```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "import pandas as pd\n",
    "import os \n",
    "import numpy as np\n",
    "from numpy import mean\n",
    "from numpy import var\n",
    "from math import sqrt\n",
    "from scipy import stats, linalg\n",
    "import warnings\n",
    "import statsmodels.api as sm\n",
    "from statsmodels.stats.power import TTestIndPower\n",
    "from statsmodels.stats.multitest import multipletests,
fdrcorrection\n",
    "from statsmodels.formula.api import ols\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "import plotly.express as px\n",
    "\n",
    "# import pingouin as pg\n",
    "\n",
    "%matplotlib inline\n",
    "\n",
    "\n",
    "# Suppress FutureWarning messages\n",
    "warnings.simplefilter(action='ignore', category=FutureWarning)\n",
    "warnings.simplefilter(action='ignore',
category=DeprecationWarning)\n",
    "\n",
    "idx = pd.IndexSlice\n",
    "\n",
    "mydir = '/Users/rye/Documents/Documents - Mac/BrainBalance'\n",
    "myfile = 'Creyos-BrainBalance-2023-11-30.csv'\n",
    "\n",
    "crm_file = 'Adult Pre Vs Post Paperwork_November 2023_De-
identified.xlsx'\n",
    "\n",
    "nonenroll_file = 'Non_Enrollment
Type_Multiple_Tests_Identifier_Only_2.xlsx'\n",
    "creyos_file_new = 'Creyos-BrainBalance- Enrollment Type-2023-12-
05.xlsx'\n",
    "#creyos_file_new = 'SCI-92-Creyos-BrainBalance-2023-12-20.xlsx'\n",
    "\n",
    "mycols = ['user_id', 'client_id', 'report', 'test_date',
'batch_name',\n",
    "          'trial_name', 'birthdate',  'enroll type','gender',\n",
```

```
    "       'age_at_test', 'SS_avg_score', 'DT_final_score',
'ML_max_score', 'RT_final_score', 'FM_final_score', 'TS_max_score']"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "Int64Index: 2389 entries, 0 to 2390\n",
      "Data columns (total 9 columns):\n",
      " #   Column                          Non-Null Count  Dtype
\n",
      "---  ------                          --------------  -----
\n",
      " 0   user_id                         2389 non-null   string
\n",
      " 1   client_id                       2389 non-null   object
\n",
      " 2   email                           2389 non-null   object
\n",
      " 3   birthdate                       2388 non-null
datetime64[ns]\n",
      " 4   Status                          2048 non-null   object
\n",
      " 5   Program Fit, No Intervention  159 non-null    object
\n",
      " 6   Hours                           170 non-null    object
\n",
      " 7   Center                          1523 non-null   object
\n",
      " 8   Unnamed: 8                      2 non-null      object
\n",
      "dtypes: datetime64[ns](1), object(7), string(1)\n",
      "memory usage: 186.6+ KB\n"
     ]
    }
   ],
   "source": [
    "#df_crm = pd.read_excel(os.path.join(mydir, crm_file))\n",
    "\n",
    "# RJ notes indicate hours completed differ from sessions completed.
\n",
    "# In all cases, sessions completed shows 0 whereas hours are greater
than zero.\n",
    "df_NoEnroll = pd.read_excel(os.path.join(mydir, nonenroll_file),
skiprows=11)\n",
```

```
    "df_NoEnroll['user_id'] = df_NoEnroll['user_id'].astype('string')\n",
    "df_NoEnroll['Status'] =
df_NoEnroll['Status'].str.strip().str.capitalize()\n",
    "df_NoEnroll['Status'] = df_NoEnroll['Status'].replace({'Could not
identify':'Can not identify' ,'Unable to identify':'Can not identify',
'Vitual':'Virtual', 'Hyrbid':'Hybrid', 'In center - control':'In center
did not enroll'})\n",
    "df_NoEnroll['Status'] =
df_NoEnroll['Status'].replace({'Virtual':'Virtual New'})\n",
    "df_NoEnroll = df_NoEnroll[(df_NoEnroll['Program Fit, No
Intervention']!='Not a program fit') | (df_NoEnroll['Status']=='Can not
identify')]\n",
    "##########\n",
    "#df_NoEnroll = df_NoEnroll[~df_NoEnroll['Status'].isin(['In center
did not enroll', 'Virtual did not enroll', 'Adult did not enroll', 'Did
not enroll'])]\n",
    "############\n",
    "\n",
    "\n",
    "## load new Creyos data with SS max score\n",
    "df_SSmax = pd.read_excel(os.path.join(mydir, 'SCI-92-Creyos-
BrainBalance-2024-01-31.xlsx'))\n",
    "df_SSmax['user_id'] = df_SSmax['user_id'].astype('string')\n",
    "\n",
    "###########\n",
    "# load old Creyos data with enroll type\n",
    "###########\n",
    "#df_SSmax['test_date'] = pd.to_datetime(df_SSmax['test_date'])\n",
    "df = pd.read_excel(os.path.join(mydir, creyos_file_new))\n",
    "df = df[mycols]\n",
    "df['user_id'] = df['user_id'].astype('string')\n",
    "#df['birthdate'] = pd.to_datetime(df['birthdate'],
errors='coerce')\n",
    "#df['test_date'] = pd.to_datetime(df['test_date'])\n",
    "df = df.sort_values(by=['user_id',
'test_date']).reset_index().drop(columns='index')\n",
    "\n",
    "##########\n",
    "df_NoEnroll.info()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "Int64Index: 2389 entries, 0 to 2390\n",
      "Data columns (total 9 columns):\n",
```

```
      " #   Column                          Non-Null Count  Dtype  \n",
      "---  ------                          --------------  -----  \n",
      " 0   user_id                         2389 non-null   string \n",
      " 1   client_id                       2389 non-null   object \n",
      " 2   email                           2389 non-null   object \n",
      " 3   birthdate                       2388 non-null   datetime64[ns]\n",
      " 4   Status                          2048 non-null   object \n",
      " 5   Program Fit, No Intervention  159 non-null    object \n",
      " 6   Hours                           170 non-null    object \n",
      " 7   Center                          1523 non-null   object \n",
      " 8   Unnamed: 8                      2 non-null      object \n",
      "dtypes: datetime64[ns](1), object(7), string(1)\n",
      "memory usage: 186.6+ KB\n"
     ]
    },
    {
     "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>user_id</th>\n",
       "      <th>client_id</th>\n",
       "      <th>report</th>\n",
       "      <th>test_date</th>\n",
       "      <th>batch_name</th>\n",
       "      <th>trial_name</th>\n",
       "      <th>birthdate</th>\n",
```

```
"      <th>enroll type</th>\n",
"      <th>gender</th>\n",
"      <th>age_at_test</th>\n",
"      <th>SS_avg_score</th>\n",
"      <th>DT_final_score</th>\n",
"      <th>ML_max_score</th>\n",
"      <th>RT_final_score</th>\n",
"      <th>FM_final_score</th>\n",
"      <th>TS_max_score</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>118799</td>\n",
"      <td>NaN</td>\n",
"      <td>152164334</td>\n",
"      <td>2023-02-17</td>\n",
"      <td>bbassessment71</td>\n",
"      <td>brain balance foxvalley</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>Male</td>\n",
"      <td>NaN</td>\n",
"      <td>8.0</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>10.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>165444</td>\n",
"      <td>petereggers</td>\n",
"      <td>23530730</td>\n",
"      <td>2019-03-12</td>\n",
"      <td>assessment3</td>\n",
"      <td>brain balance of overland park</td>\n",
"      <td>2011-05-06 00:00:00</td>\n",
"      <td>NaN</td>\n",
"      <td>Male</td>\n",
"      <td>7.0</td>\n",
"      <td>4.5</td>\n",
"      <td>16.0</td>\n",
"      <td>6.0</td>\n",
"      <td>30.0</td>\n",
"      <td>64.0</td>\n",
"      <td>7.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>165446</td>\n",
"      <td>sallysample</td>\n",
```

```
"      <td>23528820</td>\n",
"      <td>2019-03-12</td>\n",
"      <td>assessment3</td>\n",
"      <td>brain balance of overland park</td>\n",
"      <td>2006-03-08 00:00:00</td>\n",
"      <td>NaN</td>\n",
"      <td>Female</td>\n",
"      <td>13.0</td>\n",
"      <td>4.4</td>\n",
"      <td>NaN</td>\n",
"      <td>7.0</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>165446</td>\n",
"      <td>sallysample</td>\n",
"      <td>23539134</td>\n",
"      <td>2019-03-12</td>\n",
"      <td>my bbassessment91 4336</td>\n",
"      <td>brain balance of overland park</td>\n",
