M9S3RT2xvUp3Xv85xp1nNLKZeLIgpmABfN5XLl27XsE5u
ktuWzzXycllb8/
uslrRMVFXWp0QAAAABIGvnUKMXFxXtlrQ2bolW3Vc79fbft2Kt7hgw753nne8+k8PBQjXmlcBZqd908T
OO/
ekqVy9SWw+HU7riNuq3fPYWuXC6qOjfpqM8Ofa7nl7+suiG1te3EDpUMD9bgK+82HQ3FEAUzgIuWn+Xt
mNGP68CRBSof9s9dzBv2JOuOex/XnYML5l8SAQAAgOIkLi5effq/5JW1Si34Wru2/
aHqtVplzY4fPaCq1WqrT/+HvZIhP30/4xnTEfKsfGRFjRz2kg7G7ZPb7VaFyFtMR8K/
3N7tNiWlJmnfkX3qHHq1SvgHmY6EYoqCGYBRDz0+Wp2vnK7bu4WrbkVLq3dLe1LqauxT95qOBgAAAMDL
2ne6Tp9+9Iri43apRu222h+9Tjs2ztPgoYW3qC3syoVXNB0B5xHoF6jLyl9mOgaKOQpmAEYFBQUppHwD
1b/
2cW1c+6fa3dpRj7RodeETgUIo8WSyflu8QcePJ6peg6qq37RKjodywLz01HRtWbpOcTGHFVm1rGq3ayi
nr9N0LAAAigWn06nB943S9q1rtXXzHFWqUlPXdB/D35kAFBrp6emaPetbnYyJ06RxE9X/
zlsVFFS07y5n4xwAxlmWpQ4du2jof0aoOeUyiqh9e47oq6kLVP3KJHUYGKCjads1c9IS2bZtOhrOkJKQ
rKjx3+twjWQF3l1ZBylsKmrc90pNSjUdDQCAYuWy2o3Vo/cgNb68DeUygEIjJSVFD90yRGUWx2nBTW/
pii2BeuzmITp06JDpaB5FwQwAgBcs/vkv9RwcqTIR/nI4LNVrWUqVGrq1cXW06Wg4w58/
LFeZO2ooqFZpSVKJumVU6taqWv3TCsPJAAAAABR0UyZ8qsFlO6l1pYayLEt1I6rphSa3adxLb5mO5lEU
zAAAeIHTL10OZ/a7b+o0L6nN6/cYSoSzSUw6Jd8yAdlm/
pFBOnHihKFEAAAAAAqLPeu3ql5EtWyzkv5Byog/
ZSaQl7AHMwCgWAoPj9SyWbFeW+9EnDvH7NjhVB3b76tls7KXl/
uiY1WxcqS3ol2U8PD8zRUZHqHYaQWnZM+ITpbttmU5/
vlmgDvdrYzoZB0rQDmlzH93AAAAAPKHy+W65Gsk7ovXI2VdKuEXmDWzbVur1v2VL9ePioq65Gt4AgUzA
KBYem3MWK+u9+4Hr2v7+umq1tBPkpSRYWvdz36a/
c3PKl26dLZjXS6XPvloilfzmTL2lTdMR8hm6a9L9cjs5+XX85/yNv2HOH3x5iS1uKKFwWQAAAAAPCk/
ytutW7fqnSfe1sgrBmTtHz9j22I9+vLT6t6rxyVfv6CiYAYAwAseHPq43v/
Q0vKv5ypDSQryKatXn3k5R7kMszq06aBnjv9HE6Z+qk37t6p+xTp64IYnKJcBAAAAXFDt2rV1zWO3atS
HnysgxVKyj1tt+nQp0uWyRMEMAIBXWJalB4c+rgf1uOkouIDre/
TV9T36yuVyKerD703HAQAAAFCItG7XRq3btTEdw6t4yB8AAAAAAAAAIE/OewezZVnvnudt27bth/
I5T5Fm27Y++
+gjzf5sqmrWqKEGba7UoCH3yOGg5wf+Z8HPczTnm0ly2GmqVPMK3fefEQoICDAdCwAAAAAAGdxoS0yH
jjPe7YkCuaL8H8vvqwae+M0vfetkqQVG/fohUcf1/P/96bhZEDB8MXk8dq/
aqLubuUnh8PS7oPf6qEhq/
ThZ7OzNscHAAAAABAwXGhW2c7neefzp6NVrQcP35cR9dskqtWg6xZy4pVVTr2mPbu3WswWfGRkpKirV
u3KiEhwXQUnIXb7dbyBdPVq4W/
HI7MMrlauQA1LRutpUsWGk4HAAAAAACAs7nQHczrvJKiGNi5c6fqlgjJMW9aOkIb1q1TlSpVDKQqPj79
4B1tWfKD6gS7tTtRCq7TQsNHv8pdsQVIYmKiQvxO6d9/
LDWs4tSfq35Vx05dzAQDAAAAAADAOV2oYI5T5lYYZ2Pn4nycVqNGDX20Z7tuaNA02/
zvY4d1faNGhlIVD/9dskjuld9p9JURWbPl0Ws0efx7uvP+/
xhM5l1PPzlcR+JiTcc4p+EP36+DO45IHcpmm/
+59ZQW7/5Fm7YMMpRMCguP1MuvjjW2PgAAAAAAQEF1oYJ4mc5dMOMihISEKKFMsH7avE4962YWyn9E79
LRiFLcvexhc2d8pmcahGebta5cSlF/LJGKUcF8JC5W9/YMNR3jHDJzzVtWU7N/i1Gv1iGyLEt7YpP1x/
YMPXtPfaN3m0/4qeAW8wAAAAAAACadt2C2bbvjpVzcsiyXpHckOSV9bNv2q/
96/21l7ucsSUGSIm3bLn0paxZkb308Uddfe60m/
PWrnE6nrr11gJ57mOckXgyXy3XR56Qd2CFHrctzzDdvWJ+n651PVFRUvl6vuOnevqFWrg3WW99tlcNhK
7RMqP5zR1u2MgEAAAAAACigcrXFhZXZ7vSX1EhSwOmxbdv2Y+c5xylpnKSukmIkrbQsa7Zt2xv1zwUeO
eP4ByU1zXGhIiQ8PFw33nKL5syZox4e+r+Bx4wHanQyUuB+82MaVry66fqVL1M1mxXfKLa9b5J839bR
SlcwLRoXE0tGlczHQMAAAAAAAC5kNs9lMdJuk+Z22X871ZCW9I5C2ZJLSVtt217pyRZljVDUh9JG89x/
ABJz+UyT6E1cOBA7dmzRwMHDjQdpdjoe/MAvbjyN235a4OuKuerdfHpWm9HasyHozS/Tx/
T8QAAAAAAAIBCK7cFc19J05RZAj8k6TpJ/73AORUlRZ/xOkZSq7MdaFlWVUnVJS06x/
tDJA2RVOj3Kw4LC9PYsTwszJssy9Kzb76vXbt26a/
ff1PzBg11V+MmpmMBAAAAAPKJ2+3WD99O0oGDB2VZPnJYqepz/
V2KLFvJdLQiac6Sb7Rx83r5OQOV6j6la7veoLo1GpqOBcCQ3BbMZZRZKA+UFC9plqThkp4/
```

zzln2zT1XA8M7C9plm3bGWd707btiZImSlLz5s156CDypHr16qpevbrpGAAAAACAfPbtrImKrNROl7dp
JklKS0vR1CnD9cBDL8nXz99wuqIlatn3ciSU0v3XjJGUWe6Pn/
OMwgaEKyK0nOF0AExw5PK4g8osow8oc7uMNyWVuMA5MZIqn/
G6kqT95zi2v6TpucwCAMijxMRErVmzRkePHs2aZWRkaP369YqJiTGYDAAAAMgbt9ut2Ng4VanRLGvm6+
uvZm3u0PJf5xpMVjRt2LRW7Rv1ynrtcDh0S8dH9ePCWQZTATApt3cwPyMpTtKjkt6RlCTpkfOeIa2UVM
uyrOqS9imzRM6x8bBlWXWUeYf08lxmAQDkwXvvvqK1q2erQuQpHY4PUFhkS7Vp203Tp76uyhVOKiHBqa
TUqnp97GSFhISYjgsAAIACInrvDr3/+gDTMc7J7XbL7Vsrxzwsopqmf/
iE1q6caSCVZ4SFh2rc7FFGMxxPSMgxCykRpt/WLtQp/0MGEl1YWHio6QhAkZargtm27S/
OeJmrP5lt2063LOsBSfMkOSVNsm17g2VZL0haZdv27NOHDpA0w7Zttr4AAA9ZtOhnHYyeqet7BkgKlCS
t3bBUEz+Yp3tuD5UUIEk6cXKPnn3mfr3z3lRzYQEAAFCgVK5SU336v2Q6xnl98M6zsm1blvXPbp0bV8/
RXUNfU43LzO0N/P2MZ/L1ei+PeTFfr5cXDw0ZodS0FPn5/
rP1yIa9K2T7pOnDj8YZTAbAlFxtkWFZ1hLLst464/
XblmUtvtB5tm3PsW27tm3bNW3bfvn07NkzymXZtv28bdtP5iU8ACB3Zn8/
WVe1zL733MGDibq2e1C2WamSvkpM2Ka0tDRvxgMAAAAuSacuvTVn1iidOH5IGRnpWr3iGyUn7DZaLhdV
D4+8Tx8tfUr74nbJtm39tWOpVhz6RiGhwaajATAkt1tktJT02Rmv10q6N//
jAEDRNnLkcMXFxebb9aKjc3et+COb1K1d2Wyz9AxbTmfO57HGHjqonj17yuHI7Tb951a5cuQlX+N/
wsMjNWbM2Hy7HgAAAIqOeg2aq1z5Klq8YLpOnjyu5i06qH6v4aZjFUnVq1fX+1Ne1ZRPpuqPHdN1VYeW
GtL7bfXs2dN0NACG5LZgjpV0vWVZMyVZkvqdngEALkJcXKz69S6Vj1fM3bV+X2Hpz9W71bzpP8dXrRKo
n+Yd0Z23VsiaJadkyN8/
SMOG1s7HjPlj1mw+7QAAAODcyoRG6vqb7jMdo1goVaqUHnhkqOkYAAqI3BbM0yWNkHRCkq3MrTVe9VQo
AED+atWihr78OlZ7Y46ofh1/7d6TqgOHfVWzZi19Nn23mjUJ0LETGVq3IV39b25vOi4AAEChkpqUqJNx
BxVStpJ8/
PwvfAIAAEVIbgvmZyUlSbr29OsfRMEMAIWGZVm6uV9rHTmSqB27YtWkaRn1qFBaktSq5WXasHG/
ypYLVLt2kdkejAIAAIBzs21bv341SXGnUuRTtrrS5v2oyhUqqHmPG01HAwDAa3JVMNu2nWZZ1quSZkva
bdv2Mc/GAgB4QlhYCYWFVc82Cwz0U/Nm1cwEAgAAKMTWLPhep2q0VLmGV2XNDi35SrtX/
65ql19pMBkAAN6Tqyc4WZbVVNJ2SaskNbcsa6NlWR95NBkAAAAAAAXYvj07VeqMclmSQtvfoM1//
mYoEQAA3pfbLTLel3RKmQ/
4c0uaKmmwp0IBAAAAABARHqoDM183HeOsDu3dqZSM4JxvWJaS9+80njsiPNTo+gCA4iO3BXMTSS9Jevn
06/2SIj2SCAAAAAAASWNfecl0hHNyuVzq37WNft6/
UwEVamTNT238TSPvHaTbB9xkMB0AAN6T24I5RlKH0x83ljRA0m5PBAIAAAAAoDB45uFh2vLQSO3aXlPu
Sg3k3LlKV/id0KDHnzcdDQAAr8ltwfyapE9Of/ymMrfKuMMTgQAAAAAAKAz8/Pw0c/
ybWrd+vdZu3Kwre1ynmjVrmo4FAIBXXbBgtizLKekVSfdKqqXMcvlH27aXejgbAAAAAAAFXqOGDdWoYU
PTMQAAMOKCBbNt2xmWZa2XVN227Se8kAkwxu12a+mihdq3Z5c6dLtGlStXNh0J8Kjk5DSt+muXJKn5Fd
UVEOBrOBHOxe12a+Gi+dobvUvduvTkzycviouL03dRsxVWOlTXdu8pX1/+OwEAAACA/
8ntFhlBkkZYltVV0oHTM9u27T6eiQV43+HDhzXq3lvVs2y6Gpfy1cyF0+Ws11aPjHrRdDTAI9as26sVf
6xV+zaZTz///
IutatmqsZo0qmI4Gf4tNjZW9z1yk8o1PKqS4bYWvj5R9Su69MyTr5iOVuR98NmHmvT7DKW28pd2Z+iNO
9/RR8+MU4O69U1HAwAAAIACIbcFc+vT/9vsjJmdz1kAo956drheblFCwf6Z/
1nUKSt9vvYXrVrxh5q3bGU4HZC/MjLc+v33dbrn9rJZs1o1S2jiZ+vUoF5F+fg4DabDvz378sNq0S9B/
oEBkqSylaU1i+fozz/7qlmzFobTFV0pKSn6ZMV0+Q4om/
UXpoz6bj069knN+3i20WwAAAAAUFA4cnlc9bP8U8NToQAj4vdnlcv/
c3P9MP0083NDgQDP2bY9Vo0b+OeYN2ngr23bDxtIhPM5mrhb/oHZS/8GbXw1/
etPDSUqHg4nHFH6VUHZZpaPQ0dKntTRo0cNpQIAAACAgiVXdzDbtr3H00FQNI168gnFxxXssmrY4DslS
cdj9kutymR7LyE1Xb//vjzrGG8KDY/Qi6++7vV14VnR0bF6e1ys6Rg6efKkmjdOzzE/
eixd85cdUMmSCQZSFV4ul8uj1zyaHq1WqpDt/
ZTkDP08b75cf+Vu7aioqHzNV1Dl56+F03IqIyE1xzw2+pBuvPFG+fjk9gfB/lFcfh0AAAAAFB8X/
5URcBHi4w7ruW51Tcc4j3+yjU+M0ZbYk6oTWTJrNm75Hr1yi0sVw0t7Pdnonzd7fU14XuXKkerXu5TpG
LJtW+MnRik11S0/
v8wfZklNdWvXXlujRlwuy7IMJ8xp1uwTpiOck6dLw1EvPKa4fYsUXtEva7ZmvqW5s5eoYsWKHl27sMnP
X4ukpCR1vPca2XVsWY7M/
ybSjqWobaUWmjbrs3xbBwAAAAAKMwpm4LR7enfSe7PmyWd9rMoG+2rLkWS1b9HUSLkMeJplWbrxhrb6b
PpvCg/L3FI/7kjmrCCWy8XdsyNf1ZPPPqA/Vvwt/5JpSj5SSnfe9CjlsocFBgbq/
UfG6sn3ntXR8GQ5U6RadgVNePl909EAAAAAoMCgYAZO83E69cjNPZSYnKrjiUnqV6akHI7cblMOFD4R4
SV17z3ddfxEkmxbKh0SaDoSzsHX11dvjpmghIQEHTt2TBUqVODPJy9p0bS5Fnzykw4ePKigoCCFhISYj
gQAAAAABQoFM/
AvJQL8VCLA78IHAkVESCmK5cIiODhYwcHBpmMUO5ZlqXz58qZjAAAAAECBRMEMAAAAAAXcoqXLNPWHnx
To76cHB92qOnXqmI4EAAAgiYIZAAAAAAq0J156RQtTHQq6+jq5U1N1y7sT9cjVbXXL9X1NRwMAABAbOA
IAAABAAbV3714tiTupEm2vluX0kTMwSEF9+uujn6KUkZFhOh4AAAAFM4qm1PR0LVu7Tb9t2KF0/
uINAEChk5GRoSXLlmnuvCglJyebjgMY8/PSZcpocHmOeWJEBUVHRxtIBAAobA4ePaif/
16g7fu3m46CIootMlDkrNy8U3OWLlev2qFKy7A1etlyDejRWfWrVjAdDQAA5MLGzZt170tP60TjinL7+
Sh4xica2f8u9e5+jeloQJ64XK48n3v0+Akld+6pElVrZJvHbd6gwYMHy8fn0r+ki4qKuuRrAAAKHtu2N
e778fI/
5aeryl2pv3b8rWmnZmjEgMfl7+tvOh6KEApmFClp6Rmas3S53uxRR5ZlSZK61I7Uo3OX6PkhA7JmAACg
4HrotdFKv6OTSjidmYMmtfTKlEnq0q6DgoKCzIYD8uBSClzbtuW6Y7BOnDgu31IhkqTEnVvVs2Fdvf/

```
SC/kVEQBQBC1as0SN/OqrR73Mb3S2rNhcO4/
u0qdRk3Vfr3sNp0NRwhYZKFJWbtmtHpeVyVYkOxyWWlYooa37Yg0mAwAAuREdHa34yCBZ/
yuXTzvVrLrmzJ9nKBVgjmVZmvnOm6r/
+3w5Z02W36zJSvj4Hb0z+jnT0QAABdzKjSt1TY3u2WY1ylTX8aMnDCVCUcUdzPCoHTEHNGjSAa+td+zo
UT1cP+dv6/jEVD317UqVKFHCa1kAACiuLmU7gJSUFB2tXlLl/
jXPSEnRmFfG6OPxEy4t3GlsCYDCpHTp0vpk7GtZr10ul5z/
+iYMAMB7duzbqVun3Wk6xgUlRSfK3dQtp5X9c8aWw1sLRX4UHhTM8KialcrruW51vbZehtut5yZMV+8G
tpyOzLuYU9IztO1Ehr5+8Fqv5cgPo3/ebDoCAAB5cqnlbc97blNccqqcAX6SJNvtVsjqGC357y/y8/
PLj4gAAAB5VrNiDY3uMMp0jAv6Y8sKzdj4lW6p3z9r9vehNWrfqJ3u6DbISKbnlr5oZF14FgUzihSnw6
FBva7Woz8tVrNygUp321pzOEVD+na/8MkAJEnrN8ZoxYotcjrd8vUL0rXXNFOpUgGmYwEoRj4a/
ZruHvW4YiMCZPv5KHbhSn3z7kTKZQBGnDp1SqPfGqeN+4/Ix87QQFcH3dincN28Anhaamqq4g+e0H/
uGinJrdadmqn/rTfyHCTDWtVpqWkxM/
T88pfVJLShtp3YqdTAND3U9wHT0VDEUDCjyKldqZxGD+mvHQfi5ONw6IZyYaYjAYXGqj936sD+bRrUv4
wcDksnT6brsy8Wasjd3eXvz6cMAN5RoUIFzf1kqnbv3q2kpCQ9/
MfDatmsuelYAIoh27bVb+hjiu80VAFNK8q2bb2x4kftP/yZ6WhAgfLo0JF6rNc4VSub+RPMq/
5cqLf2va/HnnzQcDIMvLq/TqWc0q5Du9UitKXKBJcxHQlFEA/
5Q5FkWZYuqxChapTLwEVZvWa7enYPleP0FjMlS/
ro2u7BWvrLJsPJABRH1apVU7169bj7CYAxUQsWKa52ZwVEVJSU+XVGUKte+va3NbJt23A6oGBYs3qNyj
kbZJXLktT8squ1c81BJSUlGUyG/wnyD1KDKvUpl+Ex3I4GwCN2x8Rr+55Y1awSoeqVKfr/
Jzw8UrNmx5qOcU4JCck5ZlUrB2nqrFidSDT7pOHw8Eij6wMAgMItLw8g3X34uEIf/
TTHPDohQ9UzMi7poaZnwwNI4U359fv36OHjesQ1Lsc8IKOMunfvrqCgoDxfm/
8mgMKBghlAvsrIcGv81MWqUiZFl9cI0Nq1u/TTIl/
dd2tn+Tj5oYkxY8aajnBerVtVlm3b2e4W3L4zWUOGPKrbBt1jMBkAAMClyUtRtXjZL3pi0Ur5teiRbV4
12K
H5M+fzExYo1PKrvN20aZOmv7FE1cvXyzY/5Tis+fPny9/fP1/WAVBw0fYAyFc/
LlqrHk0t3dQhVLUrB6lfuzK6trlDPyxYYzoacsHhU16zo9KVluaWJB06nKI/
Vkfo5v63G04GAADgfR3btVGVfat0ak/
mdmF2RoYSF32u27q2oVwGTqtXr55OBUVrw94VkiS3261F677S5W1rUy4DxQR3MAPIV/
sPxOrmVqWyzepWDtLcPw8bSoSLUbJkqIY99JomT3pLaWknVblqI02Y+IT8/
PxMRwMAAPA6y7I044Oxevfjyfo96kf5OaR7b+qj9m1am44GFCivv/
uSPv90mmb8MU+WQ+pxQxd16dbZdCwAXuLRgtmyLJekdyQ5JX1s2/
arZznmJknPS7IlrbFte6AnMwHF1e6YWI2c4Pm9f0/
FJcq2S2a7o8O2bW2JTtTICZs9vj4uXaNGTfTm2zwZHQAAQJJ8fX312FC2CgPOx+Fw6Pa7b9Xtd5tOAsA
EjxXMlmU5JY2T1FVSjKSVlmXNtm174xnH1JI0UlIb27aPWpbFE5wAD6lWKVL39gz1+DrLVji1aPVOXd0
0JGu2dM0JXXd1A3W8srbH1/eECT/
Fm44AAAAAAABQIHnyDuaWkrbbtr1TkizLmiGpj6SNZxxzj6Rxtm0flSTbtj1/
eyUAj2rfspZmzT2ud747qGqRTu05nKEyYWV1Y4/CWS4XBykpKXph9CM6sH+NrIy9uv+
+fhr13HsqX7686WjA/
7N353F2z9cfx19nsm+yTfaIJUiEWmNfa4lRO1X70tKgtipauqi9llCl+NGgttqrBAlFELvYxRYkIhFZJ
ETINpPz++N8J76ZUCRz53Pn3vfz8ZiHud+5mRz3m+92PudzPiIiZWXChAn85rwL+Xh+DU1x1uvVnQv/
+HuaNWuWOjSRRmH8B28y4oHbsIrWLFw4n149e7LLHr9Qv2wpCwuqF3DlsKv46rMvqaAJ4z4bT/
uW7enSqjOz/Ut232o31l55rdRhSokqZIK5F/
BR7vVEYKM671kNwMyeItponO7uSyxjamaDgcEAffr0KUiwIlJ/
frrjQObOq2bqjNls1aktLVuo3Xsx+8OpR7Lmqq+w+frNgZ7MnTeek36zLzfePJKKCq0FKyIi0hCqq6s5
8NQ/snC/wTRtHotiPfPxBI7785lcee5ZiaMTKX6zv/iMYffcws77nEdFRRMAxr/
3HMP+cx277vGLxNGJFN5ld1/Ofr33ZpUBfQGYWz2XUx7+E7/
f+rc0sSb8aeQZdOvYle4duyeOVEpRITMH3zRE6HVeNwVWBbYG9gOGmlmHJf6Q+9XuPtDdB3bp0qXeAxW
R+teyRVP69Oig5HKRmzVrFrNnvUG3rl8v4teyRRMGrDaTkSMfThiZiIhIebl3+HC+XH9zKrLkMkCLnn1
49dPPmDNnTsLIRBqHRx+
+i823P25RchlgxVU2YtKkSQmjEmkYc+bPga9glU59F21r2bQle62+O4+MG4mZ8ev1juXfo/
6TMEopZYXM/EwEls+97g18/
A3vedbdFwDjzOwdIuH8QgHjEhEpGVVVVcv05+fMmcNqK34KLD5416mDc8opJ9G5c89l+v0jRiwxKUVER
KTonfT73zN1+qcN+ne+O248FQcetcT26fOrOWTwEbRs1Wqx7QcPPqKhQvtBulZ2Zsi556YOQ+pZZWUn7
rn1j6nD+FYfTXifz75w1txwyeNizldfJI29srLw6+BIw+tU2Zk/P15/s0ven/
TBMv35+fPns4lvsMT23sv1YtSEpwDo3KoT/33rEZ6e/dwP/v19e628TPHldarsXG+/
S4pHIRPMLwCrmtlKwCRgX2D/Ou/5D1G5/E8zqyRaZizbUSUiUkaWNYHr7vzikK2BuYttf/2t5gwbdj/
du2v6lIiIlJ+p0z+lcp9DG/
TvbD5lMk+OeoJWe+y3aJu7Y198Tq8jj1msh2xlg0b2w0y97Z+pQ5AC+EuRt2mpqqrivHP+yO3338ca6+
6+aHtNTTUrLt+ef1w9JGF0UorOOr+4jgl354S9jovrRu568cDYEezSbycAHpv0BKecc4PMKvsAACAASU
RBVCo77b5TqjClhBWsRYa7VwPHAA8CbwG3u/sYMzvTzHbN3vYg8KmZvQmMBE5294YtFRARKWNmxv4H/
pY77q3m0xnz+WpODQ+NnEe/
AXsouSwiItKAluvWg8omMO2Bu6mZO4d506Yw+brLWXuLrbVAmcj3sNVWm9OsegxvvPhvFiyYx5TJYxk5
7FR+e+IvU4cmUnBmxt7H7cufXz2HibMm8eX8LnshSuZMXcG3dt2477xw3m0ySh23HXH1KFKiSpoc1R3
fwB4oM6203LfO/Cb7EtERBLYftBOrLX2QG644QpmfzSLX/7cQcAYMWCN1WCIiImVnwz32ZtoH7/
```

POPbfQonUbtv/ZvrRqv8QSNSLyDcyMSy4+i6effpZ777uU5Xv34IZr/
kK7du1ShybSIDbbejP6rdmPW4fewqyZs9j+7O0Z89KbXDL2/9jq51tx4bb7acBSCkarb4mICN26dePkk
89IHYaIiEjZ67JyX7qs3Pe73ygiSzAzNttsEzbbbJPUoYgkUVlZyTGnHLvo9QYbbpgwGiknBWuRISIiI
iKyrKZOncqsWbOYM2dO6lBERCQhd2fMmDG8//77qUMREZE6VMEsIiIiIkWnurqawaeeyBsLPuPzbVdn2
1//gn023JrjDzsidWgiItLAXnjhJc6/+Fo6dR9IzYK5fPX5JVz4l9/
Ru3fv1KGJiAhKMIuIiIhIEfrThX/htbU607x3fzpn264f8Rybvrg+G6w/
MGlsIiLScObPn8+5F17DtrtduKh/7IL5c/ndH07n5uv/
ljg6EREBtcgQERERkSI0esJYmvfusti25ZuuuY1W335woIhERSeGRR0ayYv9dF1ucrFnzljRrRvRITJ05M
GJmIiNRSBbOIiIiIcNLvT2Hq9Ompw1jkg4nj6cbiizRZhfHUc89y8ODDE0W1pK6VlQw597zUYYiINCpV
VVXf+72ffjqTzXc8e4ntX3zxFQcccACtWrX6wb/
zu4wYMaLefpISDlQgllEREREmDp9Oh333SZ1GIt0/NfnzJs6kxZdOy7aNvkOy7aNvOxl1lzr0F0XKNfwsgWN/
XWR1OHICLS6PyQBO6cOXM48Oe/Z5X+my/
aVlNTTTM+4fHHHy9EeCIi8gMpwSwiIiIiRWejvXbi0etuZfYKXWnWpyvzXvuAypZt6LVN8SSXRUSk8Fq
1asURh+3KVdecSu9VBlG94Cs+GfcI55Sz569ShiYhIRglmERERESk6TVs0Z9CRPzo4/57JNpdP/
JDrRq3y51WCIiksCg7X/
MVltuypNPPkWrVh3YeOPLqajQklIiIsVCCWYRERERKVodl+9Jx+V7ENE5Gt
KMItIctNmfMnTL46lVasWbLHBKrRq0Sx1SCIiIiIiIiIi8j1oTomIJHX/yNcY/uCjbLrSZ6zS/
mOuvP4B3h03JXVYIiIiIiIiIiLyPSjBLCLJzPjsK6ZO/
pDBO3Vh+a4tGbBCG075BERERERERERERROR7UIJZRJJ59pUPGLRe28W2VVQYvTtH8llERERER
ERERIqbejCLlInOlV256v6pqcNYzKSJ1Szfpoa+dbZPmVnDzSO/
oHnzuUniqqtzZdfUIYiIlIWZEyfz3guv0Hq5dqy22QY0a9kidUgiIiIiIvIdlGAWKRPnnDckdQhLmD9/
PkfsvzXrrlJD0yYGwLTP5tNphU0YcvnNiaMTEZGG9Pzdw/
m0opr2P16XaTNm8f7Qm9l050FUrtg7dWgiIiIiIvI/
KMEsIsk0b96cU84yeyuH7D2LzdXowr7qCph3W5IwLrkwdmoiINKCZEyfzaUU1lbtvCUCzju1o/
as9GH31MKqOPDhxdCIiIiIi8r8owSwiSfXrP4A2XdfklLL/eRvPmzWnVqlXqkEREpIG998IrP/
xuottsyYV1LRtQfX8+TRt3jxRZCIiIiIi8l2UYBaRotC+ffvUUIYiIlLWpEyYy9YIbkvzdX8yYRsVafWj
Wsd1i2+d/MoP3/noLZpYkLhERERER+W5KMEtJmjt/AXc/
MZpJU6bRpnVr9vrxnTt0O67/6CIiEiZ6tqnNx333SbJ373y3HkMH3ozrX/VE2tSAcCciVPp0qM7/Q/
YI0lM39fMWx9NHYKIiIiSFJKMEvJmTt/
AWdeewdHD+zBGBgt69VXHbsmW1eh5kvVRRxMiOdukKp+6GDVRvKwBjL56GDVtW8D8atq3aM0m+
+ySOjQRERERREfkOSjBLybn7idEcPbK7idGRXO6+HABd27XkvVKm+ptqkQ6zVVh6pIpEQNTK2TcmTlcR4TuI
KO6KOV0v0SVrySPkZxMCxOupJnp1BKDPNnXTKqcvuuIgfuINgHahjrXFVQlucfVIQJqIhIEwRJKLpnb/
Efm+mjZvruSyiIiIiEgjogpmKahOlV0446G3G/
TvHD99HtNmz6NL2xaLXtrk7b0/7aolY3p84mb69ezRofN9Xp0q18xARERERERERkeKmBLMU1FnnXdgf+
eUKVM4b/
DenLV5V5o1qcDdufa1Tzn6tPPZfqfFezlWVVVx+dDrGjxGERERGRUqAEs5Scbt26cdQFV3PWkL
NpNmcm8ypaMGifE5ZILouIiIiIiIiMiyUYJZStJq/fpz/
j9uSh2GiIiIiIiIiIhISdMKKiIiIiIiIiIiIKyVFTBLGXjs88+47JzTmPulA+RERERER
ERERQaNSWYpSxUV1dzymH78rYDOwUA9z1xPbUkcmH78qe1W1DZowUA9z1xPbOnTUkcmYiIiIiIiIiSOOlFhlSFh6452727r2Qy
rYtFm3bedWOdKuYg7snjExEREREREERKTxUgWzNCpVNVNCpVVVVL9edmTf6Q//ys/
xLbe7Rtzg477EBFRf2MtYwYMaJefo+IiIiIiIiEhjoASzNCpLm8B9YuSjLr1HHbu13mx7x4+1DPSZjzuRr/
z5QH6GJiIiIiDS42dOn8an8zZjjzB/7hz6b7w5XVdZNXVIIiIiUmbUIPkPFP/
5hR8yp5LL0xlu30PTxTxYvYWRYRYew5cC633TSUw486ZX1P/
4JLfY8iJfeeZfn774Sn774Sn774Sn774Sn774Sn774Sn/
+3Ome+0YMjEzlSdcgk77r5n6n6tBERERERH4wd2fMc8/
Q4+AjaN6hE01atKTLyqtjKSyajemLVjI7OnTUocnIiIiZUQtMqRsNG3cmLaCHV1dU0b948dSgiIiIi32qhEi3M
S29ttsBnjX32ZNbcdlCAqERERKUdKMItIg1m4C5H1dU0b948dSgiIiIi32rqhA+ZeuEZEZqcP4n6qrq5n
btvMS2+d9PIGZjw7nzZeeSRCViIiIlKOCPJjpjNrAr4G9EOOru59X5+aHAhcKbNPf3X1oIWMSkYYYZXU1P
DeWeezOT3nqV5xQLmVVRy5G/O5s6NBEREZEldO2zEpjNdNpAq2ySuaauXOYM/
oZtjvrIqwIqMkYiIiIukULMFsn+TZnY/
M37JWq8fBkrOP9M5ZzKwX1apj2Bfhw5yEUBDzfvPjTEmRTeGpNOnT3RMz4+RCA/
+ki9nzkgdmoiIiIiJS5QiaYXwBWNbwC5sC9wb/4NZtYj93/4NZtYj93JX4K0CxiMiifRffQ0uu+
glGfDWK3Y27k1NOHpA5LRERepNF45cH7ab/HfrRdbQAArXuvQLefH80Lw/
6TODIREREpdwVrkeHu1WZ2DPAg0AS4xt3fNmsqG3v3vxc4zsx2AS4QS41t3HmNmZwGh3vxc4zsx2BaqBGcChhYpHRNJq3bo1h/
ziqNRhiIiIiDRKMz/
9lI69VlhsW9M2bZlbszBRRRCIiIiKhkD2YcfcHgAfqbDtrsd1ft9/2pwKmFjEFEREREESs/saTOY9ck0Oq/

UmxZt26QOR0pM18rOTL3tn6nDWEzNR+OpmTuXJi1bLtrm7iyY+CHTiyjWrpWdU4cgIiIiDaygCWYRERE
Rkfq0sKaGx66/g3kdWtGsT1fm/
+cVurbrwMDddkgdmpSQIeeemzqEJbz9zjscdNFltNzrYMxi4eS5jz7AJX84hZ223y5xdCIiIlLOlGAWE
RERkUZj9H8epNn269GuT7fYsMEAPn1kNB+99hbLr7V62uBECqh/v34MOWR/htzwT16fMInVe/
fk6J2qlFwWERGR5Aq5yJ+IiIiISL2a8dlMWtYmlzMdt16PsS++migikYaz1WabMuyqK+jv8xj+jyvZZ/
fdUockIiIiogpmEREREYGulZVMvfXR1GF8o6kTJtK1T28Aaj6d9Y3vWfDRNGYmiL9rZWWD/
50iIiIiIsVECWYRERERYci556UO4VtVVVVxw9VDAfjD+efw4MRpNO/
dZdHP5z71Bpeddjbbb7NtqhBFRERERMqWWmSIiIiISKNx+m9+y4CXpzL/
vueY9eLb1Nz5FLu2W1HJZRERERGRRFTBLCIiIiKNRrNmzbjh4r8zceJExo0bx48O/
xEdOnRIHZaIiIiISNlSgllEREREGp3evXvTu3fv1GGIiIiIJQ9tcgQERERERERERERkaWiBLOIiIiIi
IiIiIiILBUlmEVERERERERERERkqSjBLCIiIiIiIiIiIiJLRQlmEREREREREREREVkqSjCLiIiIiIi
IiIyFJRgllEREREREREREElooSzCIiIiIiIiIiKyVJRgFhEREREREREREZGl0jR1ACIiIiIiIvLDL
Vy4kBtvv53hzzxPiyZNOHLfvdlkww1ThyUiIiJlRglmEREREREGRRuiwk3/
H671Wo9VP9sVrajj29mEcMfY9fnnA/
qlDExERkTKiFhkiIiIiIiIiKNzBtjxvBGs7a0XnMdzIyKpk1pPWhXbn70CWpqalKHJyIiImVECWYRERERE
ZFG5tGnn4H+ay2x/
auOXZg6dWqCiERERKRcqUWGiIiIiIhIAVVVVdX777z5umv5cquf0Kpn78W2Tx3zKocccggVFd+/
lmjEiBH1HZ6IiIiUESWYRUREREECqgQCVx3Z+df/JLpn06jRecuAMx942UO2mZLzjr5pHr/
+0RERES+jRLMIiIiIiIijYyZccfll/LHC4bw1pTpNHFnnw3X41c/
PzF1aCIiIlJmlGAWERERERFphFq3bs3Fp5+WOpRERERERERESWiiqYRURERKTReWfs
WE67dAjTF86lVY1x2K4/
ZY+f7Jw6LBERERGRsqMEs4iIiIg0KtOmTePgM3+HHfRjKpo15St3znnkHgAlmUVERERREGphaZIiIiIhI
o3Lx0P+jZvcNqWgWtRJmRvPt1mPovXcmjkxERERERERERpPyogllERERERE6lVVVVVBf+c7M6fQ47znjfHv
fe+/e8SIEfUWm4iIiIhIOVOCWURERETqVaGTt5cOvYp/
jptIy5V6LtrmNTVssMrq3Pn3fxT07xYRERERkcWpRYaIiIINCpHHHQonR97l7njJgOw4PPZZVN/
4KGccfULiyERERERERErERyo8SzCIiIiLSqLRo0YJhV1/
PQdU9WOm+Mwz5+hfce97lrLH6tMKM5ppqcpaAgfRHt9iv9nDr73z4bChuAEP3AkXf5mv
qmEVERERERERERERkqSjBLCIiIiIiIiIiIiJLRQlmEREREREREREVkqBU0wm1mVmb1jZu+Z2Sn/
430/NTM3s4GFjEdEREREREREREE6k/
BEsxm1gS4HNgRGADsZ2ZLLO1tZu2A0tZu2A44DnChWLiIiIiIiIiIiNS/
QlYwbwi85+4fuPt84FZgt29431Hz+4fuPt84FZgt29431nABcDcAsYiIiIiIiIiIiIvWskAnmXsBHudcTs22LmNm6i/
+kVmNtjMRpvZ6Dlz5tR/
pCIiIiIiIiIiIiLygzUt4O0Pe2bb9jmi35oVgH8FTj33d4iIiIiIiIiIiIhIOoWsYJ4ILJ97
3Rv4OPe6HbAm8Jm8hbAm8JiZjQc2Bu/FQ/FA/ti+Kg/VA8tC+Kg/
ZDcdB+KB7FvC+mu3tVff2yglUwy7czs9HurlYgRUD7ojhoPxQP7YvioP1QHLQfiof2RXHQfige2hfFQf
uheGhfFAfth+Kg/VA8ymlfFLIHs4iIiIiIiUMCWYRURERERERERERGSpKMGcxtWpA5BFtC+Kg/
ZD8dC+KA7aD8VB+6F4aF8UB+2H4qF9URy0H4qH9kVx0H4oDtoPxaNs9oV6MIuIiIiIiIiIiIjUlEFs4
iIiIiIiIgsFSWYRURERERERERERERGSpKMEsIiIiIiIiIiIktFCWYRERERERVkmZtYqdQwiIiIikvLzF
roPCYisvSUYBaRomShqZlV5LeljEkW7RftN6jEkW7RftB5GMjofiZmbtzKxN6jhKXXYc/
NnM2qWUey9lZhuaWXXqeMqMqBBmbmU3s6ap45DvZuZNzMbsQs0ttSzOvZtQiMnumbXvvLzGjCqJTu9GjtnV3QwZ
NUOvnjQQMwxSP3sHoq8I9sm/
ZNPct9ppsCu7v7F2bWXJ910ak9T10HKMFcQLnCiHOAlVPGIt9b7fGxL1ANoPNYwzCz7sBQYOf3vVfSyX
IiA1Xhn0ZjOhcpwVzksotbPzPrb2atQUm2Qj2OzVmb2YzPr5O4Laz/
vxnRgN0a1n6+ZrQycaWbvtkVZtbVzFqb2WpmtmbiMEtersxXeeb1x2A0XUjHXevyb5tBjyYfa/9Uf9qP9NK4Ekza+fu8/
P3TjoO0nN3z6rL3wQ+ghiE0bW8/rn7wuyZbXfgPVDBSrHLFbpzl2+brGbBwcp/
pFsBy2Tmqia4ZxSU3YPZj4M9kx4eZbWJmJ5jZ8smCKyPZ8bFa6ji+D03bKUJm1sTda8zsADXxEjqWGC
6mb0PfAg85+7TUsZZSszMsgN3VeB2ZAdzO7EjgP2A9obmb/
cfd3U8ZawoyoILiGuMG7BNgDOAJYC1gP+/
nqgoOIGoAOwMLAAOBY4CRiYKs6zlrhEbAMeY2Z7Au8B/
gfuBV9z9i6RBlrjaawIHt8H3eckiE2CPnwJfJWCFJV6vSuFPbw+qyL
kDqYElT7b3w94ACgn5ndALwIjHH3eckiE2CPmw5QZ1rw7PARsAz+dl4+qyL
U1bgsjZwiJmtDzwGvAS8nT9WpN7UPc1BWaZ2dru/mrimGRJcFz+mGRJFcSz4C+At
9x9kpkdSFT7bwrsbWa7a7nsxetr+WjW0k0PNNxzEVmt9T73t9fxyqrnCjBiz2xjhf7WkBPQWjn1ZVuCMTfAzsn
9x9kpkdSFT7bwrsbWa7u/vUlEGWqlyO6kfA7Wa2B0o6B2H2A9XdSs8tfhu5SlJfvQrxE1Pz86mX3QPmkfWCOg
AOkRvZGHjB3V9KGMQdputb8cn9Q5oInAlMJm5UVgc6B37ICAt2B37v7UwnDLCm1N99mdgmApEw6AFcEwAFcDEwA1i
Uugp8TU0B1Ei0AM+sEvO/uHbPXGxI36gPxLMjMMTd18i2DwSeJI6DOcTA1/
LAz3TT3fByCeZXgKeB24mHor2z/7YBDnf3axOGWDWdJy++BPwAbADO3JavQJxw/c+8BbwL3d/
Nl2k5S1LGLwGtAYmEQMMEQMww4hB+ukpYyslWbxZXZ3kAvYE2iwqkJMAuYDvxJ1+z0zOxnwEnE88Q0YBQwAnjM
3Scr8bnscvdRBwIXEkDjy+7+4dJA5RvZZWa7a0ga2AnsQx0p5IrjnwH3e/IWF4JcvM
kH0vX6dmG30Tvb1oa4b6eXOaWOA4939YTN7CbjS3f9hZG8AQ93937qG1L/c88Y5wEruvr+/dZSLav/
Un7m0Pdfc3kgaaowRzkTKzzsB/3X29Otv7tnP5udJLgSlEswjwX2c/fR2fbXgdvd/Swzaw/

cQZxEb08Zb6nJXbwOBo519w2y7asBI929V/a6N/C0u/
dJGG7Jyh0HQ4EF7n5Utv1wYG933yF7vR1wobuvmzDcspYldJ4Gtq97LTCzTYGJ7j4hV7kmBWBmk4G13X
1qtk9WJG76uhIJ522A49z9wW//LVJo2eDlLkQSdFOgA3CNu/
8yaWAlKDsOVgUGAv2Aju5+XNqoBCBr29AE6A1sAmxIVEGtS1xLnkwYXkkxs42AVYhjYGVi4Ld2QP7P7j
4mVWzy3SwWZ1yX2H8DgQfc/SHdU9W/rJioNzFY3xdoThSzfAqc4O4LEoYnLFrT4k/
EvdMcopCiyt3nmdk0YFN3H6sEc/3LPZs/
TBStXGtm9xH5kHPN7EaiaOLvxfL5q0VGkakdpSBGT58zs1+5+xW1P3f3CURFrdSj7MBtR1RlzjSz1u7+
FdAFuDp7z+dZgn9GwlBLVe00qa2AajOrIioAjyWmqNXaCJjS4NGVidxNcy9gVTO7BngU+BVwVe6tuxDJ
TWlguZuHVYiq/p2A2/Lvcfenc9/
rQahAsgGwBUSP39o+2OPM7DjiuPk5cBpwoJk9plYBaZhZB3efAVyffWFmawDtkgZWAnKDw+2BLYmqv17
EDJh/Ze9pnzJG+Zq7f2Vm/
YBJ2UPqLUBbYkDg+bTRlRZ3fw54DiB7vtiQSJ6tSVScSRHJBsbWAbYnBgQmuftZwAtm9i+yXvO6p6p/
7v488LyZ3U3MNlqPSOq3cfcFxZI0KzcWa7zM91iPqiZrfXURMfvl8Cy5vBsw293HgtaDKYTcOechYCsz
24w4Tm7Ktm8M1M6uqM2nJKUK5iJlZpcCxxBTq+4E7gOecPeJSQMrYdkBez8x3Xw88dmfRNxozCGmSU0H
OugEWhhZpexuxLS0L4nKmoeJE+cYYvXzZ939jGRBlgEz25io2vgRsBpxszeB6Kk5nOiTvb+7qwdzIlkP
rkuJJNlw4HHiGvFm0sDKSFbRcTExMHlC7dRnMzseOMrd+1v0yb7d3VdKGGpZyvbPHsS6CisDrxDXl6+U
JKgfuQTz/
xEDwPcR9647E33h1wKecve5CcMsa7nqp7WBg4E1gEHAOe7+J4sFrVU4UQ9yx0NbIlG5FjAAuMHd78/
e09zd56eMU76WOz62JXqTTyMqZzd09/
WzqejT3P3lpIGWmNyxsgJwIpEYWx44xd3fThudAJjZZcAV7v6WmXX1Ou1Bs5lhg4F57v7XXJGk1INscH
5O7fXCzJoBZxHHyi3u/oqZrQcMd/duCUNdghLMRczMWgA7EFM6BxFJt6+AzZVEqF/
ZyHUzYGti2uC6RA/mDkQ/
qLeIRNtWdduWSP0zsx7ENOYfEQ+t7Yibvp2AdfTvv3Cy806Nu1dno9fdiOTMAOLBtB8wwN17JAyz7JnZ
SmSV5sT5qi9xjViOaDPzeMLwyoaZrQMMISqf5hMPppOA6939FjO7CFjN3XdJGGZZySUMdiNuxo8ijpW/
unsvMxsMtHf3C5MGWiKyBMEL7t41a8EwGagk+jDfAvxUCcx0cv0bRxADxecQg8T3Z0mBM4np/
+oVv4xyn/V5xD3s48DuxPVgiJntRUxrnpw0UFkkt8/+DYzOppz/
FVjO3Q8zs9OBlu5+iipp65+ZPQF8QBSwXEDMglkIbE70vf4qYXhlzcx+R9w3zc/
203TinPY8cS35OBsk0HFRAGb2W+B5d38smzH5pbtPyv28LbAfca66qJja96hFRpEys6bZdNp7sy/
MrCdxo/J+ythKUXZinE9MP3gIwMy6ETeIWxHVOOsQC3ZIPcqNYlcRSfybspvvu7IvzKw/
sAXwupLLhZG7MP0cWNnMngHeI9owPObuI7Npnn2IKbWSkLuPM7OPPHpmXmdmXYgBmfWIATGtVN8A3P0V
YDszW5U4f3UHHnb3D81sRWLmy6XpIixLlv13X+AOd38qS6I9lm1vS1zPZRnkzi8bArXX5R2ACdm05h7A
ykoup5UlzwxY392rYFGLmKOzt+wGPJAqvlKSq977JTEQP8XMDiIWU4QY7JpCDMJIEcjts45AbeXsIKLf
LETrn9o2ZBV83UdbllJuEHhroLu7b2lm3YE/
uPt4M1udmHl0S9JAy5y7n29mFdnLi4l8yIHACcBU4GUzex7QguKF8SVf31sdA3Q3s7eJPOB7wBiPRRab
QHG171GCuYjkTrjdgX3M7NfECtyjiIXlXgGuyG4UpR5lD0ITiRYZdwH3uPsU4O7si6xv3cfJgixRuQTY
KsDpwIVmNp1ojXEXMCybLqUpUwWUuzC1JBKVmwNziVHq183sLeKCNk4VBWnkBmOWA6qA7bPz0jvEFNxH
zWyUZwuiKLlc/+pMgd4l+3oeeLR2CnTOVOC87L/SQHIJg49ym3cF/
pB9vyPRykGWUp1KmdHA7CwpsBVZn2uiskaLxhWH5YE3sum0M4nLwztm1gpYQdXL9cfM+hIJ5wazewAAI
ABJREFU5Olm1hVo69GPGWK2kVotFIk657HrgP3N7GmgqZv/
28wqiWeTYbDYtUWWTe296WrAq9n3+wIvZd/3J3r/utoupJP/7N39P8B/su3diYGXvYDd3P0aFbTUP3e/
HBbNsn+M6N/fl2i9tBCYZma1zxlfJgrzG6lFRhHJTdO5g6y/
CjGiuidRSXuYu9+VMsZSlbUF2B3YhqjA6QN8Qtb/
190fThhe2chOon2I3nU7AusTD0ajsq8h7v55ugjLR3YDsTHxQLRxtvkrIul8pPZDw8tdI84kbu5eI6YV
bkBMKzzP3f+rG73CyWYXVZvZhcR1+WPiYWh1Yvrgm8Bf3P2/CcMUFq1Mfy0x7fZSYG1gBeD/
iNXPtWByPcnOSYcQ1+t/
EcfFasCF7v5UytjKXW5Q7HiiBdwMoJ27H2RmFwMruvueaaNs3PKJyqxP5mXEwO9UYF9338XMDgR+ozZ7
xcliEferidktHYhrRl9gvLufqPuq+lHnWOkAjCCu04cCl7r7rVke5DV3P0sJ5nRyzxznE88a93gs7NeG
WMvCc+/V8VHPvqnlRXZ96U8MfG0CdHT3X6aI739RgrnIZFMRZgI98lWCZnYskVD4pbt/
liq+cmFmKxM3GXsRVTgAZ7v7aemiKi9ZZc2qRKXmednmHllluRRQ/kYha0+yIrEP1iIqmPsmDK/
smdknwMbZVMIKovfyH4ibjsN0jBROLlnzLjHQ8mjtdmJg7Ejgbne/
sTYZnTLecmJmnYnVzOfltu3N14v83UWcwy7QYP3Ss1i4stLdh9fZvhZR0d8XaAKc6u6a9VUkzKwXcDax
lkUlMWg/
GrjO3d9IGVupMbOfAH8hzjsfASOBrkRP2ZtTxibBzM4CLvMlFy6rAgYS64887+43ZtuVQCuA7Bp9CJEs
e4ZoM/YUMTg5UZ97emY2jRiUf9HMjgKOINqK/srdR2sfFUYuwX880VrpXnf/
ymJxxbbuPsHM2rj7l8W2D5RgLjJZYvMG4EB3H5/
b3pnoP9szVWzlIEvWLA+0JiqaexDtArYCfu7u1/+PPy4/
UJ1EZh+gC9HPcTeiz9nqRCX5A8Bkd78mVaylLNeepz/
xb34AcBgwm7jZmwn8k2jZ84K7v5sq1nJn0Wv5EeDgrG1S/
mcTgPXcfXqS4MpENvh1LnCju7/0Xe+XhmGxsNZ12dT/
fkSFzUcWC89tSCyA+ZS7f5o00EbOzPYlqmauNLNfEMUPNwCPq9KsOJjZj4Cr3H3TrD9jK3efnf2sOTEI
0A0YpX22bLJr8q/c/
Yw621sCPyUq+dsDV7v7mAQhSh3ZMXGOu5+SvX6YqNC8y92fTxpcCTOzA4gFdq8ws5buPjfb3odYZ6cVM
A+4zd3nJwy17NnifbKvcvd+FmuNDAdOBLYFWrv74SnjLAdmNpEoHnrQzA4jciQdiJ7lo76p0jk19WAuP
hOBN4B/m9lpRM9TiKqQ92Hxnjiy7LKpHpsQCeU2wBrEjfd/