"      <td>2006-03-08 00:00:00</td>\n",
"      <td>NaN</td>\n",
"      <td>Female</td>\n",
"      <td>13.0</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>165446</td>\n",
"      <td>sallysample</td>\n",
"      <td>23583737</td>\n",
"      <td>2019-03-13</td>\n",
"      <td>my bbassessment91 4336</td>\n",
"      <td>brain balance of overland park</td>\n",
"      <td>2006-03-08 00:00:00</td>\n",
"      <td>NaN</td>\n",
"      <td>Female</td>\n",
"      <td>13.0</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>8.0</td>\n",
"      <td>12.0</td>\n",
"      <td>132.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"  </tbody>\n",
```

```
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "  user_id    client_id     report  test_date
batch_name  \\\n",
       "0  118799          NaN  152164334 2023-02-17
bbassessment71   \n",
       "1  165444  petereggers   23530730 2019-03-12
assessment3   \n",
       "2  165446  sallysample   23528820 2019-03-12
assessment3   \n",
       "3  165446  sallysample   23539134 2019-03-12  my bbassessment91
4336   \n",
       "4  165446  sallysample   23583737 2019-03-13  my bbassessment91
4336   \n",
       "\n",
       "                         trial_name            birthdate enroll
type  gender  \\\n",
       "0         brain balance foxvalley                  NaN
NaN    Male   \n",
       "1  brain balance of overland park  2011-05-06 00:00:00
NaN    Male   \n",
       "2  brain balance of overland park  2006-03-08 00:00:00
NaN  Female   \n",
       "3  brain balance of overland park  2006-03-08 00:00:00
NaN  Female   \n",
       "4  brain balance of overland park  2006-03-08 00:00:00
NaN  Female   \n",
       "\n",
       "   age_at_test  SS_avg_score  DT_final_score  ML_max_score
RT_final_score  \\\n",
       "0          NaN           8.0             NaN           NaN
NaN   \n",
       "1          7.0           4.5            16.0           6.0
30.0   \n",
       "2         13.0           4.4             NaN           7.0
NaN   \n",
       "3         13.0           NaN             NaN           NaN
NaN   \n",
       "4         13.0           NaN             NaN           8.0
12.0   \n",
       "\n",
       "   FM_final_score  TS_max_score  \n",
       "0            10.0           NaN  \n",
       "1            64.0           7.0  \n",
       "2             NaN           NaN  \n",
       "3             NaN           NaN  \n",
       "4           132.0           NaN  "
      ]
     },
     "execution_count": 5,
     "metadata": {},
     "output_type": "execute_result"
```

```
    }
   ],
   "source": [
    "df_NoEnroll.info()\n",
    "df_SSmax.head()\n",
    "df.head()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(63646, 18)"
      ]
     },
     "execution_count": 6,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# df1 = df_SSmax[['user_id', 'test_date', 'gender',
'report','SS_max_score', 'DT_final_score', 'ML_max_score',
'RT_final_score',\n",
    "#        'FM_final_score',
'TS_max_score']].drop_duplicates().dropna(subset=['SS_max_score',
'DT_final_score', 'ML_max_score', 'RT_final_score',\n",
    "#        'FM_final_score', 'TS_max_score'])\n",
    "\n",
    "# df2 = df.drop_duplicates().dropna(subset = ['SS_avg_score',
'DT_final_score', 'ML_max_score', 'RT_final_score',\n",
    "#        'FM_final_score', 'TS_max_score'])\n",
    "\n",
    "df1 = df_SSmax[['user_id', 'test_date', 'gender', 'report',
'SS_max_score', 'DT_final_score', 'ML_max_score', 'RT_final_score',\n",
    "       'FM_final_score', 'TS_max_score']].drop_duplicates()\n",
    "\n",
    "df2 = df.drop_duplicates()\n",
    "\n",
    "df_new = pd.merge(df1, df2, how='inner', left_on=['user_id',
'test_date', 'gender', 'report','DT_final_score', 'ML_max_score',
'RT_final_score',\n",
    "       'FM_final_score', 'TS_max_score'], right_on=['user_id',
'test_date', 'gender', 'report', 'DT_final_score', 'ML_max_score',
'RT_final_score',\n",
    "       'FM_final_score', 'TS_max_score'], indicator=True)\n",
    "\n",
    "df_new.shape"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "enroll type\n",
       "Adult                  347\n",
       "Adult CTRL               9\n",
       "Can not identify         8\n",
       "Child CTRL             124\n",
       "Hybrid                 860\n",
       "Initial              15499\n",
       "Test                     8\n",
       "Virtual                640\n",
       "Virtual New            768\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 7,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_comb = pd.merge(df_new.drop(columns='_merge'),
df_NoEnroll[['user_id', 'client_id', 'Status', 'Hours', 'Center']],
how='left', left_on=['user_id', 'client_id'], right_on=['user_id',
'client_id'])\n",
    "df_comb['enroll type'] = df_comb['enroll type'].mask(df_comb['enroll
type'].isna(), df_comb['Status'])\n",
    "df_comb['enroll type'].replace({'In center':'Initial', 'In center
did not enroll':'Child CTRL', 'Virtual did not enroll':'Child CTRL',
'Adult did not enroll':'Adult CTRL'}, inplace=True)\n",
    "df_comb.groupby('enroll type')['user_id'].nunique()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "# get df_comb for adults\n",
```

```
    "adult_wrong_age = df_comb.loc[(df_comb['age_at_test']<18 ) &
df_comb['enroll type'].isin(['Adult', 'Adult CTRL']),
'user_id'].values\n",
    "df_comb = df_comb[~df_comb['user_id'].isin(adult_wrong_age)]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 9,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "adult_wrong_age = df_comb.loc[(df_comb['age_at_test']<18 ) &
(df_comb['enroll type']=='Adult'), 'user_id'].values\n",
    "df_comb = df_comb[~df_comb['user_id'].isin(adult_wrong_age)]\n",
    "\n",
    "children_adult_age = df_comb.loc[(df_comb['age_at_test']>=18 ) &
(df_comb['enroll type']!='Adult'), 'user_id'].values\n",
    "df_comb = df_comb[~df_comb['user_id'].isin(children_adult_age)]\n",
    "\n",
    "children_wrong_age = df_comb.loc[(df_comb['age_at_test']<4 ) &
(df_comb['enroll type']!='Adult'), 'user_id'].values\n",
    "df_comb = df_comb[~df_comb['user_id'].isin(children_wrong_age)]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 10,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "enroll type\n",
       "Adult                 331\n",
       "Can not identify        5\n",
       "Child CTRL            118\n",
       "Hybrid                834\n",
       "Initial             14976\n",
       "Test                    3\n",
       "Virtual               609\n",
       "Virtual New           745\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 10,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_comb.groupby('enroll type')['user_id'].nunique()"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 11,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "count    25441.000000\n",
       "mean         9.639794\n",
       "std          3.183889\n",
       "min          4.000000\n",
       "25%          7.000000\n",
       "50%          9.000000\n",
       "75%         12.000000\n",
       "max         17.000000\n",
       "Name: age_at_test, dtype: float64"
      ]
     },
     "execution_count": 11,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_comb.loc[df_comb['enroll type'].isin(['Initial', 'Virtual New',
'Virtual']),'age_at_test'].describe()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "gender\n",
       "Female     5250\n",
       "Male      11071\n",
       "Other         9\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 12,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_comb[df_comb['enroll type'].isin(['Initial', 'Virtual New',
'Virtual'])].groupby('gender')['user_id'].nunique()"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [

"/var/folders/qn/v0jt4xsj1y92jrmwls18h_600000gn/T/ipykernel_4753/40801938
60.py:2: SettingWithCopyWarning: \n",
      "A value is trying to be set on a copy of a slice from a
DataFrame.\n",
      "Try using .loc[row_indexer,col_indexer] = value instead\n",
      "\n",