```
iQvdxUQls1oC1D8D3MyGEg+p3YjemHcTSc0HgIW11fzFNkJXKnIXpuHEFPJbiaq/
2e7+DOizLyLTgX8D55jZ4e4+OTuH7QnMcffp2leFkbuJWw/YH9gxaw3wlLt/
mDa68pYNDtcAY7NNFwEzzewN4PXs61Oi6kaWTWeiDQbANKAT8HegrcUCNI8CI939OZ2LkhkHnJp9vwvx
TPEksYj1fdng5FupgisxlcSzAma2GfBX4vi4z91vShmYfKsNiGft2mfAh4nnwFuya8kbRM/
T+9z9nVRBlqDZQG2Byglmtgvw IFE08W93n5MsMllM7rmwFzAuSzT/
DHjO3e8xs4XAafDNrRykflj09G+WJZfbAmcSC5D2BX5nZi96Ea6LpArmImBmPdx9cu51S+AMYorODGJK
57tEi4bnlGCuH7nRuVOIarSxRE/Ti4AXPVsoSwrLzJoSC/
g1JRJnnxJTN98DZuqGo2FkVTg3EpU2vYhqjoeAp9391f/1Z6VhmdlqRI/
H7YnptxOJBR5uzFoz6BpRQGa2JjHNfE2iZ3wz4lr9PnCNu7+WMLyyZGZbAD/KKqMqiGqonxAzkDoQsy/
eJ641l+thaOlkbRZec/fOWVuY/YiByQ5ES6vNiGrxzYHNde1Iw8zuAs5y91ey/fQjopp2V+K8NYNIMJ/
u7iPTRdr4ZdOXR7r7a9kssMFEv/
dVievyC8Q97R1qcVgczOxO4CV3P9fMdiSuD68Q9779iGTz9sAnHv2zNVBWD8zs/
4CT3f0LM9uJGKxfjWjD15QYGBtL9GLWTKPEsmtHEyJHsiGRIxmanev+Dbzj7qeqHVzhmNmWRGur04nz0
hbuXmVmaxA9sVcpxvOTEsyJZdNCfu3uv7Fo3D0QeMvdP7NYkXt14qHo/drpbVK/zOznxEPSTKIax4mE/
jjis/
8g39NR6leWDFiHuCEfSFSDtCFu+CYC44mL2COpYiwXZrYcUUXeF9iaWNxvJWJhv7eIBKZ6lzagbMT6Wn
f/WfY631amF5HI6Ub0/
f0oXaSlzcz2IipsvHYfZK0y+hAPSKsTSc2L3P0xVXQ0LDO7DcDd9zGzk4hrxrDsZ7WLoWwLzHP3I9JF2
riZ2XbE4NYGxL/
3k9x92+xnBjQH2gJd3P3tZIGWOTNbAHR195lm9iCwt7vPyn5WQSxoPRi42d3vSRhqo2dmM4G+7j7DzI4
E7iXumXoTBUKbADsDZ7j7tekilVoWi8hNdfejzexlYLC7v5D9zIAWRLGFu/
tUXc+XXVbdf42798+qxg8gWit1B3oSzxr9iaTzwUowp5ddK9oAC4kF31/
wWORvY6IQ8jfuPqYYE5ylxMz+ABxEFH5dmhWbXgZ0dvf9izHBrwRzYmY2AFjH3f+VnXzPJCpsxgMvEas
QzyjG8vdSkiVx1iFOoKsRfWibEm0x5gDnulacr3dZn8yx+Rs3M+tOVNusQ6ySuiLwirv/
LkmQZSy7uehL3PDtQuyHIWmjKi9m1o1Y3PVsMxsIPAHcRrSPGenqt1xwZrYeUXWzX7Y/BhPX5/
eBie4+O3so7Uxcr/Ug2sDM7Agigfw34CZgiLtfrmr+
+mVmHYDziWqmTsQ90rHE4q+6RyoC2UDxVUAFcZ0Y4u5d0kZVmrL71ZHAH4nqvsc9t1aORa/
flkBHYJqKVYpDdi91AbHezkDgEqJNw/NKbBaGme0KDAF+QwxwrV07OJl7T0eiv/
8HCUKUOrIB5YeIhP9N2TYj1qrq6e7PpoyvHJjZ6u7+lpn1Bj519znZLNbzgSvc/b/
FOACmBHMRyaqZdyZG8foQo0azgS+IhQdGJAyv5GQjcPsCtxOjcgtyP+vJ10nO/
sARrgUH6p2ZfQTskk3jPICYevt6nff0I85VqoZqABar025MVG+MBcbUtinRKHU6WbLfiB5oexI36O2BN
4EniWlrWnCunuVaKXXMqgG3Bi4EviKmP39A9Gt8D3jT3T9OF235MrNKoqJmO6KP/
IPEYMzLxD6aSfQo13V8GVksmjiASKw9SZyXOgBTgdHZ10PFVlFTTiwW+fsNcY/bnJj+/xCxeNYr/
+vPyg9jsdDl4UQSuTvwB2AM8IzON8XLzNoDv86+biHW32kOfExUCo70bA0SqR9ZO5kNiev0eGK9lw+Jt
pRax6KI5O59fwocTJzTLlJRS8PJ+i8/DFwO/NNjjZ2mRMV/B+CNYkss11KCObFvG3XIHpZWJnqlbU/
0DHxSCZ76Y2aDgCuIauUaok/
aMOBBd9fiJw3IYlXzMUS17FSiX90w4BF3n5QytnKQu5EYQExZa0UsitWGSKJ9SFQv35IwzLKUv0aY2bH
AlbWJGzNbgVic9FDgdnf/iyo265+ZdXL3GXW29Sfak2xMDAj3Bi5w93/qOp2OmW1LLLB1GdGDuSfRb/
Z1YJS7D00YXqNmsT7IctmUcQNOAq4hqpn6Eknn/
sR96y5q2ZOemd1KDLBMIqakr0ssdPkhcKy7P54wvJKRzWy5l/
hs3yf6Ly8kBunfIK7PE9NFKLXMbGXgY3efa2YbETMxHiaOjZWIdlcbEQPGJxZjdWBjlbXG6Ecs5n4H8Y
zRhThWPsm+Lqt7vyXpZM/ng4DfAguA89z9v2mjKn25Vnz7EQn+14DzG8uxoQRzYrl/
QD2BU4jRoQ+zn61F9F7+Uhe4wsrak/yMeCDtC0whqnMeA25x9y/
SRVeazKxz3alo2YVsJ2JfbE30lh3j7j9q+AjLR21S0swuJm72jiGOg1WJga7ViRvyUxKGWZayKbadiYr
ZCUTPLV24G0jWPmkWkaD5L3AXMQhZnXtPE2Jxs3Hu/pESzA3LzE4gBiNfy14vGmTJWjpsA+wFzFb/
5aVnZr8GKt39j9k9a0/g1drZX9n1uwvQ3d1fTBhq2TKzFsCvgL9lg8Yruvv43M/
bEBWEBwDXuftTaSItDWa2iru/l31/IPA4MI8YaFkl++/
mwJ+VlCkOZvYSsKO7T7FYQGsKsdZO7XmsJVF4NNfdJ+t6vuyy6f3z3X1q9nofd78tq9CsfdZYiRisPFi
tZIpPdm35HfBj4FFiTZ7xOj4Kz8w2IGborUvMoLwqyw0W7WevBHNiucTOiUTFx9ZZZdQJxCIqj7v7CcX
8j6ixyqYZeN1qv+zmYhdgD2J64Sbu/lyCEEuamZ0H7AA8DTwPPO3uY+u8pyfRo/
yBBCGWHTO7iJiq9q8623sBFapIazi5wcctgcOIh9ZBxHXBgOlZAqEFcK+775Aw3JJmZisCmxJJyp2Jz/
9dYDhwh7s/
nyw4wczOJaqeJpvZVUQV4ZNExbJaltSTLEnQ2t3fza4VhwDPENfvZ4n1Qz7JtxuThmVm6wAHuPvJWaXm
KUTl0/tENe0EtW2oH1lhymnuvkPWbmEDolp5Wm6Aqx2RrByn46I4mFl/d38722dvEX3k3yLOYc8R1/
apmglWf8zsSmKA9+SszVgr4vo8O/eetsTCpOq/
nJBlC8aZ2erErKQtgXbARCLBuRMwjWgvera7T0kWbJnJjp1jidZvf3f3z9JG9O2UYE4sl2B+ghiRuNnM
biAeYP9NVCJc4+63Jg20xJlZM2K15+nqA9UwzGwLYG2iOnYVop/
s58QN+pPAk+4+LXuvBlgKLOvxexGRSDuReCj9ytVHMymLXvH7AbsRK5s/
TrSReY1IGOwKbOrum6g9RmHkWshsCewDtCU+
+0FEdRpE65KjU8VYznL9sZsQbRtWJRaHbUM8CL1EJEGH6zqy7HKDXxsTvTS3JarPPieqAY9x93dTxliO
cvulhbvPy6b/nwA0IXrLziJmwUwi+gO/nDDcRi33Wa/q7mMt+pTeRNy/vkEUTrwIjK87U0/
SM7Nm7r4gu2ZsSsxe3YqscpnoIX98yhhLRe5YqU1cXkrMkpxDtEZ8ALgfeDn7uZ73ikCuCOweoJIocnm
IWBesM3EvvD6wm6739Sd3bupFfMarANOJ/v6bEmuMNCHua//k7iOTBfs/KMFcJMzsX8So6cvEasS/
cPfXs6k8p7v7vWqTUb9yyf2tiET+ckTT9OnEgijPE4sEzdLFrn5lyUyypE0boBeRFBhALLSxAvFANAf4
```

```
aTGP0pWKrNrpLuIY+JI4Fz1NHAsfqhIwLTP7LfHg2p2oKFgBqCASaDe6+z1KMBdGLsH8PnCUuz+UbTfi
er0JcIm7P1T7EJUy3nKW7ZPOxPGxItEPeG2gmbvvkjC0Ri93HDSrW42ZzaTYAvgpcKq7z0wSpCzBzHoQ
iYABwGpE24ZL3f32pIGViFxCoCNxDOxMtCFpS6zv8gd3vzNljPK13Hmsu7t/
Uudn7YgKzbbuPlT3VPXjm5LGFmu+/
IwoklidKKAY4FrQPYlssOUwYsHqj+rOKP6WPzMKuMndryp0fOXGzA4nZtOPIxL8TxBrJzQj8lSbEbMqT
/QiXIxUCeYiYWbrA2cSqxDf4O7/
Z7GA0+vElJG5SQMsQbmbjOeIZPKtwNXAq8Q0kA7Ace5+R8IwS5KZ7UicHJ8lpm5O9KzPtZl1IvpwDQB6
u/
uFyQItQ2bWmlhYdEfioXQ1YKi7n5g0sDL1bQ84Fv1O1wYmuxYPKrjs834ROLxuyyQzGwkc4u4TkgRX5n
KDxdsSg2Mv5I8Zi0WTO7v7O8mCLAG5z/
kwomr5OKJqeUWgk1qJFY9sEH8f4DF3n5xtq60kXB2Y4o1ksaDGwMy6etZfNretF7A38Khn/
eGleJjZ88AQ4E5iUHIj4lnklaSBlaDcuac7USTxWt2COTPb2t0fSxKg1Cb8byWSl9OJZ/
M3iEVipxDtr77K3lubP/krMVg5LlHYJcXMziI+7xe/
zzXDzB4ijqWTCh7cD6QEcxHJ+s1We6zQ3QY4CNjC3Q9Q9XJhZBUHH7h7x+z1TKLiaSdgHeAUd5+VMMSS
ZGZHAPsTrWC+JEbo3iT6oL1PXMjm6t99wzKztvmeaNm2FYGWqipoeLmb8pbEw8/
BRHXBPe7+ZtroyktW3XEq0Zv/
RKLfbAXRGmC4uy+XMDwBzOxNYlD4YTNbk2gP0A44vjbJJksvl2B+HLjT3S8zs0OAwUT/
2SuA39c+hErDyz34/
wS4gBgkbk60WNqZ6Ht6ecoYS0Xus14fuIpoZTWOmNK8BvBU3QpZSSt3T9WfSPz3zGZfXE8UvTQBBrv7f
UkDLTG59hh/Iaoxjyfun3Yjirn+lbW50jNfArnjojlRYLclcU3vQaxp8RHxfD4eeKV2sN7MWut6Xz/
MrDNwCXG9bkW0tHqX+Nxfcvf3v+HPHA88W4yD+0owJ5S7OTmN+AdSO+22KXHi7U4knD9WT6L6lTuZbkc
s0rGlxYIdQ9199WxK4QPuvm7iUEuama0EbA1sTNyUtyBGTt8mLmi3qMqm8Cx6kG9PJP2XJ3o7PQM86O6
fp4ytnOUSOmcSg17PAAcS7XxmEDNcznb3RxKGWTay2RWnEatoTyQGyHoRs44u0nTahpe7lvcl7qO6mFk
r4ClgFPGA9CJwoR5c64eZTQH6
uftnWduYY4gHoRuI5MyYpAGWsdw14wZidsvvsofQnxKD98sDF7j7g0kDLQG5pNlFxAyJQ81se2JhxY2A
0US7Qy1aViRyz93HAdu7+y5mdijR+mojMzuaWNPigLSRlpbc5/4BMdtrlJkNAdYj+sreRvTu/
yJpoGXsW9qYVBIt4DYjWiv1B/7i7tcrL1X/sgr/PsTn3Df7vmP242lEdfOT7j4qe3/
LYu1w0DR1AOUsO9k2B44iep9iZgcCfwI+Bg71rO+pDuL6lfs83wb+ZrF6bTNgnJn9GNiTOJilAGpvzLN
pNeOA67LtGxCLZg0kkjjXp4uy9OUSYrsCZxGLOUwmzklVwD/
M7Gl33zFhmGUrl6wcDOzj7o9bLDR3KrAv0eexKXx7Kw2pP9lg16+z6qctgAXEoou1C8MqgZnOBkTPeIB
DgBp3P97MdgX+7O7npwutdGQPnM8Ax2cz7T529+HZz9YiZlhIIrlrQBdgePb9vsBf3f12M7ufmHUhy67
2fL8pUNvK7WRiYH5bM/s3cT+rBHORyA0yfgLMMbNziCr/67LtKxF9s3VPVY+yfMdyRGXm69lg/
cHAQHefYGavEZXMSjAnUpsXMTPLNrVz9+nAsOwLM1sV0PoKBZLNePkEeD4rNu1NtCDrR5ybNiUWWxyVJ
fiLMrkMqmBOJjeatzNwrruvld2c3wKcSyR3PnT3PyYNtASZ2cr5igLLLVhjZlcRn/3LwN/d/
eFEYZa0XJXNQUQvx3+4+1N13qOpNwWW2w8PENMFh5jZlcQN9l1EIvMydx+WNNAylt3QPeLufcysPTDO3
Ttl1f8nASe4+/y0UZa+bD+sD3xFTBN8T+en4pG18vkXsbjf08Bt7n5nlkBY2d33SxheSchVi28B/
JpInN3l7s9mVYAnuvuPkgYpAJjZ/kSLmBbAc0SbmK/MbCqwlbu/lTTAEpJVYrYB5hKDvntnM0/
HAwfVVptJcTGzU4mWiPcSM1a/MLMxwO/c/T61a6hfWVL5LGJgviPQ1KMFaF/
geXfvnDTAMpe7vlcSveN/AXQCHgIeA+5z9y8ThlgWsgS/1T33ZK18VgI+c/
dPir2CXBXM6dT+o1gJeCuriDoOGO3uN2cjF4fA18noRHGWlKwNxmnADlmyZiDwtpl9RiQOjsm2feDuU9
JFWtpyVQFfEKsHD8uqoV4BbgL+6+5v699+YeX2w4rEgosQlZlHuvuTZnYi0QdK0ukMPJ7d9PUnpqJD3K
Dv7u5HF/uNRmNliy8edyqwKtEzvgnwnkXP3yfd/
Z6UcQq4+3gz25uYjfFylvRcHdid2HeyjGofPrPpze8Cc9x9lpmtRvScvThxiJJx93+Z2WSiYvAZolrzZ
8A0JZfr3V+Ac4iWSb/
NksvrA22UXC4+FgtZt3L3v5hZF3eflm3fjWirNAIWq3aWeuDuM8zsJmJw8nHg9iyZ9nPic1fVeFpNgGp
i/1QBZwM/I2bA7APcYmY3u/tB6UIsC62Bfc3sz0Q3g1FE28qXgI9qk/zF/synCubEzKwD8Fei/
+wbwJXu/
lpWUTgquwDqhFsPcqNzq7r7WDPbC7gZeA0YQ1Q9vQCMBebXVjVL4Vn0zFyHaE1yYrZ5K92cF1722R9NX
MReJkaqL8n+Oxbom02TkgaWO2d14Ouqj78RNyCdgKfd/YTaljMpYy1FuQTzCOBVj36mVwPticGx/
YiFYC/TYFjDyx0fnYg1K97KTfNsQvRf3oZYQEjHxzLKCh8uBa5199FZRU1fd3/
TYsHkue4+J22U5Ss3M3IbYIK7v5f7WUuil6a5+6PJgiwRuXNPD2I9hBoiofy5mbUDDiNLYiYNVBbJXc8
PJ64Xf/
foI7828cz3lpl1cq37UhBZId072XFTuy96Ei35HsoGLnUflUju+vEe0aL1SYsFff8OfEas0XOluz+vvF
T9yx0TRxPXjzOAXYhWMjOJ2Ug3uvuxCcP83pRgLgJmtgJRQTgqO7i3JUbE93X3D1SdVv9yB3JHokVDFT
G1rTVxo/
h7d78rZYzlyGKxuV8D0939uu96vywdM+sKfO7u83Lblsuq0Q4Hfkk8NJm7V6WKU8DMerv7xNzrbYmqgu
eA4e4+WTflhWVmk4DN3X2cmY0F9nT3183sCuBydx+jfdDwctfx04EBwAHEVPUtgLWAYe7+WsIQS0LuwX
M34Bx3X9NiMZoTiOTAO8B2rgVhi4KZvQKc5e53ZdXlg4nB4uu9iHs2Nia5BPPVwP3AvdnrbYhioelEwl
k9ZYtE7jz2JrG4+51mdhhxP7U9cKqrV3+9yh0nyxODk7/
NCry6E20YRrr7G8pzFIdscOxV4n7342wWTP9s4Ox54t534v/
+LbI0cvezTxOD+EPNbChR+HgfcBVwk7vf2hgS/BWpAyh3ZtaFmFrYJfsCeBM42bM+wTrp1r/
sIDZ3n+nud7r74e6+FjAI+AdaqKZgzKwi+++OZna8ma2XVdGSVY2/
TtYeRgrmNKAngJkNyG7+apPNtwE3Egss7p8mvPKWTRvEzPoAF5nZH81sPzMbSAxEHuHu17r7ZNBUzkLK
rtGjgKbZzfdCvl60aT9gImgfJFJ7b7QXUaW8APgDMQvmAODkbDBNlk3toj978vXCcUcTSfytiNZWByeI
```

SzK5a8bqQPcsudwWuJIYfDmd6Kkp9SBLmnUjkpP3Za/PA64gFg/
fQMnl4pIll1sR9753Z5vPJo6RrYEds4paqT+1eaZ9geWy5PJGRLL5EOBCM+umPEfRaApcBHTKBgHGAVu
Z2ebAqkouF04uYdwaqF2Talui5dsk4n73jTrvLVrqwZyALb7A2WDiAN6VaH5/
EdFz88mEIZas3Ghqe6AqG72e+f/snXe0lNX1/
j8PxV6wYUNRwd57L7Fr7JpiSTT2GqOJscUeNZrYeycaS2JU7Iq9i4qKigpYABW7YMEG8fn9sc/
I6/2h30Rm7uHOnM9aLO59Z2DtNe+8p+zz7GcTHVJvSkn9v2UNssmpJGIWJzan6wBjJL1KNEnZgLBnKDS
AVDo+xPbr6dKphNpmkMJX8wVik1SSZvkQsZhYAlgGWJdQFXwCfJwUakOBB8omtuF8RDTeNTE+PQncIul
5ogz946JezkNKGHQBpiO6anclDsXWtj1Y0gtAd+C9nHF2dCqbmXeAD9JB19rAybafSUmZwdkCLMD4OWM
NYq6ASChPZXsdSesDfyLN7YUfT2W8/ynwXNrPbUocwPwU2Jjo59I/Y5iFCTMzcC+x/
1sIeNV2X0WPi8Vsj8wbXtNRSxxvQAhXIA6ARwEbEfYL2wGnFRVzfmyPAs6pqP2vJXJTnwGXQvHJbiSKX
lR9gB6SXiMOK3tIGk5U2m+dMbz/iZJgzkDlwTwe2BO4k/gSDUjXDwDOIlQhhfpSM7E/
CNgEuBboARwLXCTpC+Bvtk/
IF2LLcBlwGzAfsCRxH3oTXthnZYyrqUnjz5lJ8dSJUG8sA6xCPBNfAMMJb/
IzcsVZACJBcDIxR3Qn7tEfiMTzUGBtSX8unoGNIz0v39osSPor4Y02PTF/
w3iFZ6H9mRK4EbicsLe6JyWXZwF62n4ha3TNxWWEgvl3xDNwezqwXJbxvRMKGagccD0CrCTpUsKnv7aW
+ikhZilMJJXPejTwarKt2ha4Jik0pwKmzRZg4Xux/
YakBwkF8zXA3umlnUjqwJJAqx+VZ+UeYN00L88CHG37/
WThc3V6T+2QrJCRZB06lkgqnw0MI+aOl9NbipiiQTga+J2eKr0N3EQknN8mbDO+7iiCluLBnAlJ8wF32
e4laTpgmO0Z02vvAguXpEH9qSiYHwD+avuWdL0z0JMwVH/
b9r9yxtkqSJofGJe8TQV0cWmu2FDS56wJTVDJKmMpItH8ue0D2ju+QpBKOUcBM6VFR+36woSioC9R9vx
vwjN+kl9wdESSxcI+xL0YZfvvmUMqtEHSPMAOxCL89uQdeCiwnu21c8bWjEgzKhlNAAAgAElEQVSa0/
ZbisZxewFb2l4jd1yFQNJviJ4ilxLClTmIjerhtm//oX9b+O9Jlkl9CP/
eo4A+tkdJehy4wKWPyCSNxvcdWZRYU91g+4qSYK4/FQ/
mOQgf5gckLUWoyeco3vD5SbmoPYDlCaHRh0QVxhPAW7a/
zhhey6A2jUYlLQF0BYam8apDKP1LgjkTknoRyoJDCJXB0bbXkrQK0SWyV0f5EnVE0gLctvvkjqWVqJTd
rESooGYnLGHGEgb2l7nSeK7QGCr34RxCkdYvnYzOBYy1/
U7mEFuepPS4BLi+Ok5J6gE8ZnsuScsSys1lynNTPyrPx9LExnMsobqZzXbvZBHwqe1iCzAJomicvCnwg
u37M4fTNEjqDXzUZvPTGehe84MvTHpIWpXwmD2xHETWn3Ro39n2OEmbEWr+zW2PzhxaoQ3JHnFx4CWi0
fW4dH0yYk9YBC7tQEpm/gxYwvb+HUWV2YxUbFv3JarybiGSy/
MR695uhBXQgRnDbHrSWmpnYEuiYnUIcS/
62f4gvafD5AWLRUYG0kD6qqQ7geuIrue3SVoSOJRIKkCUr5dT1DpRSRpMSyQMTpe0JnAX8KjtYVkDbA1
qA+PZxMB5CfAusDThWTeKaDJXaCDpOZiK8D77a0ouH0l4wX8m6Xe2i0VPRlL54I3AWZJ+ATxI+AGvCzy
T3tYd6FqSy3WnZnmxC/
CB7Z0k7Ug0dILwQusB7NmRFnzNRiojXIJInk0HvEEkDp62fXbG0JoOSfsAuwKLS3qFsHC7EXjSdmmKnJ
FKZd60wK+AuYmy5uHEJrW/7Ud+4L8o/AhSNdF0wCDgcwDbN0m61/
ZnWYMrfEslgbYcIWxZB5gVGChpILEXedB28epvAOkAZnNgBiJZ9pbtT4BLUg8FKNYYOan6ZJ9g+yr4tn
pvHqKq9c10rRwE1JnKZ7opYdd6KvF5r0FY5p4m6RXbq3akvUZRMGdC0naE/
+8mxGnF2kSi7QRiAP60bFzri6QuSWFwLvAT4HGiDKQn4/3SLikl0I1F0uxEV9TZKte6Eh5oWwC/
KIvzxlE5aNkKOMT2CklR/
g+iPGoTYAbbO+WMs9WpJA0WJvwdlwQWJHxQL0yluBcDnxRlQX2pfPZ3AufavlHSw0R10QWSrgJetn1sK
adtfyoJgw2IXhZfMP6g8jXiALm/
7WMyhtnhqTwHCwH3Eeqmp4ARxEHwL9Nbu6WEQSEDlTm9D9CLsIpZn2hu+RHRgPFU2wO//38p/
DdUxp7dge2JA67JCf/eu4GHgHvLoe+kQ+We9QU+s72DpAuBOYleCqsQe7/
dyr67viRV5iXAounSskRT8b6EaOKqkrCcNEgK5umAs8t83n5U5u+/Ae/
bPilVVNQaWM9HrLFu60j7jaJgbkcqi/U1gbPSKdENku6w/
YWkrtXynDLJ1Z3aQ7kKsJ3Hdz+fg1CjLU9snAqNZWbgdUkb2L4TwPZYSXcQVjEludxAKou56YH3JW1NK
JdvtX1Pssn4dbYAC0CM/
5J62n4JOLLt68kG4FbioKxQRypz75XAIZKGAosAV6XrqxDNF6E0PMnJb4GbUqL/
IuBJoiHjzoR1TGHiqFXRbQU8lA5adgAesb2dpGHp57IZzUjanE5J3KcFbL8j6T1Ccf5HoolvuUd1oLK5
P5bofXCpom/OvcDuhO3hWkTyrDAJULlnKxEHAhCirq2AT4lKvkvT9dJorg5UVJkbEZ/7esSe4zbCr/
xiYAPb/
8gXZaFGOkQ+ljgs6yzpPqICZhTwRUdJanZEKnvyT4AZJU1p+wvga6Iy5p1UBUBHug8lwZyHqYFvm8ilL
1ItyVbKDxpESthMSZSzzZSujQRGAk9J6gcU1UEDSYcsz0u6CThJ0txEedqSjPd+KrQPVxAno/
sQqrRz0/
XfEKXPhXamcpLdmyh1XkPSTEQ5+uNEY9ihALaHEwvAQgNI9gvXEpuj44m5YcfksXlnzUKmHAS3P5VF9i
LAn9LPqwC/
sf2EpMUoh8UTRW1Dk+hFKMMhysvvTz9PDqxMHHQVMlBRXK4FDE7J5VWIpMCDkr4CtrX9etZAm4CKSGg1
YHRKLs8PYPsQSYOBr2yX5PIkQkW9PBdR5TKNpC8Ja7Hn0nv2J3oiUfbfdaM2f2xCHAK/
IWkP4Anbl6X78Rh0LF/ZZiTN9UOIiqQViXu2C/
G8DCWqlW7OFmALkOytViDm8ZkkPUZUHr1i+52O+HyUBHP7UjsZXQbYUtI3wN8Jr5X3bY8tk1tjqExgiw
ELA30knQ48TwygI2x/njPGVqAySF5MPAt/AM4HXiYmsaI6awfS8zAuPQP/
sv18ur4moegoPth5qC3KjyI8Ak8mvMkXJhI5Z0s60fbh5TCyMVRK0I4kOmgfB+wIfEmU1P6bNE6VjVE+
FE2C+gJfSZqCmE9qB8QbEurmwo8kfa9rifxziO8+wKvAMpLWIzaiB2UIr8D45nLAuHTpwXRtXuCJdG1x
YuNamAjS51pT9C8C1HzH1wAGpJ/fAw5jfKVLITOVw8h3gf2IvMfkwMupJH0UMMr2J2VNVR/

```
Ss1JbF3UmLDEg1rH90s9LE9ZWd1FU49morGFN3Jt+wHEK/+W1CF/
gOdJ7O4w9QwfkG8L+8G7C83oTQv0/RtLTts/
IGdyPoXgwtzNp4D0O6E2Yp39CNON4kWhQc0vx7mockhYnNp9zA3MRE9uHRFLtetsPZAyvJWiblElKwbm
SIrPQQCoK2Z8QXmjn2/6sVpIjaXJgXtsvZw61pZH0PtFd+21JrxKlznMAqwF/sT28bIYai6QXCD/
4QZVrk5f5edIgeTvOTeRCh0k6kXg+hgBL2V42a4AdGEm9iOqWp2yPavPaHITab27iYHjHMg7lJ+0t5iG
SnLMRVUi1Q5eLbZ+fL7rmQtLMhNVCfyLBfDhRlbop8cwcmjG8QkLStkQly8C21nuStgR+T3iUX2n7nyW
BVn8kzUg0ox5MHEauDzwC7A2santIOaiftJA0te0x6edZbL+fO6ZWQtGTamFgAUJYNMTR+6VD7flKgjk
jknoQG6KVCJXBONsb5I2qdZDUjTgpWpkoCznJ9mN5o2pOKmVqSxOegGsQ5VGPpr9fAz4tKvLGUrkPtwI
P2z5R0jLEBmkzYDfbfbIG2eKk5M5thG3MVMALtmsKgheB5WuLv0JjkDQNsRka7NRRO13vBHR2pVdCof2
QdCpxGH9X2wNJRTPMIwil2nm2h2QIsSlIpczbAe+kP4MIleYg21+mtdN0tosNSSaSZ+aFREXFnbYHt3l
9BaLk+Rng3zUrvsL/
Tvq+b0PYVL3UNgkp6RjCY3YIcFQRS+Qn2SFeSSiWvyQSzc8TdnCDUwXfrLbfzRhm05HseRYHngZet/
1B5bWpgFOBGYj9x1l5omxtKlY/cxHVGF8TOZD5gM+InlQLpLd/
ZHvtPJG2Hhrvv4ykqYHJbX+UOawfRUkwtxOSuqQJbXdCUXCj7dFt3jNPUuKUU9QGkRTMqwGzExunO2y/
JqkL8E1HOh3qSFQSm08RSeVHCauYNYimf12B39m+IWOYLUNSyC5pe6SkJwn1zZfAusDett/
KGmALk0rTNiJK1WYkkgh/Jsasw2336mgn2R2FysJ7E+AmYDRwBlFZNOCH/3WhkSTFch/gJ4Sa/
3XgHuI+PV7dyBYmDkkLAIsSitjewCzAFMTm8zXiUPgV24OL+iwPyfv3ACIRMB8xfw8A7gTutv1OxvCai
uS5fBwxH3xGjD3PAM/YHpbe0932e9mCLHyHtKdbipgrFgV6Mn6v8QlRfTGASDYXf/
I6IelwYo7+iKgOHk4cvLya/v6a8L/+MluQLU6lkvUywv7tY+ABwqbhK2JvPoqw+nnR9t1lz1E/
KvuMmYm5uzdRUTyOsO9ZjpjPST/PZfvTLMFOBCXB3M5IOg/
Ygyhde4tIIlxp+76sgTUxleTm5sBehDXG/UA3YErgVNsPZwyxZZD0DLBGdbBUNDHbhOhG/8r3/
uNCXUglztcT41Bv4BTb86TXPiQsMkrH+YxI6lpTyUo6GtgSeJ9IdJ5eO7DMGWMzI2k2YpG9MuFD14PYG
I0EjijzRV5S2e3mwNbA6sQ8Pojwc7zR9iMZw2sq0vw8L5HI7E1YY8xMrJ92sP1mxvBamnTo0hlYiKh4W
YUYt2YF3gZeAk53amZW+HEkBfNihA/5osR8MANhsfcBkWx+GRjQVjhUyIuk6ZK/8mzEODY/MY71SL8/
mnpalIOyOpBs9hYi5osVic95CsJv+T1iTBoJ3NvWtqTQvkjamciJDADO9fiml7cAt9o+L2d8zUolwX8M
UXV3P2Ehsygxj/Qg5vETCaX/gI44PpUEczuSTiuOJvx+
+xGD8LaEovYr4gTppJJsri+VBPNjhBfdJel0e17CGqAH8LO2XoOF+lAZTOclPM8etn1N7rhaGUm/
JxYWTxNJy8slbQCcY7t33ugKqXRteiJp1onwdnyFUBMUFUE7I6knoSTYGLjI9uNF0dF+VBQfcxDz9YvV
zWlSc24O/
Jpo2LRmplA7NJXPeXHgZ8Db1U1m8vmdLf1ZzPYVmUIt8K0tTDfbj6V7MwWR+JyH2KBuSuwp7s0XZfMga
VpgDCFS6U0k0eYlVLK9iAqj8llPIkjai/DK/nO1Ki/t/
7oRVg4f2R5Y5vP6kQ4lZfsDSTMQhzNLEzmP2YlxautiiZgfSesQTcV7AqcQVWKDgM1tP50xtKZH0s+B0
4nDl0NsX5uu9wGG2z4qY3gTTUkwtwOVBOe+wFa2127jgXMM0SxicWIA/pXtt3PG3IxIGgxsb/
upNtefAfa03T9PZK2BpO2Iyesr4FrgFqKUs6hl25G0ERWhdhple1BK0JxIJG6OzBpgC1KZD2YHfgusSi
jTpgf6AicW3+X2IW0+FyQ8aHsSB799bb+v0uQvC5Xn40xgGuBo2yMUzVD+UxID9aFyGNyHaDp9RkoSdC
GaKf5HUm9i7zA0a7AtSuVZWIIQrLxo+0/
f895vK2EKP56kft0b6GL7sDavTQVMB6xAUWVOMkiaE7iDqLIYmK6JGLu+kbQVIa74OmeczUJl7tgcWBu
4z3bfCbxvJmBR2w+2e5CF71CZSyYj7JY2JXpYLGx7kbzRtQbpeTiC6MX2D9tnS3qOOKy8OW90E0en3AG
0CLXNzxTAZ6lkxwC23yB8iRYhHnCA37R/
iM1NWlhcARwmafrK9V6E2qP4azae4UQ584FE440jgYGSXk0ed4X2YXdC3TQAGJKejdeB44GzcwbWwnRO
f/
+BKHW+GNiPUBYsDZyYyqELDaLy+f4K+AehTnsf2AfoJ2nDklzOQ0W9vAWwv1NzOdtja8llSXsqmsgWfi
SVRP3GwAU1X2vb4zy+L0hvYPe0MSq0P0p//wZ4q5ZcVtAp/
byIpOOJw5jCj6T2eQLbE9YKp6XrnSvzxVTA0rZvKsnl/
KT1LIR90hs1dTLEPJKSoJMRyuZf5IqziTkQGAjcCvEMpQNKJK0HLFKSy/
mpJJenSYcsZwCXEfuPWSRtUfYcjUVhdfghcBLR2P1gSf2AGTp6chlKgrldqPimXEMs+I6RtJKkBSStT5
QiPpuUBqMJf+ZCHUn34AailO1JSbdL6gucC1xR/
Ewbi8LD7jTgadsXEUnOjYEdgEuJBGehwUiaDzgM+CdwH3FPtiOSaS+7NKnJRS15szpwvO3LU3laX6K50
KpEQ8xC46jNuwcQivGf2T6AUPrfCRwoae5s0bUolYTBZkRjuU+/
Z+MzBdGwpjARSFoE+Mr2m5XPvsqDhEKta/
tGVmjDT4k1bU096MoBwXDCSmaVXME1Gb8ErktVLLL9n8qBy/TAJuVwa5JjdaDmxf/
tOJYqir8mGputnq6VXMhEkhL3UxLilctqlRO2v6nsr78CfilpllxxFoJKXupISd1sf2n7EqIS4xLgVEL
4WGgAac4eB2D7bdvHEpa5UwFzStojWTJ1WMqg2o44mqEcTHh2XQ6cCfwFuBu4MpX0rEVYBxTqjO1Btlc
Cfgc8RDRBuSz9XmgAlQ3qAoSf41hJk6XJbKSjGdOJVX+
0QuOw/
ZrtnsTC+nxglvR3f2CCZbaFxpOUBF2AK4GfpLJb0kb2cSJZMArKZqhRpA2SiGYbL1auf277EMJjcxb4z
rhWaD8WAF5IP3eq3YNKsvljoiN3YeKYHHhZ0vJpXOqaFJu17/
w8wKy238kXYuuSxqla06yPa9dqr6cE2hjC97TYKk0Elc91dqKRXzUxU0sSvEqMTbO2f4SFtlTuz7PA/
JKmTNY+XdJcUXt9A6DW/LLM5/
VhZWBImjcm9Jk+B6xDWssW8iBpaknTp0Ox3V1pTGr7I6Ka9SmgNPBtEJXKu4slLZiuPUw8H4cRlZPL5I
tw4umSO4BWIiUQ3iRKd2YCliX804anwbgnoaZ9PmOYTUlSnm1JlO70s31b5pBahU6EOnNxYApJa9m+v/
qG4p/Z/tgeSfhh90lVFDsTiuZCPtYlleACs0q6jtgMrUSMWc9CeV4agcY3+FkYmBG4TNI+RGPF/
xBNzaa2PQC+m2QoNJbKZ30ncJykGfzdhry117chunEXJo5ngU+Ig/
ftXfHwTevUXRivDCzkYSzxXT9I0rbV8Sgl07oTSc8HMsXXNCialD1OJCTPTfs4A99U5uLlgccyhViYML
```

cCNwIbAjdUq1QlLUBYZOyeLpU1VX0YDXwlaT3bd6X5ojPJv5+ofBlre5xKU8V2p2aLQczhuxAHlEMlzU
hUJI1ONnCzA1+0WWcV6kRS+i8CfAD8zPautddsfyXpJKKp+Bvp/
eqIe46SYG4wGm98vzawE9EMYi1gPdu3p8VLbRP1aPpTqAMa31xxLeDPxPf9EMJf6EXgHsJUvfgvN4hKG
eGyhJfsHZLuIfyG7kjqj0I7kHzn1iUa0XxZu267n6RDCeVmIQNpAXGHpIWIjewvgd2AyYjNz3GK5lojX
JrS1J02SrUpiA7zfyeUzNMRi++DoTTOyshjxPNwY1qAPwN8bHuMpB0J9fLeOQNsBpL67K/AdZJGAlcD/
YjN6DHAOGI9VchEWtfeTHSgP0zSxbbfVTS9XJCYO+7riJvSSYk0L4+S9Ciwp6RbbQ+vvD4Z4dk/
wvbH2QItTIiBwEXEOPYskXDuTySWtwcu9Xgv//
KcTCTpWXla0hDgAEmv236FmC+QNC9hx3d9+iedKIn9dqXyPb8O+Iiw35uCyDsNBx6V9CShnv0YvvUJLh
ai9WUVwoqvC/CxpDWIw5nXHD7+Amay/
Rp03PFJHTTuDoekQYTv6U1EYnMlosRtR+B82x921FOKSZVKgvlq4FOiadY44uRofcL/
d4jt0uihwaQT0umIRPNq6e+e6eUlyuK88UhahlABfgaMIKx5HiASM2fZ7tB+T82IpOWIxmbbEKXpkxEn
3tfljKvZkTQ5sCahuFkeWIh4Zp4BngduKrY+7UdtbSRpHkLlPw/
wKrGGWoY4ADjS9jW5Ymw2FM2QDyDm6+WITU9f4CTbL/7Qvy20D6nK4ljCt/
FF4plYA7gLOM72kIzhNQ1p/
XoJsDlx2HIX0Zx9F+Iw8qwyJ0+aSFqWaIi5IjGPP0sknq9Oln1l311HJC1KfL49CUuMV4lmyb8gKgH+Z
Htk+dzzI2lf4AIiJ7IpIX6cDbgXOMf24KI0rz9JLFRLMk9J7Cs6EXuMtwgLy+lsr1fLY2ULdiIoCeYGU
tkUrQmca3tRSb2AR2zPJqkHoTKYP3OoTY2ks4F/2n5oAq912Ie3I5FKpaarJZIVHejnBOYvC/
P2Ifk2zk4kZ1ZLf1YgbABOtn1tvuham/R8LAwczXhFwTW2h1XeMxmxwX2yer0wcSgazhxK9D64H/5/
G5L0nnXTn02AzR3e2IV2QOHb/7WkFYmywmUIr7opgSeA/rafyhljs5B8SuciFE5TEp7Mowi1mW1/
njG8wgRIVXorA/MTivMnyqF9/
Ujr1c+J6qItiEPHOQjB0Fll7Jk0kbSk7YFtrk0BfJ0qi0uSs0FI2okQEvUibMeuB85IFgyFjPzQ9z5Vw
UxV5o/
GI2kLYCghvlue2APODLwLXGX70Y6coyoJ5nYgfYn2tb2upP2ADWxvImkr4HDby3bkL9GkSCW5Py2hHO8
OHE5sREf/8L8u1IOKPcxywM+IE7vFgJ+mgbOcjE4CSJoa+LwstNufSpXF1oTn6SBClbkqUcp5tO2/
lPmh/lTmiP2IipbdbL8jaWaixHwHosz2ZCf/60JeJH0OzOc2DeaUGl+W+WTiSJZtxxHPQ29CUXM/
cKHtR9N7yrw9CZAOJRciDo3nJ3z6X88bVXNQmRt6AnsR8/
EI4EKiQfh0tkcXu6RJF0lzESrzoUTV0V2EPdyIrIE1KZX93mrAW7ZfT16z49zGx7/
sNfJR2XOsTNj7bE4c2t8J/Nv2E+l95T41iP8jwT8z8GEzfPalG32DSYvAG4H3JZ0A/
Bq4UNJURNlOKelsAJWHc07iNOhz4A/
ABZJOkbSbpEWyBdgCVDah5wHTAHsQpcxfpsXfwYpmNIUGo+igvbKkgyX9WdLOklaRNL3tMc0wmXVw9iW
qLPa0fYDtFYgE5xqSeteSy2k+KdSH2mf5C8Lyopa0PAL4OeHZOA9wtqQZJXVSNHgqtCOSlpN0p6SDiCY
070zgbedQeopMFEnZ9xdCCXsUUd1yDKFmvjuV05YkfmaSwhzCR/
Yi4l4dQIxZSFpa0jSZwmsWanvjQ4AliQbIMxHPwyw1kUrNYiFPiIUJUXk+5gZuIBqWjiEOCgZI6i/
pzLT+LTmQ+lHbQ5xK2PRAqDEPl7RfSjZ3WD/
ZJqL2+V9EVCftRvQbWQnoJ+kbSTuU+9QY0kGMJS0qqY+k9yTdJOmwVIk0GVE51uEpCuZ2Ip0WHUeo0ro
Sfmm3EMqQ4r9cRyQtSJilj02/
T0moPBYjFB9zp79vtH1WtkCbmIoCZFngX7Z7SZodGGi7u6RuhA3AMq40nCvUlzYK2cOBDwkP5iWAYcB/
CMueY/JFWZD0EHBYzcanogZ5BvijU0fuMkfUH0mDiaqiYYpGNE8AO9u+OW1Aa/
fmgXIP2h9JiwMHAZsRpYSvEX0s/
mn7XknrEXPMDBnD7LBU5uqtiUP4LSegEN+fsJFZoSgA81K5X28Av7Z9n6ShwP62b5N0JXCe7Yczh9phq
XzGbwPr2h6Urj8BHGj74TIXTJpU1rxPAX8jrBm6EMnOI4EFCFXzvMAJtu8u97I+KGw/
B9qeKa2d7iQSmksSh5enl885Pwq7vf62l25zfRoi0TzI9tulWqn+KDVNlHQvsZa9jXg+fkLYLk0BHGH7
soxh1oWi+Ggwkqaw/aXtx4B1FebesxDJ/
Udr7yuDbn1Q+AddS3g0ImljYDDR5fk14KaU3FwIeCNboK3DbECtbHMz4KX08/
LEM1CSy+3DfsDltk+XdBXRxGEYoZIt5f/5uRK4UtG06RlglKTpiEYpj0GZIxpBWlA/
ABwm6XJgZ+DZlFyubVQXAF6Gcg9yYPt54NeSTiESA6OJcetahaXDi4TqvPDj6EQcNG4L3J1sYmrf/
Zo1z+WEsnmt9HMhEynxOTvwTUouT0Z4nN6X3rI2cHC2AJuA9BnPA3SqJZcTvUjrpTIXTJqkcUuE4vwZ2
18DXxMWJ7tKuo2wOtmRqKJ84XuqYgr/JZUE/
RrEfhvCfqG77SUVvRPOsH1atiAL1fs0HzBM0ka2b6+9bvszovl77feSXK4ztselHycHjrH9BnEIdlTaj
2xErGk7vB1ZSTDXmcrJ9yLA1sBcSV1wqe0Pbb8CvCJpakk9bL+ZN+LmIpWsbZ5OiOYiNkPvEZ/
5Y0QX25dcGjQ1lMri+2FgR0l7EQPnxanMfEeifK3QQDzet7cXcVIKUfq8qe2XJC0FlDEoA5W5YlbCJ35
WoltzHcLapzdwmu3POvpCY1IlfbZnAycDfyY2osek1/4jaV3gM9vvFpVTXmz/
vvLrNQCS5gO6Mf7gsvC/UxtXVgZ2h2+/
+52IYWpy26NSIrOoxCcNvgYeSer9scArtr9IlZJfl31FXVgdmEXSNYQSE8Jb9rOMMRX+OyYD+gJ/
lbS37TclTU5Usq5uuz/QP1UBlKZzE0llXfQIYet2CZHgPydd3xh4G8YrzNs/
ykKFJQjV7EWSriNELU8DL6cDmUIDqOz5ZgMGABsSViXAtwn+ayu/d+g9X7HIaBCSHiQW7m8Q3c7/
TJx8L0V4p00L/N72XdmCbHLSxugrSasQD/
JaROJGhJn9H3PG1yokm4yriEY0g4DhxCn3aWUj1HgUHdAPAS4gDlseI/
wahwEjgdltf54twBalohK8HTiJWJxvRHTefoOYLwamQ7OS3KwzigZOn9r+SNLcJHWa7VHp9UWBA4F3bB
9eNkbtj77bKPYA4Ejbr6bD4+naqAsLP5KkiH2LSNzfBNxq+9M27xkJbOzS8HKSQNJviQOBqYlS2wuBTY
FXbR+VM7aOTiURsCawDbAJUU0EcC5hm/Sw7bdyxVj4YST1As4CFiFsr0YTa6tHbP9W0kbA+bZ7/sB/U/
gfSJ/pXESF6t+JNe1MRB+qM23/s6yj8pPWtgsSa95ehH1MZ0JVe4TtARnDa3okbQVcTVSO3UTY5faz/
V7WwOpMSTA3gJRQ62t7rvT7jsAJwMdEV/rHCNXB5bbHZAu0yZHUF/
ir7Ucq16YllAnjbPfLFlwTUlmUzwL8lCg3HwYclBSAixBluA/bvvMH/

```
qtCHUiKDdn+MpV7fmN7hKLZ6BrEvelpe/V8UbY2qZTzPmBr2x/
mjqdVUDQ0u4awvngJeI4oof2idtiS1P1LEbYBb5Ykf/tT8as7jyhX30PSGsBvga2AfsDP2iZDC/
8b6XnYNP1ZC+hBzA+3A/
8gqlwG2C5NeSchJK1OVErOSCR2LgCuLyq0+pOekQ0JgdCqhP3bVrb7Zg2s8L2k9dXqxLjWhVA1P07YAp
0EfGX7oHwRdnwq+74liJ6zwI0AACAASURBVJzGUm1eX52w7TnZ9hdZgix8L5JmJCwzFiCUzSfb/
ihvVM1NsmmdB1icWG8tyfgk/x62b/
vef9yBKAnmOlJR2xwHzGF7l3R9e0LBvFhJKDeWVMa5DaGQfQCY2cnntzIRbgzcVya7+lL5/
p8NLExsTpcilAOjgP2JUpzTbRd7jAZR+Z7vS0xeh9h+RdI6RGltZ2BPws/
0EtvDsgXbolSelYUJS4Ynbf+1zXu6VPy6CnVE4W+9L6HimJdoRDOSSDa/
RPjGv1zKofNSeU4GEc217pR0P3AXcDGh2jynHBbXl6T+25BIzKxA2JDcb3vtrIEVgG83qMsTiX8Bo22P
zBtV85FsYjoRh/
PftHltBuBz28ViYRIiJZWXJvYgHxN7wVeqh8OSpiKaaX1d5viJozJHr0cIJfasXs8cXmECSFqBsIt5C3
jTyYO8iCjykObz+YmDmOttD22G56ckmOtIm81QF6LRxq1AH0L9cWpasEDYFpUPv84k9fhJxOe/
NHEP3iOSBkOJkoS3bc+WLcgmR9K7wGppkOxJfPYnE+qBXYiStV+X8sLGkU6lHwB+VStplvQF8Cmh1nyS
SPS/my/
KgqRfAmcC0xNlnLcCt9l+Lr1eFnwNJlW1rEx0z16SUKZ9QhyM3WT76ozhtTySOhPJ5P7AOOKgcs1kbfI
WsIntZ3LG2NFJ8/QywO2eQONdScsTScyh7R5cAfiOpdK6RAOtpYHFgL/
YPkxST9vD80bZXKRk5bces5K6JsuqZYGdbe+TNcDCt1Sejw2AExnfYHwsYTk2BHja9q0Zw2w6KnmPEwj
rvZNtX5g7rsJ3qTwf+wHbEXah3YhDyoGEsv/GIjhqLCkHuDAwNzAdMUYNazZ7DChN/
upK5bRhH8Ie4DygO6EwmELSAGCQ7Q8yhdgKvEh4NR4CTEUkDWYkTrJHEWUgQ7JF1+RI6k0cXA0FsD1ck
oGTkmJ8P0nDCa+nQp2pJCQ3Az6sJJeXB96wvYCkXxPN5N4mkpuFfNxENMJcDFgN+Amwi6IJxM9d6fBcq
B8pcdCJyB18Slgt9EuvzUoknDch5u6ixslI2hRdQpT/
vw78ISWXlwa6luTyxKFolHgUYV31paSpCVuMocTY9JjtJ3PGWPgOxxC2PTtKuhaolTMfJ+mGUh028Uia
wvaXbVSvqrxlB2CO9o+s8F/wO+Jg+OikMl85/VmfyHncWubz+lH5HD8lbMYOl/QbYp/
9KPC47YG54isEHu97fRjwW9vXSnoFuBz4JXE48AIwrAhb6k/
Fe3xb4PfAlISQZSwwXNFw9G7bd2cMs64UBXODkTQzUaa+K7AKkVgbBKxQyp/
ri6SViKZYX0haAHiXKH1egVCmzQO8Atxr+/lsgTYhFVuG/YCjCQ/
mEUTS7LCaL5ekOYl7NHO2YJuYiprgGqLZz+Hp+jrAcrZPSr/
vQDRs2i5juC2Jwht7K2KRMYhQkj+TlJrTEsqbZYE7bX9QFnuNJXkHHkrMF1/
aPiRzSIXvoaLCmR34I4DtAzKH1SGpzBVHEmujPW1/
LWlb4FTCo3xF4ANgexef6+wk9dObtudIv78DrGX7ZUkvALvbfjRrkB2clEg+hVi/
vkwkykZU92uSBgPH2L4qT5SF70PSnsDrnkCfF0lT2f68rKnqT7Id60woM5cl/
GUXAKa3vVrO2Fqdyv58EeAO23NL6g68ZHsmSWsR+am/ufj3N4TKemsocLTtKyU9Q/
RkW5hYg+1s+75mGZ+KgrnOVDZAywJjU6nzv9Mfkufmqo7GNeUUtU6kxOWjwJj00F4LPJhOTu9Of4oSrU
FUBsMbidKbi9Lv3YjTuSWITuc7EVYAhQZQ+W6/
TZST18ake4B7KqeomxN+2IV2orJo2AHYg1AHLg1cIWk9228TtgyjiY0t8J1nq1AnKou9nwBHAs8STTZW
BA5JBzLjbD+QM84CSOoBbAD0BrpKOt7225L+AEydN7qmYGtiw1PbWG4E9LF9aLJj2IMQSdycKb7CeKYH
HlE0Dn+GGKNeTh6Oc5bkcl2YDZiFaHS5FqEye0vSa4Td20hiLLopV4CF71LZd/cgDu6/
lPRaW0sfpwa+ZU3VEMYSPS0+tH0pgKSFCBuAYveWkcrn3pPx++/
liYpvgDGEbeUJ5T41hrTf6AbMaPvKdLmH7b1T1eSphOCoacankmBuHGcA/
1T4Mde+LFPYfgl4qSQ664vDz7eTpGWIEoR9gdMlfUQkcm4G7nHxqGsotkcQ9iSHpMF0CcLv6UYi6bkiY
c9QaCy3AP+Q9O+qWj8twmcgOmsfnSu4FkXEXLATkcA5F0DS3cC6RKK5zAvtQ63ceUfgGdsHSvo94617V
iEW4w+UBXf7U1HcdCesxmYkDol3Bw5O8/zkth/
LGWdHJm14OhFJywGVly4hPBmxfbekw4jkQSEztkdJ6kOsoTYDnpe0ClEheWPO2Do6lbm3G3As8Z2fj2h
UvRihMluHSEAPdWkON8lQKf/
vRHj1LwwMlvQmcCdwA6HcLGurOlKZp7ci9twG5pQ0Cvir7etr7y1rqEmC+4EPFT16hgOdJV1AHKbdl97
TmSROKtSduYHbJHUhFP413+UpCeFpU/kwd/q/
31L4X0gJnM7EguRC2/+xXetAPLWkyyX1KBNdfUkbJWw/
bfsg2wsCkxFe2J8RybTXJZ2dL8rWwvZo2w86ugrPB2wD/
AL4V97IWoJHgDuARyWdKWlNSdNK2pwofb7b9qC8IbYWlTF/KcYr/CEU/
6+nn8uc3A5UNqTdiA0pxNjUN/28HONV5OWetD+1z3w7oIvtVQmV+Zvp3s1NKNUKE0c3Irm8Ye1CUu1/
IqlT2ggtBzyUKb4C4/1/Je0MPE/4xXcnDsF+S9g4/DFbgM3FfcBgogpyy/
TzkcAfgCuJZlhHZouu8B0kba7wjcf2CNs72F6WKDk/FpiJEFyclt6v7/u/Cv8bKbk8HXASsa84mJiz/
0l4Ma+UM77Cd7H9he0nbH9k+wWir0Vvwjr0pPS2kptqHK8BFxJz9xtETupc4E8kNXlaczUFxYO5AUhal
Oh6voHtTyrXuxHdIrtlC64FkLQd8KztF9tc7w5MZ/uVPJEVCu2LpL2JxNmCxKT2CrHY/
kuznZZ2BJIH2gvERnUgsaj4h+0ZswbWokjakKg22p84+JqRUHAMBtazPbQomNufijLqAuAD24dLuiL9f
ICkPwPz2N4hc6gdlspnfAjRGGtXwg/
wo9r3XdKBwLa2l88YaoFv9w9DbHevXJuLyPO8mS+y5kFSV+Ayom/LAOBLYG3iEHggoRJ/
kGiIWVR+mVH0s7ikNg+kaouXgKfbVqtKmsb2Z6VKrD5UbMa2IhrvrlJ5bQriEGYm23tkC7LwLelgpRew
BvBysVPKT8pVHQPcA5xqe0gzjU8lwdwAJE1LnFJ8TnTrHJMWhzsAv7C9esULtVBHJE1PLAInI3yFBhEL
wodsv58ztkIhB+mZqG1KOxHlnU0xgXU0UmnaZsC8wCKEemBe4B9E4vmhoixvP9IG9VBgVeJ+3EQ0qLnF
9rE5YyuApPWBA4nqlyeJpP+bkh4HTrF9bdYAOzgVtcxphCfjE4RyfwbgJ0TJ8zm2+074fyg0moq/
7BLAfsB+tr+c0HvyRNh8SNqYaFT9OHA14c//c8KvfPFyIDxpkObvhWwPlDQHcDYxZnUG3ifGsmeA52x/
kC/S5kXSFoS6f7dkAVq7vhNxOLlBGZ/yIamLo+fXXkQO6j+EkGJa4tDsUeCftl//
```