      "See the caveats in the documentation:
https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy\n",
      "  demo_df['enroll type'] = demo_df['enroll
type'].replace({'Virtual New':'Virtual'})\n"
     ]
    }
   ],
   "source": [
    "demo_df = df_comb.loc[df_comb['enroll type'].isin(['Initial',
'Virtual New', 'Virtual', 'Child CTRL']),:]\n",
    "demo_df['enroll type'] = demo_df['enroll type'].replace({'Virtual
New':'Virtual'})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "def get_interval(df):\n",
    "    df_sorted = df.sort_values(['user_id',
'test_date']).copy(deep=True)\n",
    "    df_sorted['interval'] = df_sorted['test_date'] -
df_sorted['test_date'].shift(1) \n",
    "
df_sorted.loc[df_sorted.groupby('user_id')['test_date'].idxmin().values,'
interval'] = pd.Timedelta(days=0)\n",
    "    df_sorted['interval'] =
df_sorted['interval'].fillna(pd.Timedelta(days=0))\n",
    "    df_sorted['interval_int'] =  df_sorted['interval'].dt.days\n",
    "    cum_interval =
df_sorted.groupby('user_id')['interval_int'].cumsum()\n",
```

```
    "    df_interval = pd.merge(left=df_sorted, right=cum_interval,
how='left', left_index=True, right_index=True).rename(columns =
{'interval_int_x':'interval_int',
'interval_int_y':'cum_interval_int'})\n",
    "    df_interval['cum_interval_int'] =
df_interval['cum_interval_int'].astype('Int64')\n",
    "    df_interval['rank_interval'] =
df_interval.groupby('user_id')['cum_interval_int'].transform('rank',
method='first')\n",
    "    trt_idx =
df_interval.groupby('user_id')['cum_interval_int'].max()\n",
    "    \n",
    "    return df_interval\n",
    "\n",
    "####################\n",
    "\n",
    "my_ctrl = ['<=30 days apart', '<=60 days apart', '61-89 days apart',
'>=90 days']\n",
    "\n",
    "####################\n",
    "\n",
    "# add an argument for choosing whether to use the max() or min() on
cum_interval_int\n",
    "def get_grps(df):\n",
    "\n",
    "    test_dates = get_interval(df)\n",
    "    test_idx =
test_dates[test_dates['cum_interval_int']>0].groupby('user_id')['cum_inte
rval_int'].min()\n",
    "    test_dates['trt_grp'] = 'only once'\n",
    "
test_dates.loc[test_dates['user_id'].isin(test_idx[test_idx>=90].index),
'trt_grp'] = '>=90 days apart'\n",
    "
test_dates.loc[test_dates['user_id'].isin(test_idx[test_idx<90].index),
'trt_grp'] = '61-89 days apart'\n",
    "\n",
    "\n",
    "
test_dates.loc[test_dates['user_id'].isin(test_idx[test_idx<=60].index),
'trt_grp'] = '<=60 days apart'\n",
    "
test_dates.loc[test_dates['user_id'].isin(test_idx[test_idx<=30].index),
'trt_grp'] = '<=30 days apart'\n",
    "\n",
    "\n",
    "    test_dates['trt_grp'] = pd.Categorical(test_dates['trt_grp'],
\n",
    "                          categories=['only once', '<=30 days apart',
'<=60 days apart', '61-89 days apart', '>=90 days apart'],\n",
    "                          ordered=True)\n",
    "\n",
    "    return test_dates\n",
    "\n",
```

```
    "# problem here is that one user_id can belong to multiple program
(enroll) types\n",
    "\n",
    "def get_one_enroll_type(df, program_type, my_trt_dur):\n",
    "    df_grps = get_grps(df)\n",
    "    my_sub = df_grps.loc[df_grps['trt_grp'].isin(my_trt_dur) &
(df_grps['enroll type'].isin(program_type)),'user_id']\n",
    "    my_enroll_data =
df_grps[df_grps['user_id'].isin(my_sub)].sort_values(['user_id',
'test_date'])\n",
    "    \n",
    "    return my_enroll_data\n",
    "\n",
    "def get_one_enroll_type_all_interval(df, program_type):\n",
    "    my_sub = df.loc[df['enroll type'] == program_type,'user_id']\n",
    "    my_enroll_data =
df[df['user_id'].isin(my_sub)].sort_values(['user_id', 'test_date'])\n",
    "    df_type = get_interval(my_enroll_data)\n",
    "    return df_type\n",
    "\n",
    "def get_2_time_points(df, interval_length):\n",
    "    df_2_time_points = df.copy(deep=True)\n",
    "    df_2_time_points.loc[:,'rank_interval'] =
df_2_time_points.groupby('user_id')['cum_interval_int'].transform('rank',
method='first')\n",
    "    df_2_time_points.loc[:,'rank_interval'] =
df_2_time_points['rank_interval'].where(df_2_time_points['cum_interval_in
t']>= interval_length, pd.NA)\n",
    "    df_2_time_points =
df_2_time_points[df_2_time_points['rank_interval'].isna() |
df_2_time_points.index.isin(df_2_time_points[df_2_time_points['rank_inter
val'].notnull()].groupby('user_id')['rank_interval'].idxmin().values)
]\n",
    "    df_2_time_points.loc[:,'rank_interval'] =
df_2_time_points.groupby('user_id')['cum_interval_int'].transform('rank',
method='first')\n",
    "    df_2_time_points.loc[:,'rank_interval'] =
df_2_time_points['rank_interval'].where(df_2_time_points['cum_interval_in
t']< interval_length, pd.NA)\n",
    "    df_2_time_points =
df_2_time_points[df_2_time_points['rank_interval'].isna() |
df_2_time_points.index.isin(df_2_time_points[df_2_time_points['rank_inter
val'].notnull()].groupby('user_id')['rank_interval'].idxmin().values)
]\n",
    "\n",
    "    return df_2_time_points\n",
    "\n",
    "def make_df_2_time_points(df, enroll_type1, enroll_type2, interval1,
interval2):\n",
    "\n",
    "    df_grp1_2_time_points = get_2_time_points(df[df['enroll
type'].isin(enroll_type1)], interval1)\n",
    "\n",
```

```
    "    df_grp2_2_time_points = get_2_time_points(df[df['enroll
type'].isin(enroll_type2) &
(~df['user_id'].isin(df_grp1_2_time_points['user_id'].unique()))],
interval2)\n",
    "    df_2 = pd.concat([df_grp1_2_time_points,
df_grp2_2_time_points])\n",
    "    df_2['time_point'] =
df_2['cum_interval_int'].mask(df_2['cum_interval_int'] != 0, 1)\n",
    "    df_2 = df_2.drop(columns=['interval', 'interval_int',
'rank_interval', 'batch_name', 'report','client_id', 'trial_name',
'age_at_test'])\n",
    "    return df_2\n",
    "\n",
    "def make_jitterplot(df, program_type):\n",
    "    zscore = lambda x: (x - np.nanmean(x)) / np.nanstd(x)\n",
    "    df_sns = df.copy(deep=True)\n",
    "    df_sns[['RT_final_score', 'SS_avg_score',
'DT_final_score','FM_final_score', 'TS_max_score', 'ML_max_score']] =
df_sns[['RT_final_score', 'SS_avg_score',
'DT_final_score','FM_final_score', 'TS_max_score',
'ML_max_score']].apply(zscore)\n",
    "    df_sns = df_sns.drop(columns = ['birthdate', 'gender',
'opp_num', 'cum_interval_int']).melt(id_vars =['user_id', 'time_point',
'trt_grp'],var_name='test_name', value_name='score')\n",
    "    df_sns['time_point'] =
df_sns['time_point'].astype('string').replace({'0':'pre','1':'post'})\n",
    "    g = sns.catplot(data=df_sns, x='time_point', y='score',
hue=\"trt_grp\", col=\"test_name\", aspect=.5, alpha=.20)\n",
    "    g.set_titles(\"{col_name}\")\n",
    "    g.set_axis_labels('', '')\n",
    "    num_ctrl = df_sns.loc[df_sns['trt_grp']== '30-90 days',
'user_id'].nunique()\n",
    "    num_BB = df_sns.loc[df_sns['trt_grp']== '90 days or more',
'user_id'].nunique()\n",
    "    g.fig.suptitle(f'{program_type} Program, {num_ctrl} in CTRL,
{num_BB} in BB', y=1.06)\n",
    "    g.legend.remove()\n",
    "    g.fig.legend(handles=g.legend.legendHandles, loc=7)\n",
    "\n",
    "# remove outliers that are 4 standard deviations and beyond for the
first time point \n",
    "def outlier_removal(df, my_score_names):\n",
    "  \n",
    "    my_stats = df[my_score_names].describe()\n",
    "    outliers = (\n",
    "        abs(df[my_score_names] - my_stats.loc[\"mean\", :]) > 4 *
my_stats.loc[\"std\", :]\n",
    "    )\n",
    "    return df[my_score_names].mask(outliers,np.nan)\n",
    "\n",
    "def get_test_df(df, score_name):\n",
    "    df_one_test = df[['user_id', score_name, 'trt_grp',
'time_point', 'cum_interval_int']].copy(deep=True)\n",
```

```
    "    df_one_test = df_one_test.pivot(index=['user_id',  'trt_grp'],
columns=['time_point'], values=['cum_interval_int', score_name])
#.dropna()\n",
    "    df_one_test = df_one_test.astype('float')\n",
    "    df_one_test['diff_score'] =
df_one_test.xs((score_name,1),axis=1) -
df_one_test.xs((score_name,0),axis=1)\n",
    "    df_one_test['baseline_score'] =
df_one_test.xs((score_name,0),axis=1)\n",
    "    df_one_test = df_one_test.reset_index()\n",
    "    #drop(columns=['cum_interval_int', score_name, 'user_id'])\n",
    "    return df_one_test\n",
    "\n",
    "zscore = lambda x: (x - np.nanmean(x)) / np.nanstd(x)\n",
    "test_score_names = ['SS_max_score', 'SS_avg_score','DT_final_score',
'ML_max_score', 'RT_final_score', 'FM_final_score', 'TS_max_score']\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "def get_test_df(df, score_name):\n",
    "    df_one_test = df[['user_id', score_name, 'trt_grp',
'time_point', 'cum_interval_int']].copy(deep=True)\n",
    "    df_one_test = df_one_test.pivot(index=['user_id',  'trt_grp'],
columns=['time_point'], values=['cum_interval_int', score_name])
#.dropna()\n",
    "    df_one_test = df_one_test.astype('float')\n",
    "    df_one_test['diff_score'] =
df_one_test.xs((score_name,1),axis=1) -
df_one_test.xs((score_name,0),axis=1)\n",
    "    df_one_test['baseline_score'] =
df_one_test.xs((score_name,0),axis=1)\n",
    "    df_one_test = df_one_test.reset_index()\n",
    "    #drop(columns=['cum_interval_int', score_name, 'user_id'])\n",
    "    return df_one_test\n",
    "\n",
    "def run_ancova(test_df):\n",
    "    model = ols('diff_score ~ trt_grp + baseline_score',
data=test_df).fit()\n",
    "    return model\n",
    "\n",
    "def run_ind_ttest(df_ttest, score_name, ctrl_grp, trt_grp):\n",
    "    res = stats.ttest_ind(\n",
    "        df_ttest.loc[df_ttest['trt_grp'] == ctrl_grp, score_name],
df_ttest.loc[df_ttest['trt_grp']==trt_grp, score_name], equal_var=False,
nan_policy='omit'\n",
    "    )\n",
    "    return res\n",
```

```
    "\n",
    "def run_ind_ttest_two_tail(df_ttest, score_name, ctrl_grp,
trt_grp):\n",
    "    res = stats.ttest_ind(\n",
    "        df_ttest.loc[df_ttest['trt_grp'] == ctrl_grp, score_name],
df_ttest.loc[df_ttest['trt_grp']==trt_grp, score_name], equal_var=False,
nan_policy='omit', alternative='two-sided'\n",
    "    )\n",
    "    return res\n",
    "    \n",
    "def run_paired_ttest(df_ttest, score_name, ctrl_grp, trt_grp):\n",
    "    res_trt = stats.ttest_rel(\n",
    "                df_ttest.loc[df_ttest['trt_grp'] ==trt_grp,
(score_name,0)], df_ttest.loc[df_ttest['trt_grp'] ==trt_grp,
(score_name,1)], nan_policy=\"omit\", alternative='less'\n",
    "            )\n",
    "    res_ctrl = stats.ttest_rel(\n",
    "                df_ttest.loc[df_ttest['trt_grp'] == ctrl_grp,
(score_name,0)], df_ttest.loc[df_ttest['trt_grp'] == ctrl_grp,
(score_name,1)], nan_policy=\"omit\", alternative='less'\n",
    "            )\n",
    "    return res_ctrl, res_trt\n",
    "\n",
    "def cohend(d1, d2):\n",
    " # calculate the size of samples\n",
    " n1, n2 = len(d1), len(d2)\n",
    " # calculate the variance of the samples\n",
    " s1, s2 = var(d1, ddof=1), var(d2, ddof=1)\n",
    " # calculate the pooled standard deviation\n",
    " s = sqrt(((n1 - 1) * s1 + (n2 - 1) * s2) / (n1 + n2 - 2))\n",
    " # calculate the means of the samples\n",
    " u1, u2 = mean(d1), mean(d2)\n",
    " # calculate the effect size\n",
    " return (u1 - u2) / s"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "df_comb[test_score_names] =
outlier_removal(df_comb[test_score_names], test_score_names)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {
    "metadata": {}
   },
```

```
   "outputs": [
    {
     "data": {
      "text/plain": [
       "enroll type  trt_grp          time_point\n",
       "Child CTRL   only once        0                0\n",
       "                             1                0\n",
       "             <=30 days apart  0              118\n",
       "                             1              118\n",
       "             >=90 days apart  0                0\n",
       "                             1                0\n",
       "Virtual      only once        0                0\n",
       "                             1                0\n",
       "             <=30 days apart  0               10\n",
       "                             1               10\n",
       "             >=90 days apart  0              141\n",
       "                             1              141\n",
       "Virtual New  only once        0                0\n",
       "                             1                0\n",
       "             <=30 days apart  0               65\n",
       "                             1               65\n",
       "             >=90 days apart  0              169\n",
       "                             1              169\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 17,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "my_ctrl = ['<=30 days apart', '<=60 days apart', '61-89 days apart',
'>=90 days apart']\n",
    "bb_virtual = get_one_enroll_type(df_comb, ['Virtual New',
'Virtual'], ['<=30 days apart', '>=90 days apart'])\n",
    "ctrl_virtual = get_one_enroll_type(df_comb, ['Child CTRL'],
my_ctrl)\n",
    "df_virtual = pd.concat([bb_virtual, ctrl_virtual])\n",
    "df_virtual_2_time_points = make_df_2_time_points(df_virtual,
['Virtual New', 'Child CTRL', 'Virtual'], ['Virtual New', 'Virtual'], 1,
90).drop_duplicates()\n",
    "#df_virtual_2_time_points.groupby(['enroll type', 'trt_grp',
'time_point'])['user_id'].nunique()\n",
    "df_virtual_2_time_points['trt_grp'].replace({'<=60 days apart':'<=30
days apart', '61-89 days apart':'<=30 days apart'}, inplace=True)\n",
    "df_virtual_2_time_points.loc[(df_virtual_2_time_points['enroll
type']=='Child CTRL') & (df_virtual_2_time_points['trt_grp']=='>=90 days
apart'), 'trt_grp'] = '<=30 days apart'\n",
    "df_virtual_2_time_points.groupby(['enroll type', 'trt_grp',
'time_point'])['user_id'].nunique()"
   ]
  },
  {
```

```
   "cell_type": "code",
   "execution_count": 18,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "TtestResult(statistic=-0.6521801779063109,
pvalue=0.514653166038064, df=406.91554534658695)"
      ]
     },
     "execution_count": 18,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "def demo_ttest(df_ttest, score_name, ctrl_grp, trt_grp):\n",
    "\n",
    "    res = stats.ttest_ind(\n",
    "                df_ttest.loc[df_ttest['trt_grp'] == ctrl_grp,
score_name], df_ttest.loc[df_ttest['trt_grp'] == trt_grp, score_name],
nan_policy=\"omit\", equal_var=False,\n",
    "            )\n",
    "    return res\n",
    "\n",
    "\n",
    "df_virtual_2_time_points['age_at'] =
pd.to_datetime(df_virtual_2_time_points['test_date']) -
pd.to_datetime(df_virtual_2_time_points['birthdate'])\n",
    "df_virtual_2_time_points['age_at'] =
df_virtual_2_time_points['age_at'].dt.days\n",
    "\n",
    "demo_virtual =
df_virtual_2_time_points[df_virtual_2_time_points['time_point']==0]\n",
    "\n",
    "demo_ttest(demo_virtual, 'age_at', '<=30 days apart', '>=90 days
apart')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 19,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "Text(0.5, 1.06, 'At-home program')"
      ]
```

},
"execution_count": 19,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAjMAAAJ+CAYAAABGlIFwAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIW
XMAAA9hAAAPYQGoP6dpAABo7ElEQVR4nO3deVxVdeL/8fddBERFkQDTLAlDcyE3irRxN9NyJq
Kmcfspo41lbpNGljVJpdZUpmVmCTbWVJppln4ry9bZBJaBQtzS2z1EpHMUQJZYl7z+8P4+ZNVM6
9l+XA6/l4+FDOOZ/lfLgffHNWm2EYhgAAACzKXtUdAAAA8AdhBgAAWBphBgAAWBphBgAAWBph
BgAAWBphBgAAWBphBgAAWBphBgAAWBphBgAAWBphBjiL0aNHq1WrVrVrrtttvOuE1+fr5eeeeWVc
tX3448/qlWrVho7dmyguggAtR5hBjiDw4cP67PPPPPlNoaKg++eQTZWdnl7ndddcoyVLllRy7w
AApQgzwBmsWrVKLpdLo0aNUklJiV5//fUytzzt8+HAl9wwAcCrCDHAGK1euVMOGTDTVq1CjVr19
fr732mmngvKwBUP4QZoAw7d+7U119rr/rcTERNWtW1e9e9e/fW999////r//+97+ebf773/+qVatWnu1b
tWqlp556qtxtfPzxx/rzn/+s+Ph4h4XXXnlZo6dapycnJO2+6LL77QrbfeqoSEBLVv316DBg3So
kWLVFJS4rVd7961NXLkSG4Ux44ddcUVV+gf//fW999/r//9+97+ebf773/+qVatWnu1b
tWqlp556qtxtfPzxx/rzn/+s+Ph4h4XXnllZo6dapycnJO2+6LL77QrbfeqoSEBLVv316DBg3So
kWLVFJS4rVd7969NXLkSG3fvl0jR45Ux44ddcUVV+gf//iHCgoKdOjQIQU2aNEmdO3fWlVdeqS