```
gf+mUAcUPdgWJvJSr9luauuxkmCuE21PHSStQCijZiA8gOcjfG1Osn1PGXAbQ1pwLEr4pC1EPMzLA1MR
Tf92zRheoVBoUSTNA7yfDhy7ML7EeRHCUmm+9PMZtouVTzsiaW1gJeAj4A3bt2YOqZCQdBXhGT8ncCJh
j9EJ2KVtoq3w35NUZpsRB1ujgF8SSbVuwFCiudk5jkbVhcwkNfkRwO3ACcCL5aC4vlSSMZMD6xMWe+8B
x6Y+L4VJlGSTMQuh0lyAWE/NBswBPGn7kIzhNTWSLiH22ucQFjMbAHsC59nuU/
Id+agozQcTjeRuJQR4CxNr3s2Bs21f2Ezq2UmFygHxVoSd1UKEbc+nRHL/
DuBG229kDLMhlARzHZA0N7C+7YtTGcLUqRRnamLBvjpxknqd7XdyxtpKpA3UTMCviM7ox9vulzeqQqHQ
aqSx6BrCeuHF9PfrRMftcen12YhE80DbbxRrhsYiaWbCFmBBwrO8b1E55aeS5DkYGGH7aknrEJuhuYCR
wAmlRP3HUbHG+DmhiN3F9pD02hJEEn8ssH/
bMvNC+yNpWdsDJP2C2EssQSQIPiP2Fc8TNktjMobZNEjqWlOWSVqQaKj4U6J59W3NrjrrSFQTYpIeBva
1/Wz6fSriAH8hYKTt/iWB1hiSVc8uhP/yPMBTwKXAVbY/
zxhagW+FdzcCm9v+qnJ9MqLJ78e2v84VXzNTSTA/Dtxp+6h0fV3C438nwubnt802PpUEcx1IPo4/
sX1wKuncHbgZeKqUO7cvkhYH3iUSN/+pXL+JSDD3/95/
XCgUCg0gNULZl0hmzks00hhJJJoHEU2a3rA9KluQLUBFzTEfUfr8ITCCSNz0Bk63fXDOGFudSgL0SUJB
20fRxLdrdXNU+HFUNjw3Ao/Z/ku6/
ntCYfMEMDXwNnEAQznoal8qz8BywNG2N6mJV4iDyPmA+Ynk2SLA1rZH54u44yNpJsIGYzZgFaKa5R1Cy
TwzsIPtO/NFWGhLEnFtQ6hmh9ieovJa7RlaG3ikzB2NISUppyGekfeJypc5bY/
IGlihut6dHziPGNP+TNgzfJY3utZC0tnEevalCbxWW5M1laioJJgnkrZfCEWDvz8R5QfdiE6RjwNPAw/
YHpkl0BZAUi+gD9GcYzDR0OwDQu3RD5ijqDwKhUJOkkf/yoQic0lgdmJR/
hFws+2rM4bXtFTn6tTYbEXbW1ZeX5douHGk7XsyhVlISDqIsBW7sMzb9UfS68AWyb+0G+FVegpwCbA04
cv8R9sPZAyzJakkBk4nGmX9qm2ZuaTORIXejLZfzhZsB6eSiDyA+P73I8qWOwHPEd7vUxANRk
uZ/
ySEpGWBqwgV5lTA+cBDwN22v0jPyFjbnTKG2XRUxqf1geOAKYFXib5TfWzflTXAwndI1S97Eb7LrwBvE
hWUw4H+Lg3fG0LlOZkXuB54C9inVSrDuvzfbyn8EGlhUl349bK9LYCk3sCGRBfiPYjykZHNdkqRi8rDu
wzwBZFMvowoM/
8Z0ejhc8Lr9MqySS0UCjlI6rNOxJTxKbGJ7Zdem5VIOG8CKF1rqlKpSYRfSPqUaGwyA9H89Vts3y1pF2
Ad4J5yD/KRelgcQTwzk0m6nzis/
6j4Lk88kqYkbGEWJ56HPQjLhQtTqeyjiqZZxdItL2sA5wK0TW6m399Lfwo/
kspe7DoiAbM1Yav3VLo+DngJmFLSmLJ3m6R4jhjDriY8+qcDTgKmkvQe8DHQF0ojzDrTiajC+wdwMFGB
NyOwIvBvSXvbvjJjfC2PpGnTXgPb/5TUj6h2WZ6ofFmDqNY4jDKHNITK/qEbYY24GHCLpDeIsas/cI/
tTzKF2FCKgrkOSJrc9lfJh+gR23N/z/vKhrWOVBLMbxEqwAFE0qAf8TDPRHg2DgGGls+
+UCjkJvmcHkpY+Xzp0nym4SSP67uIDvPvE0qCzQi/
2XvStS+AewkrpdvLfJ2HdBjTFdgYWIEoUZ+FSDAPJvxm78sXYXOQVE1XEuPQR4Sv9dXptQWItewsGUNs
aVLp+edE8ux5Qnn2OjF2fViSZY0jKfpXJIQq6xBWSm8BB9p+NWdshf+f5DEr4CuiCWwvokKsE/
Bv2y+X+XzikTR1TaiVnpF7bC/
b5j3bEgc0m5U+CXmQtCpwme0FUsXkgrafavOeuYFlgbuKXUb9ScLH52rPQPKEn5VwN1iMsEtcATjN9qX
NKDwtCeaJoFJatTmwKmFu38P2KkqNatL7FgYOsr1zxnCbEkldCdXyCkSC+UtiQTg7UfJ5G1HidluzPby
FQqFjUDkM+wlwJPAs4Vm3YloErgOMK+XojSMlLpcE1gSWITrNiygT/
CZdu5mYq8tcMQkhaTaiUe92wOW2r8gcUlOQquyWJ/
zfH07X5iFs3rra3jFfdK1JZV+xGVHyfwahNpue6Dz/GtHcbygwfEKejoX6keaN2YF1iUawTak266ik/
ha7AjfZfiV3PM2MpGOJxrA3AS8ASxENyu6tvGcd4O+2e5SkfvtTmT/
mtj1C0lbAvwmV8pPEGref7WE542xmUlXq+ba3TP1DNiFZk9TmD0kzEAdhr9geXRLMhQkiaSdgNeAXxOL
vQWAUsQh8hmicMrXtX5cyncYgaWOi0/
PjhOJjZqJhx1bAYrZnzhheoVBoYSpNHPoQZf4HKppqrWj755KOAHra3rUZFxq5mdBnKmkawppkVUJNsB
xwne1Dyj3Ii6IJ4x8JBedQ4PayIWofJC1NqMbvsv107nhajcpc8XfgddtHp+s9ieZzqxIN/
roT9+iP2YItFDJReU52Jhowrq1o1LgTcAixDzzIpblf3VA0i1uJUPcvASxK9Di6irAimSVdf8r2NSXfk
Zfq5y9pQyJHtR5xn/4D7GX77xlDbEpSxeRSth9PSuariATzCKK6/iUiVziymcenkmCuI5LOJHy7eqU/
0xKD74zAH2w/VE706ktNKZ5KpNYHdiZO6o61/
Vbe6AqFQmE8kvoCVydPtCeA021fJelG4CHbfyuL8sZR2ZR+609XeW0uYBrbL5V5uv2p3Jv1gIMIy5LPC
ZurnsCZwCnlvjSWpNikHLDkRdJgYBfbD09oTkgbV9kekCfCQiEflfniFuBh23+RdBJRmXQP0fvo1FplR
qF+pP32zISyf1VgdUJk142wKVna9mv5Iiz8EMneZDvgCdtPlT1HY6iMUQsRhzIrEeNTF8Lb/
yrbl+SMsZGUBPNEUilH6ArMWVPZJL+VOYnN0QsuXTobhqSutsemnxcEdiPUzIcQ1hhjc8ZXKBQK8K2K4
Axgf+BfxOFjZ8Jbdj3bQ4t6tnGkcrXNiI7a8wP3Aw8Q5bWjMobW8lQW4zcAz9s+svLaLkRF0u5ukQ7ch
dZG0vy2h1Z+75x+dDlkKRQCSXcCfyNsxx4Efmv7LkmPAWfavrocGDceSTMSlhlL2z4ldzytTCUvNS2wF
uElb+BhouplWMbwWp5UJbYR0N920zYULwnmiaSyKdqX8Eg70/
anqcRzfqJRSjFQbwCpHOrnRCfUVYhGNe8QSuaZibKpO/
NFWCgUCuNJyo9DCdXHIoSX3bLALbaPzRlbM1PxwN6IUMKeT/
gvn0w0b5oTeNH2YhnDLACSHgGOtH1P+r127x4H/
mr7urwRFgrtQ1KTTzsh399m3ZQWCv8LkrYALiV6Kdxq+0/
p+ihgAdvv54yvmajMxfMRquXewBhgIDDQ9sisARaA7+SlDgV2JHyXIVS0sxLJ5itsH58rxlYgCU83BPY
l7DHuIBpjjs4aWDtREswTSWXAfY4ox+kjaR9CJTU7cJHts4oqrX5UTucOAE4B+hEPbifgOcLIfgrgg1L
2USgUJjUkrU2US31ENNi6NXNITU1lwX01MMT2UZKOJw4ijyNU5Vfb/
ncpF8yLpN2Aw4G9gSdtvy9pamKBvnjZxBaancp4tSGwD7FJfRW4m0gWPGX7w5wxFgqTCpIWBmZONpRTE
f79G9lesey960cl33E3UX33AlHqPxMwlvD1Pd72cxnDbHkq9+lBIi/
VN6mZpwPmIAQuw23fUNa79acyf+9ICIpuBOYj1OTTE2vZK2wfky/
```

KxtMldwAdnfQQTwb0sN0nXT4c2JMYeI+VdHMpSagflcXCdYTP9dbArwj/a4jP/
SVgSkljyuKiUCjkRtLMRLfzBYFHgAttf5A3qtagsoDuBVyTfl4POMH2m8l2dkyb9xbycBWhjNoN2ELSv
MRhfZ+SXC60CDVl8qXAecSeYjmiSdPVQDdJ29i+PlN8hcIkgaTZgCmBydJcMQy4Fqgd2otQbBYmgkrSb
DpgWdszpNzHXMR83RtYjOiBVMhIpbLlMqBruvYp8CnwlqRnicOAst5tDLXxZm3gUtsn115IvV62IwSR3
/YRa/8QG09RME8ElVOiuYGLgceIpn7L2V4vdZJ82/YMWQNtAZJp/YqE19A6wIdE6fOBtl/
NGVuhUGhN2pQUXk2MSyMYX154uu2Dc8bYKkjqQpQLjkgejbcRlS//
Ijo7L2/71aJ4aj8qz0cfQvk0NF3vTFSBLQG8C7wB9Cv9FAqtQkrk3Gx7zQm8tijwpu2P2z+yQmHSQNLe
wLHA88Ao4BPgbNtP/eA/
LPxPSFqfSB7fTwgTdwf+aPujNu+b2vaY9o+wUKNS4T01sCVwKnAlkZ961vaQrAG2EJK2AXrb/
kvuWHJQEswTSeVUb2NgP2IjdL7tpyUdBmxoe41ShtB+JN+62YF1gb4T8q8rFAqFRlJNVEo6BFjR9paV1
9cFjqHiN1toPJImt/2VpM2Awwglx/u2t8gcWkuSfMm/AKa0/VW69ivbV+SNrFBofyqHLisARwJ/
t31t7rgKhUmByvOxGiHs+jlRfTQvcSi5PbCW7eczhtlUSNof+C2hzBxJ7K+fIezF3iGUsV+Vg/
n81BSxkk4GNgWGAO8DMxD3bxxwg+1/Zgyz6Unq/
jUIFfl9RFXFfa3Uk60kmH8kki4g1LETPK2T1JvwTnvM9r9KgrlQKBRaB0m/
JBbeA4nDx3dsn9bmPVcDr9s+rDRtaixtlclJIbgisUG93/
aV5R60HxWlzWbA0baXSdcXAO60PW9RkxdaleTfeCxhA3cf8DDwAKFCK2NUoSWpiLr+DMxle8c2r58I/
KfW7K9QPyT1ADYGVgBWISq2XyXUsc8RVUYtk0CbFKmsq14AdrHdP1XZz09YxC0DXG+7X1nv1p/
K+HQgse97AJicsEbsRlSxXmH7zIxhtgvFg/lHIGkmosx2jKQZgNsI/
99HgMdtD7P9SurgOQ6Kz02hUCi0CskeaR9CMfA+Ydezv6TPgXvStS8I/
7rLc8XZCtQW3G0Tlamy5S5JZxH3pNC+dCLU41sQG9QamxKb1ep7CoVW4wFgK2ABYGlgTaLkeeak8B+QM
7hCIQeVvfRgYFFJ07RJas5OqGq/Tfa0d4zNRk0Va/
tN4ML0B0nLABsRY9OewML5oixA9KhK6tnHiMNJbI8grPnuSRVj36TrJblcf6r+y0fZvlzS9EQjzLnT9U
+h+cenomD+kUia0vYXkmYFdiK+OAsQD/
RHxAbpdtsP5IuyUCgUCjlIVj1LEovvZYj5QcBwYoG3DHAzcFBRadaXiopjPmJB9wRR2vmJ7a8r75sM+J
zoQD86T7StjaSB6cdriET/
mcDlts9t9gV4ofBD1PYZ6eduhCf5YsBlteuFQisiaRrgbqA70fD9cWA+os/
C9rYHlgqY+lBRZW5FrGevBvq3qQirvaeoYjOQ9hud0j1YCbiAyEcdDbxMrH/
fr9mQFRqLpPOBu2xf1+b6FMC4Zm3sV6UomH8klcXde7ZPkjQtMAdRhrAwsCrRcOCBMuAWCoVC61DZ2Dy
b/
tQ2RCsTc8OChDpzXEqElo1QHal8lt2BgwhVwZtAf0mPA0NtvwxsQFiXjC7zdDZ+SXhnrgvsRqyj3pPUC
RgoaQRRMVaej0JTU/GXnRXYBlhdUi/CR/Ni2/
dJeriMU4VWJq2XPgNWkrQzYXV1BFEVtpvtgfCddUBh4qiNN5MRZf/bA2MkvUyIJO6x/
RIUVWwOKvuH2mH8Z8R96UU0ZPwceB14Q9LtpfqlsSRLkqWAbSUtDzxEWFu9ZfvLvNG1H0XBPJFI2oto7
PcS8K7tz9JJ0izAF7Y/LcmDQqFQaD0qqo5pbX/a5rW5gGlsv1SSm40lqQYWAzYHtiaS+/
2BlYB7be9VKwPNGGbLI6krcU+2IZIGPYBhtlfLGlih0A5U5ouLiO/
+Y8TeYiXCJuO4lGQue4pCSyNpfsLbtBux1y5JswYj6VrisGsAkXReF9gbeJtIYJ5t+5p8EbYukvYjPn+
3ud6D8MtekVCfH2L77jKH1Jfq5ympJ/BTwgJxbmBqIvk/
BrjZLdK0tySYJwJJ3YmSznmIQfcJYBBhjzEMeKuUdxYKhUJrklSYmwF7EdUt9xPemjfZHpUxtJYmeaKt
AGwLnGf7yZLkz0NFtTmj7Y/avNYdWMT2/
XmiKxTaH0mjge62v06ClRmAPwE9gV3L3FFoRSrWVzsTic15ib33SOAq29eVxFl9qXzmKxOf8by164SQ7
g/Ax0QCbVtgT9uPfe9/WKg7yQruTNubSJoHuAK4lEhmftDmvWWd2wAk/Qa4pmpdJakzcVC8ALH/
Ww640faNrXAfSoL5R1BRGWzO+IY0Y4DViGTCZ0QS4QngnDLZFQqFQutQSZptRHjKnk/
4L59MNPybE3jR9mIZwywUslLZvG4P/
JwoK3wV6AvcYPuN6vsyhlootAvJEuM6YAfbL7RRRo0E5irClUKrImkO4Eng17bvkbQs0QxzZ2AL2/2zB
thkVOboLYhDru2BIZUxqdbMbM3kO9vF9q4ZQ25pJM0GHA+sQVhkjAT6EWuqWwjnmLKWqiOSZiaSy+uma
slziRzg08Crtj9P75saGOtKH5hmplPuADo4RwMP2T7T9iW2fwNsSCScRwOHA3/
PGF+hUCgU2h+lv39NqD5OIVRoFxE+zDcQ80ftlLtQaDnSxrU7cDFwB7AHUX67KzBY0leSupcNUaEVSKr
A14DbgRMl9UzPyKySDgWGl+RyoRVJ1WAQe+zBKbncyfYA24cDpwP75ouwOan1CAFuJfqJHAusJqm7pA2
Ag4Gn0ttHAtPlibS1qewjdiCslOYnquuPIJr99QVOq9zPQp2w/
YHtddOvUxFe5b8HbgRulfQXSVsSVODAYAAAIABJREFUzcRbIrkMJcH8o6gs8N4ExrV57UlgGuAkwjdtk
XTCWigUCoUWoDJH9CJOsQHWA263/
Wb6fUyb9xYKLUNlk7Mi0W37vPT3QbaXAHoDW9t+L1uQhUI74gRR4twFeF3Se8BthKXPMTnjKxRyUSknf
x34RtIybUrMOxOezOXQvg5I6ippJvh2XBoL/6+9e4+3dCwfP/
655mAYcj6GwSBn5ZikLyY0zhKhdE5Fx2+pJIVEiUr9lPLNKZGUHKIyKoehUJGZwTAxjmMYh3GKwbh+f9
z34jEljL32M7PX5/167dde+3nW3q9r1pp1uu7rvi6OAp6iJJuvAg6m9Ig/
ut7m7wLOaynkntb4HPEVnv9scUdmnpSZuwNDKUWP8HwBjPpARAyKiCH1x22B/
ep72O0oif3VgRMoizH0SoJ/yEtfRf/FCcAZNYE8BpgIrEZ5I/
hgZj4RESsBU1uMUZLUz+objh9TJjgD3A+MqFvYtgS+UK/n9n/1nMb/
+cuATSNircy8vnF+CqUiShqwImIV4OjM3KX+PDQzbwC2rVtv1weWovTTnN5iqNKc4ErKe6ljImIMpdBr
BcoC/nfaDGyAeQ+wBvD5iFiC0hP+
+nqciFgLGJSZ4+vPqwKnA79uKd6e1Whj8hpKG77XAtNqIrOz2DIzMx+DFyzWqA/
Udoid97MnUHZZXJKZEyl5we/BczNFoCT4B/
xnPnswz6bGA3prYA9gccqU50eBQzLzV3X7yAmZuVybsUqS2hERwzJzRkTsBBxIGYYyrZNQkHpZROwOfB
2YD/

```
gT8Bfg8mayWRqo6oLjDpn5k4jYkbKt9ixKm4wxjR0vkiiLMMAnKYsv8wEjgMOA802e9Y2IGAEskpnXRc
ShlPYjYyktrK4ArqPkO2bWmVQWSrSkMfPlAOAIyn30ocwc13JoPaX2Xz4SODMzr2gc78nHhgnmV6GuFj
0OrEXpcXMbpYH3tLrityfwWGae1F6UkqT+NuubiohYkNIO4J2U1e3TemGSsDSrxgL9mpSBTd8B7qG0xV
gZWBR4Fhidjanc0kBVk2Yzgd0oFVCjgaUpQy+vovTP/
Ht7EUrtqO0XVgAWA6Y2hr8OBajtG9QFjdfqLYDN69dI4EHKa/
b+mXljiyGqqu+ndqXskNwMeJKyQ+wi4CedYXPqW40E/
3bA8ZS84CeBGyiPkydMMOtlqcMGtqdMVH0dpeJmDKVa+dFZrjskM5/5978iSRpoXmq1OiImUna5nGGCW
b2o8YZ8H2DjzNynJgsWprQDGAkslJmnthqo1GX/7fWitsjYAdgP+H8+HtRLGq8T+wMfB56gJDU/
RVmM3B4YD5xq+5i+9RLPSwsAW1AWww7MzCm9WqU5p4qIhShzwLantDV5a2Ze4/3UPbVtzHuADYBVgWmU
avJxlN0Vd7QYXr8zwfwKzLJK8Q3g+5TKgtGUqee3Ajs6tEmSekejymMkMAq4mtI/
9pHm1OCImIfSk3lxPxCpVzUeL/
tQPqh+NTNvmeU6Lr6oZ9Rk8jeB7wI3UWbkZGbOaDUwqUV159etlIrMmZTZFasAz1B2DW8GfDkz7f3bhx
qv0fMDuwPvo8yT+l1m/rTd6NTRyEv9D/AhSsL/
7rbj6nV1QOZWlM+DOwPvycyLeinBP6jtAOYyncmP7wZ+lZknZOaEzDwaeBPl9tyjtegkSf2u8YZhSeDz
wJnAGcDBEbFjRKxez7+NssVzet0JI/
WcxuNlc8rU7WMiYv+I2Dki1oiI4SaX1Qvq9n8orZNWqr3HlwaOBp6IiNMjYpHWApRaUAeUQanAvCkzJ2
bmJOAkYKN6fP/
6877tRDmgdd6fHkW5rU8G3kJpYUVEfDgi1mgnNDV07qf3Aw83k8sRMW9EbF0LX9QFneepiFglIk7s3Na
Z+UBm/iIzP5qZSwN/
qMd7IrkMJphfkUZl8j3A8FnOPUTpGfgaeK6NhiSpR2TmlZm5GvAG4ADKa8KRwDkRcQolafCbenVfI9Sz
6nukH1O2Pk+ibOf8EPAV4Gsthia1YSfKcD8o7ffmA7YGFqBUaUq9pPP+6B3AzY3jGwPnZuZTmfkw5bVj
QfBzd19q5Dt2BQ7NzFOA+yiD/qBUNK8EL1gMUP/r3E/
rAqdDWbSMiMGZ+SSl4n+retz7qe91nnP2BhbLzFs7JyJiaERsHhHr9lJiuWNI2wHMpU4HzoqIp4FLgUe
AZYA1KL3SAHruP5MkCeobu7/Vr6/
UfmgbU94MnlivZoWmelatUO58WP15RAynLMyMAp560V+UBpBGIudxYJ6IGE1JKH8kM/
8cEYdTksxSz2g8Lp4GNoqIMcA1lB6n32hcdUfKEDN4fpex+kBELEVpQ3JTRAwDRmTmH+rpNSgtQnuqKn
NOU9uYDKEMS34fcOUsbVrXAj7RSnC9ofN/
f1PgFHhuoWtwZj4dEe8B7gbG9VJ7DLAH82yLiFGUKZHDKS+AqwIHZ+YZrQYmSZI0B4uIZSntZIYDg4Fr
geOcYaFeFBHrUXa4PAn8ITO/Wxdd7gFWsGe/
elFtLzaC8hl7bWA9StXgfcCfKbtdNsrMa3stgdNNNUkWlFlTy1LmiuySmRtHxC7A1zNzbWclzBkiYn1K
u5hLgNMoBaQfoSwKjGoxtJ4QEQdSCiQ+lZlTG8dvAD6TmWN67bFigvkVar6A1UmqGwP3UnpC3ZeZv20z
PkmSpDlVRMwLXEDZ/XVDPbwmZXDTxzPzvrZik/
pDLVK5MzMnRcSwzJwREasA82Xm+IhYGPgY8MbMfHu70Ur9rzHAbBlK+8kHgNdS+gCvBrweWDEzN20xzA
GtDh89GHgrZXD1o8Aw4IeZeX5txeCicIsarS+2Ag6k5KOmAb8ETs7MG3otudnfImJV4HjgL5TOBvMDoy
nva9/SiwtfJphfppdaGY2ImykVzD93FVWSJOl5jYTBO4AvZeaGjXPrAccAJ9Z+j9KAFRG/
oFQBjo+IfSjtk/
5KSQw8mpmPR8RywLDMvKXNWKU2RcR5wKWZ+e3GsaHACsCgzLzZz919o7ZbeDdwEzA5M+
+tSebNgA0oO42u8zlpzhERbwZuzsxpjWPDgRkm//tPvR8OoPQmn0JpjfHtzJzQi89PJpj/i85/
iDoVchRli8gU4JHMfKpxvSGUbW2Lu41NkiTphRrvqb4IbJCZ76zHO4nnTwBvtWJTA11EDMnMZ+rlUylV
mTOAu4AbKZX9twATeu2DqQQQET8HzqYsPG6SmXfU453Xi08DF2XmDf/
t7+jli4h1gR9QkmMPAbcD1wHXA3ebsJwzNN5LrQacR8lRTQM2BHYHxmXmSW3GONA17oNtgNsz86Z6fBi
lPXZPzxIxwfwyRMQmlObdSXnzdxVwJTApMydGxI6U3oHLuQ1BkiTpeRExuF58lrLN+TzgfOBHmXlPHYR
5LnB2Zn6vpTClVkTEa4A3USoFXw8sSSlmeVurgUktqI+HY4EtgOWBOynD/
H5J6VE+MyKeBUZm5m1txTnQ1N7LbwDWoTwPjaAMGb0HuL9+vzgzr20tSNFpTRIRBwEbZuYudUDsIZSFg
cHA4Zl5aZtxDnQRsSAlH7gTcCvlMfMZysLMtzLzsRbDa5UJ5leg9g1cG9gZeAdl0MBVwCbAnzJz32ZVg
iRJkp5XP8TuCXwKmIcy6C8oA2q+nJn3txed1H2NBMGOlGTAmMz8V+P8a4GVM3Nsa0FKLYuInYAPUwaYf
YjyeXtRYDJwbWbu1ovbz/tT3fr/IUp17NqUQWbHeru3p1HFfyYl//
SjiPg1MD4zD46IkyhVtYdY+Nj3Gq/f7wI+l5kb1DZvP6TsQFoK+EEvz2Uzwfwq1IqbjYG9KBXMf/
WBLEmSVETEvsAulOnmF2XmPfX4vMCmwOL1quf0+rZC9YZGguAm4IDMPLtxbHFgkcyc1HacUlsiYmhmPv
0fjq9KaSlzfWbe6efuvlUXt9anDFbchTLUbyiwImUR+IjMvNvbvX0R8X5gX8qA5GnAZzPz1oiYQJlz8R
vvp77XeK0+BpheE/k/AoZn5nsj4hvAMpn5/l69/
U0wS5IkqSsiYivKQvwalO3OdwNjKVueL8vMJ1sMT2pF3V47OTMXqz8HpZJ/
KPAb4IOZeVeLIUqtiojVgdHAYsB44PLMnNJuVANLo5fslsBXKW0wFgYWAs4BHgHGZuYNL
5b0V/
+IiEUy86Fm0jIi3k5pZXJRvY82Bc4CRnhf9b1ZbvstgOMoj5HbKEP9ro6IC4ELMvP7nWrn1gJuiQlmSZ
IkdU0dfDI/
8E7ga5QedQtQqqSuBSZSKqMeai1IqR9FxOsoA7X2a1YrR8QIypCmhVsLTmpJI+G5FiXBeSdwM7AK5TXj
AWAcpZ1Sz1UG9rXG7f1TYG/gcWA/4LRZb1/
bYrQrIr4JTMzMkyNifeDRWV475gG2ApbOzBN7tXq2WxqtMZpJ5m2ALYHTM3N8RLwJ+DWwUWbe1auPGRP
MkiRJ6qqaMDgN+CBwH2Xr7VbA0ZRJ9W/
LzMfbi1DqP7Vi+SeUoVoHUoYEDQfeDaySmbu2GJ7UikYS5zBghbrlfFlgaUql5huAQZn5lV5N3nRDRGw
GLAjsSGmNsRSl1/
VvgV9m5mUthicgIt4CTKhVzMdSEpu3A9cDl1N2hLlI3yUR8X3gmcz8bEQsD0xpVidHxBBK69z1MvMHbc
U5JzDBLEmSpK5oJAw+D7xp1sRZRHya8n70mHYilNpR+5AfDqxH6aH5JmAMcKQ9mNWLGhW1XwWmZeZxs5
```

```
xfABicmQ+3E+HAExGrUdpWLd2ozFwV2AbYGVgXWBJY0iG87auDklcBVqLcN2tTFl/
mAZ4E9szMae1FODBFxDKU5567IuI0Suu3CcDvgDMz8+
+tBjgHMcEsSZKkroqIXYGDgC9m5kWN49+hDEf5WGvBSS2IiAUz85GIWBl4M3AFcLd9ydXLanX/
ccD2lJZKlwP3ZuaDrQY2QNXKy18Ct2bm52ZtrVDbMXw6M99n1Xg7Zr3dI2Jp4KHMnBER8wGvo8y5WCkz
v9FWnL0kIt4AbEd5nloPeJrS0meHzLytxdBaZ4JZkiRJXRcR3wLeClxJqZjaBtgE2Cczr2gzNqnbGtPn
lwN2onwwXZ/ST/bEiJgnM59qN0qpXRGxFHAUsASwCDAVuKl+TcrMsS2GNyDV3rEnAO/KzH/
UY4sD+wLvBS7MzE/Y17ddEfEpoLML7BlgEqWFyZ/
q+WGZOaOt+HpB3Xm0UGbe2zg2jPJedkfgoF5fJDbBLEmSpK6rlTbbUvrMLg/
8jTIc5fJWA5P6QaNdzEmU5NmRwBHA2Mw8MCI+B1ycmde0Gqg0h6jDMLektI95A/
CHzNzfStq+01j4+jFlyN8XgA8A/
ws8SqkiH5OZT5tgbk9tG7MncCHwEDAUWIfSKuPozDy1xfB6QkR8ENgBWIsyuPpK4GTK48PF4coEsyRJk
vpco5/
mMOC1wGDg4Vn7A5osUC+JiPuBFTPzsYj4J7B3Zl4ZEZcCh3aq0aReFBHzUHrLLkAZ6NepzlyAUjl4t4n
OV6fx2rwSsDVwGbAMcAalr+wI4KjMPL7FMFVFxBqU+
+YTzQr+iBhJGZz8cWDdzLyzpRAHrMZj5X+B9wHjKI+X4ZQdeWsC38vMY1sMc45iglmSJEl9rlEZdTjwY
eAx4GbgNuA64C7gL5n5QHtRSv0nIpYFfgW8h/J4mJCZi9dz04FlM/
PxFkOUWhMRw4GvAO8HrqdUZ76e0irj0cyc3l50A09EbEKpwEzgHkqybDLwhcwcGxHz9vp2/
zY13kMdQkkg71oX7J/
JzJn1OgH8Brhg1qGY6hsRsQjwF+CTnRki9XZfijLs7yhgc1u9FUPaDkCSJEkDT/
1gtDBly+3qlArmNwKbAu+gTKbfrr0IpX53DyUZsB9wC/
BneK635vUml9WLOu1jgN2At2bmMhHxNuC4Wum/
A7AF4DDYPpSZVwKr176ya1Kqmd8LHBsRtwATIuKkzJzcZpxie+BogGaP5U7P5YiYAKxcj1nd30cau+t2
AqZn5kWd27wenwp8NyLWAv6HMqi355lgliRJUre8FjgtM2+pP98MnFqrP9bJzLvbC03qX3XRZSxwLPAZ
YHpEnAnMpFRuSr2os6X6rcBv6+XtgEvq5RHAYvCCZLT6SK1SvqZ+HVkXhjcC9qHc7pNtZdX/
GonilYG1I2IGpcL8lsx8rJFsHgV8r40YB7hBlNfmXYHT4PkEf0QMBobWx84EypA/E/
yYYJYkSVIfi4ghmfkMsBkwIiL2ysyfd87XD6rjWgtQ6kcRsTRwOKUKczzwa+BcYArwIPD7zHyktQCllt
TEZSch8wdgm3p5C56vWN4WOL2fQ+tZtRXJRfWrc8zkcgtqe4afASMpgy4fB6ZGxG3ATcBEYDXqwkyvJz
f7UmMhayNgcEQ8CPwtM2+q5zrnNwf+Xi9HP4c5x7EHsyRJkroiIo6nbC9cjJJQPg84KzMntBqY1GWN4U
DrAAcDywGnAitStqNvxPN9He9oLVBpDhERr6UsvEwCdgf2AFYBdgB2zcz7WwxPak1ELAisSmkzthal/
28CiwNLZuZaVs/2vdrzeh9Kgn95yuv4Y5T3s1cAFwD/
BLbNzAlW+ptgliRJUhdFxFBgXUqSYLt6eRiwWGY+1GZsUrc0BjSdADwKHJ2ZdzXOj6QknK/PzI/
4wVS9JiLeCNyZmVMaxxYADgTWo1T3LwJ83D7A6kWd3WAR8QZgamZOrceXpSSaNwRuysyzbB/
TPRGxAaXl20xKxfiawArAMsCKmfmaFsObo5hgliRJUtd0hqI0fp4fWC8zL28xLKlfRMRUSnXTtbVv4yB
gZk0+7wV8ANi30adc6gkR8QPgJ/WxsTcleXNpZk6pFZtPZuZT7UYptS8i/
g58BPhHJ4lsQrn7Ggn+E4AxmfmLiJgHWJSScF6d8jz1a+
+Pwh7MkiRJ6jON1gAjgV2ApSJiJqVX4DXApMy83IpNDXQRsQqloOdaeK6nY/
MD6PnAl2c5JvWKEzqPDeAtwLLAnhFxH3AdMCEiJnaqNqVe0tgFMwp4TWb+vS5SdqwXEdtm5mFtxTjQ1V
kiAPMDnaHUz9TnpKnANRExqF7X13HKCrokSZLUJxpJ4x8Be9bLHwLeA5wInBMRq5lc1kAVEZ1BP9sCQy
Ji44hYMyIWiojBjfNLAEtl5m2tBCq1pLZO+kcjYXY4cBRlWNl04M3A54AfzpJUk3pF5z3S64G/
QkliNh4PqwDbQ0lG9394vSEilqDMEflQrVJ+QZ9r+16/kBXMkiRJ6hON6uWNgDUyc/
mIWBz4KHAocDRwG2A7AA1YjcWTcylVmScAQ4EbgCspVU9jKUPMrm4lSKldB1D6LJ8fEeMoO1suBS6tg7
WWp/SYHV6Tau54Uc+YZWDfeZTk5seA0zPzkXp8N+DCenkQYKKzO5alJPtHA+Mi4mrKgN4/
O7D639mDWZIkSX2isaXzS8AGmblbRHwEeGdmbURbwe2ycx9Ww5V6lcRsTCwMbA7sBUwBXgT8OHMHMPLHN
2KT+FhGjgb0oQ1+HA/
cCNwIXU1opTc7Mp9uLUJoz1OrkT1MSnLdRWiptRlmwPCAzb3MBpnsiYjglybwYZdFrbWAkZdDf9zLz2B
bDm+OYYJYkSVKfaFQw7wUslpnHRsThwKKZuW9EnAo8mJmfbjlUqTW1RcYywKbA7zPzsZZDkloTEYdRFl
6mUx4XC1Mq+/8GHJGZj7YYntSvImI7YEhmntcZMlePbwVsWa/
2CGVA5gNtxdmrImI+YGlKsvm6zLzTBP/zTDBLkiSpz0XEUsA0ypvwn1EGpCwJfCoz/
9xmbJKk9kTEsMycERE7U6oz98zM++q5HYEfAndm5qZtxin1t4j4NCVxeUlE7AdsQKnsvwK4zURm/
6m94rcCtgbuBG4Hbq7fH/O++Hf2YJYkSdKrFhGvoVRkbk/Z7nxq7SE4vn5geg/
wQ5PLktTzOu0v3gLc0kguD8rM30TEysD99ZjVgeolPwGeqJefofRW3pnSUubhiJhEaZXxK6v7u6MO85s
J7AO8G3iA0qJkYUp7qzuBM4BftBbkHMppk5IkSZptjenlXwCOogxn2hX4eEQsFhFbAP8DzOD7Vsa0MO85s
VGGB2ATAqIj4SEUs2jm8PrFAvR78HKLUkMx+vsyzmpzw+DgC+BZwOTAReC+yJxaLd1FnQ2hf4v8zcCZg
EHEMZ3rth5zqN98DCFhmSJEnqAxExBgwGAxExBdgWuAkYBXOTaHmMysGpmPhER9wHrZOa9EXEi8NXMvMp
ygeguSoL5+My8prUgJUlzjJqc+QzlNeNeSsXmGykVgx/
IzDusYFYviojPAVtn5uhZji9PmWtxXTuR9YbaHmMysGpmPhER9wHrZOa9EXEi8NXMvMvnpxcywSxJkqR
XJSJWBa7IzCUbx2YCWwA3dbY/
S5I0q4gYBWwGDAYeA87KzFvbjUrqP7U9zLMR8X3KovwuwMmZefws13sn8FBmXtRGnL2iJvK/
BHwXeBI4G9gEmA+4OZZl9ZIkSZotjcqN0cDgiNiE0ptuU2BCZo5tNUBJ0hwrIpYDVqYMzbq0
9j2Vek6jPcx9lMX5TYBlI2Ib4I/
A7zNzMnAocCQ8n5RuIdwBLzPvjIivATMpOypuBu4BrgXOhxf0alZlBbMkSZJelYgYAexH8wCKUhMF
HKa0xpgKPu41QkgQQEQcBHwauBUZQXjPGU5I4Z2Tmwy2GJ7UmItYBTga+COwIbER5jAyjtG3YPDOfeNE
/oD4VEYOBJSjPV49TdljcYYL/
35lgliRJUp+JiIWBdYF3AW8D+qiIWBdYF3AW8D7qdM3f5sZt7SZmySpPZ0dr1ExJrA7ynDyuYFfkfZir4fcDGwV2b+q71I
```

pTlHRCxAGaC8NHBPZk6092/3REQA6wNfo7x/
nZyZR7Qb1dzBFhmSJEnqM5k5HbgMuKy+SV8G2AqY1mpgkqS2DaJsOd8Z+Edm/
jkidgOuyswDIuIp4CmTy+plETE/MBJYktL/9zbKPIsb63mTy13QaHmxK/
ApysLXpsCGwBERsSMw3fZvL84EsyRJkrqifgCaAvy07VgkSe1q9CtdELi8Xh4NXF0vDwXmAfvLqjfVhf
mvU9ox/IXSmuFGYHJETAVOycxHWgyxF3wAuDAzvxkRX6e8jwUYBSQw1uen/
2xQ2wFIkiRJkqSecSal9zKUoVmLR8SWwLuBK1qLSmpJRHRyc6Mp8yyWA44CVgHuBj4LbGByuXsaC2CDg
Svr5d0pz1dQKpn/0d9xzU2sYJYkSZIkSV3Tqfir/ZfXAn5eT50LfLD+fAJwPoDVgeoxUb/
vDIzJzIcjYn3gnMz8fET8C3gYrO7vB6cCx0XE/pS+1xdHxOKUtiUXgM9PL8YEsyRJkiRJ6g/
7AItk5s8iYlhm3gVsEBGLAo/ZW1a9qFE9+ySlqh9gM+CP9fIagL1/+8cvKcnk/YEHgKOBLYGjM/
MBe2C/OBPMkiRJkiSpmzoJmQeAqQCZOeO5k5kPthGUNKeo/Zd/
RBmODDAGGBURdwGbA4fV4yY3uygzn46IyltMt5C6cF8dmZeXs97+7+I8LaRJEmSJEndFBHDgJMpbQCO
By4EbgZuz8xnWgxNak1EDM7MmREXlg8M39Wj69AadewIqV69vsthtkTImIl4M3AvMD1wPjMfKzdqOYe
JpglSZIkSVJX1QTz/wJLASMolZiPAtOA6zPzlBbDk4rR6E9+FnBjZh4UEUOaiy62Zeiexu2/
PvAz4ClgHLAo8AxwH3BDZh7TYphzBVtkSJIkSZKkPtVJikXEKODtwHGZ+c2IGAGsDCwJrAK8Zhef8cB
Zuopjf/
vfweejYh5MvOpWa5jcrl7OgMWPwhclpkfi4hVKc9RKwNrAwuCif6XYoJZkiRJkiT1qUYi5lDgp8C99ec
zgUUoA8zGA58GHu78Wn/GKLWp0R5jKWAHYF1geESMBW4H7s3Mh//
rH9Gr0hiw+ABwSz02CZgEEBELAUPbiW7uYoJZkiRJkiT1uYh4PbMv5f/
Xl+YHXgncBo4EBgcGYeC1Zqqjd0KmEbyc2nKYswKwObAVtQFmTujIixmXl2O5EObI1dFvMAI4E9IuI+4
G/AY5n5ZDPB7/
PTf2eCWZIkSZIk9ZlgQmNU8d0QSNhvBSlZc+TwA8o1ctXRcTVMmTJEmSJEl9plEZ+C1gZmZ+qdEOYB7g6
DWlEpm28d0QSNhvBSlZc+TwA8o1ctXRcTVwMTMvKWlEOcqDvmTJEmSJEl9plEZ+C1gZmZ+qdEOYB7g6X
r+QGDNzNy7c77l0KWui4jBwFmUJPKiwI3AGcBFmXn1rNetjxv7//
ahiNgMuBJgloGK6wPbUSrJNwFOzcxPevu/
NBPMkiRJkiSpz0XE5sBpwA6Z+Y967LkfKzIi4Fvh2Zv7MBLN6UURsABxEqfZ/itKH/
EpKwvmPmfnPFsMbsCLiz8DOmTktIg4Ars7MP81ynaHAEpk5xQrylzao7QAkSZIkSdKAdCVwKfDTiNg7I
hbJzGcjYvla3fwEcA68YNiWNODV5CXAlsAdwI7ACGB7Sjvb44BT6nWjjRgHqnp77lqTywtQ7oPiIuL2i
BgTEV+IiHUy8+nMnAJgcvmlWcEsSSZIkSZK6IiKWBg6hbdsDsfDjwLTAOmA5/
NzKvcfq5e02h9MQ74fGZe2Di3GvAx4Mez2Zk5tnLsV2HPWnszqroiYNzOfrEMYh2Tml23f8/
6KiJGAMsDi1GGaf1G5Xxmp1IWAAwdwI7ACGB7Sjvb44BT6nWjjRgHqnp77lqTywtQ7oPjIuL2i
KYYJYkSZIkSZL6WUSsC3wP+CtwOTAPsAwdWau1mJoA9qL7ZpoDCidCByWmadZQf7ymGCWJEmSJEmS+l
lEDKIklD8KrAlpfqZl5jc7DuN5PFIYBRwNTAFeCQn2pcpwn2pcbwhll8ximTm9nWnYJUmSJEmSpB
ZFxGBGiWarDPW9iNiEkxxgbuAqyAbuAqygDSSfVntc7Asdl5nIm+F83KIAqAAAFuElEQVQ+E8ySJEmSJElSP4
mIocCuwBeAicBk4B/
A9cAtmXlbe9H1ezI9hoiYl9IXfmfgBrqSAzAV4HHAOuYYJZkiRJkiRJ6jrkSrL5Ywv3wmmmCVJkiRJkiRJkvpYYJJZkiRJkiRJ6gZOB
G8HrgnZQk8zDgZ5n5lRbD7EkRsRCwMbXpbYL5r1Ywv3wmmmCVJkiRJkiRJkqQuaySYLwCuyMwjIuJI4BgZOB
E4PTMPM7kpuYmg9oOQJIkSZIkSZIkSRroMnNmvmVMMEuSJEmSZIkSZIkSZIkqfseBhYGFgHWARYA1gduzMyrA
o7TGWKwO9rsVeDoiBlFaZUwCSFsOaC4pC4ypO0AJEmSJEmSGaJpIG5SUkr7SkkSpIkSZIkaSCSkCiKOBTwIPU
UwUa25kBbMkSkSZLIrmG+KskSZIkSZIkaUSCKiKOBTwIPU
VpjjAMeB9YF7s7MyzJzOjw3BFCa69iGwaI8NPcs&uI8FsBrwMeDnEbF+Zk6OiOi0
2yRobmRLTIkSZIkSZIkSZKkPtJpdRERhwDrZOY7Y7ImJeYEYzgRwR51K6CzwF/
DYzT2gnYunVsUWGJEmSJEmS1Hc6SeQlgPH18KfUJ6xgliRJkiRJkiRJkvpYR
GwDnADsnJnJnX1GODM3NmvXwN8MPM/
EmLYUqvmhXMkiRJkiRJUt+7BBGe0fEIpk5MyJWiIhDKXm1uNUODoDVjBLkiRJkiRJkiRJfSgiRJfSgiIjMzI
pYCDgG2B4ZT2mdMAx4GvpZiZl3Wu216oqtjglmSJEmSJEmSJEnqZEnqoogYCaMJEnqoogYCawFLAbMAP6QmdPajUrqGyaYJUmSJEmSJEmSJEmSJUmSJ
EmSJEmzxR7MkiRJkiRJkiRJkiRJkiRJkiRJkiRJkjRbRbTDBLkiRJkiRJkiRJkiRJkiRJkmaLCWZJkiRJkiRJkiRJkiRJkiRJ0mwxwWZGZUn6yIyqYVqbzWlXlvv
IhYMSKyfh3UOH5i5/gr/HsTXs7vRMQh9e/
vNjtxS5IkSd1mglmlJELZSZLWU3ZLKXJBLMkSZL08t0KjAS2APYPBkZKEZF5IzMNY3AIJ3VZOUuPJAiIaTTSJLZCqbjSmlmzZQ8I/bCqQyvEvIZKX90Q
iLo6Iteq5hSPigoh4JKJdqOkkY/3nIj4UkRMMrr93YUSM7M9
lCRJkjS7TDBLkiRJ

"<Figure size 1440x504 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABZgAAAHwCAYAAAArRQrgAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAA
LEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDMuMC4wLCBodHRwOi8vbWF0cGxvdGxpY
i5vcmcvqOYd8AAAIABJREFUeJzs3Xd4lFXCxuHnzKRDQjolgBTpICDNggooGLBjw2DBFREF3XU/
VECxgAIq69rQFRCsAZUFRWUpUsS6gooCChILyGFGkgymfP9kZBNSASMmXlD5ndfV67NnDnvO8+yC0me
nDnHWGsFAAAAAAAAAMAf5XI6AAAAAAAAADg1ETBDAAAAAAAAAAoFwpmAAAAAAAAAEC5UDADAAAAAAAA
AMqFghkAAAAAAAAAUC4UzAAAAAAAAAACAcqFgBgAAAAAAAACUCwUzAAAAAAAAAKBcKJgBAAAAAAAAOUS
5HSACmSdDgAAAAAAAAAVYQ5mUmsYAYAAAAAAAAlAsFMwAAAAAAAAACgXCiYAQAAAAAAAADlQsEMAAAA
AAAAACgXCmYAAAAAAAAAQLlQMAMAAAAAAAAAyoWCGQAAAAAAAABQLhTMAAAAAAAAAIByoWAGAAAAAAAA
AJQLBTMAAAAAAAAAAoFyCnA4AAAAAAAMCpaOvWrUq5/
S7FNmymGmHBuvum69TpzPZOxwIAwK+MtdbpDBUiOTnZzps3z+kYAwSqSVr5Z9i3z+kYkWNfWQ/ZDcKa0Lj
7HWnuvtbadtbadpBckzSq8NlbSI5K6K6SOos6RFjTIyvsgIHZyAAAAAAMAfcShzjw7+8n2p8erKdSAN
Hc+fO1SWX9NLQoUOdOdjgQAgCN8uQezKWPM/s7cfvzgbcpSBbcAzYdjYYmxrGOrYIMXE1Ruk1X5aAkeW
g6gcl7MuJNtd3BjCgS9xNc3Bnk9ThLOEQX7uTh9ZWi+pFxBgCiQMfl1SMFpqcQ79pj7ym8ohzbTn9ZdBf
OY79qeWuoMpdTFnPHgYy0PDoqeAffw3wADFT5z/QvZbd+mAHvSjDsMRcQB+Ss56MQcWoLNNEZK6JSbQ
/wQoLTq7sKjDw2e08xSbSXT+Zwpn9fmOSYoUpZAU1hOtcHmvAJD0gOSaOIS/ooWYB0rT4r9+IxlrE8f
LcvCE5xxXhymPH1xHzqHM7nSbOM1yVjckfmEpE9TaPiFnNyfZVbKHMkN2jwFcv9XoDpltNx3D4JULKJkB8
uuRx7mYyYMXWrnxrj3yEXcCWO8zf6dhDEx7SVxnyNvJpsCR5xAZEtT9+SSNgYJxYPdN2GiaJx20iq0EIu
LLBPK10IRlyxeyc3SfC3iPjKTyV1XbMuf9trJbVWxZxn0bJnG8QipkWQI7vU5y+njSh6TDdYWUtDK2CjeV
m39NyuWrRZ84CD4XNLHbpU3nQhYTUffQKTdqMZO2Xrz+vVcfzwbIONkpXA6rNXW2YY+kBIUnwF+tufM+gCW
OnNhGbNg44ZBmnZfh5mqZjTmBF31NKdXm/WFTXZvZKpsXmU5xTzxnbSf7Nyq13gW5nfwBnS2YavoInR9YF
e5zCP1vfNMdPsQwAwoCE6R2RjpEB9TipLuh4/yW6C0OCTbGKmsSyTmdFbYN1MpgBcF+u4lyTiUqQqZ+xpq
Cu4MtGZP/tSGvrpCQz8udmh1yefAbN3bHm9+iV9wXkyxjfrMtEIOGlAUaqlPmQTzBtxW8mdgNs7MBmTuZCK
Aw66bqWOSdKhRTnzrTrrEg0e3ZF4IQRwFH8kUb0u8xsIzjWXRWRvUzuvz0bjPHgHVOmFmZYIe6nG51pzRE4
Xvak6nTPm/WLNLk60GfBqDPjLOSGXj+ldwjFUekC8JHtAv5wzc7XHnWGAz1dtiIGSaKg/Tk8RGDZRq4KZi
RTlnapUm57vGHzgWnTn2KJiwa3hDgsB3VaMN5QEEiP0K6lJOAgOcaOkHetmsvRO5itqFJn+lbbfIb27w2P
oOwWn43zhw14ReDkIYQ7Z4vE8Q2KMUKr0dXY6MaV4Io+HbmY8QqMS1h/SH+N7qcc18LjkXMGkCJlzrhKTg
yfY74IwGb4xHxAzHMh6ua5HOT+O0m/WfoiXNfZv4uMn5tAQBsNaktohq0Ry7WGWG3pUsI8/J0kzDAcoiVOn
w0KPUixYfT0E2TRxu/W6nu+31/JnyajWcz1gmP5ofBFN0tXNCMcMfsoC4LAj6mVKq6YEqffkSbuIehQ3m
ILvFaLFuRpnb5fA+IHNPcRG5vMdfHnUprOfhOedDRpEqzqkedPJ17/0FVKp5XaxWHJGS+e7ms8qMmnCDeQ
pGMibAQhyQBxvv5zBFJXMmjkNhbQOlcUs+iRK3X+XjhAJO3ojwS36rmAtAxORBp7P53ZkHStkf3f2i9LW0
NQgK8y5ukFIAMlsUWgXLQP0JsTvJYGpOo9TiGCQR48WnVKxkGiMW7QVh0ZtzGjz0HbnTgVqpXOG3awoYxbh
SIkcSVbBT27MT3S9KUsvZKGoGHY3ewjZINKVwMnYeyFTnvWqZK3JaNEOZeZzAi3CVBfXhSuXkTt8R2yg8FdVS
gyP3S5TYeD5+T7tCnkSkNOAgKKmMHXbT2jwQmohBJ0UaE3J5uoBxfpVQnDoxdEiggiOnuHN/sdkSi4yOUC
KaFW3sOBXzYH6WVQOa3jyM3Al6mNa1PJSnkIn0TxzTI6Y3oybU/Sq8bEnSgNMQlJljNmKIEE1nJALehOcN1
rxw1X55dXPy0AD+brSKTNx1nzvmDCIdNNNSK0gbeMEKZbC7MbCWaU8q1M3A6rCIDRmA0lQ0bKhVRJQ1Rn
Fk0nXXs+0tNHL5eTsm1bTBQsHQIcPJvrA+uQ0fIgNiiAEZqVWEi1FD0Z7jwyIXhtbLyfOrr+Ls3oNeT5fdTQwg
HkVEd8A7qbcvHq87YQGuXbLE+YibqTdKIJBRn/ID8lHsH+/0NfrefVa4wsJL+tsQbvyxPwnbIuaKJ1N2xfPf
gHGhJBbT4+7HKuw1r/YbiEd1GmcWIl+JK2ub1wMMm5jR4Og4Lun+sKqkWdoiDLvCC3O9gZ3dR1AgN4Wnt7
Fk7jsZ8QgLQe6aApJFrqfNIG+Uwz5LAlihJNW5gJzgxTeb0+uT8/uKzv9wi+gpUq4Tni3KZT4b63Sc4V6uG
LAGX7L1emzCIhiTZLhXQ+7oOGwlC1Eww9ovYJ6eMWrM9w+KxiajFRtlr5VzV8aQCupQg3SaJ6cPqr5RmeLQl
pLdp93wHXGhJn4NTOsI94ATLL0rwqwEVgEGLiTAv16Ql/l0wyR6J5Xv4HbGb0lFrCpa3a88jYTpwaM5Ym
kyZnDMH5X7w1E+TbMX8cfDzTSrTVZjgMrWhVjbaKQ/+cNxMvqfCXu/3hAQ51TJEjjTu8YqH1VzZJ02iWX
a16fwHjbqoFgpuzm04tlBiGXUncZYqvHnQkUJVB6LgTdAOYgVvT/6VJvpqqmpGg/21ZIDBhOQmq/0AaJ9X
3NdwxCoZlnpwOAIINrc4mq6O5qNpeCGwXAzaPN0eoE2mxvGEIcNKO+7wgdknGqbtcEZOTOSEs8KM
n74R1p3H5XidhfPpLjGQDd1EFbAIS0EIf3ZvotZRRQ4e8uZNWxYjtbhIS2VfIGZedUfu3AQLv0HwqvWvXn
9k2Hwdl3jfZKJlGXaxLfB7WBhAFu5ki3wBbhhKmX/+GDkNNEt0BPwfR1htkyJVvG8ckAGn5gXO6U/
Zgp65s1AHZ7P1FCDWyFLnNLxdZD0NDc5ETj7Cjto0aCq7FgCRZC3wKEJ2cegBgJfGvjLi2vbqG/OVvX
AtAmWQoHmfBV5PoKaFLfbPLNObndIzf8ADifwv2zTpAwhFXnrOXeG+Hh3mZSDmmK8O0nyq21vO/pt1KyO
c3pcnZQx0xetPTHO9T6UIzGVdrOBeOaW6+3Bp/AhnGzhQIv13HNADrHMHFK6SGVCBaxb1lQVdl3ugbrP
CfNhZP4TTJbIJxPKV5FxIRnxfiaXNXvrWLmFqbzvnKEwDVkUnyoQeAGjOuz/NfHSd4bHmpk9RNeIg6S+
Apl2/JDiZYXlqS3b5bzV01F8Q6Ag5Dpwmhky/WhLqc+r/xiejSwTqlqCZ4f+o9mkFeqrpgAAaVsu/N4/
xrLyLrzZRFg6JXxAqb5Vsj7ka5ywAArSAq5QdQMSu7iR7ucJqpKP9kRQ8UqfiVDzjqnu+8AHhwn18lurcLR
uBq9DfiVhGSvhmxzs2DnvydlX0O6L1jz14yXaxVFrQHNyXbu7V5dtzl/tgoDiVXg6rSfeJWOK1yEYZv/C
4XJRZ6tHNfRt0Q7L9n3H1x2fBWW2cJK3+PZ+ze0Jx/5h3DWsKFvk+2f8jZA+vWFtWTp+5/VNjl0JRerb80W
hw+QN23J41V9vWZeIuJA3VvdvM1b/9KjF6fBfFmj3+QGVc2xVwcgzc8kFIeZBnuZNd3NFcmCt5QjKDG4DS
rJjxxKMAb/Abb1Gu61WxqeiuxbHuodR3p0NCAuL+CAUZK66c0PE7xpD98fqtMaBjjfH2dvGWOtGzWmNO
7Ahue5hBNnrMRnfM9/6ZnQC2T1ThfWZ1ru4vp0IybfkYvRuw9hv3xiAwZeGNKK7Rl47iW6bbb6vLSKrrdM
Lr+fdwHfqu91LBXeOXjjj6Vf+9/cKjVj3WD1wnRXrqBXwrV2E8ApU+LH04Plf9UubEXbXwJBp00cuqabVB
tEE/Kbd1XU/Cb7jd7P6XdfoCt+PdoaCLDKaIZa0nfDc9a3a1nqgW3I13f1p1cp9D4v8A6aSUrSz0FLnzbKHE
8A7sXQDhOJh5rpaVPMFV5AhQvu7J6F2K6EYsr7x5dgP/
nG3ITP8x3ZDdCBl5TerR8YE0gnMgm7Zos4aJ3c4ybyD8ObtXzW2ZS/
eKdzZM3NT24xNYPivqx5wKRz0vK6KN5yaPDpwBBQwmZ0SPJh4i41q0u85bYnHDVaKRqZXQHd4ERVFb
/hIJWuQOPqKmnhgvYO8wf6W3y1Cuh8vV6oQ4EWi4P3rA8A2t96LhHlcrEyfhRFQq0jhrrQYeqJKNyUwA3QWJvP
QiOgg8iB6b93d40yLP9bWaaHVP/lfXrQ3KxyLOVtKd+VvVzsfTN9z9Xz9K0Ko2TdnqDj+S1mR9rEXeb0Vy0uT
2MPm+OQ+gkl3/i3TewJb4eRyVt2LQWvwSwq/