lTpujIkSOntZeZmamamUlBR17txZHTp00ZQVWrVtq5c6ceeOABJSYmqnP
nzho5cqQ2btxY5rbr1q3TDTfcoMfbfcoHbt4sRp
7W7btk1jxozRFVdcoc6dO+vvf/+/+7Dh06pDZt2mjq1Kme7YYPH67evXvXrgw8+UI8ePXT
sSJnvWvv/66hg8froSEBLVr105r105XXXWVJk+erO+//////X6///9
dJWrZsmWebZs2aady4cZKk8847T87T+PGjdPll19e9ervo3bdqksWPH6rzzzttPQoUN1/vnn6/XXX9f
o0aO9jv689dZdGjp0qDIzM9W9e3fdfPPNKikp0SOPPKKxY8fK5XJ51fvjjjq9z9q6NChstvvt+st
/qLIyEgtXbpUd911lwYPHqwDBw7oz3/+sy666CKtWrVK9913n1f5ZcuWUlRbt27dLAgQP1l
7/8RTk5OZ4caIWLFhQ7vG75557tGrVKg0FcOB9+/bVF198oREjRujTTz89eSQGL8Oweffbds777H
Dhw3X55ZerXr16ysrKUnJycbyv23ZrRJQ4cO1Weffaa
rrrpKN954ozZt2qMBgweXeSQkNzdwkodP1iW0X66mkCR1dJFkvTwx9r6tSpysvLU1JSkoYYO
HaqoCitXr1aw4YNU2FhFMQu8ZuBmQgGL//N8lJSUGN26dTM6depkF
jh69KjX9nFxcccyYf//jHctw8/GHCxcJyuVKrzaTkMzp
07G126dDF27Njh2baoqMi49dZbTYvvsKiWJk7xK
OmDBhguF2uz37n8fPyeKEYRiG8b//22/G9/86evuSp
xq5du866f08++aRAFxdndOrUydqMazPr/z33HPG0+Z9vWa0ctqvGOQ
jH69u1rtG3b1li3bpN7GbSZFYxB4E
hx9dVXG3FxccZd93lWT5s2DC3rm4Rpt2aL/2DCv8q8bxcK/G
kBv6dvW6fDhw7r66xVHBwsSXI6nRiYBIBDSMmRcn
/I1fkj744AP9/PP/n//7pdevWnm2u1yNPNXX6R3rcYl/Hc53bYWEISS
kqKbDabZ3/atm0rSTpw4IAk6c0331RxcbGH/Ng6Wu5L7
ThhRd6vo6o6Pj9egQYP/0ww8/6Msvr/Tat/fe6o1/////Xdhtp65du3sW2i+8Z
yRQ0b1SKFStkGIa2bdumXbt26d2rrr1WnTp82zZs2NBz1Kws
4XB4rbviiiskcz4DDAclZ1B4Dq5o033pD026mlUoMGDdJLL72kcz
522LDw8XJI8p8xTU6Swdq1qyZmZmSt2wbq9qk9ddvnCQJ06dwq7oa3qVzk6t12EfQp9SzHkcIuRi
fy0qBWXFwsSdq6dpsk6bPPPtOuXsv9n1tRV7Hn3tdu31lrNMt7du314oVK7Rz5s/Z
tStWvX1+tWrXS+vXrdeDAAU+fO3Tqp2p4aab3/t9m+Hh4Ro0aP9Vj/9V
7typzz//XJLkdru6gCgCDnCI/P19r166VJ31r381r166y
2r9+vVeyxo0aDw1W3zpf25lMX69ziM/P1+SVK9evTK4eK4yb0T+1VcXKyQKqK0v79+CJ3Q
VJEVERJxxxu7KUNU7vvvuuHn/8ce3du1fr1fSyXFv27atr30z33l
R4WFhYqPj9el562vo9e/Zo/fr1eu2118DFrnuiP9+ZZtLWxWrJlXmhP0hBzpcpof15eXlK
SQk5Kz/qZpRelRh7dq1at68uV/f9ekYae/+S7WSR
UdHa1HH31UHTt21/u1UHT21AUXXBCCbzaaMjIyy1wGZVWH0zQKZoBTlJ5+eeeexr9SxY8
eralTp3oCwKnGvb3b3vd4cOHtXv1ft7l9K5dk1hOsa0lX/bK5NdU/fC0mK9/nMXDj6Q3nb1e2ha
mb1792rp0qVKSEhQ7969z9aar05vvwXxpFgIJnG7SrmkQH0e6KVQgQzNk5UzW8Ur/SROwx6jKCb7Qe

MLDwz3XomzZskU33njjaX0or9WrV8vtdmv69Onq2bOn17rdu3dLEs8YAqoxLgAGfrV//35t2L
BBzZs3LzPISCcv3E1ISNDx48f19ttvSzp5gefvn/kSCH369FH9+vX1yiuveF2rUlxcrAcfffFA
ul0vXX399wNr74x//KIfDoTlz5uinn37yLC8pKdGDDz6oRYsWlftZKYsWLfI6srJ582a98cYb
at269TkDWKdOndS8eXOtWbPG64iI2+3WP//5Tx09elR//OMfZbfb1bFjR1188cV68803tW3bN
s+2eXl5mjt3bnl33XN069T9lk4+c2fVqlWSVCHfYwCBwZEZ4FdvvPGGDMM47cLf30tOTb69e
u1bNkyJScnKzo6Wt99953S0tL0hz/8oVxHLsqjQYMGeuihhhzR58mTdfPPPN6tu3r8r8LDw7Vu3Tp
999136t69u4YMGRKQtqSTFyWnpqZq1qxZuvbaa9WnTx+FhYXpk08+0PTx0xbffn1Nj//+
/LOSkpLUr18/5efna82aNQOD5ZDDz10zrIOh0MPP/ywRo8erb/97W/q3bu3mjVrpg0bbNmjbt
m1q06aNJkyYIEmy2Wx64IEHlJKSoiFDhujqq69WgwYN9OGHH6qgoECSvO6UOpOBwfq+eefV1
pamjZs2KDIyEh9/fXX+vTTTxUeHq6cnBwdPXq0XPsXsOoPJxZAb4Vekt14MGDTrrdv3791e9evX
0xRdfaPfu3frHP/6hZs2aadmyZXr//fcD2qcBAwbo3//+a3ddWrZ2gp9/PHHHDqo6cx4kWDQPVEHjPHS1V7gMbqnmdJaPW1
WLDgtNuI/TVy5Eg999xwt26tdasWaOlS5fK4XDorrvu0rx5S+SnTp1y1f0OPf/xDffv21f/9
//pP//5j3r16qW1S5eqffvWaBWeIqk3Fo/lf0/9Xvz/S56tc/SnXdVu2TFdffbWysEL08ssvvJ06zTZ9
/pP//5j3r16qWlS5eqffv25SrfpUsXLVu2TFdffbWysrL08ssvq6CgQ0PHj9crr7zidXovIS
FBL7zwgjp06KC1a9fqjTfeUOfOnT1HZurWrO9i699FI999xzateuNdauXi7XXttazXVbt0aMVFBTkWT98+HTx79e77
MmKA33nhDdrtdH3/8cbn6DqDy2QxOBAMBIkeKeeekr53F9wVA1eADkRgMgEbqDrHRL7Oo8CW8DQuMR+Q6Kg
R0sbUIi7K2bRj2QXDyV15UpPTdOU5NwKDWrtPvS89cnR4E9FDxd5u9lqxOY7zzRTFgU0dAz
OEsfnxOD+vABHHvH1r9YJ/MLjX1oy+XXjM8E3y0ur5eBbsVe9fvdFbr43/XRs5R/+KQYQm7Il03HvAXftQOJLWP3PWt1ZnxM1RtzLJ
zVrrsDbrLo+xbiDdbxJpe8adyS1VrrryrXZh6QnBxBb1RMt3bOV5/gYQPVwc1GuBgf9+vWPmwmDGu
t1d1GfMYhZqptTpLoqg1Dl2vE1RmtvVt/XJ5v0eDjF1vKE
TZ6Mn6b3+XcYt92XvPb3S7Lq8x/3sJpgbkgAr6VXL0xkHe/hN3dyKMaV9+Tof9B/oPt5DJv
frbQ6Fed/Y68hHP6CfsK8BbN/CRR93cx8yYu5HQvn5dzxx/bCcQoI96SLTG0/CMqQJzYvWmi+
eE0J1OhUhXOSnDcZfxQq3KcoHg0a3BO/GO7nH3EkLwNc0jx9jNGLF5tGO7d8U+ijH/I1nDO/GDC
cqf+RQ/UDN1x/DLUSF5IT0dWPqlMF6pYT2+0W2dhhG4uOJFJRXQeQS3sxK39P6PDFjdSVb0M4/D
QhUyjRf8mpQC6yrg/mv/3Yt3lH56BdHuORHkNqdO/IBhXU/dIU7mkoF1j7oM7VLg8GL6dY6B7D
a6CpAr8Oxu4g1m/gCaeVnUBBcsMSTJbPIl1WZe0JnNb3dbtbsgdwEhPD8qHD77pz+nzcG6/VuF
3tq7DNxOC3TkPfJq7+Xi2Hpg2jFGDlXjqOh5Ml6eu3SrqR/eLx4ROeaJ5F9srY+Ln0fX7Dr6i+
8ivgG5qKw/Q8drvSQ7wMy/+qCJ+gV0Y9cMx1WY8IlmH1H7QPG4iN2pCPLgjBZkJ5uYj+ySwRfB
sE1Y1hVHnMwpHMs8CMjAhM6JN9CYBZ2PVGjJUtWsJC2bMVW4TBz/hh0WBAAAsdUNH7wLdqMALm2
IyBWgTA/8XW9ZdEXjVL3qH2ljKiXqPEMsjJTLr2q+y4ePaYhZSo+A0RMadafpY5OVk2V23ZN0W0
vTtxFjtpwL5exWyQF/MDUhB5bGXLAQ0PyB7eRWkRa/z1XhNJt+tb/CYa5F2kFbTePmwGMrBPbTZ
wk7FlqTi7jMtqZ6Hpkq8VgVNZ0X5wLzlPVuXsQbZHjtLu8kq3fLBlUuBQsi9vJ8k6h0jdKxeFDM
p0nDUEFq0+Vi5DJgZXB2GWHwQPVtwjDm/1V01d3zCh0B0RGGBe3TLFzvXaYdAfgBZYAP2XqGHVz
hJOAFtq1+dvLbfe2vqW2T9FtEu5aQGrzV3w7AAaqZcXbA5u0ajlhmcJbEM9x8pXWPfiNbShkAQ
ZWxAFXEJdsUm0XgzV3cGmhmbA6GbMPSbGoB1zWr3Ajpt5gB2/GAgoQI6mcwJMoU9gJlfYMZbPAg
MLDwz3XomzZskU33njjaX0or9WrV8vtdmv69Onq2bOn17rdu3dLEs8YAqoxLgAGfrV//35t2L

/OL+quwFUGS4ABgAAlkaYAQAAlkaYAQAAlkaYAQAAlkaYAQAAluZTmFm5cqUGDhyo9u3b69pr
r9Xbb7/tWbdjxw4NGzZMHTp0UM+ePZWRkRGwzgIAAPye6TDzxhtv6J577tHNN9+s1atXa+DAg
brjjjv0xRdfKDc3VykpKWrRooWWL1+u8ePHa+7cuVq+fHlF9B0AAMDcc2YMw9DcuXM1YsQIjR
gxQpJ0++23a9OmTVq/fr3Wr1+voKAgTZ8+XU6nU7sb5b58WLlyo5OTkCtkBAABQu5k6MvP
dd99p//79GjRokNfyjIwMDRkzRllZWUpISJDDT+VtGSkxM1J49e5STkxOYHgMAAJzCVJjZu3ev
JOnEiRMaNWqUrrzySyStl000364IMPJEkHDx5UkyZNvMpERUVJkg4cOBCA7gIAAHgzdZopP//k4
7LvuusujRs3TlOmTNGaNWs0duxYF7510VuxNVw6H3etvlA
/jZh5jZh7z/8xK+2yz2WSzmXtzdun2vuy3lcesKjFuFcdUmKlTp44kadSoUUpKSpIkXXrppdq
+fbuef/55hYSEqLi42KtMaYgJDQ31qYN2u03h4fV8KmtWWFjdSmmnpHczGPMyof5Xz4Oh11O
p8N0Gcm//bbymFUlxi3wTIWZ0lNIcXFxXstbtmypz76SM2aNVN2drbXutjLvr6z22oL
8/c22DNcjjsCgurq7y8Arlc7gptqyZh3MyrLWMWqADC/D+70r67XG6VlLhMlS3dV1/228pjVp
Vqy7hV1i8gpzIVZtq0aaN69epp8+bN6tKli2f5rl27dOGFF6pTp05asmSJXC6X6XHI6TvyVkZmY
qJiZGEREPneypKRyvuknfyDU3A9YRWHczGPMyo/5f26GYcgwDNNlJP/228pjVpUYt8AzdeIu
JCREo0eP1tNPP63Vq1fr+++/+/1zPPNYpJSVFycnJys/rhvFpgxY
yqq/wAAoJYzdWRGksaOHau6devqiSee0KFDh9RxhOH6devqiSee0KFDhxQbMBUJKSqSI
Wamuq5vgYAACDQTIcZSUpJSVFKSkqZ6+Lj47V06VLIkQAAFBe3B8GAAAsjTADAAAsjTADAAAs
sjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAs
jTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsj
TADAAAsjTADAAAszXSY2b9/v1q1anlkmSduzYoWHDhqlDhw7q2bOnUwdmpAACAUk
6zBb7++msFBwdr7dq1stlsnuUNGjRbm6uUlJS1LdvX6WlpenLL79UWlqaGjVqpOTk5IB2HAA
AQPIhzOzatUxMTGKioqqtGyRYsWKSgoSDNmzFDAgQNq0aKF9u7dq1deeUVhISTNX/UJ1wIwkf/jQPYJzOzN1vqTMyVQxuEVnH9GfKX0/4a+6opojzArLlmWuy8rKUkmi5XNWwbCVc1EQYCRlXYUVpvsHgKgAAyA4mwkX2P70BSTqaQBuy8P5d3N7iUqDPkfWuDPqI2KqrIkCgpzO3zJSYmKi9uzZo5ycHz+UVzmfZZlRa4cDtYuw4XqGvq7Zb2BtTbGL6cPAJYEeyfLmjHA9MfIGbSUMsq1llPTj6sYxKs6TTBgnCbrn/9nfDXV70c61rfq9vRgvYMs06kVyWFhsAc/vhiswgaZbkx+7Uj9l8Mywqk2
3bKxyi55//nkVFhXYqKCjIq0YaqvAjHMQEbNLJs2Nm2lxf/8ERJtlh5gBgc4ZbNzVV5ntRFr2tl
8L1FPvRCq4I8P3fx1Tc+y7WY
OBx2OZ0O02Uk3/fb7TZUv36IT2VLy9vt5gJYTWHlhzQEiIiouLvcrqUhpjUTtz4/ZHQyXDxfuq8Xn
bfKyMhQSEiIiouLvcrqUhpjUTtz4/ZHQyXDxfuq8Xn
bfKyMhQSEiIiouLvcrqUhpjUTtz4/ZHQyXDxfuq8Xn

mNjtW/fPi1cuJAwAwAAPGbOnBnQ+kyHmQ0bNmjp0qVauXKlevbs6VmelZWlhIQEOZ2/VZmYmK
hnn31WOTk5ps9/AQCAmi0zM1Off/65cnNzFRYWpi5duqhHjx6y2czd928qzOTl5Sk1NVX33nu
vzj//fK91Bw8eVFxcnNeyqKgoSdKBAwf8CjNOZ8W+D7P09kBfbxOsrRg38xgz85j/Z1baZ5vN
ZvqHv2d7m0yXPfX5MqbL+tl26fZ16jj8urX796c4KoOVP2uBVlRUpNtvv13r1q2Tw+FQeHi4j
h49qvT0dLVv314ZGRkKCwsrd32mwsz06dPVoUMHDRo06LR1hYX70ATLWFXXKjg42NNpX9ntNoWH1z
v3hgHAkzB9w7iZx5iVD/O/fBwOu5xOh7kydrvnb9Nlbb6X9bfthg2C5XYbql8/xHS7pdxuo0p
fyWDlz1qgPProo/rqq6/05JNPqm/fv+WrFmzNGvWrHLXV+4ws3LSmVlVZWnV
qlVlrg8JCVFxcbHXstDBeDRo06LR1hYWFp70YKjg42NNpX9ntNoWH1z
v3hgHAkzB9w7iZx5iVD/O/fBwOu5xOh7kydrvnb9Nlbb6X9bfthg2C5XYbql8/xHS7pdxuo0p

RRRdpwIABGjhwoC6++GJT9RFmAABApatfv76SkpKUlJSk3NxcvfPOO1q9erXmz5+vVq1aacCA
Abr22mvVrFmzc9bFLSEAAKBKhYeHa/DgwXrppZf0/vvv67rrrtPbb7+tPn366M9//vM5y3NkB
gAAVKpLL730nA/Ns9lsstls2rJlyznrI8wAAIBKNW7cOFOvMzgXwgwAAKhUt99+e0Dr45oZAA
BQ6V5//XWlpaVp1apVnmU7d+5UYWGh6bo4MgMAACrVvHnz9PTTT6tBgwZ65ZVX9NNPPyklJUU
PPfSQDhw4oBdffLFcdzGV4sgMAACoVMuXL9fIkSO1fv163XnnnnXruedkGIYefvhhRUREaO7c
uabqI8wAAIBKlZub63kb9g033KCjR4/qxx9/1AUXXKBbb71VmZmZpuojzAAAgEoVHR2tQ4cOS
Tr5jJmIiAjt27dPktSgQQPPl5uaaqo8zfffCNJ2rhxo8
LCwkzVxwXAAACgUrVs2VIvvvviirrnmGnXu3FkHDx7UkiVLtH79en3yySe67rrrTNVHmAEAAJX
qH//4hxo2bKg6deroq6++knTyib979+5VcnKy7rjjDlP1EWYAAEClWr16tWjYwNWH9fMAACA