```
TJkrZ0rae4xYw8X+9xK+nvhR/
E5X0pq48tsAAAAAADg5KWnZ+qKfo87HUO5uUc06eVxatLifNU97Qz9tv7rEiuYJSnnyF7d+
+B7crt9WnuclA9mPOR0BADwKV9ukQEAAAAAQJWUn5+v7777TuvWrXM6SsAJCQnT+ef30pwZ9+vQwUytX
7NUm35dLknyer36csk0ZezZqldfGaevvpjncFoAqPqc/
1UeAAAAAACnkMXLvtBjU6br4Gmd5Mo5qLi05zV17IOqU6eO09ECxhntz1WLVh3106qvdNkV12nn9h+18
P0P9dvG1erc9WbddOfrkqRV387Rgv/MUK/e/RxODABVFwWzj/
248ge9+fyLCsnxKCfI6Mpbb9G53S5wOhYAAECls27dOm3ZulWdO3VSjRo1nI4DAGXKzs7Wg5NmKKTfo4
o0RpJ0KOewBj/8pOZMec7hdIElOCRU7Tt2kySd0U5a8d9Fqt+km5q16lE054wOl2vuzAdlrZUp/
N8LAFCxKJh9aOPGjXrr0Sf0aJcL5Xa5ZK3Vsy+9quDQEHU+u/QhBAAAAIHo4MGDuvH/
hmpLfLByE6qr+qxpurrNWbrvzrudjgYApXwwd57yzrxcocXKSndouHaHJSo9PV3x8fEOpvOtrVs26MWn
bnA6xu/albZfN9zxbqnxvDyvnh9/vdxutwOpAKDqo2D2obdefEn3n3me3K6Cra6NMbqn/
bl6bMo0CmYAAIBCf3/
iEW3r1UJhsVEKk6S2TTV93nL1XPWD2p3R1ul4ACqBYSMf0p70zAq73+4tG8t97a6MLLn7P1FqPCNrr66
++mqFh4f/mWiSpJr1G5140klIiI/VhLEVdyhfvfqNK8Uhf79n9aqv9dv6L9Wmw2VFY7u2r1Xarl/
VsHEHBQcZ9b60vxISk/yai0P+AFR1AVcwjxo+Qpnp6RV2vw3btv7+k7sz9EC/
QSWG3C6XfvnhRyUnJ1dYhuNpXLdehd0rNj5eY8aPq7D7AQAASNK6zF0KiT29xFhIj3Z6ZcZbepmCGYCk
PemZqn39/RV2v9p/4trcw4f04WsvKbJJ+6Ixb16ODmxep90NOqua8ah7ymAFhYT++aB/
0s53nnI6gl+1bNNFy5aOUkx8fdU9ra22/rZS33wxXbfe/
baCgkKUc+SgUt96VDcPuEcxsYlOxwWAKiPgCubM9HQ9nnyVX17r5ff/
re37spRUI6Zo7FBujk5PqqsRN93qlwwV6aF5s52OAAAATkEn+sX6b979qquuJcZsvlcLFixQ8nc/
+jKaJGnevHk+fw0AVUdIeDW173KuVk59SOFtuyt3f6b2r12uhreNUVhiXR1J26pPp0/ShbewzY+/
GWN0+12PaNGCd/XJD7O0+be1GjD0bbndBdVHaFh1XXTZCM2fO0X9bvyrw2kBoOoIuILZn/
r3TNbjUyfp7126q1Fcgnbu36snv1ioIf36Ox0NAADAb05U4N49ari+SstUSGJs0VjeJ9/
pwzdnqGWLFiXmJicnUwgDcFyjM8/
RaW06atlbLyqv7hk6ffCTRQfIhSXW006PVb4nT+6gYIeTVqz4+NgK3e5h65YNFXavsoSEVC8ql4+qHhm
nn3/6Si8+9fUJr69Xv3GF5IiPjz3xJAA4hVEw+1BUtWoaNfAOvbfkE+1Z972iIiM17Ja/
KDYyyuloAAAAlcbTDz6qW+//m34J/kV58dVV7bcM3d61V6ly+Xi8Xq8kyVV49gWAqmX3lo3a/
fRgp2OUsj99n6J7Dioql4/KO5ytlRPuqnKHyo0bO8bpCH/I4CHDSxX9+/
ft1qWXXKTHHrnPwWQAAl1OTo5SX31DG1etVXiN6rpp6EDVq1dx29z6GwWzj0WGR+gvfS53OgYAAAEClFR
YWpunP/0vbt2/
Xjh071Lp165M+JCs9PV1DHhupTTn7ZSQ1rR6vFx8dq6gofqEPVCU16zeq0D2YK0rD3du0dMks1bxySNG
Yzfco3C11GD7ZwWQFAm0P5mPdM3SAHh4zRudc9IBCw6rp0MFMfbP4KU1+ebTT0QAEsLy8PN174x26rVZ
33ZB0pfYePqBn7npMKY8OUbsO7U98g0qIgrmS+nXHdh08nK3WDRopqIr91hsAAKAsSUlJSkpKOun51lr
1v+9u7b2ms9wRYZKkn/cd1IAH/
qZZL0/1VUwAKFKjZl3VqR6q7bNeUI1zLpdnX7r2L52hbn1vdDoaJLVs0VxPjhmsF196QYcO5ysuJlyvT
HxU8fHxTkcDEMD+nfqu+id0VavERpKk6PBIPdrxJj387GS1e/
Mlh9OVDwVzJbM7K1PPvfO22sfXVnRouEYvmKfk8y7QOa3PcDoaAABApbJq1SrtPi1KEYXlsiQF16iuzd
WlrVu3ntJvMwRw6uh06fVqtmen1n/
ziapF1VCTOx9QUEio07FQqHHjxvrnPx5zOgYAFFn9zUpdWbvkbgcul0thOQ4FqgAUzJXMSzPf0RMXXKJ
qhd+QXNGynYYvfF9nNG6i6if5VlEAAIA/Y9jI4UpLT3c6xu+6edBASdKO7duVd1HzUs9nh7l051/
vVqyfV6glxsdrwtjxfn1NAJVDVEJtdbzkOqdjAEDAGTX8QWWmZzgd43cNGVj6/
IBN6zZoc1hnnRZdq8T4z7/9UuZ8X4iNj9OY8U9U2P0omCuRfYcOqmZoRFG5fFS/
Vh208NtvdFXXCxxKBgAAAklaerpi+vVwOkaZYop9Xu3wEW2aMUtq17TEnNz129RgcIrcQf79VjdtxmK/
vh4AAECgy0zP0OiLBjkd4w/Zd/YBPfX6ixrf7Q6FBRd0gG/9OF99z+uj5M7+6f4e/
mRShd4v4ArmDdu26oYpzzsdo0x5eXlqb8JKjRtJkz9bpHfX/uD/
UAAAAJVUSHiYGjRooE2pCxSdfJZsvld7P/5Szdu19nu5DAAAAJyMGtUidee1A/
T4vFS586Vcr0dnt+ukC8881+lo5RZw33k3rltPjydf5XSM3/XwpJd0JC9PYcHBRWPvrvlOM/
46XFEREQ4mkx6aN9vR1wcAADhWq+7nqH5ahtYu+UbG5VKPy/
ooIraG07EABCBPbo7yc3MUWj3K6SgAgEqubkJtjbhpqNMxKkzAFcyV3eCrr9X9M1LVtU4DRYeFa9HmX3
TBWWc5Xi4DAABUVpGJcerUt7fTMQAEqPy8XC2dPkn7PVYmLFLK2qEuPS9X7aatnY4GAIBfUDBXMnXiEj
Turnv006aNOng4WyMu6q6QYquZAQAAgMrK4/Hozffe1ZLly5UQHa2/3zZQSUlJTscCfGrZO6/
KdX4/1a7VUJJkrdWXUx/
SpUmnKbRapMPpAADwPQrmSsgYozYNGxc93nfooKZ+9IFysw8rz3rVuc0Z6tXpLAcTAgAAACV5vV5dc+c
d2npGC4X17q51Bw5o2YMjNPHuv6pzhw5OxwN8wlqrvdmHi8plqeDnuZjkW7V62Tyd2ftaB9MBAOAfMy
VnCc/X49Pm6xHz+uthOqRstZq1prvNXPpIl3T7UKn4wEAAACSpFkfo8cmvaH3rGC4X17q51Bw5o2YMjNPHuv6pzhw5OxwN8wlqrvdmHi8plqeDnuZjkW7V62Tyd2ftaB9MBAOAfMy
tfo8cmvaE6Hij2pHIEnIT5WO995qsLut3vLxgq5j9frVW5I6VX6wZGx2rxkllw/LSrXfWvWb/
Rno0kq+HMDAMDXKJgruQUr/
quUlmcqoXrBW6uMMbq61ZkavmiORMEMAACASmL+Z58pvNf5JcaMMcqwXocSoSqZMPZxpyP8risG3assT
55cQf/b2jD3vx9oTupratasmYPJAFRVK79dqXdfeUeuHKPqdSJ1x/
2DFRMT43QsFJPnydN7n36sLTu2ye1267Lzeqp5vdOdjuUzAVcwx8bH66F5syvsfhu2ba2we5Vl/
45d+ujagaXGDx8+rH6Tn5Mx5rjXN65br8KyxMbHV9i9AAAA4F/
Jyck+vf+W9HSFt2qi8Lp1SoxvXbfupF573rx5vooG+NTT9w/
RX0Y9osOd+sodU1Pe7+bpkgYRlMsAfOLzJZ9r6bOLNLzFvQp2Byvt0B6NuPkB/
WPGP1WtWjWn40EF7255bNo/
```

NaD5xbr9rJ46kpejiUtmK61ths5v28XpeD4RcAXzmPHjnI7wh3wyb76+mLVAvRo3LzFeLamW5s94y6FU
AAAAONVUZIGbnJxc6n4HDhxQr8G3y9v/
OrlCClZy5nzzrUbcepvuvGVAhb02UNk0Pf10LX79Bc384CNtT/
tG1/5fPzVo0MDpWACqqNmTZ2lMq4eKFhwmVkvQnXUH6u0pb2vQXwc5nA6S9NlP36h3vU5qW6tgxXJYcK
j+7+x+uv/
Tf1EwwxkXXtxLf3srVXV2bVfrWknK8eTplVXfqGfK9U5HAwAAAIpERkYq9YnxGvnMP7QrL0dhXqtbz+
+m2/r3dzoa4HMhISFKubav0zEABIDw3NBS72ZvGttEM9fOcSgRjrVy/Wrd2/
KqUuPVXaGy1p5wN4JTEQVzJWeM0dNTJ+utya9q1srv5AoN0TXD/6a27ds5HQ0AAAAooWGDBpr+/
AtOxwAA4JRzsltZebd5ZJuWLCnXpq/Toq8XaXnyCl/
Fq7RbWW3Ytkn9XxvpdIwS9u3JVLeYVjqnfpsS42v2bNKNrz/
oUCrfomA+BYSEhOgvQ+50OgYAAAAAAAB84GQL3M8WfaZnX5iou1sMVpArSHsOpWvSjmla+OUnRXswl7W
VVVXVuG4Djb6ocm0NkufJ00OTnlLz+NMUGxElSfr3z0t1ydkXqu/
5vR1OV+DhTyZV6P0omAEAAAAAAIBTwHkXnqfqNapr3CvPyJVrVK1OpJ54fRwH/
FUiwUHBGn7zUL348TvyHM6Rx3rVoVU79e18gdPRfManBbMxJlnSc5LckqZYa8eXMec6SY9KspJ+sNamF
I7fIumhwmmPW2tf92VWAAAAAAAAoLJr37G92nds73QMHEdM9Rr6v+sr18pqX/
JZwWyMcUuaKKmnpG2Slhtj5lhr1xSb00TSCEnnWmuzjDGJheOxkh6R1FEFxfO3hddm+SovAAAAAAAAAO
CP8eUK5s6SfrXWbpQkY8wMSVdIWlNszu2SJh4tjq21aYXjF0taaK3NLLx2oaRkSdN9mBcAAADA77DW6r
vvvlNGZqby8/
OdjgMAAIBKwuXDeydJ2lrs8bbCseKaSmpqjPnCGPN14ZYaJ3utjDGDjDErjDEr9uzZU4HRAQAAABy1c+
dO9bzlZg38aJaGrVqudW7pjffedToWAAAAKgFfrmA2ZYzZMl6/iaRukupK+swY0/
okr5W1dpKkSZLUsWPHUs8DAAAA+POGjH5Mh667QtXCQiVJUW3b6IX33lfyBd2UmJjocDoAAAD/
+XnLL9q0e7vaNW6l2rEJTsepFHy5gnmbpHrFHteVtKOMOR9Ya/
Ostb9JWqeCwvlkrgUAAADgY0eOHNF2my93Ybl8lO16lt6YOdOhVAAAAP51OPeIHp7ytNat+EmNsqP08X
8+1sTZrzsdq1Lw5Qrm5ZKaGGMaStouqZ+klGPmvC/
pBkmvGWPiVbBlxkZJGySNNcbEFM7rpYLDAAEAAICAMGzkCKWlpzsdQ/
n5+dqzL1N1jhm3+fmaOXuWVq383pFcvycxPl4Txo5zOgYAAKhiJs9J1b3tr1XdqIJVy2fWaab31y7Tsh
+/
0fltOjuczlk+K5ittR5jzFBJ8yW5JU211q42xoyWtMJaO6fwuV7GmDWS8iXdZ63NkCRjzBgVlNSSNPro
gX8AAABAIEhLT1fs9X2djiFJipgyVZ6DBxVUvXrR2N75i9VzyCCFRUY6mKy0tHdmOR0BAABUQdkHDhaV
y0dd3rSrHl3+BgWzL29urZ0rae4xYw8X+9xK+nvhx7HXTpU01Zf5AAAAAJxY1+uv1dK3UmWTastdo4Zy
16xTmy6dK125DAAAAktsfJwe/mRShd1vw7ZNv/tcWFpuqTGPN19fbfpR/V8bWWEZjqdx3QYVcp/
Y+LgKuc9RPi2YAQAAAJz6wiIjlXznHdq/e7dyDhxUXNeucrndTscCAAABbsz4J/
z2Wi+M+4d+2rlRrRMaFY29sW6Bnp32kjp06ui3HJWRLw/5AwAAAFCFRNWsqYTTG1MuA6hUtm/
frrvvvluZmeysCcB37rr/b/pPyC96+of39N76T/
Xo92+pRo9mAV8uS6xgBgAAAAAAp6CcnBwNf3Cs1v2aKatQXdPvLj3z1Ei1a3eG09EAVEFut1ujJjyuvX
v3aseOHbqucWOFhoY6HatSoGAGAAAAAACnnJEPjVNs/
RRdcmZ9SZLX69UjY+7T9DcnKCIiwuF0AKqq6OhoRUdHOx2jUmGLDAAAgD9o7969mvn+v/
X5V1+o4MxiAADgT3l5edq+O1ex8fWLxlwul1p2vEXvvPe+g8kAIPCwghkAAOAPeH7qRL21Yqay27rlXu
tV3Mshmv7
066pZs6bT0QAACBh5eXlyu8NLjVePjFNmxgoHEgFA4GIFM4BKIzMzUz/99JOOHDnidBQAKNOmTZv02o/
vydUvQdVbxCr8nHgd6B+hoY/f63Q0AAACSkREhIzNVH6+p8T4t1++pav7XuJQKgAITKxgBuC4/
Px8PTr8LuWlf6/
aNXI1KT1CZ17QTwNuv8fpaABQwuR3p8l0r1FizB0epM25O+X1euVy8bt7AAD8ZeQDd2j4Q/
epWdsbFFWjlr5eNk3ndqqpBg0aOB0NAAIKBTMAxz3/
j9HqFPuNmrQOlRQmyatZX03TirZd1LFzF6fjATiFJScnV+j9NmdsVdgDDUuN79y2Q71795Yx5g/
db968eRUVDQCAgNOyRXO9l/
qc7hryVy36fpWuurK3Rg7nXUUA4G8UzAAct+nnL5XcLbTE2KWdgjVj+isUzAD+lIoscJOTk7Vs7lL1fv
Aa6cb/
nUyfm3FEPVtdoKlPTaqw1wKcknfkiL6aOUsHc3IkSZGhYTr72r4KCg09wZUA4IzQ0FA99eQTGjdunIYM
GeJ0HAAISBTMACpMeVcKug6sk7rVKTEW5DL6dOniCl19yEpBAH9WQkKCHr76Pj39+nM6WD9f7r1WTTxJ
emH8P52OBlSIJa+/pWp9L1NifJwkKWdPupa8/
pZ6DrrN4WQA8Pvi4uI0YcIEp2MAQMCiYAaKycjI0Lhx4zR8+HD9+MP32rNrp7r1TFZiYqLT0U4J5S1wR
z/0N+3KXKZasSFFYx/9N0tjnvqXLuxZsW9vB4A/6/KLL9UlF/
XWL7/8otjYWL5GoMrYt3OXvHVqKbSwXJak0IR47a+ZoP27dyuqZk0H0wEAAKCyomAGiklNTdW3336rO6
68SAPaxKlRNbcmzZmmmudeotvvGeZ0vCrrvgfH6f67U1Q39Dc1jPfoh61hWrRSeuBflMsAKie3263mzZ
s7HQNVXNqWrUp7+jm/
vV5WVpaCruxdajwoMUGrn3tZMTExfssCAACAUwcFM1AoIyNDCxculNK36tUb2yo82C1JalNHmvj1R/
ql9xVq0qSJwymrpvDwcL0wZbZWr16tDb+s1d13nqtvBwxwOhbgiG+/W67nJz2uPJspl62mvn0GqO+V/
ZyOBcABifXrKfYmTWeuAAAgAElEQVT6vn57vdzsbC18799Spw4lx9f9og6jHlBIeLjfsvwZme/
McjoCAASE1Dff1ecLlsvlDZEJy9XQ+waqWfNmTscC4AAKZqBQamqq8vPz1SAqqKhcPiqlZbTeSn1Nwx5
5wqF0gaFVq1Zq1aqV0zEAx2zZskVPvDBY51zrlstlJOXqo6VPKSQkTJf2udLpeACquJCICNWtWVM7P/
yPYi++UNZaZc5fpHp16pwy5TIAwD9S33xXO/
9rdUuX0ZIkT75H40c8qAlTHlFcXNwJrgZQ1bicDgBUFkuWLJHHH41GU15Z67mCOR+HVIx1IBSCQTJz8lD
pcqsJyuUCrC4L03pxXHUwFVD6HMvcqY9M2efO9Tkepctom91KH1q115N3ZynnvfXVq00Zn9LrI6VgAgE

```
rm8wXL1bX55UWPg9xB6tvhHk2b9JaDqQA4hRXMQKHu3btr/
vz52nLIakvWIdWPqSZJstZqyuoDGjZpoMMJAZxKkpP/+B7iu/
eu1U3nljxEyxijtb+uKtf9jqe8h3ICTso7fESfvvVv5cVWkysmUp75i9SiQzs16tjW6WhVSkKjhkpo1N
DpGAAAPznZ7zPz8vJ0ID1HsRG1FaQwqUvJ5xNq1NGYKTP1yaf/
qdB8fN8KVH4UzEChlJQULVy4UGE1T9PAOb/
qwma1lRjh0rpDwbrizpGKj493OiKAU0h5vhF+8eVntHXz66p5WljRWF6OV9l7jf67gm+sK7uDBw9q8+b
NatCggapVq+Z0nCrpi3fmKPya81UjNqpg4MKOWjPtY9Vq1EARsTWcDQcAwCnqZL5vzc/P1239hqr/
xaMUFRGjFz94SPnefLld/9tectWmr2SCPRTCQABiiwygUFxcnHr27KmgoCBde/
vdumfS+0p+dKqeeXeeuvW82Ol4AALAwFvv0qYvamn7r0ckSZm7c/
Tlu27FVjvN4WQ4HmutRj45Shc9cIX6z75bF913hR795xinY1U51lodys9VyNFyuVD0pedqzbKvZa3V9t
XrtfLjRdq5boNDKQEEmoyMDA0bNkyZmZllPr9//36lpaX5ORVQ8eb/Z6E6J12uqIgYSdKlXW7SC+
+PVPq+nbLWatWmr/
Ttng9UI5atJYFAxApmoJiUlBRt3rxZKSkpio6OVnR0tNORAASQsLAwvTnlQ7094zV9t/BL1a/
bWK9PvFv9+vVzOhqOY9qM1/Wf0K8Ven2cwiRZSR98/
alaftBC111xjdPxyiVtyzalPfWG0zFKsNYqN+RwqXFXSJDSvlip2V9+p2oXd1K1zo2167v1+nrKu2oeV
1suF+spAPhOamqqVq9erdTUVA0dOrRo/
MCBA7pj5Bj95omQNyRCNfZt0ZP33qH2bds4mBZVxYMjRikjvexfavjKxl826W+9Xix63KBWMw3odZ+ee
vceeYKzFRkdrjp1C77uDr59iF+znay4+Fg9MY5FAIAvUDADxcTFxWnChAlOxwAQwEJCQnTrzYN0qwY5H
QUnac4XcxV6XUyJsZAuMXp31qxTtmBOrF9XMf16OB2jlG2T31b+4SNyh/
9vG5m9875W9VaNFNmrvcKSEiRJoclnKbtpfR1Zs0PtL7nQb/
myZiz222sBcF5GRoYWLlwoa60WLFiglJQUxcbGSpIGjxyjTWffptAaBdvs5Xi9GvLkw1oy7VmFhoY6GR
tVQEZ6poZc7t+i9OcNP2rFD0t0cYcbisaiq8crqdZpGjZ4lIwxx7m6cpg4Z5TTEYAqi4IZAKD8/
Hz99NNPioqKUsOGHOzkBK/XqyVLPtGutB3qeWEfJSYmOh3JEcNG3qe09D1Ox/
hdNw8aUGps9aY1ilPLEmPGGP2welWZ830lMT5BE8Y+7bfXc8I5116mJZPfU3DbxgpKiNbh5T+rfu262n
lom2IKy+WjIhrVUfrSlQ4lBRAIUlNT5fV6JRV8HT+6ivnAgQPamBtWVC5LknG5lNepr2Z9+LFuuKavU5
GBcmvRuI0WLvtQ3/6yVGeefoFy8g5r5hf/
0tkdzz0lymUAvkXBDAABbunSBZo6ZbROSzqg7MMu7T1QW+Ofel0JCQknvhgVYtu2bbrngf6q1TpLEdFW
cx55UV3bXK+777rP6Wh+l5a+R9EplXPP6WiVnavWh7u179e9ijj9f9sqHVqTqXqdGyq6l//
+u6Slbq6weyXGxyutkq7GbZwdJNcPu3UkZ7NqJtZUSEaetm7PkLW2xA+41uuVZ1u6X1cVJ3IgMBBQlix
ZIo/HI0nyeDxavHixhg4dqtzcXHmDw0rNNxGRyty/
w98xgQoz9JbhWvr1PL265BEFB4eoV49L1bh+M6djAagEKJgBIIAdPHhQUyc/pH5XScYU/
CCUfThNox4cpH9Nmu1wusDx8Ni/
qst12QoODZck1WkoffVxqvps6KvGjRs7nA4n0rZPR306bYEy1+xTaLMo5fy8T+EZbrW65Ryno5XbhLHj
nY7wu5KTk/Wf2R+UGHtvzvsa9/
UnCju7VdFYzqer9NxDo9X7wp7+jgggQHTv3l3z58+Xx+NRUFCQevQo2FooLi5OMYd26nB+vozbXTTfrP
hI1439q1NxgT/N5XKpxzl91OOcPk5HAVDJcOoJAASwDz54V53bHy6x6i8i3C1jt2r//
v0OJgscXq9X+3O3Kji05JfkFucZpb77qkOp8Ee43C51H5isLu06ql5ajM7ueJYuuLWXjIu3i/
rLtZdfqZTYFnKlLlPu3G8UlLpMtzbsRLmMU9JPq1frwXFPauLUaTp48KDTcXAcKSkpRQeJulwupaSkFD
339LA75Zn+sA7+/I0ObVmnw+8/owFdW/
IOMQBAlcQKZgCoYpKTk096bkb6dt12Y+nxzIw9uuqqqxQcHFyByQrMmzevwu9ZER4YMUzp6Wl+f11rrX
bvSpNUs8R4zmGvPvtkgTZt3CZJuu32m/2e7WTFxyfqyXEckBpTN14xddkiwSnDBg/
R3zx3KCsrS7GxsXIXWzUInCoeeGKcFu09rOAu58uTlaG37hiqV0YM0xmtWzsdDWWIi4tTz549NXfuXPX
q1avogD9JatOqpZa8/rw+mDtPmXs36upx9yqebXQAAFUUBTMAVDF/pMDdu3ev/
n5PT51W739jeXleRUa31juzPlZycnKlLYQrWnp6ms6/
JsqR1854q4b2peeqRnyIpILS+fP3M1SrcR0ltTqk865uXKkPT1k20//
FPFCWoKAgVgfilPXT6tValJWtiF6XS5KCo2rI3jhYI597UR9N/
pfD6fB7UlJStHnz5hKrl48KCQnRtVde7kAqAAD8y6cFszEmWdJzktySplhrxx/z/
ABJT0vaXjj0orV2SuFz+ZJ+LBzfYq3lKzMAVLDo6GhdefV9eue9f6h5k4PKPuzWxi2xGjv+ZaejBZRLr
z1bs6d/Lnf4fkVEurTu+71KahyuZuflasvan/
TVstXq95fuCgpiRSYAVFUzPvxYwWedX2LMuN1ak3VA3QcNUTUj/
eWyS9T3UvY+rUzi4uI0YQLv4gEABDafFczGGLekiZJ6StomabkxZo61ds0xU9+x1g4t4xaHrbXtfJUPA
FDg8iuuU89el+mLLz5XjRrR6tixY6VeLVsVhYQG6foB3XToYI7mz/5WvW8JV0JSwYF/
0QkhSqyfrWULflSPPnxZBIBT3e9tZbUtI1OhA+spOCq6xHheULA8fW/WfpdLI2bP0BNjH1dClO/
ecRMo71wCAAAVx5crmDtL+tVau1GSjDEzJF0h6diCGQDgsPDwcF10EYdhOa1a9VDlebOVkFSjxHit0yL
006cZDqUKPHs27NLaL36Sy+1Wy25nKCYpzulIAKqQ3ytwDx06pIsGDZGt31CmcA/
xQ1t+U1D1SJnCg+RqXtVPUd4j+uiViZIUUFtZAQCAysuXBXOSpK3FHm+T1KWMeVcbY86XtF7Svdbao9e
EGWNWSPJIGm+tff/YC40xgyQNkqT69etXtXZHYAQIDZvjVN0//p/F7Cu/
Zmy9qoEqvIvV6r7RuyNf2fvzqYrGrauXa71n+9WpLUtEsL7fp1h3a5shRzQ13ZfK++/
OArNa7dQM3P54AtIFANGzlSaen++SVfQt5hrRn/
oGx8TR3ev0+h9Ruq7jX9S8xZt32Hbh50R9Hj4p9XJonxcZowdqzTMQBUItZafbPqc33z/
RcKCgrWReddoianNXc6FoAK4MuCuaz3V9tjHn8oabq1NscYM1jS65J6FD5X31q7wxjTSNJiY8yP1toNJ
W5m7SRJkySpY8eOx94bAICTllQv0bFD/or7YblLP36xUWd0/
d8q5u+X7FXvq9upRdt6x7nSOctm7nc6Qrl8//
E32u3OUvTNdSVJKxes1v5VaWo4qlPhDLcSbmisDa+sU5OzW8gdzB7YQCBKS89Q/PUD/
```