ShXIICOZPDJz9OhRzZ49Wx999WJHy8/PVqlUrTZ48WV26dJEk7dixQzNmzNDWrVrVvVqFEjDR8+X
KNGjQpohwEAgLX16dPH1LuZPvjgg7OuxjrjjDuNxk5Gj27Nlq3LixXn75ZY0aNUorVqxQ48
aNlZKSor59+yotLU1ffvml0tLS1KhRIyUnJ5tpptax222y220+lXU4/D+45msdvva5JvB1zAL
x/QIAq7viiiuq5kWT+/bt07p16/TKK6+oU6dOkqpRp06bpk08+0erVqxUSEqKgoCBNnz5dTqdT
sbGx2rdvnxYuXEiYOQu73aZG4aFy2FDhG0aFy2Cv/P7kGoXXXkdhsKC6tb6W1bVSDGzO02ZLPV3iAIADNnz
gxofeUOM+Hh4XruuefUrl07zzKbzbSbDMHTs2DFt3bpVCQkJcjp/qzIxMVHHPPvuscnJyFBERED
CO1xR2u00Ou10vv7ND2UdOmC7fqkVjDega49N/jiHBTtntNr2yZqcO5Rw3Xb51i8a6pmuMVIv
+X/Z3zKIj6mlw/9a1+qgWAEiSYRj6+OOPtX79ev38889Xr1s10XeUOM2FhYerR
o4fXsrffflvff/+9rrrqKj3xxBOKi4vzWh8VFSVJOnDggF9hxums2KMWpYf+q+IUGmbh3MLd
OAn8/85RjUOPfkPm0wHmtLts3NP+NZ2ROhpdZlt25d++1s+EGV9HTPbryHGbrdV+Oe6pqu87
+0nM1mqzXz4NRfYCzV7wC17e9nhdPNUmFhoW655RZlRZlZWUpNDRUx48fl91uV6uoqzNz9
fQUFB5a7P51uzN27cqHuvuuUd9+vRR7969NWvWrNMaDg4Oli6QVFRX52ozsdpvCw+v5XN6Mqjzd
4nDY5XQ6zJf79fSUw26+vD9lJcqhq8K2q2i/A9Xv+vVDTJetaw0/32Zw5adB37Mfb/brsox+
zWE+PtZ4dS+NHfuXO3du1fLTLbN9W7fXXXTxbxvZm4fhjTQz/L1+vDYjzw5Z5fXFc9wi+i0+yUkNbUfhnla9tXlzgcnw1/eqMyAUe9R
spU6bosssu0+zZsyVJISEhi4u9tquNMSEhoaeVkd5ud2G8vLMn34xw+GwKysysrvLyCuRyu
0rTO17XK5VVLiM13e5XZ7/jZb3p+ykuQyDrtKtrvQPU7P79Qv/xivrxVBCqAWGH++zOHLTsP
/Jj7frddlWP26+fD389KVfxfU5nKM//feecdjR07Vu3atdo2bdsduSUFBQbrmmmUnZ2tRYsWV
WyY+fe//60ZM2aoaXHY5o0aaLs7GyvZbmUDUMf/jo6ONtuMl5SyvmfwvOKEMwpfvLBLf/x
qy21PGkOny/pQtLXdaXZXUdlXtd6DGzO02quyzZjVWmf++zGGrzgN/5r6/bVeHMMfP3s1KV/9d
UFzk5OWd8lkxsbncJM1WfqxN8TislJCRo48aK
iYnh4l8AAOARHR2tzZs/d0vjRzu7nHuZhzk6xWudQ9jKfSQhrnxw
BUVpbi4ONl9eIgsb83WyWcH+PtUV5fbraO5Jwg0AFBD8QTwwGvXQGUqjXJEkqXv37fkbb+9
WprlGNQzXkmktlt9sIMwBQQ/EE8OK7CMntP9wPvfl3V3AwBQjfn6wVHCoOYSaAfDkfyjl
UAAD8Q5gJAN4+DQBA1SHMh/51H9ees1AAAgzASUL+dRI8M5mgMAgD+4YAagD+4YAAFgaYQYAAFgaYQYAAFgaYQYAAFgaY
YQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaY
QYAAFgaYQYAAFgab80GAEkOh2+/2laDkDgEGYA1Go2m01ut6+6a5L6GwsLpV3RUAPiLMAKjV
abXlmzU4dyjpsu36pFYw3oGiObzVYBvQNHn6Fmfnz5yszM1MvvviiZ9mOHTs0mOHTs0Y8YMbd26VY0
aNdLw4cM1atQovzsKABBUp+8gJ7T+cb7pcZDhHdICq5vPJ3n/961968sknvZl5uYqJSVFXFLVq0
0PLlyzV+/HjNTtXy5cv97/Ha7TU6n3fQfu52fjf2di+sjMoUHNG3aNGFExMTe6
1599VUFBQVp+vTpcjqdioN1b59+7n59uto7kn5H
YbAe6Z9ZkOM9u2bVPDhg31ptv6umnl9b+/fs945RGD8qntfTLHAqQFYdd3JkNnMT9cywf9
B6TUAABYTEuz0+fqsqMahgnMmT7M791bv3b3LXHfw4EHFxcw4EHFxcV5LYfYuKipIkHThwwK8w41T2KCDy19Ayad5bnu
wOcw43RW7K2PnkN3Nvl0EZ+njA/l/Slb1W3rlCK1Zb8DNWalh5lxbsz/ati2H3Pf37Yta/l
pOk7NwTOvCTuTBT2l6dOg6fHwfgdhsyjJoZhAJ6N1NhYaGCgoK8lgUHBwdUHNRBuSioqKfKrTbrcp
ye330rD4fdLqfT4VM5X8v7U7u9bV/q2fb/sx9v9u26JhJ/o1bwwbBfp2
ikk6GmZp63U1Aw0xISIiKi4u9lpWVlRGGmNDQUJ/qdDLsN5eWd8Ltv6pLOnjkP164fI5XarpMRlurzL
7fb8bba8P2WrvG2j9u13oPqdn1+oX34xX94qAhVAmP/Vs21/5r7fbVt0zCT/xi3o1wuAX1mzU
9lHzM+JqMahGty/tfLyCuRyuU2XN6VUDDTJMmTZSdne21PLHlZCI2rrm9cpWwSHhwObVgvscj6zt8xs
H7J/NN/1reZfLXSN/9gT0ZHRCQoI2btwol+u3NoT0uabMzaGm+TJPbYNkZCl+w9ABA4AT0yk5ycrPTPT0
dE2bNk2jR4/Wli1btJKkRNgmpFN+QZmFNjsjlC9jjhpNAUwWgQGu2OtyL+S2ySgQvTuSK0F+enpmjFjIiK9LGFtNrKVkJwGKiRjbBd6VXNSKrUfb

SUFshnAsrg1EwACz68w8/DDD5+2LD4+XkuXLvWnWqDGy8717dH5AIDT8aALAABgaYQZAABgaY
QZAABgaQG9ALiq2e02n25d5XZXAEBt4M+rEKrzjQc1JszY7TY1Cg/1PG4aAACcVPpYiLCwuj6
Vr+6PhahRYcZht+vld3aYftRz6xaNdU3XGK+XpwEAUFPU9MdC1JgwUyr7iPlbXqMa+/beKMBq
fD0VK1X/w8wAzs2X/yOtoMaFGQBl8/dUbHU/zAyg9iLMALWEP6dirXCYGUDtRZgBapmaepgZQ
O1FmAEsxtdbK30tZxU8mgGovQgzgEX4e2tlTcajGYDajTADWIQ/t1ZKUqsWjTWga4xstpp3JI
JHMwC1G2EGsBhfr3mJDK/5R3R4NANQO3FMFgAAWBphBgAAWBphBgAAWBphBgAAWBphBgAAWBp
hBgAAWBphBgAAWBphBgAAWBphBgAAWFrAw4zb7daTTz6pP/zhD7rsssv017/+Vfv27Qt0MwAA
AJIqIMzMnz9fS5Ys0UMPPaSlS5fKZrPPplltuUXFxcaCbAgAACGyYKS4u1qJFizR+/Hj16NFDr
Vu31hNPPKFDhw7pvffeC2RTAAAAkgIcZnbu3Knjx48rMTHRsywsREsywsLExt2rTRhg0bAtkUAACAJM
lmGIYRqMreffddjR8/Xps3b1ZISIhn+cSJE1VYWKhnn33WdJ2GYcjt9r1bdrv/F9UrKPGT8K9f
SuX3JHCLkQxCmyEuK6VNngmPztY7y/kKOy8B7pfrhmyN3pvOBZBiUpJJ/RnQWHGaxnbT8p0adPzpGjZ9RVWTFc4pdZOCs5yWmLx8HzasV5Y5RMrl
1ymvm5VQEkIlBIyO7LdQIYfBhxOwB4AIRdwt9E8zIeSBLAhKKvqZWPNaHYwYAFl6ydXAxWj1s59
Fr8TuPJaeNdGXAR6d7ecPdxgrUwRNdgf/XuTa8Nn9Qv1ZOxvZTPe2mqPmM8Sp3o5jC7NPMsiFo
28aqhPmLXXRtMtjv1zEbA9uUKgH7a5X1HMnm5Xq1OGa3qn5o4HT6N/5p1Yc+kHpTpEWo+hvxXVs
lmGIYRqMreffddjR8/Xps3b1ZISIhn+cSJE1VYWKhnn33WdJ2GYcjtPncXbTbJbrcr/0SxXOX
Y/lR1nHaFhtTxqay/5WnbWm1btd/+lnfYbaofGiS3263y/MRwOALzexLzv3q2bdV+07a15r8Z
zkBWVlBQIEkKCgryWh4cHKxjx475VKfNZpPDYSv39vVDg869UQWpe3a1bZV++1vebu9cn9IM
f+rd9tW7Tdt+6ay578ZAe1Z6dGY31/sW1RUpLp16wayKQAAAEkBDjPnn3++JCk7O9treXZ2tp
o0aRLIpgAAACQFOMy0bt1a9evX13/evbvX3//+1Psry8PG3fvl1dunQJZFMAAACSAnzNTFBQkIYNG6b
HHntMjRs3VrNmzfTooUuDAAAEAgBvTUbAACgslXf+6wAAADKgTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAA
AsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAA
Aszacws3LlSg0cOFDt27fXtddeq7ffftuzbseOHRo2bJg6dOigj17KiMjI2CdBQAA+D3TYea
NN97QPffco5tvvlmrV6/WwIEDdccdd+iLL75Qbm6uUlJS1KJFCy1fvlzjx4/X3LlztXz58oro
OwAAgJxmNjYMQ3PnztWIESM0YsQISdLtt9+uTZs2af369Vq/fr2CgoI0ffp0OZ1OWs5ax2Ccbat++f
Vq4cKGSk5MrZAcAAEDtZurIzHfffffaf9+/dr0KBBBXsszMjI0MzXXVlKSEhQU7eFBNmjTxKhMVFSV
bsUU5OTmB6DAAAcApTYWbv3r2SpPzUVUvTPB1sszUmRjcAeZEFBNmjTxKxHxMVFSV
JOnDgQAC6CwAAAS4M3Uaab8/HxJ0la133aVx4tnWQpQbWjNmjcaOHavnn39hYWWCgoK8ioIHBws
SSoqKvK9k86ven4K86KvcenK4bB7/Y3yYdzMZPwzmZPNSpUeNGrUKCUlJUmSLr30Y
m3fvl3P8Pff+8Qkc7FWmNMSEhob61EpFxXO6GyOqvE8bF7Q4ibqgb61EG73abw8Ho+1Tud9L0
8wboFnKsyUnkKKi4vzWt6yZttyZUt99NFHHatasmbzRz1Sd8KlteDoddYWF
1lZdXIJfLXaFt1SSMm3m1ZcwCFUCY/9UTY+ab2jJulfULyKlMhZk2bdqoXr562x6rx5s7p06
vmvXLl144YrX0IKfLfYJfxDI/nQnzTX4UHtEcTILVmXA4X4lWQ2dhGfzz3wWmJg8cdMDs27bDj
/yvvhgz3zBugWfqxF1ISIhGjx6tp9+Wqx9r9b9b333+vZ5ZRuvVrVNKSoqSk5OVn5+vadbf
fu3VqxYoUWL6sMWPGVFT/T/AQBALWfqyjR07uTTUTZYzPXzQIWpnTh9+eyKSVJ
6erpmzJcihKQkpVGKjU11N9DQAAQKDZMMwqroTZ+NYEKbcPptCs8vJ5yc49z8M8E
xs282jJmkZENAlIP8796Ysx8U1vGLVDz3wzuDwMAAJZGmAmAEBWrYs
JZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJ
ZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJZ
mOszs379frVu3LvHv5ZdfliTt2LFDw4YNU4cOHdTv2vlVq4e5ZRnBowSrpbf5bt6zXYCWY8CGHL
7VrZbDbP8gYNGig3N1cpKSlq1aqVtmzdqHT35Z0Zc9J5+e2YhCv2w8s6z8g8GdD8Lv7+dKo0bQ5
KEk6YcfftCmTZt06NAhhhYeHKzExUc2bNsXLsgm0ZQUJCmT5GZUE8aX8bi9pOJa0bnUeJp0ULoH
Op2NhY7du3TwsXLiTMAABQi+3cuVMfc1f/pezeq/3ilzPUa2IjSWX8u
pLH9EMX2a6euvvlbLLi3LXJeVla2Eh7fSzqzDkl98+Bk55syZosyZt2gnkyCBBJ17VnVg
4okntGrVVk2dOlUfffSmtZoooSmbzB4O++khTpg7VW2+9pSlY2bVrl3JycVampLqEJCKmJBa
wcPln/+8x9J0sGDB9W9kSkSROv7UuP4Bw4cMBsUwAAoIbIYWVSPM6TDgnTHzyUHWfQlimNlKESo
4jRL4l5YYyIaAa3V3fw5v+SnS3lXP7P0xaKxa2A1KErFkfbzu0XcI5fW4EvvjCcUQrySUTTO
ODuhY7NvqX/AyBs8e1N22b9ZTFz/r+OQBNZBJjiS1u2LbA0ecyyy+Nmjv8/3tgxI2E6VkM1+
eUq+p0dubXjVnJy2fG0LTpvlSRSnJtF91FlbhDh23ZjM0ZGkkMlK3Ji8Wrbm8eSGSXMgmmD/BGSm
ExUe+++666det22narV6/W5ZdfqbSumSp6HgTLMKV1kmL4KXNIexq6d2ws3mOaRVDxh+mNmqYGZ6
ut6G8vBM+lS0vh8OusLC6yssrrkMvlPncBSGLcfFbxixQAYT5Xz0xZr6pLeN2pvk/ePBgzZo1
S3l5eWrXp3mz5+vw4cgnXMqPJE+vw4cPq3/OqBiNBDWBP/5fbfvKGrvfwsaOIvVVoc
bNPMas/Jj/1Rdj5pvaOm5DhgzR0aNH9cILLgILL+jYsgVmN9xh7FgxwPbNvyDPmZ9fwOAU5Kg/JYkcThV/fbnrNHhv

m4uLidMkllygtLU3333+/wsPD9eqrr+rLL7/Ua6+9pvPOO0/p6emaNm2aRo8erS1btmjx4sVK
S0sz0wwAAKiBxo4dq7FjxyovL08nTpw47bZsX5kKM3a7XQsWLNBjjz2mSZMmKS8vT23atNHzz
z+vVq1aSZLS09M1Y8YMJSUlKTIyUqmpqUpKSgpIZwEAgPWFhYUpLCxMknT8+HHl5+erYcOGCg
kJ8ak+09fMNG7cWDNnzjzj+vj4eC1dutSnzgAAgNph3bp1mj17trZv3y7DMGS32xUfH6/Jkyc
rISHBVF218/4wAABQZT7//HONGTNGkZGRGjlypGw2m0aPHq2CggKlpKQoKyvLVH2EGQAAUKme
fvppXXPNVqwYIGuu+46GYahiRMn6o033lC3bt00d+5cU/URZgAAQKWaMGGCCvvrqK02cOFGSZLPZlJSUpIEDB6pevXpRo563B5QXYQAAFSqtm3batiGiRV6vOjIMQ/v27dPcu
ccVqYsdvtqlu3rnr16qVu3bqZqo8wAwAAKtXMmTMDWh9hBgAAVInMzFuv/12rVu3Tg6HQ+Hh4Tp69KjS09PVvn17ZWRkKCwsrNz1c
TcTAACoVI8++qi++uorPfnkk/rqq6/0n//9PVvn17ZWRkKCwsrNz1c
TcTAACoVI8++qi++uorPfnkk/rqq6/0n//8R1999ZWee+45HThwQLNmzTJVH2EGAABUqrfeek
uTJk1Sv379vE4pde/eXZXMnT9YHH3xgqj7CDAAAqFRFRFRUnfApw48aNTddHmEGAEevH
FF/XLL794LS8oKNDChQt10003marP5wuA9+zZoxuuEH33XefbrjhBknSjh07Aa2VGEnSjh07NGPGDG3dulWN
GjXS8OHDNWrUKF+bAAAVBoaKiysrLUv39/devWTY0aNdLhw4f18cf6+eff9YFF1ygu++W
9LJh+k9/PDDZ63PpzDzy+/aMQKTpx4oRnWW5urlSJqUtS3b1+lpaXpy+/VFpamho1aqTk5G
RfmgEAADXQ2rVrVrPa8s+OyzzzzzzLQ0NDPUGn1O8frlcWn8LMU089pXr16nkte/vWXEhIS5HT+lo8S
ExO1Z88e5eTk+N9TAA9TAABBQo7jdbu3evVubNm3Sd99995M9o7M5OXl1Vffee6/OP/98RGX
x5UXFyc17KoqChJ0oEDB0y0jdWcyWENhWLD...

Px8TZs2Tbt379aKFSu0ePFijRkzpqL6DwAAagizb8suZSrM2Gw2zZkzR4mJiZo0aZJuuukmHT
t2TC+99JKaNm2qiIgIpaena8+ePUpKStK8efOUmpqqpKQknzoHAABqh+bNm+uf//yn5zKWUuU
JODbDnytuKoHL5daRI8crtA2n067w8HrKzT3OeUwTGDfzasuYRUY2CEg9zP+KU7rfc17eaPqa
mQuiGmji4E7KyytQUVFJBfWw5qktn7VAzX9JOnHihHbu3KlOnTqddTuf72YCAADwx9GjR7V58
2bl5eWpUaNG6tixo+rXr+9ZHxoaes4gIxFmAAABAFZg7d67S09P1yy+/SDp5OikkJES33nqr6W
tteQwhAACoVK+88oWLVqkv//975o9e7ZsNpuee+45JSUlac6cOVq2bJmp+jgyAwAAKtXLL7+
sv/71r/rrX/+qbdu2yTAMde3Xeeedp1KlU2bt5ttHxoaes4gIxFmAAABAFZg7d67S09P1yy+/SDp
Hj16aN++fabqI8wAAIBKFRYWpkOHdpW5bseOHTrvvvvvPNM1cdpJgAAUKm6deumJ598Uv1bt/YsO
3r0qD766CM9+eSSTuuWWW0zVx5EZAABQbf7+97/r+v379Tzu+yryszEXvt6PCm2+pwKdXpl+ktcss
eZqo8jMwAAoFI1adJlWK4N7FSRhWM6XvPQgG7tXbMGOhUCBdJYqb28m/uIbCwsQ5SYFnU7mNhrO4TO
kSVOmTPGrLsIMAMAndrtNTqdvVVyu43bc7mr9Nh2yaNavc9RFmAACmNAitI7fb
UP36IT7X4XK7dTT3BIGmltqwYYN+/2rI/Px85eXlqV69eurQoYKdsttemXNT
h3KMf8i0KjGoRpyzaWy222EmVpq7dq1ZS7fu3evJk6cqP79+5uqjzADAPBJdu4J02/cBs6mRY
sWmjRpkmbOnMkTgAEAgDUVFBQozvbvbVBmOzAAAgEo1b968MpcgWOqPsIMAACsk
oVPPnzz/tAuBSzZs3N3Unk0SYAQAlWz79u2nLSqKtLmzZt13Ac+ePWrRokW56+OaQa
UOWCg4N1+eWXa+LEiZo9e7apsoQZAAAVKrXX3+9zOVHjx5TVKa
uLiTNVHmAEAAJXqnnvuO0eO6li1b6oEHHBGAABUqvfff7/M5Uc
H94x//UPfu3dpUycNHjxWVlZnvU7duzQsGHD1KFDB33
JCdH555/v+dqXICOZDDN33HGH7bQ6/bfKzc1VSKqWrooeXL
l2v8+PGaO3eul9i9f7lPHAABAzfT111+rT58+6tq1qwMYGMGKdDvv+9ekvTII/opZdeMl1fucPMv
n37tG7dOt1///3qLr74Yk2bNk2dFavXq1Xn31VQUFBWn69OmKyRo4cqYULF5
ruFAAAqLlmzJghp9OpadKkSFmwdv9OpadOmKTg4WP/4xz8kSXq1NGMGTO0bNkyU/WVO8yeeeU7t27
LbDabDMPQsWAIIT1QrgW1I9OZwuajaOneMK23W5c7U0e44HttLdjtzxPVBZ11Tm2Gk5iYqD6pq
3//+V/n5+brIECOyYNTyemfhn3B4AOCwsTD169PBa9vzg3m
qIkSQcOHFBERIs11lmxj8NxOxef6N8GDfzGDPzmP8VO3R/bTabbDabucKnllau
uY17b9LktQUJDCwsIkSZdeenqnq1KmjvXq27durcubOebffZZU/X5fDfxc8996hPz
q3bu3Zs2apaCgIK9tgoDJZ18RLGv7HabwsPr6VrejjDu7nHmJUP87/iORx2OZ0O
c2Vsv/6nbDdftrRNqfaOeW3d71O1bNlS//nPf5SYmCjcit9ttqj6fwszat
Ws1ZcoUXXbZZ5HDoeEhKi4uNnhru9IQExoa6kszkiS321UyeUVyOUyN4
C1GeNmXm0Zs0AFEOZ/xSndb5fLrZISl6myLuPkOLnc5stK8oxzbR3zmr7f5Zn/t956q2699VZ
FRESoR48eatWqlTIzM3Xxr4cKFiomJMdWm73/fv302GOpY7//GNGnSRNnZ
2V7bln4dHR1dJOcttvhkvJSWV800/Oalr7gesojBu5jFm5cf8r1iGYzzx7cVnLuRd3pc2pdo75rV1v
081c+ZMlZSU6NFHH9U//+///Pa8/cHAj27eiv0XbtXyKc/GTE92EVPQ4zn1s1RGm8fhVqfIn/+
28X0JCgpYsWSKXyyWH4+Rhx8zMzHMc/wXr5wX9Ss9eWM5/bVEL5GZWGizpXdPfL12T2yWDWVBfnDmSsuyxxnsCKMWCfSjWVt0nHEFWXjcXT9b
yRHXr1vXKEkFBQerUqZOaNdxXVfxuajwxr4VHXnDNEzD5p5gCYFBJvFK
afTo0dqyZYsWL16stLQ0CAOBM7HabGoWHmH3/SJal9uto7knDnCRV6Iknp7zR4vWr
NEvv/yi9957T57XqSkJD388MZBu/XjBkzlHSJUpMjiSKWmpiopKSMgHQYA1F52uu0OU1
0vv7ND2UfMX08VHa7jfUNBuDjv70osmyGIahqG264azblVMlF6sczx8fFaunS
EAMAn2UdOaP/hl3AGDyJI3gIX3yWBTTdeLTB8l0cTSKBfmSK1tAb9VuDwqnFR5f
XEWcmmvatKnn38ePPH1nf+r4aNmyokJAQeNnY3PY7HSVxYWjQKeu/gPEAeKJA9
HhNnjxZCQkJpurimoAgCpx8unD5V7weYMWMUGRmpkfN9AjzvEkSRaafTo0SooKFBSqLN3Xo9+Wtd
qysrJM1cenAgBQqRqE1pGEqnKsOAoffkQW8Eddt4Pr0cTAY9WmcU2GsiFbXEPrCRW2T6UjQY
jjTfUrVs3zZ0711R9HGnCt2G8yu6IHabXlmLxCS87Ry5BLPhv3uBx47O0J5Yl56DQ18MNROO46FyhW1z
c92QIUM0ceJEURmqA6nS7HQCULy/Xb0SGc2SmJqhTp45yrEjBMcUpExltd
jsAAM6lbdu2Wrokb56pXr16m6iPMWFx1u90OAIBzmTBhgkaJEy
dq/PjxstlsLL75YqampL5XDSzppX1el2OOwAAzq2ZjxEcWC2+PlHoz
RkZEaPHiwbrzxRgUFBZmqjzADCBbm6zVtf0WZ1/3n+sHrAnpgRgeqXiD
dG1z6m3hvuA6w+qJMAMAFuXPNO7TVpuuwFPw4+2tR6Uw7t2tUU9TbnH3hz3Wvv7cL
S3HKa7AI8xUMQ4TA4A1+HuKSXmsFeEV+7fVnOvmbo+mpHIl3Gw4aKgtGIS3iqsCEG
aqCV8Pedb2Q8UAUNl8/Xkd1Tj0ZPlcHqURaIQZAPATrxqBhqqZGhw6AEAP/BKEaD
qEWbg922GAKr2Vl/UHnxWykaYqcUCdZshd1IBvgnEHETTwGfl7AgztZi/txlGR9TT4P6tfb7w
Eajt/J2DPJqh9uCzcnZ+hZn58+crMZ7NGPGDNhO0DNWrUKL87i
orj6+HxmjopgMrGoxlQXnxWyubzSbR//etfevLJJ72W5Kiu1RS1atNDy5cs1fvx4zZ07V8
uXL/e7o0B1Ybfbh5HTaffPT089bA0BVMH1k5tChQ5o2bZZ3btyomyomJgomKCNH36d36dDm

dTsXGxmrfvn1auHChkpOTA9ZpoKoE4vUTXGcEAIFlOsxs27ZNDRs21Jtvvqmnn35a+/fv96zL
yspSQkKCnM7fqk1MTNSzzz6rnJwcRUREBKbXQBXx9/UTXGcEAIFnOsz07t1bvXv3LnPdwYMHF
RcX57UsKipKknTgwAGfw4zTWbGH5v19k2kg2rbZbD79tu4pYzN/DYs/ZUvLSb+ddjFd3Ob7U1
Olk0c4DMO3Z3P42nZpmcO5BTrwk/mL8Gy/lvd1zGqj6j7//ZnD/s7BKpv/p2xuqX5Xcdv+jFt
12O/qfJo8oHczFRYWKigoyGtZcHCwJKmoqMinOu12m8LD6/ndt/KoylveHA67nE6H+XK/nu5w
2M2X96fsqeXr1w8xXVY6GUb8DTP+PELen7b9/X75Oma1jZXmvy+fiUDNwcqe/w6bNftd5W37M
W5Vut+/hpjqfFt4QMNMSEiIiouLvZaVhpjQ0FCf6nS7DeXlmT+cb4bDYVdYWF3l5RXI5XJaF
tnatvlcqukxGW6vMvt9vxttrw/ZU8tn59fqF9+MVe+dL9fWbPTp9M1UY1DNbh/a5++Z/603eq
icF3TNaZKxsxKAhVArDD//ZnDgZqDlT3/XYY1+13lbfsxblW637/Oi/LOkcr6BeRUAQ0zTZo0
UXZ2ttey0q+jo6N9rrekpHICxskfRpUbZkoZhm+nTDxlDJku70/Z0nLSyf9wfB23QznHfbrNs
LS//nzPfGn7vEa/HlGpwjGrbawy/32Zw/7OwSqb/6dsbql+V3Hb/oxbddjvqvw/8lwCegIsIS
FBGzdulMv1W+rLzMxUTEwMF//WYL7cqlydz70CAKwloEdmkpOTl6ermnTpmn06NHasPmWLFi9
erLS0tEA2g2qi9PHaXP8BAKhKAQ0zERERSk9P4wZM5SUlKTIyEilpqYqKSkpkM2gmvDn8o1
/dHaAIDK41eYefjhh09bfr6VL/pTLSwmO9f847Vr+qO1AQCVhwsXAACApRFmAACApRFmAACA
ACApQX0AuDazG737fH43KLsH1/GjzFHWfx9nQGAqkOYCYBAvEkZZ5pTeFl6dH7BAHEeKi+fJYnPE1AdEGb
EYWYAAKyMMCMOMwMAYGWEmVNwmBkBkAAOvh9hsAAGBphBkAAGBphBkAAGBp
hBkAAGBphBkAAGBphBkAAGBphBkAAGBphBkAAGBphBkAAGBpvM4AqAJ2u01Op7p7V+l3C7DbndRLV3AwBOQ
5gBKlGD0pyuw3Vrx/iU3TLf073K7dTT3BIEGQLVDmAEAEqUiw0+c3t3ft7
v7Uj6qcaiGXHOp7HYbYQZAtRPwON2uzVv3jwwzZMeXl56ty5s+6Vv3jwtW7ZMeXl56ty5s+6//35ddNFFgW4KsKzsXPN
vaC99O7u/b3f3tTwAVFcBP2k/f/58LVmyRLfffrvvLVy/LLmyRA8/0ygq9DMvvH
n9N2AKqHgB6ZKS4u1Ji3TnkXjYvz7+Rq5eUeKPHoltNQxmfiL9KNJc4eKYSIK
91aF1UDCIyAhpmdO3vh/fq/+eHCJaHtoRkxKCQSK83hAtftmfg9s+Q0eUxyQtLEFc9mu1ZkSSkh7nBhKvPkE4JKAaMBYTLt9brsfAnOxnrlySTu7rlcoSV0F
s01jVdYyQO0ACWZTMMI2BX87377rsaP368Nm/erJCQ3W7erzK0WgArGNCAyaUS0wNC4i0TV6tKxvTkrjMQdA1e9DsMTKQ8+artMwzN00avP
pB5JNdrtN+SeK5fLh4sY6TrtCQ+r4VN6fsrRd+W1btd/+lnfYbaofGiS3263y/MRwOAJzhKS8
899mk+x2u+XG1aptW7W7XftG2t+W9GQI/MFBQUSJKCgoHB+vHB+vYsWM+1Wm32xQeXs/vvpWXwxGA0fJOpzbWKd2ebaRbkrm1DjdKM
869UQWVp21rtW3Vfvtb3l7Jp3HMzn+rjqtV27Zqv2nbN5U9/80IaM9Kj8b8/mLfoqIi1a1bm
BNAQAASApwmDn//PLs7Oz1aRJk0OtA2BQAAICnAYaZl169evX4x2m++/3g42f0Hxzfq27erSmN
27erSpUsgmwIAAJAU4GtgmoKCNDxH4eAHzTrOQ4QKoNJB2n/E9ePEfGwZvAW8JQpcB9RDp13aPQNAJAvpkDMvBBnUYAQEhCLaDSLKYbAvaHKk9eLs96NpD5+YuJ+Tg13wQ++m8N3IzKiUCz8s+v7e8/hUXPWgLlrkWXlIeaEaJLEP4IUUQDxHlVzhIgUaW3sv84tS2SeWABHU5sbUb00sWYu54p7Dy90lYCgyh+0i8NxlDsQdyQyvbzU0JoSeQKfc69Otu10pQdYZJlpLyXhkeQ2U6Vu2qXEM9ka2ZBmvkGP9kX8N0ooMC75H34T9kr3GrjIn1F+rFJxEGIzyYHxVKr
6zOzcuVMdO3ZUVlaW1/KtW7dK/WoXvUFtuzfBPRHajH82JQDm4zWKFhfjPZbPp1/BvVw9eC+MGhWfr4dfgnGPbXhGIadGi2XT0sbVvzt7EuKAGIBfAx7MHWD8b4LDBbkQ0k70+8aVKSNrvMmK
DMPQypUrNXQVHXq1Enz589fX//79NXbsWLlc4c0V/+8hdt27ZN9913nx5//HHG53QHAMS84X4eNMXS3ZuF/t0Ri6gQWYUgu+cNdgXA

```
TpU3377bRXuHYCzYf6jShlAABQVFRl//etfjf/7v//zWr5o0SIjLi7OOHTokNGzZ09jzJgxXu