```
PJa8ZJOl5Tv8ejHhfO0r2lrGff/fmTzejwKrR5ZlCfeL6nKJ+2d15yOAMAP9h/
cq3c+el2HDx1RvvWox7m91LZFpzLnTn3vRdUKb6pbL3hEuZ4cfbBsijY1/
EU9u17m59QAKpovC+Ztkor/
JFxX0o7iE6y1xZcCTJb0ZLHndhT+50ZjzFJJ7SWVKJgBAKgK0nfv16K538q6cuXNMzIK1o5f0lU91uhg
hlWDhvUrbbl8qso9lKMd6buUcGuTorG4Pqfp0Lb98hzMU1D14KLx4KbVlbU1XfGNajoRFUAAcgcFqfWF
PTXv5RekS65WtYZNlJuVoT3vvq7zL+/
rdDwAkCQdyTmsZ6eM1W0XPayYyAR5vV7N+vIVHco+qHM6dC8xd+eebXLnRqh7lyslSWEh4br+/
Lv10twH1f3s3gpy+7KeAuBrvvwbvFxSE2NMQ0nbJfWTlFJ8gjGmtrV2Z+HDyyX9XDgeIym7cGVzvKRzJ
T3lw6yOy8jI0Lhx4zRy5EjFxsY6HQcA4CfZh3I0Z+Yy9RmYoOCQ6pKk7xdnqUZoY7U4o74iqoXI5XY5n
NJ/0rbsUtr4XT5/
naysLAVfXb3UeFS7BGVv2KuotglFY0d+PaCdS9cpM3STz3MBwFFBIaFKvvNurVmySOmfL1J4RIR63nCj
ImL4WQFA5TB/2Qe6qssdioks+L7J5XLpmq536uV5I0oVzD/
8vEIdT+9R6h6nxTfTrj3bVLdWA39EBuAjPiuYrbUeY8xQSfMluSVNtdauNsaMlrTCWjtH0j3GmMtVsM9
ypqQBhZe3kPSKMcYryaWCPZir9OGAqampWr16tVJTU3X6aQ20dPYHcnut4hs31OBhf1d4eLjTEQEAPvD
l4jU67+poBYf8r0Ru3yNG8179TZ26NnUwmTNan9lGael7Kux+aVvKLqvDwsKUtn6fdFbtEuPZ67IUdVa
tosc5Ow5Ka7MVGlenwjIVl1i/
1oknney94hNOPAnAKSUoJFRnXNzH6RgAKplNWzfovokpJ57oYxk7DurFQbeWGs/al1Uq3/59B9Sr2V/
UqHbLEuNrt32nL995T0FBrGAGTmU+/
RtsrZ0rae4xYw8X+3yEpBFlXPelpDa+zFaZZGRkaOHChbLWauabb2lgyw56tFknGWO0KTNd9//
ldj2f+maJA58AABUrPj5Ry2ZW3CF/
27ee3L12pO9Q+0sTS40fPJDj00P9kuqVfs3yio+vuHtNGPt0hd3rRFL+erPW7sxQSO1qkqS8XdnqFNRK
NdZFaf03m+QyRp1qttKEz2crNDTUb7kAAACOZ968eU5HkCS9Me1trV+/Uk2T2pUYr9MwXi+/
8XqJMWutbu9/j7IO9Cxa8fzrzh/Vokt9PTbuFb9lBuAb/IqoEkhNTZXX65W1Volet65t/r9/
nBvExqtHZoKWLVmqC3p0P85dAAB/xpPjJjjyum+lTtN3659V3aZhRWNer1WTBu30+r/
mOJIpULz29BSNfGqUvv/0J8lKbWu30rhnx1AmAwAAnITrU67RoP5/
VXREvBJj6irPk6tZ30zUtbeWPrTPGKN/
vjJW4x97RvtWH5E1XjVuXVeP3FdqzSGAUxAFcyWwZMkSeTweeTweNYiOKfX8mTWT9NG331EwA0AVdP21
N+rj29+TO3i7ajcM0+GDHn37sdGDd49yOlqVFxISogkPPXniiQAAACglNDRUL057Wi8/
N0WLf0iXO0S6adi1an9muzLnR0ZG6okJj/g5JVDxcnNztXr1aiUmJiopKanMOYF21hoFcyXQvXt3zZ8/
X9ZabczIKPX8Nzu3ql2vqx1IBiCQpKena8rkZ7Qnbbvate+qfjcMcDpSQAgODtZrr8zWtDdf0Q/
zv1Z0VLyeG32f6tWr53Q0AAAA4LgiIyN1/0P3Oh0j4Hz+6ef6z9tzZT1enXFBW1130/
VyuQLnYHAnzZ7+nr56b4E6VG+kpTmZ2ln9iB594UlFRESUmFf8rLWhQ4c6lNZ//vD/
+4wx1xljNvoiTKBKSUmRy+WSMUYZQUav/bhcXq9XkvRz2k59kbdf55x3nsMpAVRl69b9rL/
d3Uf1Ev6ji7quVeauF3T7bVcU/VsE3woNDdXggffo5X+matxjz1MuAwAAACjTGy+/obUv/agRcX/
XqNoPKOHTaD3Mux/9YtOmTVr93jKNbn+zrmjSVXe2vlyDY3voqQfHlJhX/Ky1BQsWKDMz06HE/
vO7BbMx5syyPiS1lnSa/yJWfXFxcerZs6eMMbr6xhS1+MsNGv3bSo1ev0Jf143SU1Ne4YA/AD71/
LOjdP2VVrExIZKkpo3DdWbr7crK2ulwMgAAAACAJOXl5Wn1/FXqf3o/
uV1uSdI5tc9SnbSaWrd2ncPpqr53prypgc17lxirExWvI1uySowdPWtNkrxer1JTU/
2W0SnH2yJjhSTrryCBLiUlRZs3b1ZKSopiY2N1PvstA/CxHTt2aN5/
ZishobY8eXsUFFTyd45NGodJNs2hdAAAAACA4rZv365GIQ1KjZ8d21nfff2tmjVv5v9QASTf41FQkLv0
E7ZkfXr0rDVJ8ng8Wrx4cZXfJuN4BXO+pO8krTlmvImks32WKEDFxcVpwoQJTscA4IARI4YpPd2/
Re6m335WQmyGup9XXau/82jVqjRd2btxiTlZe/Pk9Qbr9ttv9mu2PyI+PlHjxvFvJwAAAAD/
GvXAKGWmlz5Hy5c8Ho/s1nzd3KR/ifH/7lyupe9/rs8/+7xobMhtd/k128mKjY/
TmCfHnHhiJdR3wA16Y9QU3XXGFUVjmdn75a4VWWLe0bPWPB6PgoKC1KNHD39H9bvjFcwrJf1grR1UfNA
YM1DSOT5NBQABJD09TddcHuW319uxc69c+Vm68pIESVLdOtKBg3lasChdvS6MlyTl51vN/CBT99/
bQ9HREce7naNmzmGFNQAAAAD/y0zP0GMX+H/v48lHpujTLZ/
pgvoFZ3Vt2rtZqw78pBdvfdbvWcrjkU9PzXJZkpo3b66vuzfX2E+m6/
yYltp6JF3fe7dp9MslFz2lpKRo4cKFkiSXy6WUlBQn4vrV8QrmNEmvG2NulvSptXZz4fhHkti/
AQBOUd8s/1XdupYstLueHatnJm7Wtp0uhYRY5eUFq0/
y2ZW6XAYAAACAQDOwz22a9cVsPfTfR+WSWzViojSy/3CnYwWMAXcN1L7++/
Tfr75Wp7rddGvr1qXmHD1rbe7cuerVq5diY2MdSOpfxyuYkyVNlzRNUj9JmyXJWrtL0i7fRwOAwLB1a5
r+OdF/
K3Ez0g+oQ9sgRdcILjHucgVrf3YjmcMFh4rOnJMlKauMOwAAAAAAnGCM0dVd+0pdnU4SuGrUqKFeyRcf
d07xs9YCwfEK5gxJEyUZSS8aY54s9py11jYu+zIAwB9Rr16iX7fIOHSont6evlC33RQqYwrK5D3pOUpM
jNWNNzTxW46KMHPOfqcjAAAAAABQQqCdtXa8gnmspJGSIiVFSeJ90gBQBVSrFqrzz+
+gyW+sVO1Eo4PZVvneaup3LdvrAwAAAACAP+Z3C2Zr7bOSnjXGLJE02aW1JLe/ceVnH+xAKA0a60+/
ODf+nzR+woJDdf1Nw9VmzPaOh3rlNS8aW01a1JLe/ceVnh4sMLCgk98EQAAAAAAwDGOtZbu/
AJXCQ/fdoYbu/2pAhzDl5FnNePZW/
dL77+p77Y1ORzslGWMUE8ObUwAAAAAAQPm5nA4AACdjzZrVqp69XOe2DJcxRmEhLt14QYgWfTBZXq/
X6XgAAAAAAAABiYIZwCnhq2UL1bGRLTVes/pBpaWlOZAIAAAAAAAAJ9wiA0BJ+/fvl8fjUsrsrNNR/
rQHhw9TRnrlLWcHD7y56PNdu3fqwkY5qpsQVmLOmo2ZGjXi/+R2u/
2aLS4+UU+MD5wTYQEAAAAAAMpCwQycpKysLI0dNkTR2bsUbKx2KEp3j/6HGp9+utPRyi0jPU13XFJZi/
KSuayN0VOvbFKHJnmKiSw4kO77Xw+pXlJNpVye4Pd0r3xceYt5AAAAAAAAf6FgBk7S4/
feofub5Co6PF6SlJfv1f33DdYLM+f7ffVsIDLGaMjNF+ntOf+VJzdL+V4pKam2brisrdPRAAAAAAAAAh
```

YFM3ASdu3apTqe9KJyWZKC3S71qWX12adL1a3HhQ6mCxzVI0I1sN/
5TscAAAAAAAABAIQpmOGLU8PuVmb6nwu63YdvOCrtXWbKzs3VlQrak+BLjkcFWD9x/
n6LjE497feO6tSssS2x8gsaMf6rC7gcAAAAAAACUFwUzHJGZvkeP9GpegXesyHuVZq3VY5Omy1orY0zR
+MfrM5V6d19VDw/16esX99iCtX57LeBErLXyeq3cbpfTUQAAAAAAgAMomIGTYIzRlT266v/mLlP/
M2qpWohbM37crVYtW/q1XAYqC2ut5nz0rdIz9igk2OrwEbe6nd9OTZvUdD
oaAAAAAADwIwpm4CS1a1JfTetfp8Xf/aycg3kacE0XxVSPcDoW4IiP/
7NSTRoe1GUXF2wbY63VtLdXKDGhu6Kj+XsBAAAAAECgoGAGVFCOzV/
+k1at36ggt1vJ53RQy9NK75scERqiS89u60BCoHLZnbZbfS6KK3psjNGVl0RryaerddUVnRxMBgAAAAA
A/
ImCGZD0j+kfqVvtID1+QZJyPF5N+uoLbdnVVMldznA6GgJAfHyiZs5JczpGmbZuTVO9eqUPsczOzi81V
iMqWKt/3q98s98f0SQV/
NkBAAAAAADnUDAj4K3bukuNwvN1YZM6kqSwYLfu6dpQ989bo16dWsvl4vAy+Na4cROcjvC7kpOTNXnyG
6XG7xx8tfLytig4+H9/
P1aszNHox19Q9+4X+TMiAAAAAABwkE+bM2NMsjFmnTHmV2PM8DKeH2CM2WOMWVn4MbDYc7cYY34p/
LjFlzkR2L5fv0ndG8eWGm9YI0Tp+w85kAio/
O5/4GlNnx2kTZuPKPtwvj7/+ogO5XRSt24XOh0NAAAAAAD4kc9WMBtj3JImSuopaZuk5caYOdbaNcdMf
cdaO/SYa2MlPSKpoyQr6dvCa7N8lRf+tWHbTt08dafTMSRJ+7MyldAyRA1iq5UY/
277Xs2d+Q0rmIEyNGzYSFNfW6x33nlDqzf8oiuuvUadO5/
ldCwAAAAAAOBnvtwio7OkX621GyXJGDNDUh9JxzaYy9JL0gJr7f7Cay9IulzS2z7Kigqlatl9zPiSVW/
j979x0nVXn9cfxzdulF6SIggggidsWGsSui2Yy3FkoLtthN1Fhij/
40ioo12Bsqgg2NFcEuoChVRKSDSN3d8/vj3CXDgqjLzt7Zme/
79eLlzp3PcVvOc57z+Kzke18BegGPZCZWq9dWad2q3N33p2TTsMIBb4u+SuR9mx42JaNqoLwDsTZrJ
Bpw7csN9uKUe3ostf/jLtEEQAqF+/
Pscdd1LaYYiIiIiIhIirKZYG4LfGpvvxAcpmgnk40MZtNnMOpxZEyt7Yc5va
9RjPoUf3ZOnSpWmHVrAGDBjAyJEjGTBgAKWlpRx66pFc8909jOw9m+c6DKPXVQcx6vqrv/
yDZDnuzoOvvUC9PbfCkkV167VtxuNTopN3TopN3XM1wzqdw8TG9bju+ZNGfLwxg/
4qMVXvfFq0Mp6rENL3r3ouaF6HWneaw/
q7rkLH7/4UgpRi4iIiIiEh1cyMTVVOe5V+zfQOdh0ckqS/
snLzCeeFlk78nMeefPmxZqZmvQ12ZQAqYJZ0
jxmzZu4Q7Q7774t7dAK0Iwv2DxxL+q2YMk1+2oiPH3pDATkrcVGZ/aN/KkWM8HZmvMkPdZ2ucs/
cF5xwTT8ee5/K5RrhGmtsUDk3xmd5jAqK/GvXUQgfjRGiRAnRcpFqDBfjw3Nn7Bh0vRtC2w97oy7ruuwBr
g9bhqOHPCuxTeCbhJ9YAcYSyaQLZ3PG+vWtW4OSwHAD0LUmWRi2Fo8wyv/
ZvcrSqj6M09Xuq5UXH+WZFMMTSbCIOIuNUSN0QSBq7JaaNxfuqsfjWaQP/
FTvAGdyspr3cpbhBf1KBN3rDJjJTmPaBqdRzZJj7wQOs9/
NfdlINdfXvpEe/D6M63TX7AQ8LS6d6lDkdyE/
SzWEJbOJFjkxLiOY7Tbp0nGqEPzNmebIzeVzQkvpZwxd/IWfAD8/CGcX/gV8X7r5LJP4/VLRcUaJ4rz/Q3
2NmagxY1SPxa+LTdcpDS48CbW8e+c5FqtvRGfEsodyt8WPKdkg0a8P/
LdjbGZNm0rPIX+RQRHrAu7r/yUcMVs3c2vSlLjjM9enX1EvfxTgXUOVyxRIdAGjPVgRpxjovI6JGsD9Ryue
kQs4S3BHp0CTHmxMG/
XTKVBhnNPF13bzjl23+dqkYdSSDB2YY35f1yWtbjs6/jdxWk0tmn+CTIJvrI3rOjUxv4rpuUf9Kmw4mm
fvxW0QE+FGMLA3iIQuQ5GDrjUNNi6VTTf1e+yggQYhzIWYp4UZOa8h+F+o7cSo4Mx1EPx3KNGYmTLIh
Q2nyw2CqdNc0v/uMJo25SK2R+vK3S5ExDdeJt6xpXPGgIu+WvA0aQpLLwPt9vjKE4jNxwGOOq/
kDuIGW1xYIbc8NYObY0b9ZRGlZZiUzh+Gds1VOuSxmwFTc/
zNgfkhEy7wLtXR86z5h0thcy2PkYRNzLDGLdaStsdAjZSAv6TXD9VP5MPQP2dR1NLhK2

t0cWmy+fPFUqyZtee2TmSu0FFMVc3Z9+M3H7Np2pxW2b9liC8ZM+ZqN2ndLISopZEowiySOOv5k7v7Xf
C5+9xUas4R5xQ35cc31eXHiPDabs4BP5haxtN0mXHTs8WmHKiIFoM/
hx9Hn8OO47bbbGDRoEL1791YVSEqaN2/OnnvuyaBBg+jZsyfNmjWjWbNmPHP7Y2mHJlItWrduzcB+9/
DDDz9QUlJC27ZtV/
n6+vXr07lz51W+RkREah4rLqO0rJTiov8t0FtSWoLVKluhpdjrr7+e9wnmsd+N4+gBf0zld8+bN4+jmx
/Bdu22XW7759O/
4J7XHqBu3bqpxCWFSwlmkYSZ0fcv5+FnnsvSpUuXLVQwZcoUvvvzyS47p1o3WrVunHKVI1ZgzZw5jxoyh
U6dONG/ePO1wZBX69OnDxIkT6dOnT9qhFDTtBxFYa6210g5BRERSdNgxB/B8v7s5cOuTl217/
qO7OeLkg/hgxLDlWorttttuKUZaPTq1XY/Ld740td9/
yb2XMXvhHJrWbwLA5HmTWVx3CU8cPcPSC1mH6Nv715ZdohSBYowSxSgZktSy4DtGnThjZt2qQYkUjVcXeu
ufoCJo0fStvWC/h+Wn1arLU9l1/xL/XHzFHNmzfnhhtuSDuMgqf9ICIiIoWuxw7bMXvmHO57/
BKKS+tRVryI/Y/Yi22334Rb1u3RarqWYBuWz77wjzuHmgbfBIgCnduO6nP37v6QdlhQoJZhFRArI008/
ii8ZwgF71wWiL+ZXX79F/3tv5/gT8nsKm4iIiIrJ7e+/
ei9/69Vti+spZikl1rNlyT8488L+0wRADQ8twiIgVk6GtPscUmdZ0BOBRTuCGmqgy8
8fNtpoI1UvixQgVTCLiOSZXr1WrCgoN2fkCgoN2fWaHrvvvmIv8U8//
eRnv29VP68yBg8eT/
T0RERERE0qeWYiKFSwlmEZE8s6oE89dff82PP/6Y9ugWtE6tWtWqwB9f4SbFn/ZP17Zc/eJKAEJ9H7q3xO/
Nq+4IDhrnk8bq/tbRLjyp0VGVNYIfGN/M7GGQL1BbzAc0bEtpHXN93R7ag4vOWh72/I+ZvAd4WHZYG2ZvKH/
Avh/zAXNfSh3zo03a6h9zSs4t+TmVQ4F5UkuEBOgRWIBYGgBwmSd4tZe5qMyB2ZvhD7WQOE3P2AT8iI4bUbe
xU0g/XUMlkIjFf16dDXIWVKXOSnOTvQt+oNGT1q5N3ht/bzjf5Y5+PYe8bV6x2kfyPQCzqd5k25aZI7uRr+V
+mj8fd0DAPT5SyAI3sdRPdrhHxXiWWhTPTxa59HjHCN/
PJh5X4S99DDffsWJOtNxXAQDwl4o3hgNpxkyJmpNePL+eq6WLiWaL4JjyTjEhBw2PuPxNjUD8N77O/
Uo73Iazrm4zRpLAGvkwSYYn8wvl/hvoDVAxVNr5JTSXbbh7/
TlPaYcc4TPTMFkS4Yx6F1hJOpmQ6WFPjtIRcTbK8c5qivwF24+TN/xtDkB7zfWwQ9/
4i4iIiIiIiIuVMpwaDJHbnw/J1nTyTA0CxMFY7SBkPm0HyTtwfGQn9fe1/
mijlOZ3r8FuhhevdrcRQdUPyvmaFPUeREREREPvI0rxaDzGdESF+zQ/
e7Ph3SYl+rFUq/ZMJl6JAiyTIVTpoRrqrhs+ctFV17TXlGVIJa0iUa3/
Pu8iKd8UkHlNxNUOkl30Bs5fkZoa1xBfYwfwTULaAwN+OvJnhp7ajj+/
yhRVFjLQl+wtttaPpY3+OMWiuP4b4iZ/ZPHuNKGzPJOUmw7b4ruIXq/
FZjvIl9JawKW9LovbZy1ZrLwWk6A1ZOcNNj/
iX/BQUfLhQs6l3/
bJoW3sw5vIsA6IsEwKt8gWUI4uYtgRIqFDkfxq2EwkgAJwIizDmMTe4Zm+tqY2OmY83Bb/
F+IiIuX/KT7uI6vPu/
dvStrGROjjBhXyIzMpwGqUivcKK1wP4HKYdaDO+sLYyFn1ZYBjaAKSXpuAr0Z8A13N+aI4LBNsdaBs9Zo5
SRWN56NIuLUL2Oe36ttERERERLLXmv7tV2QQnuBWHNJbpOSYK77XEOlE7Qn3fnF1lvpT0G+FO5Vf6Vpww4
a4VHs7SRvWZVgTP0+IlREw/7j6FE/
wJRHe9g74S3QeGJBx7btBWpSxKaaKQ2VgrHTC3Muv+LBB0a5mKIHsd5pbeexNS6C7RXX5Zr2XLfaPq3cy9
y0pqoofHZkuKiIZZ3/
LaN62Xv3ExxRKWvvoNyfY9gAzfIltD5hbu8S0tKXJjqIpywCUeE6b+PSxK0c81XY+n58tt2qtFOAW/
8YN8UFZ2k+4JDUWPeG0oaHkrnG/VlF+DndOy/nn0ufV/XrfOLlTwMEQNV2+IMTTUnz/
DDm6XFxfN+2ruFIqsNQRAoTWrNvhnWtVwuIVldkLVotcEMmLmaxnZlqMffntH6wZHAz+UpRMPL1XTAgRdX
2jxj8GS1pmSb+XcYuZYUdWR0jLewHx83cwUkerxYpsAsHrq4avaWRgHMsYSGTW44upcm+1LbpDBWZ+3gn5
NMUEuEGgM07EWnZQhy7AnkRbyzYV6cIWBXC9/WLeXIAOhg4tz9oVmhfnDsVPpbqPJFFGfx9rsx87/Hw08/
m+Z1+yx7zz3/MnefJWfXE66vrfLIbEmzBgwNEsJjgOYpBxkb2HMjVHCdZG8BdO6P7F7kz79M/
q/SyGmxWhVBfaoSYHrpXehn8N/
5k8HaMMLQ8BLmVm6+QyKG0CbfKozWMfxrbgUphE0YSEjqdywRTq3rVg6BU0ex9N+IdGUZSndBVqD3b5kfW
89zSGDrbC3NmGGiw+R3Ns0K+QNCPnwdYQaSUOWqtZdXtZEn+7k6ZqmHxKOwPgGSvhZwYPqGP+cS9XIhw3G
7TKnEgrDEn+hNoyS7Iwi8tu3SEiJnwO/VXvRnKCTVjNFkIMKlj2GV4KbZqAm4jX/
s6/bAAYyq2hhhv0kOGwdx6E88OFXHdW8GGb3iHpi2FI/
wkqLk3kjRz9zk3Rt7s2sSvmcpqHXGYMbBeQz3FH76A+nSPwRKBfvH2TNAR2gHRPmI/
+s1F1RZc/GfTJ3+x8pg8VVj7VPHNdj5T4stmvuXNNXfxdM6LZSe/
thCsUNmWIlHAXvIbIYNSdv0t0eWG/eOT/
FZjv9TgDdgi4VYZrFdMumHGdpznMqSsdN6hFvX+ENRIvPwiVYXHxNO8UV4CI4KMg/
4R0XcIUWB5XqshzrHsfUgLx6DGr0OuxlT4RpT0dFhEHY8NR9RJaS/
mrZwlbQnEhXDu+p61oqOHVd4HkHEi4Bs80Q5IX2ZfW/
NzhexGcIjkZbrE8jowfEJP7rxvtJY2qTFAGuJqJv

e6L5V2mGIiIhIAdEifyIiIiIiIiIiIiJSKUowi4iIiIiIiIiIiEilKMEsIiIiIiIiIiIpWiBLOIiIi
IiIiIiIiIVIoSzCIiIiIiIiIiIiJSKUowi4iIiIiIiIiIiEilKMEsIiIiIiIiIiIpWiBLOIiIiIiIi
IiIiIVIoSzCIiIiIiIjlk1OjRPPDoo4waPTrtUERERER+Ua20AxARERERERERoKSnhuLPP46v6TShbrwt
F9z9O10Vzuf+m6ykuLk47PBEREZGVUgWziIiIiIhIDril392M6bY19Xffh4Yd16f+7vvw5YZbcUu/
u9MOTURERORnKcEsIiIiIiKSA94Z9RX11uu83Lb663Xh7ZFfphSRiIiIyC9TiwwRERERERZFq0qtXr599
7su581lnvz6Y2bJt7s7HI4b/
7Pet6udVxuDBg6v054mIiEj+U4JZRERERESkmqwqgfvI089w44h3abD1Dsu2LRr+Dtdd8Fd+f+AB1RGe
iIiIyG+mFhkiIiIiIiI54MiDD2L/
Bkbpkw+w8M0hlD35APs3KlZyWURERHKaKphFRERERERyxGVnncl5Cxfy3Xff0bZtW+rXr592SCIiIiKr
lNUKZjPrZWZfmdk3ZnbBKl53qJm5mXXP2HZh8n1fmdle2YxTRERERERQkV9SvX5/1119fyWURERGpEbJW
wWxmxcDtwJ7AZGC4mQ1091EVXtcYOAMYlrGtG3AEsBHQBnjVzLq4e2m24hUREREREREREGR3yabFczb
AN+4+zh3XwI8CqysediVwHXAooxtBwCPuvtidx8PfJP8PBEREREREREREHJEdlMMLcFvs14PDnZtoyZ
bQGs4+4v/NbvFREREREREREZF0ZTPBbCvZ5sueNCsCbgbO+a3fm/
Ez+prZCDMbMX369EoHKiIiIiIiIiIiIiK/XTYTzJOBdTIetwOmZDxuDGwMvGFmE4DtgIHJQn+/
9L0AuHs/d+/u7t1btmxZxeGLiIiIiIiIiIiIyKpkM8E8HOhsZh3NrA6xaN/
A8ifdfa67t3D3Du7eAXgf2N/
dRySvO8LM6ppZR6Az8EEWYxUREREREREREREGR36hWtn6wu5eY2WnAEKAY6O/
uI83sCmCEuw9cxfeONLPHgVFACfBndy/NVqwiIiIiIiIiIiIi8ttlLcEM4O6DgEEVtl32M6/
dpcLjq4CrshaciIiIiIiIiIiIiKyWbLbIEBEREREREREREZE8pgSziIiIiIiIiIiIFSKEswiIiIiIiI
iIiIiUilKMIuIiIiIiIiIiIhIpSjBLCIiIiIiIiIiKVogSziIiIiIiIiIiFSKEswiIiIiIiIiIiI
iUilKMIuIiIiIiIiIiIhIpSjBLCIiIiIiIiIiKVogSziIiIiIiIiIiFSKEswiIiIiIiIiIiUil
KMIuIiIiIiIiIiIhIpSjBLCIiIiIiIiIiKVogSziIiIiIiIiIiFSKEswiIiIiIiIiIiUilKMIu
IiIiIiIiIiIhIpSjBLCIiIiIiIiIiKVUivtAEREREREREREKRyfvzxR2667EYWTV5AWZEzb8o8ysr
KKCpSXalUDyWYRUREREREREaiB356ITLuDctc+kVZeWAHzc6BOuOf9qLr7+kpSjk0KhoQwRERERERE
REZEaaPiw4WxftA2tGrZctm2L1puz9JvF/
PjjjylGJoVEFcwiIiIiIiIiIiIp6tWrV6W+b+60udy61Q0rbF9jUSN69+5NgwYNVjc0AAYPHlwlP0fyk
xLMIiIiIiIiIiIiKapsAnfSpEk88ZdH2LhVt+W2f193Gq8//
zq1ain1J9mnFhkiIiIiIiIiIiI1UPv27SnpCs+Oe4EyL2Ph0oXcMaof2xy8rZLLUm3M3dOOoUp0797dR
4wYkXYYIiIiIiIiIiIi1cbdGfrKUF5/5jVq1anFIccfyiabbpJ2WJIf7Fe9SAlmEREREREREREREangV
yWY1SJDREREREREREERCpFCWYREREREREREREqRQlmEVEREREREREREESkUrKaYDazXmb2lZl9Y2YXr
OT5k83sczP7xMzeNrNuyfYOZrYw2f6Jmd2ZzThFREREREREERE5Lerla0fbGbFwO3AnsBkYLiZDXT3U
RkvG+Dudyav3x+4CeiVPDfW3TfPVnwiIiIiIiIiIisnqyWcG8DfCNu49z9yXDxsCn
sV4REREREREREREERKQKZTPB3Bb4NuPx5GTbcszsz2Y2FrgOOCPjqY5m9rGZvWmYWYjF
i5cWJWxi4iIiIiIiIiIiMgvyFqLDMBWsm2FCmV3vx243cz6AJcAxwLfA+3dfaaZbQU8a2YbVah4xt37A
f1+7meLiIiIiIiIiISPZks4J5MrBOxuN2wJRVvP5R4EAAd1/
s7jOTrz8ExgJdshSniIiIiIiIiIiIiFRCNhPMw4HOZtbRzOoARwADM19gZp0zHvYGvk62t0wWCcTM1gM
6A+OyGKuIiIiIiIiIiIiIiI/
EZZa5Hh7iVmdhowBCgG+rv7SDO7Ahjh7OLu2ApMJtojwGwE3CFmZUApcDJ7j7rF37lylpyrPgis
8FAi9/+f1QtWgAz0g5CtB9yhPZDbtB+yA3aD7lB+yE3aD/
kBu2H3KD9kBu0H3KD9kNu0H7IDdoPuSGX98MMd+9VVT/M3NW6OFeY2Qh37552HIVO+yE3aD/
kBu2H3KD9kBu0H3KD9kNu0H7IDdoPuUH7ITdoP+QG7YfcoP2QGwppP2SzRYaIiIiIiIiIiIiI5DElmEV
ERERERERERESkUpRgzi390g5AAO2HXKH9kBu0H3KD9kNu0H7IDdoPuUH7ITdoP+QG7YfcoP2QG7QfcoP
2Q24omP2gHswiIiIiIiIiIiIiUimqYBYRERERERERERERGRSlGCWURERERERERERERERQqRlmERERERERERERE
REakUJZhFRERERKTKmFn9tGMQEfmtzKyujl8iIpWjBLOI1BgWaplZUea2NGOSZftF+0EEHZNymZk1NrO
GaceR75LPwN/
MrHHascjyyq+fzGwbM2uRdjyFwMzWNLNaacchq2ZmxcmXfYBD0oylEJlZ04x9ICI1lBLMeUQ3tdXHzIr
0flcvMyvyUOLuZeXb3d3TjKvQmFk9M1vHzJqUb0v2i/
ZDDtHxKT2ZnwUNvuSGjJvWC4G7k23aL1Us4z3tARzo7j+aWR291zml/
Ph0H6AEcxZlFENcBayXZizyq5R/No4ASgB0/KoeZtYauAco+6XXSnqS/
Ed3VfhXv5p0HFKCuYZKTngbmFlXM2sASrRlm5nVN7NdzayZu5eVv9816QNf05S/
t2a2HnCFmY0xszvMrJWZNTCzLma2ciqdDgfeAJ4GrzWwNM9vSzC4ws56qPEifmdUDnQ/
SklTgnJycn+tUHHzR+SId7l6afFkbGJJ8rX1R9crf0xbA22bW2N2XZF4v6TOQLnf3pLJ8FPAtxACMzt9
Vz93Lknu0A4FvQAUquSyjeGUC0CzZtkT3e9mT8Z7uCKyRHJ+Kda7ILRmDZbsCfyP5fJjZ9mZ2lpmtk1p
wBSL5bHRJO45fQ9N1ahAzK3b3UjM7CvgLMbr6NTDDzMYCE4Fh7j49zTjziZlZ8oHuDJwP7AG0NrN/
A9cCRwJ1zOxAzZdx+TZqx5yoiKgnuJC75bgIOAk4BNgS2BKWZ2kbu/
lVaQ+S65SSoi3v8ziQqDs4hKwObAUuA44BRgaEphFqyMc8PWwGlmdjAwBngFeBH4xN1/
TDXIPFd+riCq1E4HTgS+MrNxwAfEPxCq1qEw0t0XpxaZlM8EKyMqzDsBlwAXZgzAZB7LZDVUOCe8D2w
LvJc5A0/vde5JClo2A441s62AN4CPgC8zPydSZcrv82oB88xsM3f/
NOWYZEVFxL3fn4DR7v6dmR1NVPv3AA4zswPdfVqQeajjFzUJsDjZnYQ8BWRB9kOGO7uH6UaZAWm81rN
kfEHNhm4AvieuHDZEGgKtAYucvd3Ugwzr5RfjJvZLcC6RODgbeAmRBwBkwXFinEtMB9WBtYqZWTNgrLs3
TR5vQ1ys9wEmEwnPUuBEJdGqXsZxZ2/
gBnffKNneHXib+AwsJAa91gF+r4vw6pWRYP4EeBd4nLhBOiz5b0PgBHfvn2KYeS1jH1wKbA3MIs7R6xI
XgGOB0cAAd38/

vUgLV5I4+AxoAHxHDL48TwzMz0gztnySVJ0dBrQFNiYqnYqBecAM4FKdq9NlZr8HziXuH6YDbwGDgTfc
/
XslPldfxrXT0cD1RFHQQ8SA48fuPjHVAGWlzKwdsDPQhvh8rEkk1xx41t0fTDG8vGVmdxN9rz8nZhl9l
fybqPNF+jKOZyOBM939VTP7CPi3u99tZoOAe9z9aZ0/qlbG/
cVVQEd372NmPYmWb12J69nj3P2LVAPNoARzDWNmzYFX3H3LCtvbA1slz81PJbg8lJFg/ho40t1HJNs/
Bx539yvNbE3gCeLA+nia8eaTjJPZMcDp7r51sr0LMNTd2yaP2wHvunv7FMPNWxmfgXuApe5+SrL9BOAw
d98rebwHcL27b5FiuAUrSeq8C+xZ8RxgZj2Aye4+KaOCTbLAzL4HNnP3ack+6UBcBLYiEs67AWe4+5Cf
/
ymSTcmg5X5EErQH0AS4191PTDWwPJR8BjoD3YENgKbufka6UUnStqEYaAdsD2xDVENtQZxD3k4xvLxiZ
tsC6xN//+sRA77lg/B/c/
eRacUmq2axMOMWxL7rDgxy95d1HVX1kuKhdsQAfSegDlG8MhM4y92XphiesGwti0uJa6aFRPFEL3dfbG
bTgR7u/rUSzFUr4z78VaJIpb+ZvUDkPq42s4eIQon/y5X3Xi0yaojy0QtiRHWYmZ3q7neUP+/
uk4iKWqlCyQe6MVGZOdvMGrj7AqAl0C95zdwkwT8rxVDzUfm0qZ2BEjPrRVQBnk5MWSu3LfBDtUdXIDI
uotsCnc3sXuB14FTgroyX7kckOKUaZVxMrE9U9PcGHst8jbu/m/
G1boqyJBn8Wkr0+S3vgz3ezM4gPjN/
BC4DjjazN9QuoPqZWRN3nwU8kPzDzDYCGqcaWB7IGBReE9iJqP5rS8x8GZC8Zs00Y5Tg7gvMbAPgu+Rm
9RGgETEY8EG60eUXdx8GDANI7ie2IRJoGxOVZ5IjkgGxzYE9icGA79z9SmC4mQ0g6TGv66iq5+4fAB+Y
2TPELKMtiaR+Q3dfmiuJs0Jjsa7LEo+1p0qTllc3EjNfTkiSywcA8939a9AaMFUt43jzMrCzme1AfEYe
TrZvB5TPrCjPnaRKFcw1jJndCpxGTLV6EngB+K+7T041sDyWfJBfTcGkMwAAIABJREFUJKadTyDe+3OJ
i4+FxNSpGUATHVSrXlIpewAxTe0nosrmVeJgOpJYCf19d788tSALgJltR1RxbAJ0IS7+JhG9NV8i+mT3
cXf1YE5B0pPrViJR9hLwJnFuGJVqYAUkqe64iRiQPKt8CrSZnQmc4u5dLfpkP+7uHVMMteAk+
+YgYi2F9YBPiPPKAiULqkZGgvlOYuD3BeJ6dV+iJ/ymwDvuvijFMAtWRhXUZsAxwEZAT+Aqd7/
UYgFrFUpUgYzPQiMiWbkp0A140N1fTF5Tx92XpBmnhIzPxu5EX/
LpROXsNu6+VTIdfbq7f5xqoHkm43OyLnAOkRxbB7jA3b9MNzoBMLPbgDvcfbSZtfIKrUCTGWF9gcXufn
NGQaSspmRAfmH5ecLMagNXEp+TR9z9EzPbEnjJ3ddKMdQVKMFcA5lZXWAvYnpnTyLxtgD4nZIJVSsZza
4N7EJMI9yC6MHchOgRNZpItu1csW2JVC0zW5uYzrwJcfPamLgI7A1srr/
97EmOOaXuXpKMZq9FJGm6ETepGwDd3H3tFMMsaGbWkaTKnDhOdSLODWsQLWbeTDG8gmFmmwM3EFVQS4i
b1O+AB9z9ETO7Eeji7vulGGbByEgcHEBcmJ9CfE5udve2ZtYXWNPdr0810DyRJAqGu3urpA3D90ALog/
zI8ChSmKmI6OP42BicPgqYmD4xSQxcAXRAkA94ldTxnt9LXHd+iZwIHEeuMHMDiGmN3+faqACLLe/
ngZGJNPObwbWcPfjzezvQD13v0CVtFXPzP4LjCMKVq4jZr+UAb8j+l4vSDG8gmZm5xPXS0uS/
TSDOJ59QJxHpiSDBPpcVDEz+yvwgbu/kcyQ/Mndv8t4vhFwJHGcujGXWveoRUYNY2a1kmm1A5N/
mFkb4sJlbJqx5aPkYLmEmJbwMoCZrUVcMO5MVOZsTizgIVUkY1S7F5HAfzi5EH8q+YeZdQV2BD5Xcjk7
Mk5WfwTWM7P3gG+IVgxvuPvQZMpne2J6raTE3ceb2bcevTPvM7OWxGDMlsRAmFasrwbu/
gmwh5l1Jo5drYFX3X2imXUgZrzcml6EBceS/
x4BPOHu7ySJtDeS7Y2Ic7ishoxjyzZA+fl4L2BSMr15bWA9JZfTkyTQDNjK3XvBsvYwf05ecgAwKK348
klGBd+JxOD7D2b2B2IxRYiBrh+IARhJWcb+agqUV872JPrNQrT8KW89VsT/emhLJWUM/
u4CtHb3ncysNXCxu08wsw2JGUePpBpogXP3f5pZUfLwJiL3cTRwFjAN+NjMPgC0iHjV+4n/
XU+dBrQ2sy+JfN83wEiPBRaLIbda9yjBXANkHIRbA4eb2V+I1bjfIhaW+wS4I7lwlCqU3BRNJlpkPAU8
5+4/AM8k/0j62E1JLcg8lJEEWx/4O3C9mc0gWmM8BTyfTJ/
SFKosyjhZ1SOSlb8DFhGj1p+b2WjiJDdeFQbVL2MgZg2gF7Bncjz6ipiK+7qZveXJ4ihKLle9ClOh90v
+fQC8Xj4VOsM04Nrkv1INMhIH32Zs3h+4OPl6b6KVg1RShaqZEcD8JDmwM0mfa6LKRgvHpW8d4ItkWu1
s4rTwlZnVB9ZV9XLVMbNORAJ5hpm1Ahp59GOGmGWkdgs5oMLx6z6gj5m9C7Ry96fNrAVxL/
I8LHdOkdVTfj3aBfg0+foI4KPk665E719X24X0ZL737v4s8GyyvTUx8HIIcIC736silqrl7rfDstn0bx
B9+zsRLZfKgOlmVn5f8VNKYa6UWmTUABlTd54g6btCjLIeTFTSHu/
uT6UZY75KWgMcCOxGVOO0B6aS9AB291dTDK8gJAfW9kQfu72BrYibpLeSfze4+9z0IiwcyQXFdsTN0Xb
J5gVE0vlk7YfqlXFuuIK40PuMmGK4NTHF8Fp3f0UXfdmTzCoqMbPrifPxFOLGaENiKuEo4Bp3fyXFMAu
exQr1/Ynpt7cCmwHrAncSq6BrkeQqkhyPjiXO0wOIz0QX4Hp3fyfN2ApZxmDYmUTLt1lAY3f/
g5ndBHRw94PTjbJmy0xWJv0ybyMGfKcBR7j7fmZ2NHC22urlHosF2/
sRs1qaEOeKTsAEdz9H11JVo8LnpAkwmDg/Hwfc6u6PJjmPz9z9SiWY05Nxn/FP4v7iOY+F/
RoSa1h4xmv1+ahCK2t5kZxXuhKDXtsDTd39xDTiWxUlmGuIZHrCbGDtzEpBMzudSCyc6O5z0oqvUJjZe
sSFxyFERQ7AP9z9svSiKhxJlU1nolrz2mTz2klVuWRR5oVD0p6kA7EPNiUqmDulGF5BM7OpwHbJtMIio
vfyxcRFyPH6fGRPRtJmDDHI8nr5dmJQ7GTgGXd/qDwZnWa8hcLMmhOrmi/O2HYY/
1vk7yni2HWdBugrz2LRyhbu/
lKF7ZsS1fydgGLgQnfXTK8cYGZtgX8Q61e0IAbqRwD3ufsXacaWb8xsH+Aa4pjzLTAUaEX0lf1PmrEJm
NmVwG2+4sJlvYDuxHojH7j7Q8l2JdCyIDk3H0skzN4jWou9QwxKTtb7nj4zm04Mxn9oZqcAJxEtRE919
xHaR1UvI7l/JtFSaaC7L7BYWLGRu08ys4bu/
lOuvf9KMNcQSWLzQeBod5+Qsb050YO2TVqxFYIkabMO0ICoaF6baBmwM/
BHd39gFd8uv0GFRGZ7oCXR1/EAou/ZhkQV+SDge3e/
N61Y81lGa56uxN97N+B4YD5x8TcbuJ9o1zPc3cekFWshs+i1/
BpwTNIuKfO5ScCW7j4jleAKRDLwdTXwkLt/9Euvl+yzWFzrvmT6/
wZEpc23FgvPbUMsfvmOu89MNdAazsyOICpo/m1mfyIKHh4E3lTFWfrMbBPgLnfvkfRprO/
u85Pn6hADAGsBb2l/
rZ7kXHyqu19eYXs94FCiin9NoJ+7j0whRMmQfB6ucvcLksevEhWaT7n7B6kGl8fM7ChiYd07zKyeuy9K
trcn1tWpDywGHnP3JSmGWvBs+T7Zd7n7Bhbri7wEnAPsDjRw9xPSjDPfmdlkolhoiJkdT+RDmhD9yt9a

```
WaVz2tSDueaYDHwBPG1mlxF9TyEqRMbC8n1yZPUl0z+2JxLKDYGNiAvxV4iT301EJbPaAlQtA9zM7iFu
VtciemQ+QyQ1BwFl5ZX8uTZqly8yTlYvEVPJHyWq/+a7+3ug9z5HzACeBq4ysxPc/fvk2HUwsNDdZ2g/
ZUfGRd2WQB9g76Q9wDvuPjHd6ApXMiBcCnydbLoRmG1mXwCfJ/9mEtU3snqaE20wAKYDzYD/
AxpZLEbzOjDU3YfpOJSK8cCFydf7EfcQbxOLVr+QDEqOTiu4PNOCuDfAzHYAbiY+Gy+4+8NpBiYrtTVx
X11+v/cqcc/3SHIO+YLoe/
qCu3+VVpB5aD5QXpBylpntBwwhCiWedveFqUUmy8m4D2wLjE8Szb8Hhrn7c2ZWBlwGK2/nIKvPopd/
7SS53Ai4glh8tBNwvpl96Dm4BpIqmHOYma3t7t9nPK4HXE5M25lFTO8cQ7RoGKYEc9XIGLG7gKhK+5ro
bXoj8KEnC2ZJ9phZLWIBv1pE8mwmMY3zG2C2LkCqR1KR8xBRddOWqO54GXjX3T9d1fdK9TGzLkSvxz2J
abiTiQUfHkpaM+jckEVmtjEx3Xxjol98beIcPRa4190/
SzG8gmNmOwKbJBVSRURV1D7ErKMmxKyLscQ55nbdFFVO0mrhM3dvnrSEOZIYkGxCtLLagagW/
x3wO50zqp+ZPQVc6e6fJPtoE6Kadn/ieDWLSDD/
3d2HphdpzZdMYx7q7p8lM7/6Er3eOxPn4+HEdewTammYPjN7EvjI3a82s72J88InxLXuBkSyeU9gqkfv
bA2QVQEzuxM4z91/
NLPexAB9F6LtXi1iUOxrohezZhilLDlvFBP5kG2IfMg9yXHuaeArd79QLeCyw8x2Ilpa/
Z04Ju3o7r3MbCOiH/
b6uXhsUoI5RyVTRf7i7mdbNPTuDox29zkWq3NvSNwgjS2f7iZVy8z+SNwwzSYqc5xI6I8n3vtxmf0dpe
okSYHNiYvz7kRlSEPiAnAyMIE4qb2WVoyFwszWIKrIOwG7EIv7dSQW9htNJDHVw7SaJCPY/
d3998njzJYybYlkzlpE399v04s0v5nZIUS1jZfvg6RVRnviZmlDIrF5o7u/oeqO6mNmjwG4+
+Fmdi5xrng+ea58YZTdgcXuflJ6kdZsZrYHMbC1NfG3fq677548Z0AdoBHQ0t2/
TC3QAmZmS4FW7j7bzIYAh7n7vOS5ImIB677Af9z9uRRDrfHMbDbQyd1nmdnJwEDiOqkdURC0PbAvcLm7
908vUgGwWERumrv/2cw+Bvq6+/DkOQPqEsUV7u7TdA5ffUll/
73u3jWpGj+KaKnUGmhD3Ft0JZLOxyjBnL7kPNEQKCMWeB/
uscjfdkTR49nuPjIXk5z5wswuBv5AFHndmhSV3gY0d/
c+uZjcV4I5R5lZN2Bzdx+QHJCvIKptJgAfEasSz8rFsvh8kiRzNicOql2IXrS1iLYYC4GrXavPV6mkX+
bXmRdyZtaaqLzZnFg5tQPwibufn0qQBSy52OhEXADuR+yHG9KNqnCY2VrEoq7/MLPuwH+Bx4jWMUNd/
Zazzsy2JCpwjkz2R1/ivDwWmOzu85Mb1ObEeVo3pdXIzE4iEsj/
Ah4GbnD321XJX7XMrAnwT6KqqRlxXXQ6seirrotSlgwO3wUUEeeHG9y9ZbpR5afkGnUocAlR4femZ6yN
Y9Hvtx7QFJiu4pT0JddP1xFr63QHbiHaNHygxGZ2mNn+wA3A2cTg1mblg5IZr2lK9PUfl0KIUkEykPwy
kfB/ONlmxLpUbdz9/
TTjy3dmtqG7jzazdsBMd1+YzFr9J3CHu7+Si4NfSjDXAEk1877EyF57YiRpPvAjsRjB4BTDyzvJqNwRw
OPESN3SjOfa8L9EZ1fgJNciBFXKzL4F9kumdB5FTMH9vMJrNiCOX6qKqgYWK9ZuR1RzfA2MLG9TolHrd
CSJfiP6oR1MXKyvCYwC3iamsGnBuSqW0UKpaVIVuAtwPbCAmAY9jujd+A0wyt2npBdtYTKzFkRlzR5E/
/
ghxEDMx8T+mU30J9e5ezVZLJrYjUiuvU0ck5oA04ARyb+Xc626plBYLPJ3NnFNW4doAfAysYDWJ6v6Xv
vltLBa5PIFIIrcGLgZGAu/pWJObzGxN4C/
Jv0eItXbqAFOIasGhnqw5IlUjaSWzDXF+nkCs7zKRaEGptStySMb17qHAMcTx7EYVslSPpP/
yq8DtwP0ea+rUIqr9mwBf5FpiuZwSzDnq50Yjkhun9YjeaXsS/
QPfVpKn6phZT+AOolq5lOib9jwwxN21GEo1sVjhfCRRLTuN6F33PPCau3+XZmyFIOPCohsxha0+sThWQ
yKRNpGoXn4kxTALTua5wcxOB/
5dnrwxs3WJRUmPAx5392tUtVn1zKyZu8+qsK0r0Z5kO2IguB1wnbvfr/
NzOsxsd2KRrduIHsxtiJ6znwNvufs9KYZXo1msCbJGMnXcgHOBe4mqpk5E0rkrca26n9r1pMvMHiUGV7
4jpqVvQSxyORE43d3fTDG8vJHMaBlIvLdjif7LZcTA/
BfEeXlyehEKgJmtB0xx90Vmti0xA+NV4nPRkWhxtS0xSHxOLlYI1lRJa4wNiMXbnyDuKVoSn5Opyb/
bKl5jSXqS+/GewF+BpcC17v5KulHlt4zWe0cSyf3PgH/WlM+FEsw5KuMPqw1wATFiNDF5blOi9/
JPOullV9Ke5PfEzWkn4AeiUucN4BF3/zG96PKPmTWvODUtObH1JvbDLkR/
2ZHuvkn1R1g4yhOTZnYTcfF3GvEZ6EwMcm1IXKBfkGKYBSeZatucqJidRPTg0om8miRtk+YRiZpXgKeI
wceSjNcUEwucjXf3b5Vgrj5mdhYxCPlZ8njZAEvS0mE34BBgvvovV56Z/
QVo4e6XJNepbYBPy2d8JeftlkBrd/8wxVALkpnVBU4F/pUMFHdw9wkZzzckqgiPAu5z93fSiTQ/
mNn67v5N8vXRwJvAYmKQZf3kv78D/qbETPrM7CNgb3f/wWIRrR+IdXXKj1/1iCKjRe7+vc7hqy+Z4r/
E3acljw9398eSKs3ye4uOxCDlMWojk3uS88r5wK7A68QaPBP0+cguM9uamJW3BTFj8q4kB5iz77sSzDk
qI7lzDlH9sUtSIXUWsaDKm+5+Vi7/
cdVUyfQDr1j1l1xw7AccREw33N7dh6UQYt4ys2uBvYB3gQ+Ad9396wqvaUP0Jx+UQogFx8xuJKauDaiw
vS1QpMq06pEx6LgTcDxx89qTOB8YMCNJJNQFBrr7XimGm9fMrAPQg0hU7ku8/2OAl4An3P2D1IIrcGZ2
NVH99L2Z3UVUEr5NVCyrXUkVSZIFDdx9THKOOBZ4jzhvv0+sGTI1s8WYVB8z2xw4yt3PS6o1LyAqoMYS
1bST1LahaiSFKJe5+15Jy4WtiWrl6RmDW42JhOV4fSbSZ2Zd3f3LZH+NJvrHjyaOXcOI8/
k0zf6qOmb2b2Jg97yktVh94rw8P+M1jYhFSdV/OUWWLBpnZhsSs5F2AhoDk4kkZ29gOtFK9B/u/
kNqwRaQ5HNzOtHu7f/cfU66Ef08JZhzVEaC+b/ESMV/
zOxB4kb2aaIy4V53fzTVQPOcmdUmVn+eod5Q2WdmOwKbEdWx6xM9ZecSF+tvA2+7+/TktRpcybKkz+
+NRDLtHOIGdYGrn2ZqLHrEHwkcQKxy/ibRQuYzInGwP9DD3bdXe4zsyGgfsxNwONCIeO97ElVqEK1L/
pxWjIUqozd2MdG2oTOxKGxD4oboIyIJ+pLOH6svY+BrO6Kn5u5EFdpcoirwNHcfk2aMhSZjn9R198VJC
4CzgGKiv+w8YvbLd0R/4I9TDLdGy3ivO7v71xa9Sh8mrlm/IIolPgQmVJydJ+kys9ruvjQ5V/
QgZqruTFK5TPSOPzPN
GPNFxuekPHF5KzErciHRCnEQ8CLwcfK87u9yQEbR13NAC6Kw5WViDbDmxPXvVsABOs9XjYzjUlvi/
V0fmEH09e9BrCtSTFzLXuruQ1MLdhWUYM5xZjaAGEn9mFid+E/u/
```

```
nkyvefv7j5QbTKqVkZyf2cikb8G0Ux9BrFAygfEgkHzdAKsOkkykyRx0xBoSyQHuhELb6xL3BwtBA7N5
ZG7fJFUPj1F/P3/RByH3iU+BxNVEZgeM/
srcQPbmqguWBcoIpJoD7n7c0owZ0dGgnkscIq7v5xsN+I8vT1wi7u/XH5DlWa8hSrZH82Jz0YHoh/
wZkBtd98vxdBqvIzPQO2KFZnJLIodgUOBC919dipBynLMbG0iGdAN6EK0bbjV3R9PNbA8kZEYaEr8/
e9LtCFpRKzncrG7P5lmjBIyjl+t3X1qhecaExWajdz9Hl1HVY2VJY0t1nj5PVEYsSFRNNHNtYB7KpLBl
uOJRaq/rTiD+Ge+5y3gYXe/K9vxFRIzO4GYNT+eSO7/
l1gzoTaRj9qBmEV5jufgQqRKMOc4M9sKuIJYlfhBd7/TYiGnz4lpJItSDTAPZVx4DCOSyY8C/
YBPiakhTYAz3P2JFMPMO2a2N3HAfJ+YxjnZkx7XZtaM6MvVDWjn7tenFmgBMrMGxKKiexM3qF2Ae9z9n
FQDK0A/
d7Nj0fN0M+B71yJCWZe83x8CJ1RslWRmQ4Fj3X1SKsEVsIwB4t2JQbHhmZ8Xi4WSm7v7V6kFmQcy3ufj
iarlM4iq5Q5AM7UPyw3JwP3hwBvu/
n2yrbyacEPgB68hiwbVBGbWypMesxnb2gKHAa970htecoOZfQDcADxJDEZuS9x7fJJqYHko47jTmiiM+
KxicZyZ7eLub6QSoJQn/
B8lEpgziHvxL4gFYn8g2l4tSF5bniu5mRioHJ9S2HnDzK4k3usPf825wsxeJj5H52Y9uN9ICeYaIOk5W
+KxWndD4A/Aju5+lKqXsyOpQBjn7k2Tx7OJ6qfewObABe4+L8UQ846ZnQT0IdrA/
ESM2o0i+qKNJU5si/
Q3X73MrFFmj7RkWwegnqoMqlfGBXo94kboGKLS4Dl3H5VudIUlqfS4kOjJfw7Rc7aIaA/
wkruvkWJ4Bc/MRhEDwa+a2cZEi4DGwJnliTapvIwE85vAk+5+m5kdC/
QletDeAVxUfjMq1Svj5n8f4DpiYLgO0VppX6L36e1pxpgvMt7rrYC7iBZW44mpzRsB71SskpX0ZFxHdS
WS/
m2SWRcPEEUuxUBfd38h1UDzTEZ7jGuIiswziWumA4jCrQFJeyvd46Ug43NRhyim24k4l69NrGXxLXE/
PgH4pHyQ3swa6Dy/
+sysOXALcZ6uT7SyGkO85x+5+9iVfM+ZwPu5OKCvBHMOyrhYuYz4wymffluLOBi3JhLOU9SnqGplHGD3
IBbt2MliAY973H3DZIrhIHffIuVQ85aZdQR2AbYjLtDrEiOpXxInuEdUcZN9Fv3H9ySS/usQ/
Z7eA4a4+9w0YytUGUmdK4jBrveAo4k2PrOImS3/
cPfXUgyzYCQzKy4jVtSeTAyOtSVmG92oqbXVK+P83Ym4dmppZvWBd4C3iBulD4HrdQNbNczsB2ADd5+T
tIw5jbgpepBI0oxMNcAClXGueJCY1XJ+cjN6KDFgvw5wnbsPSTXQPJCROLuRmB1xnJntSSysuC0wgmhv
qIXLckDGPfYZwJ7uvp+ZHUe0u9rWzP5MrGNxVLqR5peM930cMcPrLTO7AdiS6C37GNGz/
8dUAy1gP9PGpAXR9m0Hoq1SV+Aad39AOaiqlVT3tyfe407J102Tp6cT1c1vu/
tbyevr5Wong1ppByArSg7AdYBTiP6nmNnRwKXAFOA4T3qf6oNdtTLezy+Bf1msaFsbGG9muwIHEx9yqW
LlF+nJNJvxwH3J9q2JhbO6E4mcB9KLMv9lJMX2B64kFnf4njge9QLuNrN33X3vFMMsSBnJyr7A4e7+ps
VCcxcCRxD9HmvBz7fSkKqTDHT9JamE2hFYSiy6WL4grJKY6dia6BUPcCxQ6u5nmtn+wN/c/Z/phZY/
khvP94Azk9l1U9z9peS5TYnZFZKCjGN/S+Cl5OsjgJvd/XEze5GYbSGrr/
w43wMob992HjEYv7uZPU1cwyrBnAMyBhenAgvN7Cqiwv++ZHtHome2rqOqUJLbWIOozvw8GaA/
Buju7pPM7DOiklkJ5pSU50DMzJJNjd19BvB88g8z6wxoXYUsSGa6TAU+SIpK2xFtxzYgjks9iIUW30qS
+zmZXAZVMOecjBG+fYGr3X3T5EL9EeBq/p+9s46Ws7re/
+dJgkMIwV0S3N2lOBSnLVIoFC9aaCnW4lJocSue4hR3CO5OCBAgggQJTnANPL8/9hnycn+Bfktm5tw7c
z5rZeXedyZZe80775F9nv3sSPAMt/3XrIG2IJJmqyoMVGleI+kM4rMfAJxi+/
ZMYbYsFcXNloSn41m2H+jwnlKK02Aq9+Emonzwn5JOJxbcVxLJzJNtX5810DYlLe7usD2TpEmBl233Ts
r/
PwN72v46b5StT7oPiwKfEyWDw8rY1DlIFj4XE839HgQus31FSiTMZnuzjOG1BBW1+PLAH4nk2ZW2H05q
wD/
Znj9rkAUkbU7Yw4wHPEJYxHwu6R1gRdvPZw2whUhqzImAL4nD3l+nStNXgC1rqrNC50HSfoT94XVEdeo
nkgYB+9i+odg11JeUVD6MOIyfDOjhsPvsAzxqe/
KsAbY5lXl9CsI3fhugN9AfuBu4wfZnGUNseVJyXx3HnWTjMyvwoe23Ort6vCiYOx+1L8uswPNJGbU78L
jti9KJxlYwOhmdKc6WItlgHAiskZI2iwEvSPqQSCDsmq69ZPvtfJG2LhWVwCdEN+HrkyrqKeBC4DbbL5
TvfWOp3IdZiIaLEOrMnWzfL+lPhDdUIQ+TA/
ekBeBcRDk6xGJ9A9u7dPaFR1dFP2wgtx8wO+EX3x0YpvD9vd/
2tTnjbHdsvyLp10QVxoCU9Jwb2IC4b4WxpLYJTWXOQ4AvbH8saQ7Cd/
a4zCEWANsXS3qTUA0+RCg2fwO8W5LLdeco4AjCKukvKbm8KDBRSS53LhSNqyewfZSkKW2/
m66vT9gp3QI/UDsX6oDtDyRdSBxK3gP8JyXUfk987kU1npfuwCji/
qwJHA78hqh+2QS4RNJFtrfMF2LLMyGwqaSDCNeC+wiLyieB12oJ/
s6+xysK5k6KpF7A8YQH7bPA6bafTqrC+9KkWAbhOlA5sZvd9lBJGwMXAU8DgwgF1GPAUODrmqq50FgU3
pkLEbYkf0qXVywL9caTPvtdiIltAHFyfUL6eyjQJ5VNFZpIZazqxWgFyInEgqQ38KDtPWt2MzljbUUqC
eZbgIEOX9MzgUmJg7HNiAawJ5eDsOZSeTZ6E30qnq+Ue3Yn/JdXJhoJlWdjLElih5OAc20/ntQ1fWw/
p2iS/KXtL/JG2Z5UKiFXBl61Pazy2viEn6Zs35ktyBahMu5MS/RB+JZIKH8kaRJgW1IiM2ugBeAHc/
h2xDxxisM/fkFif/e8pN4ufV4aQhLNDU7PTO1eTEdY8PVPB5Zl7ZSJytwxjLBjvV/RyPcU4EOiJ8/
pth8tOaj6UnkediHmjUOAdQkbmZFEFdIFtnfLGOb/mZJg7sRImplQEd6XHvhViBPyTW2/
VFRq9afygE9G2DSsSZS6TUgsHPe3fWXOGNsNRbO5PwLv2T7vv72/8POQNBXwke2vKtd6JlXadsD2xAZK
ttfMFWe7I2kG269Xfl+FUBg8Atxs+82yQG8skt4AlrP9sqShwEa2n5F0GnCq7UHlHjSXytx9MDAP8Fui
XH15YAHgettPZwyxJahsQNcHjrA9n6IxzZ5EkmAwsKpLI9jsSHoKOMz2lUlZvgNxQPxvd2Lvxq5EJcF8
JnAjcF36fWVCHPQekXAuvrKdgMpJ8LYTAAAgAElEQVT49RzRyP0KSdsSa6jVgP1cPPrrSuUZmZE4lPxL
EnNNQ9gw3GX72ZLT6Bykg7GBxBp3RKqAmSsdmj1KrHdf/+n/pfC/
UlnDPkgc3J8t6WxC4HgDcAZwoe1Lu0Jyv1vuAApjRtKURJnhlOkPwHPA3k4+wWUgrj/
```

p4ZbtkbavsL2d7QWA1YGzKE1rGoKkbunvtSTtIWmRpKIlKcafIVnDFBrGgcB0AJLmSYvBWrL5MuACosH
i5nnCa19SCSGSZgKOlfRXSZtJWow4gNzR9rm234RS1tlI0tx8H9AjLcS/
Y3Tzps2A16HcgwzU1kMbEyrlb4ADiOqX3wJ7p0O0wthRa/6zEaObx+1CJPFXJCytfpchrgI/
mCvmBqZJyeWJgdOJg5eDCV/NQh1IibOpiQTlDen3vwOnEc3CFy/
J5c5DSi5PQKx1r06XDyeej5WAtZKitlA/
armmTYGeKbm8JJFs3gr4h6SpS06j09ADOBbonQ4BXgZWlLQcMHtJLjeGSsJ4QqDWf2oVwubtDWKN+2yH
93ZaigdzJ0I/bHK2A/FQr0cY4h9LeG/
enzHElqVywjopsGY60R5JdE29LiX1/5k1yBamkoyZn9ikrgJ8JulFomHKGoQ9Q6EBpBLyIbZfTpeOI5Q
3gxT+ms8SG6aSOMuDiMXFAsAiwKqEwuBj4KOkVBsK3FM2sw3nA6Lhromx6THgBknPEOXoHxX1cvNJiYM
eQE+iw/
Y4xGHYyrYHS3oWmAp4J2ecXZ3KxuYt4L10yLUycIztASk5MzhbgIXaXLECMUdAJJQntL2KpNWBv5Lm88
LPpzLO/xJ4Ou3f1iUOX34JrE30b3kkY5iF/
58pgDuJvd5cwIu2r1H0tZjP9oi84bUctcTxGoRQBeLgdySwFmG/sDlwfFEx58f2SODUitr/
ciIP9SlwLhSf7Eah6DvVD5hB0kvEIeUMkoYTFfUbZwzvf6IkmDsRlYf1CGAn4Fbiy/
VEur4ncDKhECnUl5qx/d7AOsDlwAzAocBZkr4A/
mn7yHwhtgXnATcBswELEvegL+GDfXLGuFqaNPaclNRP3Qg1xyLAMsTz8AUwnPAlPzFXnAW2AY4h5oapi
PvzZyLxPBRYWdLhxT+wcaRn5XurBUn/
ILzSJiXmbRit8iw0lwmAa4HzCUurO1JyeUpgZtvPZo2utTiPUDD/
kfj+35wOKhdldM+EQpOpHGw9ACwl6VzCn7+2fvolIV4pjCWVz/
pD4MVkV7UZcGlSaU4ITJItwMIYsf2apHsJBfOlwM7ppa1JCsGSQKsflefkDmDVNB9PCRxs+91k33NJek
/
tgKyQkWQT+g2RVD4FeIWYN15IbykCigbgaOB3QqrqNnAdkXB+k7DN+LqrCFiKB3MnQ9JswG22+0jqCbx
iu3d67W1g7pI8qD8VBfM9wD9s35CudwdmJozW37T9n5xxtgOSZgdGJX9TAT1cGis2lPQ5a0yTVrLKWIh
INH9ue89mx1f4vvHiSGDytAipXZ+bUBdcQ5Q/
X0F4xXf6BUhXJNks7ELci5G2/505pEIFSbMAWxAL8puTh+B+wGq2V84ZWysiaXrbbyiax/
0B2ND2CrnjKoCk3xM9RM4lhCrTERvWA2zf/FP/tvB/
J1kl9SM8fA8C+tkeKelh4AyX3iGdFo3uMzIvsY662vYFJcFcfyoezNMRPsz3SFqIUJNPV3zh85PyTjsC
ixPCoveJCoxHgTdsf50xvLZAHZqMSloAGAcYmsaqLqHyLwnmToakPoTSYF9CdXCw7ZUkLUN0j+zTVb5c
XZG0ILftfrljaRcqZThLEWqoaQk7mG8IU/
vzXGk8V2gMlftwKqFM659OS2cEvrH9VuYQ25qk+jgHuKo6PkmaAXjI9oySFiXUm4uUZ6Z+VJ6NhYlN6D
eEAmca232TTcAntos1QCdD0Sx5XeBZ23dnDqdlkNQX+KDDRqg7MFXNC77QuZC0LOEze1Q5gKw/
6aC+u+1RktYjlPzr2/4wc2iFCskKcX7geaKx9ah0fVxi/1cELU0gJTN/
DSxge4+uosxsRSoWrbsSlXg3EMnl2Yi1bi/CBmivjGG2NGn9tA2wIVGhOoS4D/
1tv5fe02Xyf8UioxORBtcXJd0KXEl0QL9J0oLAfkRyAaKEvZys1olK8mASInFwgqQVgduAB22/
kjXA1qc2WJ5CDKbnAG8DCxP+dSOJJnOFBpKegQkJL7R/pOTygYQP/KeS/
mi72PNkIpUSXgucLGkT4F7CD3hVYEB621TAOCW5XHdqlhfbAu/
Z3lrSVkRjJwhvtBmAnbrSArCVSCWFCxAJtJ7Aa0QC4Unbp2QMreWQtAuwHTC/
pGGEbdu1wGO2SyPkTFQq8SYBtgRmIkqbhxOb1UdsP/AT/0XhZ5CqiHoCg4DPAWxfJ+lO259mDa4A/
CCBthghZFkFmBoYKGkgsfe413bx6G8A6fBlfWAyImH2hu2PgXNS7wQo1hg5qfpkH2n7Yvi+Ym8Woor19
XStHATUkcrnuS5hy3oc8VmvQFjjHi9pmO1lu9LeoiiYOxmSNif8f9chTjFWJpJtRxKD8idlA1tfJPVIi
oPTgF8ADxOlITMz2j/
tnFIK3TgkTUt0Sp2mcm0cwhNtA2CTslBvHJVDlo2AfW0vkRTlFxLlUusAk9neOmec7UwleTA34fO4IDA
n4YV6ZirJPRv4uKgM6kvls78VOM32tZLuJ6qKzpB0MfCC7UNLaW1zqSQO1iD6V3zB6APKl4hD40dsH5I
xzC5P5RmYC7iLUDk9DrxKHABvmt7aKyUOCk2mMo/3A/oQNjGrE40tPyCaLx5ne+CP/y+F/
wuVcWcH4LfE4dZ4hIfv7cB9wJ3lsLdzULlf1wCf2t5C0pnA9ET/
hGWIfd72ZY9dX5Iy8xxg3nRpUaKJ+DWEUOLikrDsHCQFc0/
glDKPN4fKvP1P4F3bR6dqilrT6tmIddVNXWl/
URTMnYDKwn1F4OR0cnS1pFtsfyFpnGrJTpn46k7tYV0G2NyjO6FPR6jSFic2UYXGMQXwsqQ1bN8KYPsb
SbcQNjEludxAKou7SYF3JW1MKJdvtH1Hssn4XbYAC6Q5YmbbzwMHdnw9WQHcSByQFepIZc69CNhX0lBg
HuDidH0ZovkilOYnudgduC4l+c8CHiOaMW5D2MYUxo5a5dxGwH3pkGUL4AHbm0t6Jf1cNqWZSJvUCYh7
NIfttyS9Q6jN/0I07i33pw5UNvmHEj0PzlX0ybkT2IGwOVyJSKAVMlO5X0sRhwEQAq6NgE+Iyr1z0/
XSaK4OVJSZaxGf+2rEHuMmwqv8bGAN2xfmi7JQIx0eH0oclHWXdBdR/
TIS+KKrJDa7GpX998dAb0kT2P4C+JqoiHkrVQDQle5BSTB3LiYCvm8il75gtURbKUloEClxMwFR3jZ5u
jYCGAE8Lqk/
UFQIDSIdsDwj6TrgaEkzEeVqCzLaC6rQHC4gTkt3IdRpp6XrvydKoAtNpHKy3ZcoeV5B0uRESfrDREPY
oQC2hxOLwUIDSBYMlxMbpSOIOWGr5LV5a80+phwAN5fKgnse4K/
p52WA39t+VNJ8lAPisaK2uUn0IZThEGXmd6efxwOWJg65Ck2morpcCRicksvLEImBeyV9BWxm+
+WsgbYAFVHQcsCHKbk8O4DtfSUNBr6yXZLLnYCKenlGorplYklfEnZiT6f37EH0P6LstetGbd5Yhzj8f
U3SjsCjts9L9+Mh6Fresq1ImuOHEJVISxL3bFvieRlKVCldny3AFifZWi1BzN+TS3qIqDgaZvutrvhsl
ARz56B2WroIsKGk74B/Ex4s79r+pkx4jaEyqc0HzA30k3QC8AwxqL5q+/
OcMbY6lYHzbOI5+DPwL+AFYlIr6rMmkJ6FUen7/x/bz6TrKxIKj+KD3XxqC/SDCL/
AYwhf8rmJZM4pko6yfUA5hGwMlZK0A4lu2ocBWwFfEuW1V5DGqLJJyoOiWdA1wFeSxifmkdqh8JqEurn
wM0nf6Voi/1Tiew/wIrCIpNWIDeneGcJre1JyoDswKl26N12bFXg0XZuf2MAWxoL0udbU/PMANc/
xFYAn0s/vAPszusKlkJHKIeTbwG5E7mM84IVUlj4SGGn747KOqg/

pOamthboTlhgQa9f+6eeFCUur2yiq8WxU1q0m7k1/4DCF//
JKhDfwdOm9XcaioYvxHWF3eDvhd70Oofz/TNKTtk/
MGdzPoXgwdxLSYHwY0JcwVP+YaM7xHNGs5obi5dU4JM1PbERnAmYkJrv3icTaVbbvyRhey9MxMZPUgjM
mVWahgVRUsr8gvNH+ZfvTWpmOpPGAWW2/
kDnUtkXSu0Sn7TclvUiUPE8HLAf83fbwsjFqLJKeJbzgB1WujVfm5fwkj8eZiFzoK5KOIp6NIcBCthfN
GmAXRlIfoqrlcdsjO7w2HaH6m4k4EN6qjEF5SXuJWYgk5zRE5VHtwOVs2//
KF11rIWkKwm7hESLBfABRhbou8bzslzG8AiBpM6KCZWBHqz1JGwJ/
IvzJL7J9WUmg1R9JvYkG1IOJQ8jVgQeAnYFlbQ8ph/OdC0kT2f4s/
Tyl7Xdzx9QuKPpPzQ3MQQiJhjh6vXSpPV5JMHdCJM1AbI6WIlQHo2yvkTeq9kFSL+IEaWmiVORo2w/
ljar1qJStLUz4A65AlEs9mP5+CfikKMgbS+U+3Ajcb/
soSYsQm6X1gO1t98saZBuTEjw3EZYxEwLP2q6pCZ4DFq8tBAuNQdLExMZosFN37XS9G9DdlR4JheYg6T
jiAP62jgeRikaYfyMUa6fbHpIhxJYglTRvDryV/
gwilJqDbH+Z1ks9bRcbkgwk38wziUqKW20P7vD6EkTZ8wDgipr1XuF/
J33Xf0XYUz3fMREp6RDCZ3YIcFARSOQlWR9eRCiWvyQSzc8Q9m+DU8Xe1Lbfzhhmy5GseeYHngRetv1e
5bUJgeOAyYj9xsl5omxvKjY/MxKVGF8T+Y7ZgE+J/lNzpLd/
YHvlPJG2Fxrtv4ykiYDxbH+QOayfRUkwZ0ZSjzTJ7UAoDK61/
WGH98ySVDnlZLVBJAXzcsC0xCbqFtsvSeoBfNeVTo26CpXE5uNEUvlBwiZmBaLp3zjAH21fnTHMtiGpZ
Be0PULSY4QS50tgVWBn229kDbBNSWVqaxFla72JZMLhxFh1gO0+Xe1ku6tQWYSvA1wHfAicSFQUPfHT/
7rQKJJiuR/wC0LJ/
zJwB3GPHq5uaAtjh6Q5gHkJVWxfYEpgfGIT+hJxGDzM9uCiQms+yft3TyIZMBsxZz8B3ArcbvutjOG1F
Mlz+TBiHviUGHcGAANsv5LeM5Xtd7IFWfietH9biJgj5gVmZvTe4mOi6uIJItlcvMnrhKQDiLn5A6ISe
Dhx6PJi+vtrwv/
6y2xBtjmVytXzCMu3j4B7CKuGr4i9+EjC5uc527eXfUZ9qOwrpiDm7L5E9fAowrpnMWIeJ/
08o+1PsgQ7FpQEcydB0unAjkQp2xtEMuEi23dlDayFqSQ41wf+QFhj3A30AiYAjrN9f8YQ2wJJA4AVqg
OoopHZOkRX+mE/
+o8LdSGVOl9FjEF9gWNtz5Jee5+wyCjd5zMhaZyaSlbSwcCGwLtEovOE2kFlzhhbGUnTEAvupQlPuhmI
TdII4G9lnshHKr9dH9gYWJ6YuwcRvo7X2n4gY3gtRZqXZyWSmX0Ja4wpiDXTFrZfzxhe25IOXLoDcxGV
LssQ49XUwJvA88AJTg3NCj+PpGCej/
Agn5eYByYjLPXeI5LNLwBPdBQKFfIhqWfyV56GGL9mJ8avGdLvD6Y+FuWArA4kW725iHliSeJzHp/
wW36HGI9GAHd2tC0pNBdJ2xD5jyeA0zy66eUNwI22T88ZXytSSe4fQlTa3U3Yx8xLzB8zEPP3UYTK/
4muODaVBHMnIJ1iHEz4/fYnBubNCEXtV8Sp0tEl2VxfKgnmhwhvunPSifeshD3ADMCvO/
oOFsaeygA7K+GBdr/tS3PH1c5I+hOx0HiSSFyeL2kN4FTbffNG196kMrZJicRZN8LjcRi
hLCiKgiYjaWZCWbA2cJbth4u6ozlU1B/TEXP0c9VNalJ0rg/
8jmjctGKmULs0lc95fuDXwJvVzWby+p0m/
ZnP9gWZQm17kiVML9sPpfsyPpH4nIXYqK5L7CHuzBdl6yBpEuAzQpTSl0ikzUooZfsQlUXls+4ESPoD4
ZN9eLUKL+31ehFWDh/
YHljm8PqRDiNl+z1JkxEHMwsT+Y1piTFq42KBmB9JqxCNxGcGjiWqwwYB69t+MmNoLY2k3wAnEAcv+9q
+PF3vBwy3fVDG8MaakmDOSCXBuSuwke2VO/jiHEI0j5ifGJS3tP1mzphbEUmDgd/
afrzD9QHATrYfyRNZ6yNpc2Iy+wq4HLiBKOssatkmkjalIpRPI20PSomao4gEzoFZA2wzKvPAtMDuwLK
EQm1S4BrgqOK73BzSRnROwod2ZuLA9xrb76o0+Ws6lWfjJGBi4GDbryoao3xbEgT1oXII3I9oNH1iShb
0IJopfiupL7GPGJo12Dak8hwsQAhUnrP91x957/cVMIWfT1LA7gz0sL1/
h9cmBHoCS1CUmZ0CSdMDtxDVFQPTNRFj1neSNiLEFF/
njLNVqMwZ6wMrA3fZvmYM75scmNf2vU0PsvADKvPIuITV0rpE74q5bc+TN7rWJz0LfyN6rl1o+xRJTxO
HlNfnjW7s6JY7gDanthEaH/g0lfEYwPZrhFfRPMRDD/D75ofY2qTFxgXA/pImrVzvQ6g/
is9mYxlOlDXvRTTiOBAYKOnF5HdXaA47EEqnJ4Ah6bl4GTgCOCVnYG1K9/
T3n4mS57OB3QiVwcLAUaksutAgKp/vlsCFhErtXWAXoL+kNUtyuflU1MsbAHs4NZez/
U0tuSxpJ0Xz2MLPpJKoXxs4o+ZrbXuUR/
cC6QvskDZJheai9PfvgTdqyWUF3dLP80g6gjiIKfxMap8n8FvCXuH4dL17ZZ6YEFjY9nUluZyXtH6FsE
16raZOhpg/
UhJ0XELZvEmuOFuYvYCBwI0Qz086mETSasA8Jbmcn0pyeeJ0yHIicB6x55hS0gZln9E4FNaG7wNHE43c
95HUH5isqyeXoSSYs1LxU7mUWAAeImkpSXNIWp0oS3wqKQ8+JPyZC3Uk3YOridK2xyTdLOka4DTgguJr
2jgUfnbHA0/aPotIcq4NbAGcSyQ4Cw1G0mzA/
sBlwF3EPdmcSKi94NKwJge1BM7ywBG2z0+latcQTYaWJZphFhpHbb7dk1CM/9r2noTK/
1ZgL0kzZYuuDakkDtYjGst98iMboPGJxjWFsUDSPMBXtl+vfPZV7iWUauM0N7JChV8Sa9iagtCVw4Hhh
I3MMrmCazE2Ba5M1Suy/W3lsGVSYJ1ysNWpWB6oefB/P36l6uGvicZmy6drJR8ylqTE/
QSEWOW8WtWE7e8qe+mvgE0lTZkrzkJQyUEdKKmX7S9tn0NUYZwDHEeIHAt1Js3VowBsv2n7UMIad0Jge
kk7JiumLksZUDsBjsYo+xAeXucDJwF/
B24HLkplPisR9gGFOmN7kO2lgD8C9xFNUc5LvxfqTGWjOgfh6/
iNpHHT5DbC0ZTpqKpfWqFx2H7J9szEQvtfwJTp70eAMZbcFhpLUhX0AC4CfpHKb0kb2oeJpMFIKBujRp
E2SyKabzxXuf657X0Jr80p4QdjWqE5zAE8m37uVvv8K8nmj4ju3IWxYzzgBUmLpzFpnKTarH3fZwGmtv
1WvhDbkzQ+1RpnfVS7Vns9JdE+I7xPi53SWFD5XKclGvlVkzO1ZMGLxLg0dfMjLFSp3JungNklTZAsfX
qkOaL2+hpArfFlmcPrw9LAkDRfjOkzfRpYhbR+LeRB0kSSJk0HYju40pTU9gdE9erjQGnc2wAq1XZnS5
ozXbufeDb2JyolF8kX4djTI3cAhe89Hl8nynkmBxYl/NSGpwF6ZkJN+0zGMFuSpEDbkCjn6W/
7pswhtQPdCIXm/MD4klayfXf1DcVHs/

nYHkH4YfdLFRTbEIrmQh5WJZXiAlNLupLYGC1FjFVPQXlWGoFGN/
uZG+gNnCdpF6Kx4rdEY7OJbD8BP0w2FBpH5XO+FThM0mT+YRPe2uu/
IjpzF8aOp4CPicP237ri45vWptsyWiFYaD7fEN/
zvSVtVh2HUkJtKiLpeU+m+FoGRaOyh4mk5Glp32bgu8ocvDjwUKYQC/8/
NwLXAmsCV1crUiXNQVhk7JAulXVUffgQ+ErSarZvS/
NEd5JvP1Hx8o3tUSpNFZtOzRaDmLu3JQ4nh0rqTVQifZis36YFvuiwvirUgaTynwd4D/
i17e1qr9n+StLRRBPx19L71RX3GCXBnAmNNsNfGdiaaA6xErCa7ZvTYqa2oXow/
SnUAY1urrgScDjxHOxLeA49B9xBmK0X/
+UGUCkpXJTwk71F0h2EB9EtSQlSaALJh25VoinNl7XrtvtL2o9QbxaaTFpQ3CJpLmJDuymwPTAusRE6T
NFg61WXBjV1p4NibXyi2/y/CSVzT2Ihvg+UBlqZeIh4Fq5Ni/
EBwEe2P5O0FaFe3jlngK1AUqH9A7hS0gjgEqA/
sSk9BBhFrKEKGUjr2OuJTvT7Szrb9tuKhpdzEnPGXV1xc9qZSPPxSEkPAjtJutH28Mrr4xJe/a/a/
ihboIWODATOIsavp4iE8yNEYvm3wLke7eFfnpGxJD0nT0oaAuwp6WXbw4h5AkmzEvZ7V6V/
0o2S2G8qle/5lcAHhOXe+ESOaTjwoKTHCAXtR/
C9V3CxC60fyxDWez2AjyStQBzMvOTw7xcwue2XoOuOTeqicbcMkgYR3qfXEYnNpYiSt62Af9l+v6ueXn
RWKgnmS4BPiOZZo4gTpdUJD+AhtkvzhwaSTkx7Eonm5dLfM6eXFygL9cYjaRFCDfgp8Cphy3MPkaA52X
aX9oBqNSQtRjQ3+xVRnj4ucQJ+Zc64Wh1J4wErEuqbxYG5iOdlAPAMcF2x9GkOtfWQpFkIhf8swIvEum
kRIvl/oO1Lc8XYaigaIO9JzNOLERuga4CjbT/3U/+20HhSdcWhhH/
jc8TzsAJwG3CY7SEZw2sZ0pr1HGB94qDlNqIZ+7bEIeTJZS7ufEhalGiGuSQxdz9FJJ4vSRZ9ZY9dRyT
NS3y+MxOWGC8SDZI3IaoA/
mp7RPnc8yNpV+AMIv+xLiF0nAa4EzjV9uCiNK8vSRxUSzJPQOwjuhF7ijcIu8qetler5auyBTsWlARzB
iobpBWB02zPK6kP8IDtaSTNQKgOZs8caksj6RTgMtv3jeG1LvtQdxVS6VTPWiJZ0Yl+emD2skhvDsnDc
VoiSbNc+rMEYQVwjO3L80XXvqRnY27gYEarCy61/UrlPeMSG93HqtcLY4ei+cx+RM+Du+H/tyFJ71k1/
VkHWN/hjV1oMAq//
q8lLUmUGC5C+NZNADwKPGL78ZwxtgrJr3RGQuk0AeHJPJJQndn25xnDK3QgVeUtDcxOqM0fLQf19SOtU
T8nqoo2IA4bpyMEQieXcafzIWlB2wM7XBsf+DpVEZckZ4OQtDUhHOpDWI1dBZyYLBgKGfmp732qgJmwz
B2NRdIGwFBCaLc4seebAngbuNj2g105F1USzBlJX65dba8qaTdgDdvrSNoIOMD2ol35y9UZqST3JyGU4
1MBBxCb0g9/+l8XxpaKNcxiwK+JU7z5gF+mwbSclHYCJE0EfF4W3s2lUl2xMeF7OohQZi5LlHUebPvvZ
V6oP5W5YTeikmV7229JmoIoNd+CKLk9xsn/
upAPSZ8Ds7lDgzmlppdlHhk7kk3bYcSz0JdQ19wNnGn7wfSeMl9nJh1GzkUcFM9O+PO/
nDeq1qAyJ8wM/IGYh18FziQagve0/
WGxSeqcSJqRUJgPJSqNbiPs4F7NGliLUtnfLQe8Yfvl5Dc7yh38+8veIh+VfcbShLXP+sRh/
a3AFbYfTe8r96kB/Jfk/
hTA+63wuZfu85lIi8JrgXclHQn8DjhT0oREKU8p72wAlYd2euKU6HPgz8AZko6VtL2kebIF2OJUNqOnA
xMDOxIlzV+mxeA+isY0hQaj6Ki9tKR9JB0uaRtJy0ia1PZnrTDBdWF2JaordrK9p+0liATnCpL61pLLa
R4p1IfaZ7kJYXlRS1z+DfgN4d84C3CKpN6SuikaPRWahKTFJN0qaW+iGc1bY3jbqZT+ImNFUvj9nVDDH
kRUtRxCqJlvT2W1JYmfkaQuh/
CSPYu4T3sSYxWSFpY0cabwWoXaHnlfYEGi6fHkxLMwZU2UUrNZyBNioSOVZ2Mm4GqiUelnxCHBE5IekX
RSWu+WPEj9qO0ZjiMseiAUmQdI2i0lm7usp2wLUfv8zyKqkrYneowsBfSX9J2kLcp9qj/
pEMaS5pXUT9I7kq6TtH+qQBqXqBbr8hQFc2bSCdJhhDptHMI/7QZCJVL8l+uIpDkJE/
Vv0u8TEKqP+QgFyEzp72ttn5wt0BalogZZFPiP7T6SpgUG2p5KUi/
CCmARVxrOFepLB5XsAcD7hAfzAsArwLeEXc8h+aJsbyTdB+xfs+
+pKEMGAH9x6s5d5ob6I2kwUU30iqIpzaPANravT5vR2r25p9yD5iJpfmBvYD2irPAlonfFZbbvlLQaMb
dMljHMLktljt6YOHjfcAwK8T0IC5klihIwH5V79RrwO9t3SRoK7GH7JkkXAafbvj9zqF2Wymf8JrCq7U
Hp+qPAXrbvL3NA56Oyxn0c+CdhzdCDSHYeCMxBqJpnBY60fXu5j/
VBYfE50Pbkab10K5HQXJA4tDyhfM75UVjsPWJ74Q7XJyYSzYNsv1mqlOqLUsNESXcS69ebiGfjF4Td0v
jA32yflzHMulBUHpmQNL7tL20/
BKyqMP2ekkj6P1h7XxmI64PCU+hywq8RSWsDg4muzy8B16UE51zAa9kCbQ+mAY5AUz4AACAASURBVGol
nOsBz6efFye+/
yW53Bx2A863fYKki4mmDq8QStliAZCXi4CLFM2bBgAjJfUkmqY8BGVuaARpcX0PsL+k84FtgKdScrm2a
Z0DeAHKPWg2tp8BfifpWCJB8CExXl2usHR4jlCcF34e3YgDxs2A25NFTO17X7PlOZ9QNq+Ufi5kICU+p
wW+S8nlcQmf07vSW1YG9skWYAuQPuNZgG615HKiD2mNVOaAzkcar0SozQfY/
hr4mrA32U7STYTNyVZE1eSzP1INU/g/UknQr0DsrSHsF6ayvaCiZ8KJto/
PFmShep9mA16RtJbtm2uv2/6UaPZe+70kl+uI7VHpx/GAQ2y/
RhyAHZT2H2sR69gub0FWEsxNonISPg+wMTBjUhuca/
t928OAYZImkjSD7dfzRtxapBK29dPJ0YzExugd4jN/
iOhs+7xLo6aGUVmI3w9sJekPxGB6dio134ooZys0EI/27u1DnJ5ClECva/
t5SQsBZfxpMpU5YmrCH35qonTtLcLSpy9wvO1Pu/
rCo7OSPttTgGOAw4lN6SHptW8lrQp8avvtonjKh+0/
VX69FEDSbEAvRh9YFv53amPK0sAO8P33vhsxRI1ne2RKZhaVeH6+Bh5Iyv1vgGG2v0iVkV+XfURdWB6Y
UtKlhBoTwl/204wxFf474wLXAP+QtLPt1yWNR1StLm/7EeCRVAFQms6NJZW10AOElds5RIL/
1HR9beBNGK0wb36UhQoLEMrZsyRdSQhZngReSAcyhTpT2eNNAzwBrEnYlADfJ/cvr/
zepfd4xSKjyUi6l1jEv0Z0Pj+cOAlfiPBSmwT4k+3bsgXZ4qRN0leSliEe8JWIBI4Ig/u/
5IyvHUg2GRcTTWkGAcOJU+/

```
jy6ao8Si6oe8LnEEctDxEeDe+AowAprX9ebYA25CKUvBm4Ghiob4W0YX7NWKeGJgOy0pys84oGjl9Yvs
DSTORVGq2R6bX5wX2At6yfUDZJDUX/bBB7J7AgbZfTAfGPTsoDAs/
k6SKfYNI3F8H3Gj7kw7vGQGs7dLsMjuSdicOAyYiSm7PBNYFXrR9UM7YujqVhMCKwK+AdYgqIoDTCLuk
+22/kSvGwo8jqQ9wMjAPYXX1IbGeesD27pLWAv5le+af+G8K/wPpM52RqEj9N7GOnZzoOXWS7cvK2ik/
aT07J7HO7UPYx3QnlLV/s/1ExvBaGkkbAZcQ1WLXEba4/W2/
kzWwOlMSzE0kJdWusT1j+n0r4EjgI6I7/UOECuF8259lC7TFkXQN8A/
bD1SuTUIoFUbZ7p8tuBajskCfEvglUXL+CrB3UgHOQ5Tj3m/71p/
4rwp1ICk4ZPvLVPr5ne1XFY1GVyDuzcy2l88XZfuSyjrvAja2/
X7ueNoFRVOzSwnri+eBp4ly2i9qBy1J2b8QYR3weknyN5eKd93pRMn6jpJWAHYHNgL6A7/umAwt/
G+kZ2Hd9GclYAZiXrgZuJCobnnCdmnG20mQtDxRGdmbSO6cAVxVlGj1Jz0faxKCoGUJy7eNbF+TNbDCG
ElrquWJ8awHoWp+mLADOhr4yvbe+SLs+lT2eQsQ+YuFOry+PGHZc4ztL7IEWfhRJPUmLDPmIJTNx9j+I
G9UrUuyY50FmJ9YYy3I6AT/jrZv+tF/
3IUoCeYmUFHeHAZMZ3vbdP23hIJ5vpJQbiyppPNXhEr2HmAKJ6/
fyuS4NnBXmQDrR+W7fwowN7FJXYhQEowE9iBKc06wXewxGkTlO74rMaHta3uYpFWIMtvuwE6Er+k5tl/
JFmwbUnlO5iYsGR6z/Y8O7+lR8e8q1BGFv/WuhKJjVqIpzQgi2fw84Rn/
QimLzkflGRlENNi6VdLdwG3A2YRy89RyQFxfkgpwTSJBswRhQ3K37ZWzBlaobVQXJ5L+Aj60PSJvVK1H
sojpRhzIf9fhtcmAz20Xm4VOQkoqL0zsOT4i9n3DqgfCkiYkGmp9Xeb1saMyN69GiCN2ql7PHF5hDEha
grCLeQN43cmDvAgnmk+ax2cnDmGusj20FZ6dkmBuAh02Rj2Ixhs3Av0IJchxaQEDYWVUbkqdSerxo4nP
f2HiHrxDJA+GEqUKb9qeJluQLYykt4Hl0sA5M/
G5H0OoCbYlSth+V0oNG0c6pb4H2LJW2izpC+ATQrH5GJHofztflO2NpE2Bk4BJiZLOG4GbbD+dXi+Lvw
aTqlmWJjppL0go1D4mDsWus31JxvDaGkndiWTyI8Ao4oByxWRr8gawju0BOWPs6qT5eRHgZo+h4a6kxY
lE5tCmB1eoWimtSjTRWhiYD/
i77f0lzWx7eN4oW4uUsPzeZ1bSOMmqalFgG9u7ZA2wAPzg2VgDOIrRDcW/
IWzGhgBP2r4xY5gtRyXHcSRhtXeM7TNzx1X4IZXnYzdgc8IatBdxQDmQUPZfWwRGjSPl+uYGZgJ6EuPT
K61mjwGlyV9TqJxC7EJYBJwOTEUoDsaX9AQwyPZ7mUJsB54jfBv3BSYkkge9idPtkURpyJBs0bUwkvoS
h1lDAWwPl2Tg6KQW303ScML7qVBnKknJ9YD3K8nlxYHXbM8h6XdEQ7k3iQRnIQ/
XEU0w5wOWA34BbKtoCvEbV7o9F+pHSiB0I3IInxB2C/
3Ta1MTCed1iDm7KHMykTZH5xAWAC8Df07J5YWBcUpyeexQNEo8iLCs+lLSRIQtxlBiXHrI9mM5Yyx8zy
GEXc9Wki4HaiXNh0m6ulSEjT2Sxrf9ZQflqypv2QKYrvmRFf4LfyQOgw9OCvOl05/VibzHjWUOrx+Vz/
ETwlrsAEm/J/bUDwIP2x6YK75C4NG+1/sDu9u+XNIw4HxgU+Jw4FnglSJmqS8V3/
HNgD8BExDClW+A4Ypmo7fbvj1jmHWlKJgzIWkKolR9O2AZIrk2CFiilEHXF0lLEc2xvpA0B/
A2UQK9BKFQmwUYBtxp+5lsgbYYFVuG3YCDCQ/
mV4nE2f41ny5J0xP3Z4pswbYwFXXBpUTjnwPS9VWAxWwfnX7fgmjctHnGcNsOhS/
2RsSiYxChIh+Q1JqTECqcRYFbbb9XFn6NJfkI7kfME1/
a3jdzSIUxUFHjTAv8BcD2npnD6pJU5ogDifXQTra/
lrQZcBzhT74k8B7wWxef66wkFdTrtqdLv78FrGT7BUnPAjvYfjBrkF2clEg+llizvkAky16t7s8kDQYO
sX1xnigLY0LSTsDLHkNfF0kT2v68rKPqT7Ia606oMxclPGbnACa1vVzO2Nqdyn58HuAW2zNJmgp43vbk
klYiclH/
dPHurzuVNdZQ4GDbF0kaQPRem5tYd21j+65WGZuKgrlJVDZDiwLfpJLnK9Ifkvfmso4mNuVktU6k5OWD
wGfpYb4cuDedpt6e/
hRFWgOoDJDXEqU4Z6XfexEndgsQXc+3JuwACg2g8r1+kygrr41HdwB3VE5W1yf8sAtNoLKI2ALYkVAIL
gxcIGk1228StgwfEhtc4AfPVaFOVBZ/vwAOBJ4imm4sCeybDmNG2b4nZ5ztjqQZgDWAvsA4ko6w/
aakPwMT5Y2uJdiY2PzUNphrAf1s75csGXYkhBHXZ4qvEEwKPKBoFD6AGJteSF6O05fkcl2YBpiSaHK5E
qE2e0PSS4TF2whiHLouV4CF0VT22DMQh/
VfSnqpo5WPU9Peso5qCN8QfSzet30ugKS5CCuAYvGWkcrnPjOj99uLE9XdAJ8RNpVHlvtUf9L+ohfQ2/
ZF6fIMtndOVZLHEQKjlhmbSoK5+ZwIXKbwY659ica3/
TzwfEl01heHp283SYsQpQm7AidI+oBI6FwP3OHiWdcwbL9KWJPsmwbYBQj/
p2uJpOeShD1DobHcAFwo6YqqUj8tyicjOm0fnCu4NkTEHLA1kcQ5DUDS7cCqRKK5zAfNoVb2vBUwwPZe
kv7EaNueZYiF+T1l8d1cKsqbqQh7sd7EwfAOwD5pbh/
P9kM54+zKpM1PNyJx+UTlpXMIb0Zs3y5pfyKJUMiI7ZGS+hHrpvWAZyQtQ1REXpsztq5OZc7tBRxKfN9
nI5pTz0eozVYhEtBDXRrEdQoq5f/dCI/+uYHBkl4HbgWuJpSbZT1VRyrz80bE/
trA9JJGAv+wfVXtvWXd1Cm4G3hf0ZNnONBd0hnEQdpd6T3dSWKkQl2ZCbhJUg9C3V/
zXZ6AEJi2lA9zt//+lkI9SEmc7sQC5Uzb39qudSSeSNL5kmYok199SZsmbD9pe2/
bcwLjEl7YnxIJtZclnZIvyvbB9oe273V0GZ4N+BWwCfCfvJG1BQ8AtwAPSjpJ0oqSJpG0PlECfbvtQXl
DbB8qY/1CjFb3Q6j9X04/
lzm6CVQ2p72IzSnEuHRN+nkxRqvIyz1pLrXPe3Ogh+1lCYX56+m+zUQo1gpjRy8iubxm7UJS7H8sqVva
FC0G3Jcpvran5v8raRvgGcInfiri8Gt3wsbhL9kCbC3uAgYTVY8bpp8PBP4MXEQ0xDowW3SF75G0vsIv
Htuv2t7C9qJE2fmhwOSEwOL49H792P9V+N9IyeWewNHEPmIfYq6+jPBiXipnfIUfYvsL24/a/sD2s0Q/
i76ETejR6W0lD9UYXgLOJObs14jc02nAX0lK8rTOagmKB3MTkTQv0QF9DdsfV673IrpI9soWXBsgaXPg
KdvPdbg+FdDT9rA8kRUKzUPSzkTybE5iohtGLL7/3monqJ2d5If2LLFhHUgsMi603TtrYG2KpDWJKqM9
iEOv3oSaYzCwmu2hRcHcXCoKqTOA92wfIOmC9POekg4HZrG9ReZQuyyVz3hfokHWdoQ34Ae177qkvYDN
bC+eMdS2J+0XhtieqnJtRiLX83q+yFoHSeMA5xF9Wp4AvgRWJg5/
BxIq8XuJZphF6ZcRRQ+Lc2rjf6qyeB54smNlqqSJbX9aKsPqQ8VabCOi4e4yldfGJw5gJre9Y7YgC9+T
Dlb6ACsALxQrpbyknNQhwB3AcbaHtNLYVBLMTUTSJMTpxedEB8/P0mJxC2AT28tX/
```

FALdUTSpMSicFzCa2gQsUC8z/
a7OWMrFJpNeh5qG9RuRKlnS0xqXYlUprYeMCswD6EkmBW4kEg831dU5c0jbVb3A5Yl7sd1RL
OaG2wfmjO2dkfS6sBeRNXLY0TC/3VJDwPH2r48a4BdnIpy5njCm/FRQrU/GfALovT5VNvXjPl/
KDSSisfsAsBuwG62vxzTe/JE2HpIWptoTv0wcAnhy/8bwqt8/nIQnJ80Z89le6Ck6YBTiLGqO/
AuMYYNAJ62/V6+SFsXSRsQyv7tk91n7frWxKHkGmVsyoekHo7+Xn8g8k3fEuKJSYgDsweBy2y//BP/
TWEsUfRam5vIP71ku6XtxkqCucF0PI2QtAShkJqM8ACejfC6Odr2HWUQbgxpETIv4Zs2F/
GQLw5MSDT92y5jeIVCoc2QNAvwbjpo7MHoUud5CCul2dLPJ9ouFj5NRNLKwFLAB8Brtm/
MHFIBkHQx4RU/PXAUYY/
RDdi2Y7Kt8H8nqc3WIw61RgKbEom1XsBQosHZqY7m1IWMJCX534CbgSOB58rhcH2pJGTGA1YnLPXeAQ5
NfV0KnZBkkzElodKcg1hDTQNMBzxme9+M4bU0ks4h9tWnEvYyawA7Aafb7ldyG/
moKM0HE83kbiTEdnMT69z1gVNsn9lKCtrOQOVgeCPCxmouwrLnEyKxfwtwre3XMobZEEqCuYFImglY3f
bZqTRholSeMxGxeF+eOF290vZbOWNtJ9JmanJgS6JL+hG2+
+eNqlAotAtpDLqUsF54Lv39MtF9e1R6fRoi0TzQ9mvFmqGxSJqCsAaYk/
Arv6YonvJSSfTsA7xq+xJJqxCbohmBEcCRpUz951GxxvgNoYrd1vaQ9NoCRBL/
G2CPjuXmheYiaVHbT0jahNg7LEAkCT4l9hHPEPZKn2UMs2WQNE5NYSZpTqKh4i+JhtU3tbr6rKtQTYhJ
uh/Y1fZT6fcJiUP7uYARth8pCbTGkGx6tiX8l2cBHgfOBS62/XnG0Ap8L7K7Fljf9leV6+MSzX0/
sv11rvhalUqC+WHgVtsHpeurEt7+WxMWP7u32thUEswNJPk5/
sL2Pqm8cwfgeuDxUvbcXCTND7xNJHC+rVy/jkgwP/Kj/
7hQKBTqSGqKsiuRzJyVaKoxgkg0DyKaNb1me2S2INuAirJjNqIE+n3gVSKB0xc4wfY+OWNsZyoJ0McIB
W0/RePecaqbpMLPo7L5uRZ4yPbf0/U/EWqbR4GJgDeJwxfKIVfzqHz/
FwMOtr1OTaxCHEDOBsxOJNDmATa2/
WG+iLs+kiYnbDCmAZYhqljeIpTMUwBb2L41X4SFKkmw9StCNTvE9viV12rPz8rAA2XOaAwpSTkx8Xy8S
1S8TG/71ayBFapr3NmB04nx7HDCouHTvNG1D5JOIdawz4/
htdo6rKVERCXB3CA6flEUDf7+SpQk9CI6SD4MPAncY3tElkDbAEl9gH5Es47BRFOz9wj1R39guqL6KBQ
KOUje/
EsTqswFgWmJBfoHwPW2L8kYXstSnaNTc7MlbW9YeX1VogHHgbbvyBRmAZC0N2EldmaZq+uPpJeBDZKPa
S/Cs/RY4BxgYcKX+S+278kYZttRSQ6cQDTL2rJjqbmk7kRFXm/
bL2QLtotTSUbuSXz3+xPly92Apwnf9/GJ5qKl1L+TIGlR4GJChTkh8C/
gPuB221+k5+Mb290yhtlyVMam1YHDgAmAF4keU/
1s35Y1wMIPSJUvfyB8l4cBrxNVk8OBR1wavNedyjMyK3AV8AawS7tUg/X4728p/
BzSQqW6EOxjezMASX2BNYmuxDsSJSUjWu30IheVh3oR4AsimXweUW7+a6L5w+eE5+lFZcNaKBSaSVKhd
SOmik+IzWz/
9NrURMJ5HUDpWkuVTnUSNpH0CdHkZDKi6ev32L5d0rbAKsAd5R7kIfWt+BvxvIwr6W7igP6D4rs89kia
gLCEmZ94FnYkbBfOTCWzDyqaZxUbt3ysAJwG0DG5mX5/J/0p/
Ewqe68riSTMxoSN3uPp+ijgeWACSZ+VvVqn4Wli7LqE8ObvCRwNTCjpHeAj4BooTTDrTDei8u5CYB+i6
q43sCRwhaSdbV+UMb62R9IkaX+B7csk9ScqXRYnql5WICo19qfMH3Wnsl/
oRVghzgfcIOk1Ytx6BLjD9seZQmwoRcHcQCSNZ/ur5E30gO2ZfuR9ZeNaRyoJ5jcINeATRPKgP/
GQT074Nw4BhpbPvlAo5CJ5ne5HWPh86dKIpuEkj+vbiG7z7xLKgvUIz9k70rUvgDsJC6WbyzzdfNJBzD
jA2sASRJn6lESCeTDhOXtXvghbg6RuuogYgz4gfK0vSa/NQaxfp8wYYtuSys8/
JxJozxDqs5eJMev9kjBrHEnNvyQhTFmFsFB6A9jL9os5Yyv8kOQxK+ArovlrH6IqrBtwhe0Xyhw+9kia
qCbKSs/HHbYX7fCezYjDmfVKf4Q8SFoWOM/2HKlKck7bj3d4z0zAosBtxS6jviSB49O173/
yg5+acDGYj7BHXAI43va5rSgwLQnmBlAptVofWJYwvJ/B9jJKTWvS+
+YG9ra9TcZwWxJJ4xCq5SWIBPOXxAJxWqL88yai5O2mVnuoC4VC56ZyCPYL4EDgKcK/
bsm0IFwFGFVK0htHSl4uCKwILEJ0nRdRMvhdunY9MUeXOaKTIGkaojnv5sD5ti/
IHFJLkCrrFie83+9P12YhrN3Gsb1Vvujaj8o+Yj2i7P9EQnE2KdGB/
iWiud9QYPiYvB0L9SPNF9MCqxINYFtSddYVST0ttgOusz0sdzytjKRDiYaw1wHPAgsRTcrurLxnFeDft
mcoSf3mU5k7ZrL9qqSNgCsIlfJjxLq2v+1XcsbZqqQq1H/Z3jD1DFmHZEtSmzckTUYcgg2z/
WFJMBf+JyRtDSwHbEIsBu8FRhKLwgFEE5WJbP+ulO40BklrE52fHyYUIFMQDTw2AuazPUXG8AqFQhtSa
erQjyj130vRWGtJ27+R9DdgZtvbteLCIzdj+kwlTUxYkyxLqAsWA660vW+5B/
lQNGD8C6HiHArcXDZGzUHSwoRq/DbbT+aOp52ozBH/
Bl62fXC6PjPRfG5ZosHfVMT9+Uu2YAuFDFSekW2I5osrK5o0bg3sS+z59nZp7lc3FM3iliKU/
QsA8xL9jC4mrEimTNcft31pyW3kpfr5S1qTyEetRtynb4E/2P53xhBbjlQhuZDth5OS+WIiwfwqUUX/
PJETHNHKY1NJMDcBSScRPl590p9JiAG5N/Bn2/eVU776UlOKp7Kp1YFtiNO7Q22/
kTe6QqFQAEnXAJckf7RHgRNsXyzpWuA+2/8sC/
TGUdmgfu9VV3ltRmBi28+X+bm5VO7LasDehF3J54S11czAScCx5Z40lqTapByu5EPSYGBb2/
ePaS5IG1jZfiJPhIVCHirzxA3A/bb/LuloohrpDqLP0XG1ioxC/
Uh76ykIVf+ywPKEoK4XYVOysO2X8kVY+CmSvcnmwKO2Hy/7jPpTGZ/mIg5kliLGph6Ep//Fts/
JGWMjKQnmBlEpURgHmL6muEk+LNMTG6VnXTp3NgxJ49j+Jv08J7A9oWbel7DG+CZnfIVCob1JioITgT2
A/
xCHjt0Jf9nVbA8t6tnGkcrX1iO6a88O3A3cQ5TajswYWltTWZhfDTxj+8DKa9sSVUg7uE26cRfaF0mz2
x5a+b17+tHlgKVQAEm3Av8krMbuBXa3fZukh4CTbF9SDokbj6TehGXGwraPzR1PO1PJQU0CrET4yBu4n
6h4eSVjeG1NqgxbC3jEdss2EC8J5gZR2SDtSnimnWT7k1TuOTvRNKWYqjeAVCL1G6I76jJE05q3CCXzF
EQp1a35IiwUCoXvVSD7EQqQeQhfu0WBG2wfmjO2VqbigbOWoYb9F+G/
fAzRxGl64Dnb82UMs+2R9ABwoO070u+1+/Yw8A/

```
bV+aNsFBoPElJPsmYfH9bdXNaKPxfkbQBcC7RP+FG239N10cCc9h+N2d8rURlDp6NUC33BT4DBgIDbY/
IGmAB+EEOaj9gK8J3GUJJOzWRbL7A9hG5Ymx1ksB0TWBXwh7jFqIp5odZA2sSJcHcICqD8NNEiU4/
SbsQaqlpgbNsn1zUafWjcmK3J3As0J94oLsBTxPm9uMD75VSkEKh0FmQtDJRPvUB0WTrxswhtTSVxfcl
wBDbB0k6gjiAPIxQlV9i+4pSOpgPSdsDBwA7A4/
ZflfSRMRiff6ymS20MpVxak1gF2Kz+iJwO5EweNz2+zljLBQ6A5LmBqZIlpMTEr79a9lesuyz60clt3E
7UXH3LFHuPznwDeHre4TtpzOG2fZU7tO9RA7qmqRm7glMR4hahtu+uqxx60tl3t6KEBBdC8xGKMknJda
vF9g+JF+UjadH7gBalfRgjwvMYLtfunwAsBMxGB8q6fpSplA/KguIKwmf642BLQn/a4jP/
XlgAkmflQVHoVDIhaQpiM7ncwIPAGfafi9vVO1BZTHdB7g0/
bwacKTt15P17Gcd3ltoPhcTCqntgQ0kzUoc0PcryeVCG1BTJp8LnE7sIRYjGjVdAvSS9CvbV2WKr1DIj
qRpgAmAcdMc8QpwOVA7qBeh2CyMBZXEWU9gUduTpTzHjMQ83ReYj+h3VMhIparlPGCcdO0T4BPgDUlPE
YcBZY1bf2pjzcrAubaPqb2QertsTggfv+8X1vwQG09RMDeAysnRTMDZwENEU7/
FbK+WOky+aXuyrIG2AcnIfknCf2gV4H2iBHov2y/
mjK1QKLQXHcoLLyHGo1cZXWp4gu19csbYLkjqQZQOvpr8Gm8iKl7+Q3R6Xtz2i0X91Bwqz0Y/
QgE1NF3vTlR+LQC8DbwG9C89FArtQErmXG97xTG8Ni/wuu2Pmh9ZoZAfSTsDhwLPACOBj4FTbD/+k/
+w8D8haXUieXw3IU7cAfiL7Q86vG8i2581P8JCjUo190TAhsBxwEVELuop20OyBtgmSPoV0Nf233PHko
OSYG4QlZO+tYHdiE3Rv2w/KWl/
YE3bK5TShOaRfOymBVYFrhmTn12hUCg0gmqiUtK+wJK2N6y8vipwCBXP2ULjkTSe7a8krQfsT6g63rW9
QebQ2o7kSf4FMIHtr9K1LW1fkDeyQqG5VA5clgAOBP5t+/
LccRUKuak8G8sRIq7fEBVHsxKHkb8FVrL9TMYwWwpJewC7E+rMEcReegBhKfYWoYz9qhzG56emipV0DL
AuMAR4F5iMuH+jgKttX5YxzJYmKftXIBTkdxEVFXe1U++1kmCuM5LOINSxYzzBk9SX8FJ7yPZ/
SoK5UCgUWh9JmxKL8IHEoeNbto/
v8J5LgJdt71+aNzWWjsrkpBRcktis3m37onIPmkNFcbMecLDtRdL1OYBbbc9alOSFdiT5OB5K2L7dBdw
P3EMo0crYVGg7KgKuw4EZbW/
V4fWjgG9rzf4K9UPSDMDawBLAMkR19ouEOvZporqobZJonZHKeupZYFvbj6SK+tkJW7hFgKts9y9r3Pp
SGZv2IvZ59wDjEVaIvYiq1Qtsn5QxzKZQPJjriKTJiXLbzyRNBtxE+P8+ADxs+xXbw1JXz1FQvG8KhUK
h1Um2SLsQ6oF3CZuePSR9DtyRrn1BeNmdnyvOdqC2+O6YrEwVLbdJOpm4J4Xm0Y1Qjm9AbFRrrEtsWqv
vKRTaiXuAjYA5gIWBFYmy5yn+X3t3Hnf5XP9//
PGcxTL2beyDQbKWNaRIlJ1QEW2iovVbKolQqZQ2P6UUkUiLUKksFYayhDAYJsYyBjOWwcg6nr8/3u/
Dx1VaxnXOZ+Y6z/vtdt2uc30+57purzlnzjmfz+vzer9etbr/
qjaDi+i1xnnzzcCakuYfkNRcmlJV+1zCp9cxDjWdqljbk4Hj6xeS1gO2pbwvvR9Yvb0oA8o8qlpB+xfK
hUls30lpx/eHulLsfGNM+QAAIABJREFU2bo9yeXB1ey/fJjtH0laiDIEc0zd/igM/
femVDAPMknz2n5c0pLAuyj/oV5GeZE/SDlZ+p3ti9qLMiIieqm26HkF5UB8PcrngoA7KAd76wG/
Bj6RSs3B1ajoGEs5wLuCsszzEdtPNe43F/APyjT66e1E278kXVtvnk5J8h8D/
Mj2d4b6wXjEi+mcV9TbC1P6ka8F/LCzPaLfSJofuAAYTRnufhkwljJbYS/b12bly+BoVGbuSjmG/
Qlw+YBVYJ37pCq2BfUcY1h9DjYGvkfJPR0OTKAc807rtB+L7pH0XeB822cM2D4P8MxQHezXlArmQdY42
Jtq+yhJCwDLUJYmrA68mjKE4KK8CUdEDH2Nk5y/1a/OydEmlM+E1SgVms/
URGhOigZR47EcDXyCUmUwGbhc0mXARNsTgDdSWpdMz+dzK/
ag9NDcCtiPcuw0VdIw4FpJd1JWieW1EUNWo8fsksDuwGskrUzppfkD23+SdEnen6Jf1WOkGcDGkvahtL
c6lLISbD/b18ILPvvjpem818xFWfq/F/CYpAmUwog/2L4JUhXbhsY5Q+ci/
AzK87IyZSDjP4BJwF2SfpeVL91T25G8EthT0obAOEpLq7ttP9FudL2TCuYukbQ/
ZbDfTcB9tmfUq0tLAI/bfjRJhIiI/tGo8FjA9qMD9i0PzG/
7piQ3u6tWEawF7AzsRknuXw5sDPzR9v6dJaEthtnXJI2kPB+7U5IHywG3296s1cAiuqzxOfF9yv/
7v1DOJTamtMn4fE0y5xwi+pakVSn9TRemnFcnadZlkn5OudB1FSXpvBVwAHAPJYF5rO3T24uwf0n6EOX
x94Dty1H6Zb+KUn1+kO0L8vkxeJqPpaQVgO0pLQ/
HAPNREv+PAb92nwzrTYK5CySNpizvXJHyRnwFcAOlPcbtwN1Z6hkR0V9qJeZOwP6UVS0XUnps/
sr2Qy2G1tdqj7SNgD2B42xfmSR/7zUqNxe1/
eCAfaOBNWxf2E50Eb0laTow2vZTtUBlEeAQYAVg33xmRL9ptLvah5LYXIlynj0FOM32GUmcDa7GY74J5
TFeqbOdUjR3IPAwJYm2J/
B+23950T8Yg662fzvG9g6SVgROAU6kJDTvH3DfHNsOMknvBk5vtqySNJxygfhllPO9DYCzbZ/
dD89BEsyDqFF1sDPPD6d5DNiMklSYQUkmXAF8Ox+AERFDXyNxti2lr+x3Kf2Xv0IZ+LcscKPttVoMM6I
1jZPYvYC3UJYY3gqcBZxp+67m/
VoMNaLrakuMM4C9bY8fUCE1BVg+hSrRjyQtA1wJvMP2HyStTxmEuQ+wi+3LWw1wiGl8Nu9CucC1F3BL4
/2oM9Bs89p7doTtfVsMua9JWgo4EngtpUXGFOA8yrHUbyidY3IMNUgkLU5JLm9VV0d+h5Lruxq41fY/
6v3mA552Y+7LUDas7QCGqMOBcbaPsX2C7XcD21ASztOBzwAntxhfRET0jur3d1AqQL5GqUb7PqUP85mU
z43OVe+IvlJPYEcDPwB+D7yPsgx3X+BmSU9KGp0ToxjqamXgbcDvgC9WqG+PpaU9GngjiSXo9/
UFWBQzqdvrsnlYbavsv0Z4JvAB9uLcGjqzAUBzqHMEPkcsJmk0ZLeCHwK+Gu9+xRgwXYi7W+Nc4e9KW2
UVqWspD+UMuzvLOAbjeczBoHt+21vVX8cRelT/
nHgbOAcSV+W9CbK8PC+SC5DEsyDqnHANxl4ZsC+K4H5gaMofdTWqFddIyJiCGt8NqxMuaoNsDXwO9uT6
8+PDbhvRF9onOy8ijJ5+7j6/
RO21wFWAXazPbW1ICN6xBVlmfMIYJKkqcBvKa18jmgzvog2NJaUTwKelbTegGXmwyk9mXOhfhBIGilpM
XjuPelp4KvAU5Rk8+XAYZT+8EfXx/xtwK9aCrmvNc4dDuX584k7bf/
Q9puBkZQCR3i+6CVeIknDJI2oP24LHFCPW7ejJPVfDpxAuRBDvyT3R/
znu8QsOAE4vSaQzwMmAKtRDgwftP24pJWAe1uMMSIieqQegHyPMs0Z4H5gTF3O9jrgk/V+aQEQfaXx//
1iYFNJa9q+obF/
```

CqUyKmJIkrQKcLTtXerPI23fCGxbl+CuByxJ6ak5vcVQI9p2GeX46ZuSzqMUda1AuWj/
9TYDG2LeDqwOfELSEpR+8DfU7UhaExhm+/
r686rAacAvW4q3bzXamCxAab23DDCtJjM7F1tm2p4BL7hYEy9RbX/YOYY9gbLC4kLbEyj5v2/
Bc3NEoCT3h/
w5XnowD7LGi3xr4K3A4pSpz48Ch9v+RV1ScoLt5dqMNSIiekvS3LaflLQTcDBlMMq0TmIhol9JejPwBW
Be4I/AX4BLmsnmiKGoXmjcwfYPJO1IWV57BqVNxnmNlS4RfU/
SSOBDlAsv8wJjgM8Dv0nybHBIGgMsYvtaSUdQ2o+Mo7SuuhS4lpLbmFnnT6U4oiWNOS8HAV+kPEfvsX1
dy6H1jdp/
+SjgZ7YvbWzvy9dFEsxdUK8gPQasSel7czulsfe0ehVwD2CG7R+2F2VERPTKwIMMSQtSWgK8hXK1+9R+
mCwc0dS4KL8GZXDT14F7KG0xVgYWBZ4FtnFjQnfEUFQTZzOB3SmVUNsAS1EGXl5O6aF5VXsRRvRebb+w
ArAYcG9j6OtIgNq+Ibqg8Rm9BbB5/
RoLPEj5rD7Q9k0thhhVPY7albIqcjPgCcrKsPOBH3QGzsXgaST3twOOp+T/
PgTcSHmNPJ4Ec7wkdQDB9pQpqy+jVN+cR6lWfnTAfUfYfuaf/0pERAwV/
+nqtaQJlNUtpyfBHP2mcXC+H7CR7f1q0mBhSkuAscBCtk9pNdCILvp3nxO1RcYOwAHA/8trIfpF4/
PhQOADwOOUpOaHKRchtweuB05J65jB9R/
ek+YHtqBcCDvY9pR+rdScXUlaiDLza3tKW5PX2746z1N31JYxbwfWB1YFplEqya+jrKy4s8Xwei4J5kE
w4OrFl4BjKJUG21AmoN8G7JjhTRERQ1+j4mMssCVwBaWH7CPNKcKS5qL0ZF48J0fRjxqvlf0oJ6yftX3
rgPvkwkv0hZpM/
jLwDeBmyqwc236y1cAiWlJXe91GqcicSZlXsQrwDGWF8GbAZ2yn9+8ganw2zwe8GXgnZXbU72z/
qN3ooqORg3ot8B5Kwv/
utuPqZ3U45laU87+dgbfbPr+fkvvD2g5giOhMhNwL+IXtE2yPt300sAnlcX5ra9FFRETPNA4gRgOfAH4
GnA4cJmlHSS+v+99IWe45va6AiegrjdfK5pQJ3N+UdKCknSWtLmlUkssx1NUWAFBaJq1U+44vBRwNPC7
pNEmLtBZgRI/
VAWVQKjBvtj3B9kTgh8CGdfuB9ef924lySOsck36V8lifBLyG0roKSftKWr2d0KKh8zy9C3i4mVyWNI+
krWuxSwyyznuUpFUkndh5nG0/YPuntt9neynggrq9L5LLkATzoGhUJt8DjBqw7yFK/
8AF4Lk2GhERMcTZvsz2asArgYMonwVHAWdJOpmSPPh1vXs+G6Iv1eOi71GWQE+kLOt8D3Ao8LkWQ4vot
Z0ow/
2gtNubF9gamJ9SqRnRLzrHRLsBtzS2bwScbfsp2w9TPjMWhJxjD6ZGbmNX4AjbJwNTKYP+oFQ0rwQvuB
gQvdd5ntYBToNywVLScNtPUCr+t6rb8zwNrs77zd7AYrZv6+yQNFLS5pLW6afEcseItgMYYk4DzpD0NH
AR8AiwNLA6pXcaQN/9J4uI6Gf1IO+v9evQ2httI8qB4Yn1bqnSjL5UK5Q7J60/kTSKclFmS+CpF/
3FiCGikcx5DJhL0jaUhPJ7bf9Z0pGUJHNEX2i8Jp4GNpR0HnA1pc/
plxp33ZEyxAyeX1Ecg0DSkpQ2JDdLmhsYY/
uCunt1SjvQvqrMnN3UNiYjKEOS3wlcNqAl65rAB1sJbujr/L/
fFDgZnrvINdz205LeDtwNXNdP7TEgPZgHnaQtKdMjR1E+FFcFDrN9equBRURERMxmJC1LaSUzChgOXAM
cl7kV0W8krUtZ2fIEcIHtb9QLLvcAK6RXf/
Sb2lJsDOV8ei1gXUrl4FTgz5RVLhvavqbfkjjdVBNlosyVWpYyS2QX2xtJ2gX4gu21MiNh9iBpPUq7mA
uBU
ylFpO+lXBTYssXQhjxJB1OKIj5s+97G9huBj9o+r99eJ0kwD5Lmh1qdrroRcB+lT9RU279tM76IiIiI2
YmkeYBzKCu+bqyb16AMcPqA7altxRbRbbUo5S7bEyXNbftJSasA89q+XtLCwPuBV9l+U7vRRvRWY4DZ0
pRWkw8Ay1D6AK8GvAJY0famLYY5pNXBo4cBr6cMq34UmBv4ju3f1FYMuRjcokbri62Agym5p2nAz4GTb
N/YbwnOXpK0KnA88BdKB4P5gG0ox7Kv6ceLXkkwv0T/6WqppFsoFcw/
yZXViIiI6HeNxMFuwKdtb9DYty7wTeDE2vcxYkiS9FNKJeD1kvajtE26kpIceNT2Y5KWA+a2fWubsUa0
RdKvgItsf62xbSSwAjDM9i05xx4ctd3CXsDNwCTb99Uk82bA+pQVRtfm/Wj2IenVwC22pzW2jQKeTPK/
N+pzcBClL/kUSmuMr9ke34/
vTUkwz4LOf5Q6LXJLyrKRKcAjtp9q3G8EZZnb4lnWFhEREfGC46hPAevbfkvd3kk8fxB4fao2YyiTNML
2M/X2KZTKzCeBycBNlKr+W4Hx/XaCGiHpJ8CZlAuOG9u+s27vfE58BDjf9o3/7u/Ef0/
SOsC3KQmyh4A7gGuBG4C7k7CcPTSOoVYDfkXJR00DNgDeDFxn+4dtxjiUNR7/
NwB32L65bp+b0hq7r+eHJMH8EkjamNLU25SDwcuBy4CJtidI2pHSR3C5LE2IiIiIfidpeL35LGW586+A
3wDftX1PHYJ5NnCm7W+1FGZEz0laANiEUi34CmA0pXjlja0GFtFj9bVwLLAFsDxwF2WY388p/
clnSnoWGGv79rbiHGpq7+VXAmtT3oPGUAaM3gPcX7//
yfY1rQUZdFqTSDoE2MD2LnU47OGUCwPDgSNtX9RmnEOZpAUpeb+dgNsor5ePUi7KfMX2jBbDa1USzIOg
9hBcC9gZ2I0yfOByYGPgj7b3b1YpRERERPS7ejK7B/BhYC7KoD9RBtV8xvb97UUX0V2NJMGOlITAebb/
0di/DLCy7XGtBRnRIkk7AftSBpi9h3JuvSgwCbjG9u79uAS9l+ry//dQqmPXogwzOzaPe3saVfw/
o+Savivpl8D1tg+T9ENKZe3hKXIcXI3P7bcBH7e9fm3t9h3KyqMlgW/38/
y1JJi7oFbfbATsSalgvjIv7oiIiOhnkvYHdqFMOT/f9j11+zzApsDi9a5n9fsSwxj6GkmCm4GDbJ/
Z2LY4sIjtiW3HGdEGSSNtP/
0vtq9KaSdzg+27co49uOqFrfUogxV3oQz1GwmsSLn4+0Xbd+dxb5+kdwH7UwYjTwM+Zvs2SeMp8y1+ne
dpcDU+o78JTK9J/O8Co2y/Q9KXgKVtv6tfH/skmCMiIiKi6yRtRbn4vjpl2fPdwDjK0ueLbT/
RYngRPVeX2U6yvVj9WZQq/
pHAr4F9bE9uMcSI1kh6ObANsBhwPXCJ7SntRjW0NPrJvg74LKUNxsLAQsBZwCPAONs3vljSP3pD0iK2H
2omLiW9idLK5Pz6HG0KnAGMyXM1uAY87lsAx1FeH7dThvpdIelc4Bzbx3SqnVsLuCVJMEdERERET9QhK
PMBbwE+R+lXNz+lWuoaYAKlQuqh1oKM6BFJL6MM1TqgWa0saQxlUNPCrQUX0YJGwnNNSoLzLuAWYBXKZ
8UDwHWUNkp9Vx042BqP94+AvYHHgAOAUwc+vmmL0S5JXwYm2D5J0nrAowM+N+YCtgKWsn1iv1bQdkOjN
UYzyfwG4HXAabavQJ8EtgQ9uT+/

X1kgRzRERERPRMTRycCuwDTKUswd0KOJoysf6Nth9rL8KI3qgVyz+gDNY6mDIsaBSwF7CK7V1bDC+i5x
qJnM8DK9Rl58sCS1EqNV8JDLN9aL8mcLpB0mbAgsCOlNYYS1J6Xf8W+Lnti1sMLwBJrwHG1yrmYynJzT
uAG4BLKCvBcnG+CyQdAzxj+2OSlgemNKuTJY2gtMhd1/
a324pzdpAEc0RERER0XSNx8Algk4HJM0kfoRybfrOdCCN6r/
YgPxJYl9JHcxPgPOCo9GCOftOoqP0sMM32cQP2zw8Mt/
1wOxEOPZJWo7SrWqpRnbkq8AZgZ2AdYDQwOsN321cHJK8CrER5btaiXHyZC3gC2MP2tPYiHHokLU1535
ks6VRKu7fxwO+An9m+qtUAZyNJMEdEREREz0jaFTgE+JTt8xvbv04ZlPL+1oKL6DFJC9p+RNLKwKuBS4
G705M8+lWt7D8O2J7SSukS4D7bD7Ya2BBVqy9/Dtxm+
+MDWyvUdgwfsf3OVI23Y+DjLmkp4CHbT0qaF3gZZb7FSra/1Fac/
ULSK4HtKO9R6wJPU9r57GD79hZDa10SzBERERHRU5K+ArweuIxSOfUGYGNgP9uXthlbRDc1ptAvB+xEO
UFdj9JT9kRJc9l+qt0oI9ojaUngq8ASwCLAvcDN9Wui7XEthjck1f6xJwBvs/23um1xYH/
gHcC5tj+Yvr7tkvRhoLP66xlgIqWFyR/r/rltP9lWfENdXXG0kO37Gtvmphy/
7ggc0u8Xh5NgjoiIiIieqhU321J6zS4P/JUyKOWSVgOL6LJGq5gfUhJoRwFfBMbZPljSx4E/
2b661UAjZgN1EObrKK1jXglcYPvAVNIOnsZFr+9Rhvx9Eng38H/
Ao5Qq8vNsP50Ec3tq25g9gHOBh4CRwVVhlH2z6lxfCGPEn7ADsAa1KGVV8GnER5beSicJUEc0RERER
0VaOv5tzAMsBw4OGBfQKTNIh+Iel+YEXbMyT9Hdjb9mWSLgKO6FSkRfQbSXNResvOTxno16nOnJ9SPXh
3Ep0vTeMzeSVga+BiYGngdEpv2THAV20f32KYUUlanfLcfLBZwS9pLGVg8eAdWzf1VKIQ1LjdfJ/
wDuB6yivlVGUVXhrAN+yfWyLYc5WkmCOiIiIiK5qVEgdCewLzABuAW4HrgUmA3+x/
UB7UUb0hqRlgV8Ab6e8FsbbXrzumw/
uxFkOMaIWkUcChwLuAGyjVma+gtMp41Pb09qIbeiRtTKnCNHAPJWE2Cfik7XGS5un3Jf9taahw7HU5JIO
9aL9Q/Y3tmvY+AXwPnDByKGS+dpEWAvwAf6swNqY/
5kpRhf18FNk97t2JE2wFEREREExNBWT5AWpiy9fTmlgvlVwKbAbpQJ9du1F2FET91DSQgcANwK/
Bme6695Q5LL0W86rWOA3YHX215a0huB42qV/w7AFkCGwA4i25cBL6+9ZdegVDO/AzhW0q3Aek/
tD2pzTiD7YGjAZo9ljs9lyWNB1au21LdPwgaK+p2AqbbPr/
zeNft9wLfkLQm8FrKgN6+lwRzRERERPTCMsCptm+tP98CnoFqaL5+ilp+HVUvCG8I7Ed53CelhVXvNRLFKwNrSXqSUmF
ZlOrNiH7TWVb9euC39fZ2wIX19fZ2wIX19hhgMXhBMjoGSa1Svrp+HVUvCG8I7Ed53CelhVXvNRLFKwNrSXqSUmF
+q+0ZjWTzlsC32ohxCBtG+UzeFTgVnk/uSxoOjKyvm/
GUIX9J7pMEc0RERER0kaQRtp8BNgPGSNrT9k86++sJ63WtBRjRI5KWAo6kVGeD/wSOBuYAjwI/
N72I60FGNGCmrjsJGUuAN5Qb2/
B8xXL2wKn9Ti0vlVbkZXfvzrbklxuQW3R8GNgLGXQ5WPAvZJuB24GJgCrUS/
M9HuCc7A0LmJtCAyX9CDwV9s3132d/
ZsDV9Xb6nGYs530YI6IiIirpN0PGWp4WKUhPKvgDNsj281sIguagwbEppL7YmpQewgcWB0bbXTAXt4Kr9rvejJPeXp
3x+z6Acw14KnAP8HdjW9vhU+SfBHBERERE9ImkksA4lWbBdvT03sJjth9qMLaIbGkOaAeBY62Pbmxf
ywl4XyD7ffmBDX6iaRXAXfZntLYNj9wMLAupbJ/EeAD6QMc/
aizCkzSK4F7bd9bty9LSTRvANxs+4y0j+kOrwGsACwNrGh7gRbDm60wkRwRERERPdEZk
NL4eT5gXduBXtBhWRNdJupdS5XRN7d84DJhZk897Au8G9m/0KI8Y8iR9G/
hBfV3sTUngXGR7Sq3YfML2U+1GGdE+SVcB7wX/+1kkiJ6HcXY3k/gnAebZ/
KmkuYFFKwvnllPeoX+a5KNKDOSIiIiK6otEeYywC7CkpMp2Key5Bp7r7dg8
Ef0N8JkB2yL6wQmd1wXwGmBZYA9JU4FrgfGSJnSqaRNiP6SWMFzJbAAravqhcoO9aVtK3tz7cV41BW54cA
zAd0BlE/U9+P7gWuljSs3jef35Qr5xERERERg66RNP4usEe9/
R7g7cCJWFmSVktyOYSVkZ2BP9sCIyRtJGkNSkGtLeqxTVQpDYz/Ek7qibVVstIl+saAgX2
Xwc+M6ApFpEv+gcG70CuBJKIrPxelgF2B5KMrr34Q19kpagzA55T61SfkGP6/
S8fqFUMEdERETEoGtUL28zIbNsqt1C2Dq8IrG5Ktecp+UmLA+8DjgCORuByyjVT+Mog8yuA/
XIiPYcROmz/BtJ11FWtPwEXS7tatAVcFSHay1P6TE7tBvH/
oiQ43w+cZvuRunsvFgW5AZ9Ty6+iJ8eCm3zFaUtLhERRREThHoGks7Pw2Sk7bPw2Y/
+WQ43oGUkLAxsBwaAqYAmwD72j6y6Ah+NjKluyqeSgKi/x
lh7MERERETHoGks7Pw2sb3t3Se8F3mJ7K0/
B9xdOunWcnIQt1Gr6YxLs2dgENNj3Tb3MMVMwWZ5DCW62Jb3dFR6IRxPcuM9l3h5wgRSbDNIYfQeAzOGjHlgT02lS9KX8zuRwMyF7TWlMBvHQ0mjthfSfwM/cEqQ+adUuVjp0u
vjhl5oXpc9b+pcLkHeCzHo91iOVB5HOYJ3Vcna69Ga9nDS0/
SbXUq3x8hG6k3MEGIBVrS4rK1SFZX3m3iHvDZN38TT3MvS7fGLvdFMgwBVo5zQF8p/
2Y6spvsD4eXzDUYkDh9i9j0brRjw0Uq7MN0/
Dvm/BEnRYzEBdBp+1pnVY2u0C3n9Yx8eSdAL5Dm3e1FeI89/1+FPxRsNBYcE0fUUp0iVS8GmkhFQ5
.

k3QIsC9wNTCG8llxPSWRc7rth1sML6I1ktYGTgI+BewIbEh5jcxNad2wue3HX/
QPxKCRNBxYgvJe9RhldcWdSe7/sySYIyIiIqIrJC0MrAO8DXgjcD9lAvfHbN/
aZmwREdF7ndUuktYAfk8ZVjYP8DvKcvQDgD8Be9r+R3uRRsw+JM1PGZy8FHCP7Qnp/
9sdkgSsB3yOcsw6yfYX241qzpAWGRERERHRFbanAxcDF9cD9qWBrYBprQYWERFtGUZZdr4z8Dfbf5a0O
3C57YMkPQU8leRy9DNJ8wFjgdGUHsC3U+ZY3FT3J7k8yBotL3YFPky56LUpsAHwRUk7AtPT8u3FJcEcE
REREV1XT4SmAD9qO5aIiGhHo2fpgsAl9fY2wBX19khgLkh/
2ehP9YL8FygtGf5Cac9wEzBJ0r3AybYfaTHEoe7dwLm2vyzpC5RjV4AtAQPj8t70rw1rO4CIiIiIiIiI
6Cs/o/
RehjI4a3FJrwP2Ai5tLaqIlkjq5Oe2ocyxWA74KrAKcDfwMWD9JJe7o3HxazhwWb39Zsp7FZRK5r/
1Oq45SSqYIyIiIiIiIqKrOlV/tf/ymsBP6q6zgX3qzycAvwFIhWD0GdXvOwPn2X5Y0nrAWbY/
IekfwMOQ6v4uOwU4TtKBlJ7Xf5K0OKVlyTmQ96YXkwRzRERERERERPTKfsAitn8saW7bk4H1JS0KzEhv
2ehHjQraJyhV/QCbAX+ot1cH0v+3+35OSSYfCDwAHA28Djja9gPpf/
3ikmCOiIiIiIiIiG7rJGUeAO4FsP3kczvtB9sIKmJ2Ufsvf5cyFBngPGBLSZOps2L2AAAKYElEQVSBzY
HP1+1JcHaJ7aclHUVpk/EaSg/
mM21fUvfnsX8RymMTEREREREREd0maW7gJEobgOOBc4FbgDtsP9NiaBGtkTTc9kxJP2xWCYgN4CpWX
DipQK2mNaDHPIk7QS8GpgHuAG4HrbM9qNas6RBHNEREREREREEdF1NMP8fsCQwhlKJ+SgwbDjB0cAqwCuB6fV3MsA
s+krj//tVwLOS5rL91ID7JLncHZ3hivsAF9t+v6RVKe9PKwNQtCkvz/
SRLMEREREREREETHoGsmYI4AfAffVn38GLEIZYHY98BHg4c6v9TLGiDY12McGgAOMkjQOuAO4z/
bD//aPxCxrDDFd9hPN5SLMysBmwBaUCzJ3SRpn+0Nw9e9I3Cc7eWS3P/
4KzDD9hPN5H7em/
69JJgjIiIiIiIiYlA1Wl3sDIxv7Fob+IPt84FHgc20KYEa2oic0jgKuB39p+EPgulNcOsBGwKb
A1pZI57WMGWSNhvCSlXc8TwLcp1cuXS7oCmGD71pZCnFKNkyF9EREREREEDKpGdeBXgJm2P91BzAX8H
TdfzCwhu29O/
tbDj2i6yQNB86m7vPvBw4CTgfUy+WJIEDgOGMyHrAdpYp8Y/AU2x/Y/
+fJcEcERERERERERERERV0haXPGVH4AH23+r256rxJR0DfA12z9t/
B9t9bDG9IkvRRTEMGBjJ2lvSIISEsc+FzkJ4iyArwG+HRmLaSFNdyHXYA+1GRDi/
2fD2g4gIiIiIiIiIoasy4CLbSgt4yIt9DTCkKIioasy4CLgB9J2lvSIISEsc+FzkJ4iyArwG+HRmLaSFNdyHXYA+1GRDi/
2fD2g4gIiIiIiIiIoasy4CLgB9J2lvSIraflbR8rW5+HDgLXjBwK2LIqwlMgNcBdwI7AmOA7SktbY8DT
q73VRsxDkX1sdy1Jpfnpzz+x0m6Q9J5kj4paW3bO0uo9WCuaIiIiIiI6BpJSwGHU5aejwBwKeB
aYB04GP2b48S9Cj3zRaX1wHfML2uY19qwHvB75ne0IqaLtH0oLA4sDqwKuBDYD1gGtsb91mbHOI2gjI
iIiIiouskjQXWAhajDNS6wPa0dqOKaE+tpj0JeBo4xPa9jX23AXsM7Mkc3SNpPhJttP1AGMI2x/Jq17/
jtJMEdERERERERERLRA0jrAt4ArgUuAYtAgK1tr9ZiaEPWi62YaAwnnnQB83vapqR7BHBERERERE
RER0QJJwygYJ5fcBawD3AncBP7J9YRPAReB4LbAlcAUwBHr9VON+IygrLBa3Pb8STB7eWS3P/
vJcEcERERERERERHRQ5JGArsvr296IY+SfNQesLvDOwGDKMkmzcG/mh7f0kjbD/
TYphzjCSYIyIiIiIiIiIeqAzNE7SnsDHgdOBVwBvoSSZ5wZ+bPvQFsPsPsO5IWAjYC9qRUMF+ZCub/
XhLMERERERERERERPdBIMJ8DXGr7i7i5KOAp4BTgJOBE6zfVwSnDnGnGNZ2ABEREREREREREf3A9sx6zRw
br29EzDO9kTgTgTkqrDCh9giNme0kwR0RERERERERE9LdPkj9IkeSmtMRarg/
1uA56WNIz8IiIiIiIiIioh/UzsXioh/UthePA1+TtEtl/
F74Hzg78DPbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+xHbd+x

```
+i82n0RJHO8HbAlcSWmJ8bSkzwMvp7TAOBOYCKxa/9bJkpYC3gccR6mM/ld/
PyIiIiJitiD7xVbtRURERERERERERES8uLTIiIiIiIiIiIiIiIhZkgRzRERERERERERERMySJJgjIiIi
IiIiIiIiYpYkwRwRERERERERERERsyQJ5oiIiIiIiIiIiIiYJUkwR0RERERERERERMQsSYI5IiIiIiIi
IiIiImbJ/wfmn8cfvivDmAAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<Figure size 1440x504 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "cv_scorings = pd.DataFrame()\n",
    "n_jobs = -1\n",
    "models = [LogisticRegression(random_state = seed, n_jobs = n_jobs),\n",
    "          MultinomialNB(),\n",
    "          BernoulliNB(),\n",
    "          GaussianNB(),\n",
    "          LabelPropagation(n_jobs = n_jobs, alpha = 0, kernel='knn'),\n",
    "          LabelSpreading(n_jobs = n_jobs, kernel = 'knn'),\n",
    "          RandomForestClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "          SGDClassifier(random_state = seed,n_jobs = n_jobs,loss='log'),\n",
    "          DecisionTreeClassifier(random_state = seed),\n",
    "           MLPClassifier(random_state=seed),\n",
    "          #XGBClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "          GradientBoostingClassifier(random_state = seed),\n",
    "          #SVC(random_state = seed,kernel='linear',probability=True),\n",
    "          KNeighborsClassifier(n_jobs = n_jobs),\n",
    "          #NearestCentroid(),\n",
    "          QuadraticDiscriminantAnalysis(),\n",
    "          LinearDiscriminantAnalysis()]\n",
    "\n",
    "cv_df_2 = pd.DataFrame()\n",
    "for score in scoring:\n",
    "    entries = []\n",
    "    for model in models:\n",
    "        # Pipeline to oversample using smote\n",
    "        pipeline = make_pipeline(smote_, model)\n",
    "        \n",
    "        model_name = model.__class__.__name__\n",
    "        '''\n",
    "        if model_name == 'SVC' or model_name == 'XGBClassifier':\n",
    "            # got to transform into a matrix\n",
    "            accuracies = cross_val_score(pipeline,  x.as_matrix(),
y.as_matrix(), scoring= score, cv=CV)\n",
    "            \n",
    "        else:\n",
    "            accuracies = cross_val_score(pipeline, x, y, scoring= score,
cv=CV)\n",
    "        '''\n",
    "        accuracies = cross_val_score(pipeline, x, y, scoring= score, cv=CV)
\n",
    "        for fold_idx, accuracy in enumerate(accuracies):\n",
    "            entries.append((model_name, fold_idx, accuracy))\n",
    "            \n",
    "    cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx',
score])\n",
    "\n",
    "    # Plot different models\n",
    "    plt.figure(figsize = (20,7))\n",
    "    ax = sns.boxplot(x='model_name', y=score,
data=cv_df,boxprops=dict(alpha=.8),linewidth=1)\n",
    "    sns.stripplot(x='model_name', y=score, data=cv_df, \n",
```

```
    "                  size=6, jitter=True, edgecolor=\"black\", linewidth=.5)\
n",
    "    ax.set_xticklabels(ax.get_xticklabels(), rotation=75, fontsize =12) \
n",
    "    ax.spines['top'].set_color('none')\n",
    "    ax.spines['right'].set_color('none')\n",
    "    ax.spines['left'].set_smart_bounds(True)\n",
    "    ax.spines['bottom'].set_smart_bounds(True)\n",
    "    plt.xlabel('Model',  fontweight = \"bold\")\n",
    "    plt.ylabel(score,  fontweight = \"bold\")\n",
    "    #plt.gcf().subplots_adjust(top = 0.15)\n",
    "    #plt.gcf().subplots_adjust(bottom = 0.15)\n",
    "    plt.tight_layout()\n",
    "    plt.savefig(score+'_base_models.png')\n",
    "    plt.show()\n",
    "    \n",
    "    cv_scorings[score] = cv_df.groupby('model_name')[score].agg('mean')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {},
   "outputs": [],
   "source": [
    "cv_scorings.reset_index(inplace=True)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "from sklearn.model_selection import cross_val_predict\n",
    "\n",
    "profits = []\n",
    "profits_norm = []\n",
    "\n",
    "for model in models:\n",
    "    revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "    revenues = []\n",
    "    revenues_norm = []\n",
    "    pipeline = make_pipeline(smote_, model)\n",
    "    \n",
    "    for fold_train, fold_valid in CV.split(x,y):\n",
    "        pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
    "        y_prob = model.predict_proba(x.iloc[fold_valid])[:,1]\n",
    "        t = 0.5\n",
    "        y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "        cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
    "        revenue = cm[1][1] * revenue_answer\n",
    "        expenses = cm[:, 1].sum() * expense_answer\n",
    "        net_revenue = revenue - expenses\n",
    "        r_real = np.sum(y.iloc[fold_valid].values)*8\n",
    "\n",
    "        revenues.append(net_revenue)\n",
    "        revenues_norm.append(net_revenue/r_real)\n",
    "        \n",
    "\n",
    "    profits.append(np.average(revenues))\n",
```

```
    "    profits_norm.append(np.average(revenues_norm))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 19,
   "metadata": {},
   "outputs": [],
   "source": [
    "profit_df = pd.DataFrame({'model_name':[x.__class__.__name__ for x in
models], 'profit':profits, 'profit_norm':profits_norm})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "# add to cv dataframe\n",
    "cv_final_board = cv_scorings.merge(profit_df, how =
'inner',on='model_name')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>model_name</th>\n",
       "      <th>accuracy</th>\n",
       "      <th>precision</th>\n",
       "      <th>recall</th>\n",
       "      <th>f1</th>\n",
       "      <th>profit</th>\n",
       "      <th>profit_norm</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>8</th>\n",
```

```
"      <td>LogisticRegression</td>\n",
"      <td>0.843519</td>\n",
"      <td>0.488400</td>\n",
"      <td>0.845400</td>\n",
"      <td>0.617311</td>\n",
"      <td>210.2</td>\n",
"      <td>0.509285</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>GradientBoostingClassifier</td>\n",
"      <td>0.881632</td>\n",
"      <td>0.597244</td>\n",
"      <td>0.675038</td>\n",
"      <td>0.628470</td>\n",
"      <td>205.8</td>\n",
"      <td>0.498822</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7</th>\n",
"      <td>LinearDiscriminantAnalysis</td>\n",
"      <td>0.833121</td>\n",
"      <td>0.469248</td>\n",
"      <td>0.849246</td>\n",
"      <td>0.603079</td>\n",
"      <td>200.4</td>\n",
"      <td>0.485502</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>11</th>\n",
"      <td>RandomForestClassifier</td>\n",
"      <td>0.884521</td>\n",
"      <td>0.635084</td>\n",
"      <td>0.531448</td>\n",
"      <td>0.576100</td>\n",
"      <td>180.2</td>\n",
"      <td>0.436633</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12</th>\n",
"      <td>SGDClassifier</td>\n",
"      <td>0.784043</td>\n",
"      <td>0.405298</td>\n",
"      <td>0.887934</td>\n",
"      <td>0.554179</td>\n",
"      <td>159.4</td>\n",
"      <td>0.385756</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
"      <td>LabelSpreading</td>\n",
"      <td>0.804832</td>\n",
"      <td>0.418597</td>\n",
"      <td>0.791327</td>\n",
"      <td>0.545785</td>\n",
"      <td>156.0</td>\n",
"      <td>0.378035</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>KNeighborsClassifier</td>\n",
"      <td>0.814658</td>\n",
"      <td>0.430256</td>\n",
"      <td>0.740950</td>\n",
```

```
"      <td>0.542594</td>\n",
"      <td>153.2</td>\n",
"      <td>0.371324</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5</th>\n",
"      <td>LabelPropagation</td>\n",
"      <td>0.805988</td>\n",
"      <td>0.419274</td>\n",
"      <td>0.775792</td>\n",
"      <td>0.542709</td>\n",
"      <td>153.2</td>\n",
"      <td>0.371239</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>GaussianNB</td>\n",
"      <td>0.822726</td>\n",
"      <td>0.439500</td>\n",
"      <td>0.647587</td>\n",
"      <td>0.522018</td>\n",
"      <td>137.6</td>\n",
"      <td>0.333305</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>9</th>\n",
"      <td>MultinomialNB</td>\n",
"      <td>0.829063</td>\n",
"      <td>0.450028</td>\n",
"      <td>0.608974</td>\n",
"      <td>0.514358</td>\n",
"      <td>134.2</td>\n",
"      <td>0.324981</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>BernoulliNB</td>\n",
"      <td>0.814060</td>\n",
"      <td>0.418191</td>\n",
"      <td>0.624434</td>\n",
"      <td>0.498419</td>\n",
"      <td>122.6</td>\n",
"      <td>0.296917</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>DecisionTreeClassifier</td>\n",
"      <td>0.826759</td>\n",
"      <td>0.434388</td>\n",
"      <td>0.534615</td>\n",
"      <td>0.478810</td>\n",
"      <td>112.8</td>\n",
"      <td>0.272766</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>10</th>\n",
"      <td>QuadraticDiscriminantAnalysis</td>\n",
"      <td>0.764205</td>\n",
"      <td>0.394314</td>\n",
"      <td>0.690422</td>\n",
"      <td>0.486680</td>\n",
"      <td>88.0</td>\n",
"      <td>0.211718</td>\n",
"    </tr>\n",
```

```
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "                          model_name  accuracy  precision    recall
f1  \\\n",
       "8              LogisticRegression  0.843519   0.488400  0.845400
0.617311   \n",
       "3      GradientBoostingClassifier  0.881632   0.597244  0.675038
0.628470   \n",
       "7      LinearDiscriminantAnalysis  0.833121   0.469248  0.849246
0.603079   \n",
       "11         RandomForestClassifier  0.884521   0.635084  0.531448
0.576100   \n",
       "12                    SGDClassifier  0.784043   0.405298  0.887934
0.554179   \n",
       "6                   LabelSpreading  0.804832   0.418597  0.791327
0.545785   \n",
       "4             KNeighborsClassifier  0.814658   0.430256  0.740950
0.542594   \n",
       "5                 LabelPropagation  0.805988   0.419274  0.775792
0.542709   \n",
       "2                       GaussianNB  0.822726   0.439500  0.647587
0.522018   \n",
       "9                    MultinomialNB  0.829063   0.450028  0.608974
0.514358   \n",
       "0                      BernoulliNB  0.814060   0.418191  0.624434
0.498419   \n",
       "1          DecisionTreeClassifier  0.826759   0.434388  0.534615
0.478810   \n",
       "10  QuadraticDiscriminantAnalysis  0.764205   0.394314  0.690422
0.486680   \n",
       "\n",
       "    profit  profit_norm  \n",
       "8    210.2     0.509285  \n",
       "3    205.8     0.498822  \n",
       "7    200.4     0.485502  \n",
       "11   180.2     0.436633  \n",
       "12   159.4     0.385756  \n",
       "6    156.0     0.378035  \n",
       "4    153.2     0.371324  \n",
       "5    153.2     0.371239  \n",
       "2    137.6     0.333305  \n",
       "9    134.2     0.324981  \n",
       "0    122.6     0.296917  \n",
       "1    112.8     0.272766  \n",
       "10    88.0     0.211718  "
      ]
     },
     "execution_count": 22,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "cv_final_board.sort_values(by=['profit'],ascending = False,inplace=True)\
n",
    "cv_final_board\n",
    "#print(cv_final_board.sort_values(by=['profit'],ascending = False).index)\
n",
    "#print(cv_final_board.sort_values(by=['f1'],ascending = False).index)"
   ]
  },
```

```
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Select k best models"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "['LogisticRegression', 'RandomForestClassifier',
'GradientBoostingClassifier', 'LinearDiscriminantAnalysis']\n"
     ]
    }
   ],
   "source": [
    "k = 4\n",
    "k_best_models = cv_final_board[:k]['model_name'].values\n",
    "k_best_models = [model for model in models if model.__class__.__name__ in
(k_best_models)]\n",
    "print([model.__class__.__name__ for model in k_best_models])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 89,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "Int64Index([8, 3, 7, 11], dtype='int64')"
      ]
     },
     "execution_count": 89,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "cv_final_board[:4].index"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Ensemble"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Voting\n",
    "from sklearn.ensemble import BaggingClassifier\n",
```

```
    "from mlxtend.classifier import EnsembleVoteClassifier"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
```

```
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
"Fitting clf4: lineardiscriminantanalysis (4/4)\n",
"Fitting 4 classifiers...\n",
"Fitting clf1: logisticregression (1/4)\n",
"Fitting clf2: randomforestclassifier (2/4)\n",
"Fitting clf3: gradientboostingclassifier (3/4)\n",
```

```
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n"
     ]
    }
   ],
   "source": [
    "\n",
    "n_splits = 5\n",
    "CV = StratifiedKFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = ['accuracy','precision','recall','f1','profit', 'profit_norm']\
n",
    "\n",
    "vote_df = pd.DataFrame()\n",
    "\n",
    "\n",
    "for score in scoring:\n",
    "    \n",
    "    v_claf = EnsembleVoteClassifier(clfs=k_best_models, voting='soft', 
verbose=0)\n",
    "    pipeline = make_pipeline(smote_, v_claf)\n",
    "    \n",
    "    if score != 'profit' and score != 'profit_norm':\n",
    "        entries = []\n",
    "        model_name = v_claf.__class__.__name__\n",
    "        accuracies = cross_val_score(pipeline, x, y, scoring= score, 
cv=CV)\n",
    "        for fold_idx, accuracy in enumerate(accuracies):\n",
    "            entries.append((model_name, fold_idx, accuracy))\n",
    "        cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 
```

```
score])\n",
    "        vote_df[score] = cv_df.groupby('model_name')[score].agg('mean')\n",
    "    \n",
    "    else:\n",
    "        profits = []\n",
    "        profits_norm = []\n",
    "        revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "        revenues = []\n",
    "        revenues_norm = []\n",
    "\n",
    "        for fold_train, fold_valid in CV.split(x,y):\n",
    "            pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
    "            y_prob = v_claf.predict_proba(x.iloc[fold_valid])[:,1]\n",
    "            t = 0.5\n",
    "            y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "            cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
    "            revenue = cm[1][1] * revenue_answer\n",
    "            expenses = cm[:, 1].sum() * expense_answer\n",
    "            net_revenue = revenue - expenses\n",
    "            r_real = np.sum(y.iloc[fold_valid].values)*8\n",
    "\n",
    "            revenues.append(net_revenue)\n",
    "            revenues_norm.append(net_revenue/r_real)\n",
    "        profits.append(np.average(revenues))\n",
    "        profits_norm.append(np.average(revenues_norm))\n",
    "        if score =='profit':\n",
    "            vote_df[score] = profits\n",
    "        else:\n",
    "            vote_df[score] = profits_norm"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 27,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>accuracy</th>\n",
       "      <th>precision</th>\n",
       "      <th>recall</th>\n",
       "      <th>f1</th>\n",
       "      <th>profit</th>\n",
       "      <th>profit_norm</th>\n",
```

```
       "    </tr>\n",
       "    <tr>\n",
       "      <th>model_name</th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>EnsembleVoteClassifier</th>\n",
       "      <td>0.875865</td>\n",
       "      <td>0.562949</td>\n",
       "      <td>0.802866</td>\n",
       "      <td>0.658296</td>\n",
       "      <td>232.8</td>\n",
       "      <td>0.564263</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "                         accuracy  precision    recall        f1  profit
\\\n",
       "model_name
\n",
       "EnsembleVoteClassifier  0.875865   0.562949  0.802866  0.658296   232.8
\n",
       "\n",
       "                         profit_norm  \n",
       "model_name                            \n",
       "EnsembleVoteClassifier     0.564263  "
      ]
     },
     "execution_count": 27,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "vote_df"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Grid search\n",
    "---"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 28,
   "metadata": {},
   "outputs": [],
   "source": [
    "def sampling(data, column, seed):\n",
    "    #random.seed(seed)\n",
    "    \n",
```

```
   "    num_of_1=len(data.loc[data[column]==1])\n",
   "    idxs=random.sample(set(data.loc[data[column]==0].index), num_of_1)\n",
   "    new_data_0 = data.loc[data.index.isin(idxs)]\n",
   "    \n",
   "    sample = pd.concat((new_data_0, data.loc[data[column]==1]), axis=0)\n",
   "    \n",
   "    y=sample[column]\n",
   "    x=sample.drop(columns=column)\n",
   "    \n",
   "    return( x , y)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 29,
  "metadata": {},
  "outputs": [],
  "source": [
   "x_t, y_t = sampling(train, 'Response', seed)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 83,
  "metadata": {},
  "outputs": [],
  "source": [
   "grid_dict={\n",
   "    'LogisticRegression':{'C':[100, 10, 1, 0.1, 0.01, 0.001, 0.0001],\n",
   "                          'penalty':['l1','l2']},\n",
   "    'LabelPropagation':{'kernel': ['knn'],\n",
   "                        'n_neighbors':[3,5,10,20,50],\n",
   "                        'max_iter': [1000, 1250, 1500, 1750, 2000]},\n",
   "    'LabelSpreading':{'kernel': ['knn'],\n",
   "                        'max_iter': [1000, 2000],\n",
   "                        'n_neighbors':[3,5,10,20,50],\n",
   "                        'alpha': [1e-4, 1e-3, 1e-2, 1e-1],\n",
   "                        'max_iter': [1000, 1250, 1500, 1750, 2000]},\n",
   "    'RandomForestClassifier':{'bootstrap': [True, False],\n",
   "                       'max_depth': [10, 30, None],\n",
   "                       'max_features': ['auto', 'sqrt'],\n",
   "                       'min_samples_leaf': [1, 2],\n",
   "                       'min_samples_split': [2, 5, 10],\n",
   "                       'n_estimators': [10, 50, 100, 200]},\n",
   "    'SGDClassifier':{'solver': ['svd', 'lsqr'],\n",
   "                     'shrinkage':[None],\n",
   "                     'n_components': [None]},\n",
   "    'GradientBoostingClassifier':{\"loss\":[\"deviance\"],\n",
   "                     \"learning_rate\": [0.05, 0.1, 0.2],\n",
   "                     \"min_samples_split\": np.linspace(0.1, 0.5, 10),\n",
   "                     \"min_samples_leaf\": np.linspace(0.1, 0.5, 10),\n",
   "                     \"max_depth\":[3,5],\n",
   "                     \"max_features\":[\"log2\",\"sqrt\"],\n",
   "                     \"criterion\": [\"friedman_mse\"],\n",
   "                     \"subsample\":[0.5, 1.0],\n",
   "                     \"n_estimators\":[10, 100]},\n",
   "    'KNeighborsClassifier':{'weights': ['uniform', 'distance'],\n",
   "                     'n_neighbors':[3,5,10,20,50],\n",
   "                     'metric': ['euclidean', 'manhattan', 'minkowski']},\n",
   "    'LinearDiscriminantAnalysis':{'solver': ['svd', 'lsqr'],\n",
   "                     'shrinkage':[None],\n",
   "                     'n_components': [None]},\n",
   "    'MLPClassifier':{'hidden_layer_sizes': [(1),(2),(2,2),(3),(3,3),],\n",
   "                        'solver': ['sgd', 'adam','lbfgs'],\n",
```

```
    "                   'alpha': 10.0 ** -np.arange(1, 5),\n",
    "                   'momentum':np.arange(0,0.1, 0.2),\n",
    "                   'max_iter': [200,500],\n",
    "                   'learning_rate': ['constant']}\n",
    "    \n",
    "}"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 71,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Add MLP to best models\n",
    "k_best_models.append(MLPClassifier(random_state=seed))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 79,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "\n",
      "  0%|          | 0/1 [00:00<?, ?it/s]\u001b[A"
     ]
    },
    {
     "ename": "ValueError",
     "evalue": "Parameter values for parameter (min_samples_leaf) need to be a
non-empty sequence.",
     "output_type": "error",
     "traceback": [
      "\
u001b[0;31m-----------------------------------------------------------------------
------\u001b[0m",
      "\u001b[0;31mValueError\u001b[0m                                Traceback
(most recent call last)",
      "\u001b[0;32m<ipython-input-79-735bba7fcc44>\u001b[0m in \
u001b[0;36m<module>\u001b[0;34m()\u001b[0m\n\u001b[1;32m      1\u001b[0m \
u001b[0mgrid_search_dict\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m{\u001b[0m\
u001b[0;34m}\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m      2\
u001b[0m \u001b[0;32mfor\u001b[0m \u001b[0mmodel\u001b[0m \u001b[0;32min\
u001b[0m \u001b[0mtqdm\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m[\u001b[0m\
u001b[0mk_best_models\u001b[0m\u001b[0;34m[\u001b[0m\u001b[0;36m2\u001b[0m\
u001b[0;34m]\u001b[0m\u001b[0;34m]\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m:\
u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 3\u001b[0;31m     \
u001b[0mmodel_gs\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mGridSearchCV\u001b[0m\
u001b[0;34m(\u001b[0m\u001b[0mmodel\u001b[0m\u001b[0;34m,\u001b[0m\
u001b[0mgrid_dict\u001b[0m\u001b[0;34m[\u001b[0m\u001b[0mmodel\u001b[0m\
u001b[0;34m.\u001b[0m\u001b[0m__class__\u001b[0m\u001b[0;34m.\u001b[0m\
u001b[0m__name__\u001b[0m\u001b[0;34m]\u001b[0m\u001b[0;34m,\u001b[0m\
u001b[0mcv\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mCV\u001b[0m\u001b[0;34m,\
u001b[0m\u001b[0mn_jobs\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m-\u001b[0m\
u001b[0;36m1\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mscoring\u001b[0m\
u001b[0;34m=\u001b[0m\u001b[0;34m'f1'\u001b[0m\u001b[0;34m,\u001b[0m\
u001b[0mverbose\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;36m1\u001b[0m\
u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m
4\u001b[0m     \u001b[0mmodel_gs\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mfit\
u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mx_t\u001b[0m\u001b[0;34m,\u001b[0m\
```

```
u001b[0my_t\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\
u001b[1;32m      5\u001b[0m     \u001b[0mgrid_search_dict\u001b[0m\u001b[0;34m[\
u001b[0m\u001b[0mmodel\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m__class__\
u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m__name__\u001b[0m\u001b[0;34m]\u001b[0m\
u001b[0;34m=\u001b[0m\u001b[0mmodel_gs\u001b[0m\u001b[0;34m.\u001b[0m\
u001b[0mbest_estimator_\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",

"\u001b[0;32m/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/
_search.py\u001b[0m in \u001b[0;36m__init__\u001b[0;34m(self, estimator,
param_grid, scoring, fit_params, n_jobs, iid, refit, cv, verbose, pre_dispatch,
error_score, return_train_score)\u001b[0m\n\u001b[1;32m   1185\u001b[0m
return_train_score=return_train_score)\n\u001b[1;32m   1186\u001b[0m         \
u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mparam_grid\u001b[0m \
u001b[0;34m=\u001b[0m \u001b[0mparam_grid\u001b[0m\u001b[0;34m\u001b[0m\
u001b[0m\n\u001b[0;32m-> 1187\u001b[0;31m         \u001b[0m_check_param_grid\
u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mparam_grid\u001b[0m\u001b[0;34m)\
u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m   1188\u001b[0m \
u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m   1189\u001b[0m     \u001b[0;32mdef\
u001b[0m \u001b[0m_run_search\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mself\
u001b[0m\u001b[0;34m,\u001b[0m \u001b[0mevaluate_candidates\u001b[0m\
u001b[0;34m)\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",

"\u001b[0;32m/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/
_search.py\u001b[0m in \u001b[0;36m_check_param_grid\u001b[0;34m(param_grid)\
u001b[0m\n\u001b[1;32m    381\u001b[0m             \u001b[0;32mif\u001b[0m \
u001b[0mlen\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mv\u001b[0m\u001b[0;34m)\
u001b[0m \u001b[0;34m==\u001b[0m \u001b[0;36m0\u001b[0m\u001b[0;34m:\u001b[0m\
u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m    382\u001b[0m
raise ValueError(\"Parameter values for parameter ({0}) need \"\n\u001b[0;32m-->
383\u001b[0;31m                                          \"to be a non-empty
sequence.\".format(name))\n\u001b[0m\u001b[1;32m    384\u001b[0m \u001b[0;34m\
u001b[0m\u001b[0m\n\u001b[1;32m    385\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\
n",
      "\u001b[0;31mValueError\u001b[0m: Parameter values for parameter
(min_samples_leaf) need to be a non-empty sequence."
     ]
    }
   ],
   "source": [
    "grid_search_dict={}\n",
    "for model in tqdm(k_best_models):\n",
    "    model_gs = GridSearchCV(model,grid_dict[model.__class__.__name__],cv =
CV,n_jobs=-1,scoring='f1',verbose=1)\n",
    "    model_gs.fit(x_t,y_t)\n",
    "    grid_search_dict[model.__class__.__name__]=model_gs.best_estimator_"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### voting with models from grid search"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Voting\n",
    "from sklearn.ensemble import BaggingClassifier\n",
    "from mlxtend.classifier import EnsembleVoteClassifier"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 79,
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n",
      "Fitting clf3: gradientboostingclassifier (3/4)\n",
      "Fitting clf4: lineardiscriminantanalysis (4/4)\n",
      "Fitting 4 classifiers...\n",
      "Fitting clf1: logisticregression (1/4)\n",
      "Fitting clf2: randomforestclassifier (2/4)\n"
     ]
    },
    {
     "ename": "KeyboardInterrupt",
     "evalue": "",
     "output_type": "error",
     "traceback": [
      "\
u001b[1;31m---------------------------------------------------------------------------
------\u001b[0m",
      "\u001b[1;31mKeyboardInterrupt\u001b[0m                         Traceback
(most recent call last)",
      "\u001b[1;32m<ipython-input-79-8166be0328b5>\u001b[0m in \
u001b[0;36m<module>\u001b[1;34m\u001b[0m\n\u001b[0;32m     16\u001b[0m         \
u001b[0mentries\u001b[0m \u001b[1;33m=\u001b[0m \u001b[1;33m[\u001b[0m\
u001b[1;33m]\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m     17\u001b[0m         \u001b[0mmodel_name\u001b[0m \u001b[1;33m=\
u001b[0m \u001b[0mv_claf\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m__class__\
u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m__name__\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m---> 18\u001b[1;33m         \
u001b[0maccuracies\u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mcross_val_score\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mpipeline\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0mx\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0my\u001b[0m\u001b[1;33m,\
u001b[0m \u001b[0mscoring\u001b[0m\u001b[1;33m=\u001b[0m \u001b[0mscore\
u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mcv\u001b[0m\u001b[1;33m=\u001b[0m\
u001b[0mCV\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m     19\u001b[0m         \
u001b[1;32mfor\u001b[0m \u001b[0mfold_idx\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0maccuracy\u001b[0m \u001b[1;32min\u001b[0m \u001b[0menumerate\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0maccuracies\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m     20\u001b[0m             \u001b[0mentries\u001b[0m\u001b[1;33m.\
u001b[0m\u001b[0mappend\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mmodel_name\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mfold_idx\u001b[0m\
u001b[1;33m,\u001b[0m \u001b[0maccuracy\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\model_selection\\
_validation.py\u001b[0m in \u001b[0;36mcross_val_score\u001b[1;34m(estimator, X,
y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, error_score)\
u001b[0m\n\u001b[0;32m    400\u001b[0m                                \
u001b[0mfit_params\u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mfit_params\u001b[0m\
u001b[1;33m,\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m    401\u001b[0m                                \
u001b[0mpre_dispatch\u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mpre_dispatch\
```

```
u001b[0m\u001b[1;33m,\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[1;32m--> 402\u001b[1;33m
error_score=error_score)\n\u001b[0m\u001b[0;32m    403\u001b[0m     \
u001b[1;32mreturn\u001b[0m \u001b[0mcv_results\u001b[0m\u001b[1;33m[\u001b[0m\
u001b[1;34m'test_score'\u001b[0m\u001b[1;33m]\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    404\u001b[0m \u001b[1;33m\
u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\model_selection\\
_validation.py\u001b[0m in \u001b[0;36mcross_validate\u001b[1;34m(estimator, X,
y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch,
return_train_score, return_estimator, error_score)\u001b[0m\n\u001b[0;32m
238\u001b[0m             \u001b[0mreturn_times\u001b[0m\u001b[1;33m=\u001b[0m\
u001b[1;32mTrue\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mreturn_estimator\
u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mreturn_estimator\u001b[0m\u001b[1;33m,\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m
239\u001b[0m             error_score=error_score)\n\u001b[1;32m--> 240\
u001b[1;33m         for train, test in cv.split(X, y, groups))\n\u001b[0m\
u001b[0;32m    241\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    242\
u001b[0m     \u001b[0mzipped_scores\u001b[0m \u001b[1;33m=\u001b[0m \
u001b[0mlist\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mzip\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[1;33m*\u001b[0m\u001b[0mscores\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\parallel.py\u001b[0m in \u001b[0;36m__call__\u001b[1;34m(self,
iterable)\u001b[0m\n\u001b[0;32m    918\u001b[0m                 \u001b[0mself\
u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m_iterating\u001b[0m \u001b[1;33m=\
u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\
u001b[0m_original_iterator\u001b[0m \u001b[1;32mis\u001b[0m \u001b[1;32mnot\
u001b[0m \u001b[1;32mNone\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m    919\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[1;32m--> 920\u001b[1;33m             \u001b[1;32mwhile\u001b[0m \
u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mdispatch_one_batch\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0miterator\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0m\u001b[0;32m    921\u001b[0m                 \u001b[1;32mpass\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    922\u001b[0m
\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\parallel.py\u001b[0m in \u001b[0;36mdispatch_one_batch\u001b[1;34m(self,
iterator)\u001b[0m\n\u001b[0;32m    757\u001b[0m                 \
u001b[1;32mreturn\u001b[0m \u001b[1;32mFalse\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    758\u001b[0m             \
u001b[1;32melse\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[1;32m--> 759\u001b[1;33m                 \
u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m_dispatch\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mtasks\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    760\u001b[0m
\u001b[1;32mreturn\u001b[0m \u001b[1;32mTrue\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    761\u001b[0m \u001b[1;33m\
u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\parallel.py\u001b[0m in \u001b[0;36m_dispatch\u001b[1;34m(self, batch)\
u001b[0m\n\u001b[0;32m    714\u001b[0m         \u001b[1;32mwith\u001b[0m \
u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m_lock\u001b[0m\u001b[1;33m:\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m
715\u001b[0m             \u001b[0mjob_idx\u001b[0m \u001b[1;33m=\u001b[0m \
u001b[0mlen\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\u001b[1;33m.\
u001b[0m\u001b[0m_jobs\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m--> 716\u001b[1;33m             \
u001b[0mjob\u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\
u001b[0m\u001b[0m_backend\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mapply_async\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mbatch\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0mcallback\u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mcb\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
```

```
u001b[0m\u001b[0;32m    717\u001b[0m            \u001b[1;31m# A job can
complete so quickly than its callback is\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    718\u001b[0m
\u001b[1;31m# called before we get here, causing self._jobs to\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\_parallel_backends.py\u001b[0m in \u001b[0;36mapply_async\
u001b[1;34m(self, func, callback)\u001b[0m\n\u001b[0;32m    180\u001b[0m     \
u001b[1;32mdef\u001b[0m \u001b[0mapply_async\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mself\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mfunc\u001b[0m\u001b[1;33m,\
u001b[0m \u001b[0mcallback\u001b[0m\u001b[1;33m=\u001b[0m\u001b[1;32mNone\
u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    181\u001b[0m         \
u001b[1;34m\"\"\"Schedule a func to be run\"\"\"\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m--> 182\u001b[1;33m         \
u001b[0mresult\u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mImmediateResult\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mfunc\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m
183\u001b[0m         \u001b[1;32mif\u001b[0m \u001b[0mcallback\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m    184\u001b[0m             \u001b[0mcallback\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[0mresult\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\_parallel_backends.py\u001b[0m in \u001b[0;36m__init__\u001b[1;34m(self,
batch)\u001b[0m\n\u001b[0;32m    547\u001b[0m         \u001b[1;31m# Don't delay
the application, to avoid keeping the input\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    548\u001b[0m
\u001b[1;31m# arguments in memory\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m--> 549\
u001b[1;33m         \u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\
u001b[0mresults\u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mbatch\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m    550\u001b[0m
\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m    551\u001b[0m     \u001b[1;32mdef\u001b[0m \
u001b[0mget\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\u001b[1;33m)\
u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\parallel.py\u001b[0m in \u001b[0;36m__call__\u001b[1;34m(self)\u001b[0m\
n\u001b[0;32m    223\u001b[0m         \u001b[1;32mwith\u001b[0m \
u001b[0mparallel_backend\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_backend\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0mn_jobs\u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_n_jobs\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m    224\u001b[0m             return [func(*args, **kwargs)\n\
u001b[1;32m--> 225\u001b[1;33m                     for func, args, kwargs in
self.items]\n\u001b[0m\u001b[0;32m    226\u001b[0m \u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m    227\u001b[0m     \u001b[1;32mdef\u001b[0m \
u001b[0m__len__\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\externals\\
joblib\\parallel.py\u001b[0m in \u001b[0;36m<listcomp>\u001b[1;34m(.0)\u001b[0m\
n\u001b[0;32m    223\u001b[0m         \u001b[1;32mwith\u001b[0m \
u001b[0mparallel_backend\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_backend\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0mn_jobs\u001b[0m\u001b[1;33m=\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_n_jobs\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m    224\u001b[0m             return [func(*args, **kwargs)\n\
u001b[1;32m--> 225\u001b[1;33m                     for func, args, kwargs in
self.items]\n\u001b[0m\u001b[0;32m    226\u001b[0m \u001b[1;33m\u001b[0m\
```

```
u001b[0m\n\u001b[0;32m    227\u001b[0m     \u001b[1;32mdef\u001b[0m \
u001b[0m__len__\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mself\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\model_selection\\
_validation.py\u001b[0m in \u001b[0;36m_fit_and_score\u001b[1;34m(estimator, X,
y, scorer, train, test, verbose, parameters, fit_params, return_train_score,
return_parameters, return_n_test_samples, return_times, return_estimator,
error_score)\u001b[0m\n\u001b[0;32m    526\u001b[0m             \
u001b[0mestimator\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mfit\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mX_train\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[1;33m**\u001b[0mfit_params\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\n\u001b[0;32m    527\u001b[0m
\u001b[1;32melse\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\n\u001b[1;32m--> 528\u001b[1;33m             \
u001b[0mestimator\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mfit\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mX_train\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0my_train\u001b[0m\u001b[1;33m,\u001b[0m \u001b[1;33m**\u001b[0m\
u001b[0mfit_params\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\n\u001b[0m\u001b[0;32m    529\u001b[0m \
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    530\u001b[0m     \
u001b[1;32mexcept\u001b[0m \u001b[0mException\u001b[0m \u001b[1;32mas\u001b[0m \
u001b[0me\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\imblearn\\pipeline.py\
u001b[0m in \u001b[0;36mfit\u001b[1;34m(self, X, y, **fit_params)\u001b[0m\n\
u001b[0;32m    237\u001b[0m         \u001b[0mXt\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0myt\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mfit_params\u001b[0m \
u001b[1;33m=\u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m_fit\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mX\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[0my\u001b[0m\u001b[1;33m,\u001b[0m \u001b[1;33m**\u001b[0m\
u001b[0mfit_params\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\n\u001b[0;32m    238\u001b[0m         \
u001b[1;32mif\u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\
u001b[0m_final_estimator\u001b[0m \u001b[1;32mis\u001b[0m \u001b[1;32mnot\
u001b[0m \u001b[1;32mNone\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\n\u001b[1;32m--> 239\u001b[1;33m             \
u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0m_final_estimator\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0mfit\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mXt\
u001b[0m\u001b[1;33m,\u001b[0m \u001b[0myt\u001b[0m\u001b[1;33m,\u001b[0m \
u001b[1;33m**\u001b[0mfit_params\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\n\u001b[0m\u001b[0;32m
240\u001b[0m         \u001b[1;32mreturn\u001b[0m \u001b[0mself\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    241\u001b[0m
\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\mlxtend\\classifier\\
ensemble_vote.py\u001b[0m in \u001b[0;36mfit\u001b[1;34m(self, X, y,
sample_weight)\u001b[0m\n\u001b[0;32m    172\u001b[0m \u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m    173\u001b[0m                 \u001b[1;32mif\u001b[0m \
u001b[0msample_weight\u001b[0m \u001b[1;32mis\u001b[0m \u001b[1;32mNone\
u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[1;32m--> 174\u001b[1;33m                     \u001b[0mclf\
u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mfit\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mX\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\
u001b[0m\u001b[0mle_\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mtransform\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0my\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m)\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\n\u001b[0m\
u001b[0;32m    175\u001b[0m                 \u001b[1;32melse\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\n\
u001b[0;32m    176\u001b[0m                     clf.fit(X,
self.le_.transform(y), \n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\ensemble\\
forest.py\u001b[0m in \u001b[0;36mfit\u001b[1;34m(self, X, y, sample_weight)\
u001b[0m\n\u001b[0;32m    317\u001b[0m             \u001b[1;32mfor\u001b[0m \
```

```
u001b[0mi\u001b[0m \u001b[1;32min\u001b[0m \u001b[0mrange\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[0mn_more_estimators\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m:\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m
318\u001b[0m                tree = self._make_estimator(append=False,\n\
u001b[1;32m--> 319\u001b[1;33m
random_state=random_state)\n\u001b[0m\u001b[0;32m    320\u001b[0m
\u001b[0mtrees\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mappend\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mtree\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    321\u001b[0m \
u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\ensemble\\base.py\
u001b[0m in \u001b[0;36m_make_estimator\u001b[1;34m(self, append, random_state)\
u001b[0m\n\u001b[0;32m    129\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0;32m    130\u001b[0m         \u001b[1;32mif\u001b[0m \
u001b[0mrandom_state\u001b[0m \u001b[1;32mis\u001b[0m \u001b[1;32mnot\u001b[0m \
u001b[1;32mNone\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[1;32m--> 131\u001b[1;33m             \
u001b[0m_set_random_states\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0mestimator\
u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mrandom_state\u001b[0m\u001b[1;33m)\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\
u001b[0;32m    132\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    133\
u001b[0m         \u001b[1;32mif\u001b[0m \u001b[0mappend\u001b[0m\u001b[1;33m:\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\ensemble\\base.py\
u001b[0m in \u001b[0;36m_set_random_states\u001b[1;34m(estimator, random_state)\
u001b[0m\n\u001b[0;32m     51\u001b[0m     \u001b[0mrandom_state\u001b[0m \
u001b[1;33m=\u001b[0m \u001b[0mcheck_random_state\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[0mrandom_state\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m     52\u001b[0m     \
u001b[0mto_set\u001b[0m \u001b[1;33m=\u001b[0m \u001b[1;33m{\u001b[0m\
u001b[1;33m}\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[1;32m---> 53\u001b[1;33m     \u001b[1;32mfor\u001b[0m \u001b[0mkey\
u001b[0m \u001b[1;32min\u001b[0m \u001b[0msorted\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mestimator\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mget_params\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mdeep\u001b[0m\u001b[1;33m=\u001b[0m\
u001b[1;32mTrue\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0m\u001b[0;32m     54\u001b[0m         \u001b[1;32mif\u001b[0m \
u001b[0mkey\u001b[0m \u001b[1;33m==\u001b[0m \u001b[1;34m'random_state'\u001b[0m
\u001b[1;32mor\u001b[0m \u001b[0mkey\u001b[0m\u001b[1;33m.\u001b[0m\
u001b[0mendswith\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;34m'__random_state'\
u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m     55\u001b[0m             \
u001b[0mto_set\u001b[0m\u001b[1;33m[\u001b[0m\u001b[0mkey\u001b[0m\u001b[1;33m]\
u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mrandom_state\u001b[0m\u001b[1;33m.\
u001b[0mrandint\u001b[0m\u001b[1;33m(\u001b[0mMAX_RAND_SEED\
u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\base.py\u001b[0m
in \u001b[0;36mget_params\u001b[1;34m(self, deep)\u001b[0m\n\u001b[0;32m    181\
u001b[0m         \"\"\"\n\u001b[0;32m    182\u001b[0m         \u001b[0mout\
u001b[0m \u001b[1;33m=\u001b[0m \u001b[0mdict\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[1;32m--> 183\u001b[1;33m         \u001b[1;32mfor\u001b[0m \u001b[0mkey\
u001b[0m \u001b[1;32min\u001b[0m \u001b[0mself\u001b[0m\u001b[1;33m.\u001b[0m\
u001b[0m_get_param_names\u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\
u001b[0m\u001b[0;32m    184\u001b[0m             \u001b[0mvalue\u001b[0m \
u001b[1;33m=\u001b[0m \u001b[0mgetattr\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mself\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mkey\u001b[0m\u001b[1;33m,\
u001b[0m \u001b[1;32mNone\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m    185\u001b[0m             \
u001b[1;32mif\u001b[0m \u001b[0mdeep\u001b[0m \u001b[1;32mand\u001b[0m \
u001b[0mhasattr\u001b[0m\u001b[1;33m(\u001b[0mvalue\u001b[0m\
```

```
u001b[1;33m,\u001b[0m \u001b[1;34m'get_params'\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\site-packages\\sklearn\\base.py\u001b[0m
in \u001b[0;36m_get_param_names\u001b[1;34m(cls)\u001b[0m\n\u001b[0;32m    151\
u001b[0m         \u001b[1;31m# introspect the constructor arguments to find the
model parameters\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m    152\u001b[0m         \u001b[1;31m# to
represent\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[1;32m--> 153\u001b[1;33m         \
u001b[0minit_signature\u001b[0m \u001b[1;33m=\u001b[0m \u001b[0msignature\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[0minit\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m
154\u001b[0m         \u001b[1;31m# Consider the constructor parameters excluding
'self'\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m    155\u001b[0m         parameters = [p for p in
init_signature.parameters.values()\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36msignature\u001b[1;34m(obj, follow_wrapped)\u001b[0m\n\u001b[0;32m
3081\u001b[0m \u001b[1;32mdef\u001b[0m \u001b[0msignature\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[0mobj\u001b[0m\u001b[1;33m,\u001b[0m \u001b[1;33m*\u001b[0m\
u001b[1;33m,\u001b[0m \u001b[0mfollow_wrapped\u001b[0m\u001b[1;33m=\u001b[0m\
u001b[1;32mTrue\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   3082\u001b[0m
\u001b[1;34m\"\"\"Get a signature object for the passed callable.\"\"\"\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m->
3083\u001b[1;33m     \u001b[1;32mreturn\u001b[0m \u001b[0mSignature\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0mfrom_callable\u001b[0m\u001b[1;33m(\u001b[0m\
u001b[0mobj\u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mfollow_wrapped\u001b[0m\
u001b[1;33m=\u001b[0m\u001b[0mfollow_wrapped\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m
3084\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   3085\u001b[0m \
u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36mfrom_callable\u001b[1;34m(cls, obj, follow_wrapped)\u001b[0m\n\
u001b[0;32m   2831\u001b[0m         \u001b[1;34m\"\"\"Constructs Signature for
the given callable object.\"\"\"\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m   2832\u001b[0m         return
_signature_from_callable(obj, sigcls=cls,\n\u001b[1;32m-> 2833\u001b[1;33m
follow_wrapper_chains=follow_wrapped)\n\u001b[0m\u001b[0;32m   2834\u001b[0m \
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   2835\u001b[0m     \u001b[1;33m@\
u001b[0m\u001b[0mproperty\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36m_signature_from_callable\u001b[1;34m(obj, follow_wrapper_chains,
skip_bound_arg, sigcls)\u001b[0m\n\u001b[0;32m   2282\u001b[0m         \
u001b[1;31m# If it's a pure Python function, or an object that is duck type\
u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\
u001b[0m\n\u001b[0;32m   2283\u001b[0m         \u001b[1;31m# of a Python
function (Cythonfunctions, for instance), then:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m-> 2284\
u001b[1;33m         \u001b[1;32mreturn\u001b[0m \
u001b[0m_signature_from_function\u001b[0m\u001b[1;33m(\u001b[0m\u001b[0msigcls\
u001b[0m\u001b[1;33m,\u001b[0m \u001b[0mobj\u001b[0m\u001b[1;33m)\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m
2285\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   2286\u001b[0m     \
u001b[1;32mif\u001b[0m \u001b[0m_signature_is_builtin\u001b[0m\u001b[1;33m(\
u001b[0m\u001b[0mobj\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m:\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36m_signature_from_function\u001b[1;34m(cls, func)\u001b[0m\n\
u001b[0;32m   2193\u001b[0m     return cls(parameters,\n\u001b[0;32m   2194\
u001b[0m                \u001b[0mreturn_annotation\u001b[0m\u001b[1;33m=\
u001b[0m\u001b[0mannotations\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mget\
u001b[0m\u001b[1;33m(\u001b[0m\u001b[1;34m'return'\u001b[0m\u001b[1;33m,\
```

```
u001b[0m \u001b[0m_empty\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m,\u001b[0m\
u001b[1;33m\u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n\u001b[1;32m-> 2195\
u001b[1;33m                __validate_parameters__=is_duck_function)\n\u001b[0m\
u001b[0;32m   2196\u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   2197\
u001b[0m \u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36m__init__\u001b[1;34m(self, parameters, return_annotation,
__validate_parameters__)\u001b[0m\n\u001b[0;32m   2798\u001b[0m            \
u001b[1;32melse\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m   2799\u001b[0m                params =
OrderedDict(((param.name, param)\n\u001b[1;32m-> 2800\u001b[1;33m
for param in parameters))\n\u001b[0m\u001b[0;32m   2801\u001b[0m \u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m   2802\u001b[0m         \u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_parameters\u001b[0m \u001b[1;33m=\u001b[0m \
u001b[0mtypes\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mMappingProxyType\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mparams\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;32m~\\Anaconda3\\lib\\inspect.py\u001b[0m in \
u001b[0;36m<genexpr>\u001b[1;34m(.0)\u001b[0m\n\u001b[0;32m   2798\u001b[0m
\u001b[1;32melse\u001b[0m\u001b[1;33m:\u001b[0m\u001b[1;33m\u001b[0m\
u001b[1;33m\u001b[0m\u001b[0m\n\u001b[0;32m   2799\u001b[0m
params = OrderedDict(((param.name, param)\n\u001b[1;32m-> 2800\u001b[1;33m
for param in parameters))\n\u001b[0m\u001b[0;32m   2801\u001b[0m \u001b[1;33m\
u001b[0m\u001b[0m\n\u001b[0;32m   2802\u001b[0m         \u001b[0mself\u001b[0m\
u001b[1;33m.\u001b[0m\u001b[0m_parameters\u001b[0m \u001b[1;33m=\u001b[0m \
u001b[0mtypes\u001b[0m\u001b[1;33m.\u001b[0m\u001b[0mMappingProxyType\u001b[0m\
u001b[1;33m(\u001b[0m\u001b[0mparams\u001b[0m\u001b[1;33m)\u001b[0m\u001b[1;33m\
u001b[0m\u001b[1;33m\u001b[0m\u001b[0m\n",
      "\u001b[1;31mKeyboardInterrupt\u001b[0m: "
     ]
    }
   ],
   "source": [
    "\n",
    "n_splits = 5\n",
    "CV = StratifiedKFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = ['accuracy','precision','recall','f1','profit', 'profit_norm']\
n",
    "\n",
    "vote_df_grid = pd.DataFrame()\n",
    "\n",
    "\n",
    "for score in scoring:\n",
    "    \n",
    "    v_claf = EnsembleVoteClassifier(clfs=list(grid_search_dict.values()),
voting='soft', verbose=1)\n",
    "    pipeline = make_pipeline(smote_, v_claf)\n",
    "    \n",
    "    if score != 'profit' and score != 'profit_norm':\n",
    "        entries = []\n",
    "        model_name = v_claf.__class__.__name__\n",
    "        accuracies = cross_val_score(pipeline, x, y, scoring= score,
cv=CV)\n",
    "        for fold_idx, accuracy in enumerate(accuracies):\n",
    "            entries.append((model_name, fold_idx, accuracy))\n",
    "        cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx',
score])\n",
    "        vote_df_grid[score] = cv_df.groupby('model_name')
[score].agg('mean')\n",
    "    \n",
    "    else:\n",
    "        profits = []\n",
    "        profits_norm = []\n",
```

```
   "         revenue_answer, expense_answer = 11, 3\n",
   "\n",
   "         revenues = []\n",
   "         revenues_norm = []\n",
   "\n",
   "         for fold_train, fold_valid in CV.split(x,y):\n",
   "             pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
   "             y_prob = v_claf.predict_proba(x.iloc[fold_valid])[:,1]\n",
   "             t = 0.5\n",
   "             y_pred = [0 if v < t else 1 for v in y_prob]\n",
   "             cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
   "             revenue = cm[1][1] * revenue_answer\n",
   "             expenses = cm[:, 1].sum() * expense_answer\n",
   "             net_revenue = revenue - expenses\n",
   "             r_real = np.sum(y.iloc[fold_valid].values)*8\n",
   "\n",
   "             revenues.append(net_revenue)\n",
   "             revenues_norm.append(net_revenue/r_real)\n",
   "         profits.append(np.average(revenues))\n",
   "         profits_norm.append(np.average(revenues_norm))\n",
   "         if score =='profit':\n",
   "             vote_df[score] = profits\n",
   "         else:\n",
   "             vote_df[score] = profits_norm"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 78,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "    .dataframe tbody tr th:only-of-type {\n",
      "        vertical-align: middle;\n",
      "    }\n",
      "\n",
      "    .dataframe tbody tr th {\n",
      "        vertical-align: top;\n",
      "    }\n",
      "\n",
      "    .dataframe thead th {\n",
      "        text-align: right;\n",
      "    }\n",
      "</style>\n",
      "<table border=\"1\" class=\"dataframe\">\n",
      "  <thead>\n",
      "    <tr style=\"text-align: right;\">\n",
      "      <th></th>\n",
      "      <th>accuracy</th>\n",
      "      <th>precision</th>\n",
      "      <th>recall</th>\n",
      "      <th>f1</th>\n",
      "      <th>profit</th>\n",
      "      <th>profit_norm</th>\n",
      "    </tr>\n",
      "    <tr>\n",
      "      <th>model_name</th>\n",
      "      <th></th>\n",
      "      <th></th>\n",
      "      <th></th>\n",
      "      <th></th>\n",
      "      <th></th>\n",
      "      <th></th>\n",
```

```
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>EnsembleVoteClassifier</th>\n",
       "      <td>0.879322</td>\n",
       "      <td>0.571755</td>\n",
       "      <td>0.802866</td>\n",
       "      <td>0.665068</td>\n",
       "      <td>236.4</td>\n",
       "      <td>0.57286</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "                          accuracy  precision    recall        f1  profit
\\\n",
       "model_name
\n",
       "EnsembleVoteClassifier  0.879322   0.571755  0.802866  0.665068   236.4
\n",
       "\n",
       "                          profit_norm  \n",
       "model_name                             \n",
       "EnsembleVoteClassifier      0.57286  "
      ]
     },
     "execution_count": 78,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "vote_df_grid"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# MLP\n",
    "---"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 57,
   "metadata": {},
   "outputs": [],
   "source": [
    "mlp_param_grid = {'hidden_layer_sizes': [(1),(2),(2,2),(3),(3,3),],\n",
    "                  'solver': ['sgd', 'adam','lbfgs'],\n",
    "                  'alpha': 10.0 ** -np.arange(1, 5),\n",
    "                  'momentum':np.arange(0,0.1, 0.2),\n",
    "                  'max_iter': [200,500],\n",
    "                  'learning_rate': ['constant']}\n",
    "    \n",
    "model = MLPClassifier(random_state=seed)\n"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 58,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Fitting 5 folds for each of 1280 candidates, totalling 6400 fits\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent
workers.\n",
      "[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    5.5s\n",
      "[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   11.5s\n",
      "[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:   25.3s\n",
      "[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed:   50.1s\n",
      "[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed:  1.2min\n",
      "[Parallel(n_jobs=-1)]: Done 1792 tasks      | elapsed:  1.7min\n",
      "[Parallel(n_jobs=-1)]: Done 2442 tasks      | elapsed:  2.6min\n",
      "[Parallel(n_jobs=-1)]: Done 3192 tasks      | elapsed:  3.8min\n",
      "[Parallel(n_jobs=-1)]: Done 4042 tasks      | elapsed:  5.2min\n",
      "[Parallel(n_jobs=-1)]: Done 5182 tasks      | elapsed:  6.9min\n",
      "[Parallel(n_jobs=-1)]: Done 6400 out of 6400 | elapsed:  8.2min finished\
n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=0,
shuffle=False),\n",
       "       error_score='raise-deprecating',\n",
       "       estimator=MLPClassifier(activation='relu', alpha=0.0001,
batch_size='auto', beta_1=0.9,\n",
       "       beta_2=0.999, early_stopping=False, epsilon=1e-08,\n",
       "       hidden_layer_sizes=(100,), learning_rate='constant',\n",
       "       learning_rate_init=0.001, max_iter=200, momentum=0.9,\n",
       "       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,\n",
       "       random_state=0, shuffle=True, solver='adam', tol=0.0001,\n",
       "       validation_fraction=0.1, verbose=False, warm_start=False),\n",
       "       fit_params=None, iid='warn', n_jobs=-1,\n",
       "       param_grid={'hidden_layer_sizes': [1, 2, (2, 2), 3, (3, 3)],
'solver': ['lbfgs'], 'alpha': array([0.1   , 0.01  , 0.001 , 0.0001]),
'momentum': array([0.1   , 0.01  , 0.001 , 0.0001]), 'nesterovs_momentum':
[True, False], 'max_iter': [500, 600], 'learning_rate': ['constant'],
'activation': ['identity', 'logistic', 'tanh', 'relu']},\n",
       "       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',\
n",
       "       scoring='f1', verbose=1)"
      ]
     },
     "execution_count": 58,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
```

```
    "ml_grid = GridSearchCV(model,mlp_param_grid, cv = CV, n_jobs=-
1,scoring='f1',verbose=1)\n",
    "ml_grid.fit(x_t, y_t)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 60,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'activation': 'tanh',\n",
       " 'alpha': 0.0001,\n",
       " 'hidden_layer_sizes': 1,\n",
       " 'learning_rate': 'constant',\n",
       " 'max_iter': 500,\n",
       " 'momentum': 0.1,\n",
       " 'nesterovs_momentum': True,\n",
       " 'solver': 'lbfgs'}"
      ]
     },
     "execution_count": 60,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "ml_grid.best_params_"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 62,
   "metadata": {},
   "outputs": [],
   "source": [
    "#pipeline = make_pipeline(smote_, ml_grid.best_estimator_)\n",
    "#accuracies = cross_val_score(pipeline, x, y, scoring= 'score', cv=CV)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 63,
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "n_splits = 5\n",
    "CV = StratifiedKFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = ['accuracy','precision','recall','f1','profit', 'profit_norm']\
n",
    "\n",
    "#vote_df_grid = pd.DataFrame()\n",
    "\n",
    "mlp_df = pd.DataFrame()\n",
    "\n",
    "for score in scoring:\n",
    "    model = ml_grid.best_estimator_\n",
    "    #v_claf = EnsembleVoteClassifier(clfs=list(grid_search_dict.values()),
voting='soft', verbose=1)\n",
    "    pipeline = make_pipeline(smote_, model)\n",
```

```
    "    \n",
    "    if score != 'profit' and score != 'profit_norm':\n",
    "        entries = []\n",
    "        model_name = v_claf.__class__.__name__\n",
    "        accuracies = cross_val_score(pipeline, x, y, scoring= score,
cv=CV)\n",
    "        for fold_idx, accuracy in enumerate(accuracies):\n",
    "            entries.append((model_name, fold_idx, accuracy))\n",
    "        cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx',
score])\n",
    "        mlp_df[score] = cv_df.groupby('model_name')[score].agg('mean')\n",
    "    \n",
    "    else:\n",
    "        profits = []\n",
    "        profits_norm = []\n",
    "        revenue_answer, expense_answer = 11, 3\n",
    "\n",
    "        revenues = []\n",
    "        revenues_norm = []\n",
    "\n",
    "        for fold_train, fold_valid in CV.split(x,y):\n",
    "            pipeline.fit(x.iloc[fold_train],y.iloc[fold_train])\n",
    "            y_prob = model.predict_proba(x.iloc[fold_valid])[:,1]\n",
    "            t = 0.5\n",
    "            y_pred = [0 if v < t else 1 for v in y_prob]\n",
    "            cm = confusion_matrix(y.iloc[fold_valid], y_pred)\n",
    "            revenue = cm[1][1] * revenue_answer\n",
    "            expenses = cm[:, 1].sum() * expense_answer\n",
    "            net_revenue = revenue - expenses\n",
    "            r_real = np.sum(y.iloc[fold_valid].values)*8\n",
    "\n",
    "            revenues.append(net_revenue)\n",
    "            revenues_norm.append(net_revenue/r_real)\n",
    "        profits.append(np.average(revenues))\n",
    "        profits_norm.append(np.average(revenues_norm))\n",
    "        if score =='profit':\n",
    "            mlp_df[score] = profits\n",
    "        else:\n",
    "            mlp_df[score] = profits_norm"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 64,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
```

```
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>accuracy</th>\n",
       "      <th>precision</th>\n",
       "      <th>recall</th>\n",
       "      <th>f1</th>\n",
       "      <th>profit</th>\n",
       "      <th>profit_norm</th>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>model_name</th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "      <th></th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>EnsembleVoteClassifier</th>\n",
       "      <td>0.818685</td>\n",
       "      <td>0.446353</td>\n",
       "      <td>0.8454</td>\n",
       "      <td>0.582579</td>\n",
       "      <td>184.4</td>\n",
       "      <td>0.446785</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "                         accuracy  precision  recall        f1  \
profit  \\\n",
       "model_name                                                      \
                                                                    \n",
       "EnsembleVoteClassifier  0.818685   0.446353  0.8454  0.582579  \
184.4   \n",
       "\n",
       "                         profit_norm  \n",
       "model_name                            \n",
       "EnsembleVoteClassifier     0.446785  "
      ]
     },
     "execution_count": 64,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "mlp_df"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Specific Model (trial)"
   ]
  },
  {
```

```
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "#model = bagging_clf =
BaggingClassifier(GradientBoostingClassifier(random_state = seed))#,
random_state=seed, n_jobs = -
1)#SVC(probability=True)#RandomForestClassifier(random_state = seed,n_jobs =
n_jobs)\n",
    "#v_claf = EnsembleVoteClassifier(clfs=k_best_models,
voting='soft',verbose=1)\n",
    "\n",
    "pipeline = make_pipeline(smote_, clf_gscv)\n",
    "pipeline.fit(x_train,y_train)\n",
    "predictions = pipeline.predict(x_test)\n",
    "y_scores = clf_gscv.predict_proba(x_test)[:, 1]\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "y_scores = pipeline.predict_proba(x_test)[:, 1]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "predict_metrics(y_test,predictions)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,
y_scores)\n",
    "roc_auc = auc(false_positive_rate, true_positive_rate)\n",
    "roc_auc"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def plot_roc_auc(roc_auc):\n",
    "    plt.figure(figsize=(5,5))\n",
    "    plt.title('AUROC Curve')\n",
    "    plt.plot(false_positive_rate,true_positive_rate, color='red',label =
'AUC = %0.2f' % roc_auc)\n",
    "    plt.legend(loc = 'lower right')\n",
    "    plt.plot([0, 1], [0, 1],linestyle='--')\n",
    "    plt.axis('tight')\n",
    "    plt.ylabel('True Positive Rate')\n",
```

```
    "    plt.xlabel('False Positive Rate')\n",
    "    plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "plot_roc_auc(roc_auc)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.metrics import precision_recall_curve"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "precision, recall, thresholds = precision_recall_curve(y_test, y_scores)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.metrics import average_precision_score\n",
    "average_precision = average_precision_score(y_test, y_scores)\n",
    "average_precision"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": false
   },
   "outputs": [],
   "source": [
    "from sklearn.utils.fixes import signature\n",
    "\n",
    "step_kwargs = ({'step': 'post'}\n",
    "               if 'step' in signature(plt.fill_between).parameters\n",
    "               else {})\n",
    "plt.figure(figsize=(10,7))\n",
    "plt.step(recall, precision, color='b', alpha=0.2,\n",
    "         where='post')\n",
    "plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)\
n",
    "\n",
    "plt.xlabel('Recall')\n",
```

```
    "plt.ylabel('Precision')\n",
    "plt.ylim([0.0, 1.05])\n",
    "plt.xlim([0.0, 1.0])\n",
    "plt.title('Precision-Recall curve: AP={0:0.2f}'.format(\n",
    "          average_precision))\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def adjusted_classes(y_scores, t):\n",
    "    \"\"\"\n",
    "    This function adjusts class predictions based on the prediction threshold (t).\n",
    "    Will only work for binary classification problems.\n",
    "    \"\"\"\n",
    "    return [1 if y >= t else 0 for y in y_scores]\n",
    "\n",
    "def precision_recall_threshold(p, r, thresholds,t=0.5):\n",
    "    \"\"\"\n",
    "    plots the precision recall curve and shows the current value for each\n",
    "    by identifying the classifier's threshold (t).\n",
    "    \"\"\"\n",
    "    \n",
    "    # generate new class predictions based on the adjusted_classes\n",
    "    # function above and view the resulting confusion matrix.\n",
    "    y_pred_adj = adjusted_classes(y_scores, t)\n",
    "    print(pd.DataFrame(confusion_matrix(y_test, y_pred_adj),\n",
    "                       columns=['pred_neg', 'pred_pos'], \n",
    "                       index=['neg', 'pos']))\n",
    "    \n",
    "    # plot the curve\n",
    "    plt.figure(figsize=(10,7))\n",
    "    plt.title(\"Precision and Recall curve ^ = current threshold\")\n",
    "    plt.step(r, p, color='b', alpha=0.2,\n",
    "             where='post')\n",
    "    plt.fill_between(r, p, step='post', alpha=0.2,\n",
    "                     color='b')\n",
    "    plt.ylim([0, 1]);\n",
    "    plt.xlim([0, 1]);\n",
    "    plt.xlabel('Recall');\n",
    "    plt.ylabel('Precision');\n",
    "    \n",
    "    # plot the current threshold on the line\n",
    "    close_default_clf = np.argmin(np.abs(thresholds - t))\n",
    "    plt.plot(r[close_default_clf], p[close_default_clf], '^', c='k',\n",
    "             markersize=15)\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "precision, recall, thresholds = precision_recall_curve(y_test, y_scores)"
   ]
  },
```

```
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "precision_recall_threshold(precision, recall, thresholds,  t = 0.5)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "#get_profit(model, x_test)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "---"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": false
   },
   "outputs": [],
   "source": [
    "fig = plt.figure(figsize=(15,8))\n",
    "ax1 = fig.add_subplot(1,2,1)\n",
    "ax1.set_xlim([-0.05,1.05])\n",
    "ax1.set_ylim([-0.05,1.05])\n",
    "ax1.set_xlabel('Recall')\n",
    "ax1.set_ylabel('Precision')\n",
    "ax1.set_title('PR Curve')\n",
    "\n",
    "ax2 = fig.add_subplot(1,2,2)\n",
    "ax2.set_xlim([-0.05,1.05])\n",
    "ax2.set_ylim([-0.05,1.05])\n",
    "ax2.set_xlabel('False Positive Rate')\n",
    "ax2.set_ylabel('True Positive Rate')\n",
    "ax2.set_title('ROC Curve')\n",
    "\n",
    "for w,k in zip([1,5,10,20,50,100,10000],'bgrcmykw'):\n",
    "    lr_model = LogisticRegression(class_weight={0:1,1:w})\n",
    "    \n",
    "    lr_model.fit(x_train,y_train)\n",
    "    pred_prob = lr_model.predict_proba(x_test)[:,1]\n",
    "\n",
    "    p,r,_ = precision_recall_curve(y_test,pred_prob)\n",
    "    tpr,fpr,_ = roc_curve(y_test,pred_prob)\n",
    "    print('weight:',w, '\\tPR Value:',average_precision_score(y_test,
pred_prob))\n",
    "    ax1.plot(r,p,c=k,label=w)\n",
    "    ax2.plot(tpr,fpr,c=k,label=w)\n",
    "ax1.legend(loc='lower left')    \n",
    "ax2.legend(loc='lower left')\n",
    "\n",
    "plt.show()"
```

```
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Neural Networks\n",
  "---"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### 1. MLP w gird search"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "from sklearn.model_selection import GridSearchCV\n",
  "from sklearn.neural_network import MLPClassifier\n",
  "mlp = MLPClassifier(max_iter=100,random_state=seed)\n",
  "\n",
  "parameter_space = {\n",
  "    'hidden_layer_sizes': [(3), (5,5)],\n",
  "    'activation': ['tanh', 'relu'],\n",
  "    'solver': ['sgd', 'adam', 'lbfgs'],\n",
  "    'momentum':np.arange(0, 1.2, 0.2),\n",
  "    'nesterovs_momentum': [True,False],\n",
  "    'alpha': 10.0 ** -np.arange(1, 5),\n",
  "    'learning_rate': ['constant','adaptive'],\n",
  "}"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": []
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=2, verbose=1)\n",
  "clf.fit(x_train, y_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "scrolled": true
 },
 "outputs": [],
 "source": [
  "# Best paramete set\n",
```

```
    "print('Best parameters found:\\n', clf.best_params_)\n",
    "\n",
    "# All results\n",
    "means = clf.cv_results_['mean_test_score']\n",
    "stds = clf.cv_results_['std_test_score']\n",
    "for mean, std, params in zip(means, stds, clf.cv_results_['params']):\n",
    "    print(\"%0.3f (+/-%0.03f) for %r\" % (mean, std * 2, params))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "predictions = clf.predict(x_test)\n",
    "predict_metrics(y_test,predictions)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "y_scores = clf.predict_proba(x_test)[:, 1]\n",
    "false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,
y_scores)\n",
    "roc_auc = auc(false_positive_rate, true_positive_rate)\n",
    "roc_auc\n",
    "\n",
    "plot_roc_auc(roc_auc)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "average_precision = average_precision_score(y_test, y_scores)\n",
    "\n",
    "step_kwargs = ({'step': 'post'}\n",
    "               if 'step' in signature(plt.fill_between).parameters\n",
    "               else {})\n",
    "plt.figure(figsize=(10,7))\n",
    "plt.step(recall, precision, color='b', alpha=0.2,\n",
    "         where='post')\n",
    "plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)\
n",
    "\n",
    "plt.xlabel('Recall')\n",
    "plt.ylabel('Precision')\n",
    "plt.ylim([0.0, 1.05])\n",
    "plt.xlim([0.0, 1.0])\n",
    "plt.title('Precision-Recall curve: AP={0:0.2f}'.format(\n",
    "          average_precision))\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
```

```
   "outputs": [],
   "source": [
    "get_profit(clf, x_test)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Roc Curve"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# the classifiers\n",
    "architectures = {\"3\": (3), \"5\": (5), \"3x3\": (3,3), \"6x6\": (6,6)}\
n",
    "clfs_ = {}\n",
    "\n",
    "for key, value in tqdm(architectures.items()):\n",
    "    mlp = MLPClassifier(learning_rate_init=0.001, hidden_layer_sizes=value,
\n",
    "                        max_iter=500, random_state=seed)\n",
    "    mlp.fit(x_train, y_train)\n",
    "    clfs_[\"mlp_\"+key] = mlp\n",
    "\n",
    "print(\"Candidate models: \", clfs_.keys())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.metrics import roc_curve, roc_auc_score, auc\n",
    "\n",
    "c=1\n",
    "plt.figure(figsize=(5,5))\n",
    "plt.plot([0, 1], [0, 1], 'k--')\n",
    "thresholds = []\n",
    "for key, value in clfs_.items():\n",
    "    y_pred=value.predict_proba(x_test)[:, c]\n",
    "    fpr, tpr, t = roc_curve(y_test, y_pred)\n",
    "    thresholds.append(t)\n",
    "    auroc = roc_auc_score(y_test, y_pred, average=\"weighted\")\n",
    "    plt.plot(fpr, tpr, marker='.', label = key + \" (AUROC
{:.2f}\".format(auroc) + \")\")\n",
    "\n",
    "plt.xlabel('False positive rate')\n",
    "plt.ylabel('True positive rate')\n",
    "plt.title('ROC curve: moons unbalanced classification problem')\n",
    "plt.legend(loc='best', title=\"Models\")\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
```

```
    "# Precision - Recall Curve\n",
    "\n",
    "Good for unbalanced Problems"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.metrics import precision_recall_curve,
average_precision_score\n",
    "\n",
    "c=1\n",
    "plt.figure(figsize=(5,5))\n",
    "plt.plot([0, 1], [0.5, 0.5], 'k--')\n",
    "for key, value in clfs_.items():\n",
    "    y_pred = value.predict_proba(x_test)[:, c]\n",
    "    precision, recall, _ = precision_recall_curve(y_test, y_pred)\n",
    "    auroc = auc(recall, precision)\n",
    "    plt.plot(recall, precision, marker='.', label = key + \" (AUPR
{:.2f}\".format(auroc) + \")\")\n",
    "\n",
    "plt.xlabel('Precision')\n",
    "plt.ylabel('Recall')\n",
    "plt.title('PR curve on unseen data')\n",
    "plt.legend(loc='best', title=\"Models\")\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Tuning with parfit"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "#import scikitplot as skplt\n",
    "\n",
    "model = RandomForestClassifier(random_state = seed,n_jobs = n_jobs)\n",
    "model.fit(x_train,y_train)\n",
    "\n",
    "y_prob = model.predict_proba(x_test)[:, 1]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "model = RandomForestClassifier(random_state = seed,n_jobs = n_jobs)\n",
    "paramGrid = {\n",
    "    'min_samples_leaf': [1,3,5,10,15,25,50,100,125,150,175,200],\n",
    "    'max_features': ['sqrt', 'log2', 0.4, 0.5, 0.6, 0.7],\n",
    "    'n_estimators': [60],\n",
    "    'n_jobs': [-1],\n",
```

```
    "    'random_state': [seed]\n",
    "}\n",
    "\n",
    "clf = GridSearchCV(model, paramGrid, n_jobs=-1, cv=2, verbose=1)\n",
    "clf.fit(x_train,y_train)\n",
    "#clf.fit(x_train, y_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "y_scores = clf.predict_proba(x_test)[:, 1]\n",
    "\n",
    "average_precision = average_precision_score(y_test, y_scores)\n",
    "\n",
    "step_kwargs = ({'step': 'post'}\n",
    "               if 'step' in signature(plt.fill_between).parameters\n",
    "               else {})\n",
    "plt.figure(figsize=(10,7))\n",
    "plt.step(recall, precision, color='b', alpha=0.2,\n",
    "         where='post')\n",
    "plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)\
n",
    "\n",
    "plt.xlabel('Recall')\n",
    "plt.ylabel('Precision')\n",
    "plt.ylim([0.0, 1.05])\n",
    "plt.xlim([0.0, 1.0])\n",
    "plt.title('Precision-Recall curve: AP={0:0.2f}'.format(\n",
    "          average_precision))\n",
    "plt.show()\n",
    "\n",
    "get_profit(clf, x_test)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Deriving Classification Threshold"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import GridSearchCV\n",
    "from sklearn.metrics import make_scorer, average_precision_score,
precision_recall_curve\n",
    "mlp_param_grid = {'hidden_layer_sizes': [(3), (6), (3, 3), (5, 5)], \n",
    "                  'learning_rate_init': [0.001, 0.01]}\n",
    "\n",
```

```
    "model = MLPClassifier(random_state=seed)\n",
    "clf_gscv = GridSearchCV(model, mlp_param_grid, cv=5, n_jobs=-1, \n",
    "                        scoring=make_scorer(average_precision_score))\n",
    "\n",
    "# fit \n",
    "clf_gscv.fit(x_train, y_train)\n",
    "\n",
    "# analyze\n",
    "print(\"Best parameter set: \", clf_gscv.best_params_, \"\\n\")\n",
    "means = clf_gscv.cv_results_['mean_test_score']\n",
    "stds = clf_gscv.cv_results_['std_test_score']\n",
    "for mean, std, params in zip(means, stds, clf_gscv.cv_results_['params']):\
n",
    "    print(\"%0.3f (+/-%0.03f) for %r\" % (mean, std * 2, params))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# generalization ability in terms of profit"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "#predictions = clf_gscv.best_estimator_.predict(x_test)\n",
    "#predict_metrics(y_test,predictions)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "clf = clf_gscv.best_estimator_\n",
    "get_profit(clf, x_test)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# ----------------------------------------------------------------"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Logistic"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": false
   },
   "outputs": [],
```

```
   "source": [
    "clf = LogisticRegression(random_state = seed, n_jobs = n_jobs)\n",
    "clf.fit(x_train, y_train)\n",
    "predictions = clf.predict(x_test)\n",
    "predict_metrics(y_test, predictions)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "get_profit(clf, x_test)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import scikitplot as skplt\n",
    "y_prob = clf.predict_proba(x_test)\n",
    "skplt.metrics.plot_ks_statistic(y_test, y_prob,figsize=(10,5))\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### !!fazer uma generalizacao do teste ks com profit para decidir
threshold!!"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Plots "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Adjusting Threshold"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "clf = RandomForestClassifier()\n",
```

```
    "clf.fit(x_train, y_train)\n",
    "predictions = clf.predict(x_test)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "y_scores = clf.predict_proba(x_test)[:, 1]\n",
    "# for classifiers with decision_function, this achieves similar results\n",
    "# y_scores = classifier.decision_function(X_test)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "p, r, thresholds = precision_recall_curve(y_test, y_scores)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def adjusted_classes(y_scores, t):\n",
    "    \"\"\"\n",
    "    This function adjusts class predictions based on the prediction
threshold (t).\n",
    "    Will only work for binary classification problems.\n",
    "    \"\"\"\n",
    "    return [1 if y >= t else 0 for y in y_scores]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def precision_recall_threshold(p, r, thresholds, t=0.5):\n",
    "    \"\"\"\n",
    "    plots the precision recall curve and shows the current value for each\
n",
    "    by identifying the classifier's threshold (t).\n",
    "    \"\"\"\n",
    "    \n",
    "    # generate new class predictions based on the adjusted_classes\n",
    "    # function above and view the resulting confusion matrix.\n",
    "    y_pred_adj = adjusted_classes(y_scores, t)\n",
    "    print(pd.DataFrame(confusion_matrix(y_test, y_pred_adj),\n",
    "                       columns=['pred_neg', 'pred_pos'], \n",
    "                       index=['neg', 'pos']))\n",
    "    \n",
    "    # plot the curve\n",
    "    plt.figure(figsize=(8,8))\n",
    "    plt.title(\"Precision and Recall curve ^ = current threshold\")\n",
    "    plt.step(r, p, color='b', alpha=0.2,\n",
    "             where='post')\n",
```

```
    "    plt.fill_between(r, p, step='post', alpha=0.2,\n",
    "                     color='b')\n",
    "    plt.ylim([0, 1]);\n",
    "    plt.xlim([0, 1]);\n",
    "    plt.xlabel('Recall');\n",
    "    plt.ylabel('Precision');\n",
    "    \n",
    "    # plot the current threshold on the line\n",
    "    close_default_clf = np.argmin(np.abs(thresholds - t))\n",
    "    plt.plot(r[close_default_clf], p[close_default_clf], '^', c='k',\n",
    "             markersize=15)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "precision_recall_threshold(p, r,0.5)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):\
n",
    "    plt.figure(figsize=(8, 8))\n",
    "    plt.title(\"Precision and Recall Scores as a function of the decision
threshold\")\n",
    "    plt.plot(thresholds, precisions[:-1], \"b--\", label=\"Precision\")\n",
    "    plt.plot(thresholds, recalls[:-1], \"g-\", label=\"Recall\")\n",
    "    plt.ylabel(\"Score\")\n",
    "    plt.xlabel(\"Decision Threshold\")\n",
    "    plt.legend(loc='best')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "plot_precision_recall_vs_threshold(p, r, thresholds)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def plot_roc_curve(fpr, tpr, label=None):\n",
    "    plt.figure(figsize=(8,8))\n",
    "    plt.title('ROC Curve')\n",
    "    plt.plot(fpr, tpr, linewidth=2, label=label)\n",
    "    plt.plot([0, 1], [0, 1], 'k--')\n",
    "    plt.axis([-0.005, 1, 0, 1.005])\n",
    "    #plt.xticks(np.arange(0,1, 0.05), rotation=90)\n",
    "    plt.xlabel(\"False Positive Rate\")\n",
    "    plt.ylabel(\"True Positive Rate (Recall)\")\n",
```

```
   "    plt.legend(loc='best')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "fpr, tpr, auc_thresholds = roc_curve(y_test, y_scores)\n",
   "print(auc(fpr, tpr)) # AUC of ROC\n",
   "plot_roc_curve(fpr, tpr, 'recall_optimized')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "threshold = 0.5\n",
   "\n",
   "predicted_proba = clf.predict_proba(x_test)\n",
   "predictions = (predicted_proba [:,1] >= threshold).astype('int')\n",
   "\n",
   "accuracy = accuracy_score(y_test, predictions)\n",
   "accuracy"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "predict_metrics(y_test, predictions)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# Deep Learning"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "from tensorflow.keras import models, Sequential\n",
   "from tensorflow.keras.layers import Dense, Activation\n",
   "from sklearn.model_selection import train_test_split\n",
   "from sklearn.preprocessing import MinMaxScaler\n",
   "from tensorflow.keras import optimizers"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
```

```
    "def plot_history(history):\n",
    "    acc = history.history['accuracy']\n",
    "    val_acc = history.history['val_accuracy']\n",
    "    loss = history.history['loss']\n",
    "    val_loss = history.history['val_loss']\n",
    "    x = range(1, len(acc) + 1)\n",
    "    \n",
    "    plt.figure(figsize = (12,5))\n",
    "    plt.subplot(1, 2, 1)\n",
    "    plt.plot(x, acc, 'b', label = 'Trainning acc')\n",
    "    plt.plot(x, val_acc, 'r', label = 'Validation acc')\n",
    "    plt.title('Trainning and Validation accuracy')\n",
    "    plt.legend()\n",
    "    plt.subplot(1, 2, 2)\n",
    "    plt.plot(x, loss, 'b', label = 'Trainning loss')\n",
    "    plt.plot(x, val_loss, 'r', label = 'Validation loss')\n",
    "    plt.title('Trainning and Validation loss')\n",
    "    plt.legend()\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "input_dim = x_test.shape[1]\n",
    "\n",
    "model = Sequential()\n",
    "model.add(Dense(10, input_dim=input_dim, activation='relu'))\n",
    "model.add(Dense(10, activation='relu'))\n",
    "model.add(Dense(1, activation='sigmoid'))\n",
    "\n",
    "model.summary()\n",
    "model.compile(loss='binary_crossentropy', optimizer = 
'adam',metrics=['accuracy'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "epochs = 165\n",
    "history = model.fit(x_train, y_train,epochs=epochs,batch_size=30,
validation_split = 0.1)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "plot_history(history)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
```

```
 "outputs": [],
 "source": [
  "loss, accuracy = model.evaluate(x_train, y_train, verbose = False)\n",
  "print(\"Trainning Accuracy: {:.2%}\".format(accuracy))\n",
  "\n",
  "loss, accuracy = model.evaluate(x_test, y_test, verbose = False)\n",
  "print(\"Test Accuracy: {:.2%}\".format(accuracy))"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "predictions = model.predict(x_test)\n",
  "predictions = [int(np.round(x[0])) for x in predictions]"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "scrolled": false
 },
 "outputs": [],
 "source": [
  "predict_metrics(y_test, predictions)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Ensemble"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 1. Models"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "# knn\n",
  "\n",
  "#create new a knn model\n",
  "knn = KNeighborsClassifier()\n",
  "#create a dictionary of all values we want to test for n_neighbors\n",
  "params_knn = {'n_neighbors': np.arange(1, 25)}\n",
  "#use gridsearch to test all values for n_neighbors\n",
  "knn_gs = GridSearchCV(knn, params_knn, cv=5)\n",
  "#fit model to training data\n",
  "knn_gs.fit(x_train, y_train)"
 ]
},
{
 "cell_type": "code",
```

```
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "#save best model\n",
  "knn_best = knn_gs.best_estimator_\n",
  "#check best n_neigbors value\n",
  "print(knn_gs.best_params_)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "# Random Forest\n",
  "\n",
  "#create a new random forest classifier\n",
  "rf = RandomForestClassifier()\n",
  "#create a dictionary of all values we want to test for n_estimators\n",
  "params_rf = {'n_estimators': [50, 100, 200]}\n",
  "#use gridsearch to test all values for n_estimators\n",
  "rf_gs = GridSearchCV(rf, params_rf, cv=5)\n",
  "#fit model to training data\n",
  "rf_gs.fit(x_train, y_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "#save best model\n",
  "rf_best = rf_gs.best_estimator_\n",
  "#check best n_estimators value\n",
  "print(rf_gs.best_params_)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "# Logistic Regression\n",
  "\n",
  "#create a new logistic regression model\n",
  "log_reg = LogisticRegression()\n",
  "#fit the model to the training data\n",
  "log_reg.fit(x_train, y_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "print('knn: {:.2%}'.format(knn_best.score(x_test, y_test)))\n",
  "print('rf: {:.2%}'.format(rf_best.score(x_test, y_test)))\n",
  "print('log_reg: {:.2%}'.format(log_reg.score(x_test, y_test)))"
 ]
```

```
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## 2. Voting"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "from sklearn.ensemble import VotingClassifier\n",
  "#create a dictionary of our models\n",
  "estimators=[('knn', knn_best), ('rf', rf_best), ('log_reg', log_reg)]\n",
  "#create our voting classifier, inputting our models\n",
  "ensemble = VotingClassifier(estimators, voting='hard')"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "#fit model to training data\n",
  "ensemble.fit(x_train, y_train)\n",
  "#test our model on the test data\n",
  "ensemble.score(x_test, y_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "predictions = ensemble.predict(x_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "predict_metrics(y_test, predictions)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Esemble\n",
  "---"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### 1. Bagging"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "models= [LogisticRegression(random_state = seed, n_jobs = n_jobs),\n",
    "         MultinomialNB(),\n",
    "         BernoulliNB(),\n",
    "         GaussianNB(),\n",
    "         PassiveAggressiveClassifier(random_state=seed, n_jobs = n_jobs), #
parece ser bom\n",
    "         NuSVC(random_state=seed, gamma='scale'), # bom\n",
    "         LabelPropagation(n_jobs = n_jobs, alpha = 0, kernel='knn'),\n",
    "         LabelSpreading(n_jobs = n_jobs, kernel = 'knn'),\n",
    "         RandomForestClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         SGDClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         DecisionTreeClassifier(random_state = seed),\n",
    "         XGBClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         GradientBoostingClassifier(random_state = seed),\n",
    "         RidgeClassifier(random_state = seed),\n",
    "         SVC(random_state = seed,kernel='linear'),\n",
    "         KNeighborsClassifier(n_jobs = n_jobs),\n",
    "         NearestCentroid(),\n",
    "         QuadraticDiscriminantAnalysis(),\n",
    "         LinearDiscriminantAnalysis()]\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.ensemble import BaggingClassifier\n",
    "#compare_models = pd.DataFrame({'acc':0})\n",
    "acc_simple = []\n",
    "rec_simple = []\n",
    "simple_names=[]\n",
    "\n",
    "acc_bag = []\n",
    "rec_bag = []\n",
    "bag_names = []\n",
    "\n",
    "for model in tqdm(models):\n",
    "    \n",
    "    model_name = model.__class__.__name__\n",
    "    model.fit(x_train, y_train)\n",
    "    predictions = model.predict(x_test)\n",
    "    #print(\"Original Results of: \", (model_name), '\\n')\n",
    "    #print(classification_report(y_test,predictions))\n",
    "    \n",
    "    acc_simple.append(np.round(accuracy_score(predictions, y_test),3))\n",
    "    rec_simple.append(np.round(recall_score( y_test, predictions),3))\n",
    "    simple_names.append(model_name)\n",
    "    \n",
    "    bagging_clf = BaggingClassifier(model, random_state=seed)\n",
    "    \n",
    "    bagging_clf.fit(x_train, y_train)\n",
    "    predictions = bagging_clf.predict(x_test)\n",
    "    \n",
    "    \n",
```

```
    "    \n",
    "    #print(\"Bagged Results of: \", (model_name), '\\n')\n",
    "    #print(classification_report(y_test,predictions))\n",
    "    \n",
    "    acc_bag.append(np.round(accuracy_score(predictions, y_test),3))\n",
    "    rec_bag.append(np.round(recall_score( y_test, predictions),3))\n",
    "    bag_names.append('bag_'+model_name)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "simple_compare = pd.DataFrame({'acc':acc_simple, 'rec': rec_simple},
index=simple_names)\n",
    "bag_compare = pd.DataFrame({'acc':acc_bag, 'rec':rec_bag},
index=bag_names)\n",
    "compare_models = pd.concat([simple_compare, bag_compare])\n",
    "\n",
    "print(compare_models.sort_values(by='acc',ascending=False)[:5])\n",
    "print(compare_models.sort_values(by='rec',ascending=False)[:5])\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "bagging_clf = BaggingClassifier(RandomForestClassifier(),
random_state=seed)\n",
    "bagging_clf.fit(x_train, y_train)\n",
    "predictions = bagging_clf.predict(x_test)\n",
    "predict_metrics(y_test, predictions)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "clf = bagging_clf\n",
    "get_profit(clf, x_test)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### 2. Voting"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Aqui estÃ£o os modelos normais\n",
    "\n",
    "names = []\n",
```

```
  "for model in models:\n",
  "    model_name = model.__class__.__name__\n",
  "    names.append(model_name)\n",
  "model_simple_tuple_list = list(zip(names,models))"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "from sklearn.ensemble import VotingClassifier\n",
  "# Aqui estÃ£o os modelos em Bagging\n",
  "bag_models = []\n",
  "bag_names = []\n",
  "for model in models:\n",
  "    model_ = BaggingClassifier(model, random_state=seed)\n",
  "    model_name = 'bag_'+model.__class__.__name__\n",
  "    bag_models.append(model_)\n",
  "    bag_names.append(model_name)\n",
  "    \n",
  "model_bagged_tuple_list = list(zip(bag_names,bag_models))"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "model_tuple_list = model_simple_tuple_list + model_bagged_tuple_list"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "ensemble = VotingClassifier(estimators=model_tuple_list, voting='hard')"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "#fit model to training data\n",
  "ensemble.fit(x_train, y_train)\n",
  "#test our model on the test data\n",
  "ensemble.score(x_test, y_test)\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "predictions = ensemble.predict(x_test)\n",
  "predict_metrics(y_test, predictions)"
 ]
```

```
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# PCA"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "pca = pd.read_excel('df_PCA.xlsx', index_col=0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "pca.head(1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "x_train, x_test, y_train, y_test = split_data(pca,balance = True,
random_state = seed, test_size = 0.2, strat = True)\n",
    "x_train, x_test = scale_data(x_train, x_test)\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "n_jobs = -1\n",
    "models= [LogisticRegression(random_state = seed, n_jobs = n_jobs),\n",
    "         MultinomialNB(),\n",
    "         BernoulliNB(),\n",
    "         PassiveAggressiveClassifier(random_state=seed, n_jobs = n_jobs), #
parece ser bom\n",
    "         NuSVC(random_state=seed, gamma='scale'), # bom\n",
    "         QuadraticDiscriminantAnalysis(),\n",
    "         RandomForestClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         SGDClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         DecisionTreeClassifier(random_state = seed),\n",
    "         XGBClassifier(random_state = seed,n_jobs = n_jobs),\n",
    "         GradientBoostingClassifier(random_state = seed),\n",
    "         RidgeClassifier(random_state = seed),\n",
    "         SVC(random_state = seed,kernel='linear'),\n",
    "         KNeighborsClassifier(n_jobs = n_jobs),\n",
    "         QuadraticDiscriminantAnalysis(),\n",
    "         LinearDiscriminantAnalysis()]"
   ]
  },
```

```
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "n_splits = 5\n",
    "CV = KFold(n_splits=n_splits, random_state=seed)\n",
    "cv_df = pd.DataFrame(index=range(n_splits * len(models)))\n",
    "scoring = 'accuracy' #accuracy #precision #recall #f1\n",
    "entries = []\n",
    "\n",
    "x = x_train\n",
    "y = y_train \n",
    "\n",
    "for model in models:\n",
    "  model_name = model.__class__.__name__\n",
    "  accuracies = cross_val_score(model, x, y, scoring= scoring, cv=CV)\n",
    "  for fold_idx, accuracy in enumerate(accuracies):\n",
    "    entries.append((model_name, fold_idx, accuracy))\n",
    "cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', scoring])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Plot different models\n",
    "plt.figure(figsize = (20,7))\n",
    "ax = sns.boxplot(x='model_name', y=scoring,
data=cv_df,boxprops=dict(alpha=.8))\n",
    "sns.stripplot(x='model_name', y=scoring, data=cv_df, \n",
    "              size=6, jitter=True, edgecolor=\"black\", linewidth=.5)\n",
    "ax.set_xticklabels(ax.get_xticklabels(), rotation=75, fontsize =12) \n",
    "plt.xlabel('Model',  fontweight = \"bold\")\n",
    "plt.ylabel(scoring,  fontweight = \"bold\")\n",
    "plt.show()\n",
    "\n",
    "print(np.round(\n",
    "        cv_df.groupby('model_name').agg('mean').\n",
    "        sort_values(by=[scoring], \n",
    "                    ascending=False)[scoring],decimals=3)*100)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "clf = LogisticRegression(random_state = seed, n_jobs = n_jobs)\n",
    "clf.fit(x_train, y_train)\n",
    "predictions = clf.predict(x_test)\n",
    "predict_metrics(y_test, predictions)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Save and Export Model"
   ]
```

```
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "from sklearn.externals import joblib"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "# Insert here final model\n",
   "\n",
   "model = LogisticRegression(random_state = seed, n_jobs = n_jobs)\n",
   "model.fit(x_train, y_train)\n",
   "predictions = model.predict(x_test)\n",
   "predict_metrics(y_test, predictions)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "# save the model to disk\n",
   "filename = 'finalized_model.sav'\n",
   "joblib.dump(model, filename)\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "# load the model from disk\n",
   "loaded_model = joblib.load(filename)\n",
   "result = loaded_model.score(x_test_scaled, y_test)\n",
   "print(result)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
```

```
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```