ufffZZIy4uzli+fLlhGIYxe/Zso3379saPP/7oVXefPn2M8ePHV/yOADCN+Y+qxpEZBERQUJA
yMjI0cOBAZWdna8OGDVq6dKk+/PBDSdLevXt14MABXXPNNV7lrr32Wq+vMzMzdemllyo6Olol
JSUqKSmR3W5X9+7d9dlnn1Xa/gAoP+Y/qpqzqjuAmuM///mPZs6cqe+++0716tVTq1atVK9eP
UlSnTp1JEkRERFeZSIjI72+Pnr0qPbt26e2bduW2UZBQYHq1q1bAb0H4A/mP6oSYQYB8f333+
v2229Xnz599Oyzz+rCCy+UJL300kv6z3/4zkvnpOT41Xu9183aNBAl19+uVJTU8tsJygoqAJ
6D8AfzH9UNU4zISC2bt2qoqIijRkzxvODTDr525okRUVF6cILL9R7773nVW7X19++eXa
s2ePYmJi1L59e8+fN998U8uWLZPD4aj4nQFgCvMfVY0wg4Bo27atrnE6nHn30Ua1bt04ffvihx
o8fr48++kjSycPDEyZM0Nq1a3X//ffr008/Ptb0V8fFtStJsttPfhRHjhjxwpt9utkSNH6q233l
JmZqbuu+8+vfDCC7r44ouravcAnAXzH1XNZhiGUdWdQM3wjvvaN68efr+efr+++/VsGFDdejdwq
v//0/DR8+XPfdd5+GDh2qpUuXKiMjQwcOHNAll1yioUOHatq0aXrqqad09dVXSzp5yPrxxxbt5b9X
Zmamqk1KJFCw0fP1w33nhjFe8hgDNh/qMqEWZQaVavxXq02bdp4/Yb10Ucfdp4/Yb10UcfacyYMXrjjTfUu
nXrKuwdgIrE/EdFIsygSyg0vztb3/X3r179eSTT+q0qm0Jun888/X3r179eSTT+qiiy7Siy++WNXdA1CBmP
+oSIQZVJrc3Fw9/vjj+uSTT3TkyBdd9556t+/vyZMmOC5hRNAzcT8R0UizAAAAAEvjbiYAAGB
phBkAAGBphBkAAGBphBkAAGBphBkAAGBphBkAAGBphBkAAGBphBkAAGBp/x+XaKFw
0n+TYgAAAABJRU5ErkJggg==",
      "text/plain": [
       "<Figure size 600x600 with 4 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "df_virtual_2_time_points['trt_grp'] =
pd.Categorical(df_virtual_2_time_points['trt_grp'], \n",
    "                         categories=['<=30 days apart',  '>=90 days
apart'],\n",
    "                         ordered=True)\n",
    "df_virtual_2_time_points['age_years'] =
round(df_virtual_2_time_points['age_at']/365.25)\n",
    "\n",
    "\n",
    "sns.set_theme(style=\"darkgrid\")\n",
    "\n",
    "g = sns.displot(\n",
    "    df_virtual_2_time_points, x=\"age_years\", col=\"trt_grp\",
row=\"gender\",\n",
    "    binwidth=1, height=3, facet_kws=dict(margin_titles=True),\n",
    ")\n",
    "\n",
    "\n",
    "custom_colnames = [\"CTRL\", \"BB\"]\n",
    "for i, ax in enumerate(g.axes[0]):\n",
    "    ax.set_title(custom_colnames[i], fontsize=14)  # Adjust the
fontsize\n",
    "\n",
    "\n",
    "g.set(ylabel='')\n",
    "g.set(xlabel='age')\n",
    "g.fig.suptitle(f'At-home program', y=1.06)\n"
   ]
  },
```

```
  {
   "cell_type": "code",
   "execution_count": 20,
   "metadata": {
    "metadata": {}
   },
   "outputs": [],
   "source": [
    "my_ctrl = ['<=30 days apart', '<=60 days apart', '61-89 days apart',
'>=90 days apart']\n",
    "bb_init= get_one_enroll_type(df_comb, ['Initial', 'Initial New'],
['<=30 days apart', '>=90 days apart'])\n",
    "ctrl_init = get_one_enroll_type(df_comb, ['Child CTRL'],
my_ctrl)\n",
    "df_init= pd.concat([bb_init, ctrl_init])\n",
    "df_init_2_time_points = make_df_2_time_points(df_init, ['Initial',
'Initial New', 'Child CTRL'], ['Initial', 'Initial New'], 1, 90)\n",
    "\n",
    "df_init_2_time_points['trt_grp'].replace({'<=60 days apart':'<=30
days apart', '61-89 days apart':'<=30 days apart'}, inplace=True)\n",
    "df_init_2_time_points.loc[(df_init_2_time_points['enroll
type']=='Child CTRL') & (df_init_2_time_points['trt_grp']=='>=90 days
apart'), 'trt_grp'] = '<=30 days apart'\n",
    "df_init_2_time_points.groupby(['enroll type', 'trt_grp',
'time_point'])['user_id'].nunique()\n",
    "\n",
    "df_init_2_time_points['age_at'] =
pd.to_datetime(df_init_2_time_points['test_date'], errors='coerce') -
pd.to_datetime(df_init_2_time_points['birthdate'], errors='coerce')\n",
    "df_init_2_time_points['age_at'] =
df_init_2_time_points['age_at'].dt.days"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 21,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Chi2ContingencyResult(statistic=1.8408452811848117,
pvalue=0.17485190743884685, dof=1, expected_freq=array([[ 61.3916501,
98.6083499],\n",
      "       [131.6083499, 211.3916501]])) \n",
      " Chi2ContingencyResult(statistic=0.29060083476050125,
pvalue=0.589835758308618, dof=1, expected_freq=array([[ 118.13996033,
254.86003967],\n",
      "       [1318.86003967, 2845.13996033]]))\n"
     ]
    }
```

```
   ],
   "source": [
    "print( stats.chi2_contingency([[54,106], [139, 204]])\n",
    "      , '\\n'\n",
    "      , stats.chi2_contingency([[113,260], [1324, 2840]])\n",
    ")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "gender  trt_grp        \n",
       "Female  <=30 days apart     54\n",
       "        >=90 days apart    106\n",
       "Male    <=30 days apart    139\n",
       "        >=90 days apart    204\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 22,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_virtual_2_time_points.groupby(['gender',
'trt_grp'])['user_id'].nunique()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "gender  trt_grp        \n",
       "Female  only once             0\n",
       "        <=30 days apart     113\n",
       "        >=90 days apart    1324\n",
       "Male    only once            0\n",
       "        <=30 days apart     260\n",
       "        >=90 days apart    2840\n",
       "Other   only once            0\n",
```

```
       "        <=30 days apart       0\n",
       "        >=90 days apart       2\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 23,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_init_2_time_points.groupby(['gender',
'trt_grp'])['user_id'].nunique()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "Text(0.5, 1.06, 'In-center program')"
      ]
     },
     "execution_count": 24,
     "metadata": {},
     "output_type": "execute_result"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAjMAAAJ+CAYAAABGlIFwAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIW
XMAAA9hAAAPYQGoP6dpAABfXklEQVR4nO3deXQUVcLG4be7swOBhAlBEWTRELawaDCCbCKyiW
NER8EwgsCAKItKEAU/QVlUEAQRQmKjAIiiCOyCIqOgyDbDCKLuAAqW5AkBMhm0vX9waSHNgF
S3Uk3RX7POR7pqrq3bt/uSt7culVlMwzDEAAAgEXZ/d0AAAAAbxBmAACApRFmAACApRFmAACA
pRFmAACApRFmAACApRFmAACApRFmAACApRFmUO79+uuvql+/vvr06ePvppSJvLw8f
fXWW/5uBgCUGcIMcJnr06ePZs2a5e9mAECZIcwAl7njcwjAwrF+d8/N/dxtzKm9Ct5O4Vr8tr
b6//uZ1ed8zxNIAeO6TWdVnyw0uNB8bdMkYPKeTgqY16OSYqAIAnCWakGGhK4Q5JhkjXsvyKE
zZyo/t9Q1JvP7PP+zt5vz5a4Bi1PNOh51I72A12ncD6xYSkGXCpbMdR0fpyAvQ5cNyeGgsdyh
mBT8yx3yczcz2gqqTfvo1JM7i+zbBXpNT93ZN5EfZhI3TB3/RrCcXX6TQ/rDlHEaR87L/0n
QsFbDBkYf1JRvpkP2dZZtX/ewzgIT5Y1Xb/6w+VhXcK7EJPSi3ecAAY13Kt7DoJqBb6rpE0xaW
zmoh78nmwazjJ0qL2zmOvLCsPP0tN+SUJ/3YH6TFfVupJqfsW/CPSLLqAnzsatypCT2f1Y8KOv
OjOaR0tlrvMmapyL1THTOhzffHyzA/uxNQnJakdXGNlKv+bLO1ZZRfWmpp5JB2qsuFMDq/zaEJcL0e9CLTIEoGM+hXt6KqUDLdD+p
VeKYz4qF+6CXDBjP0mbGqwkDWLq2LB7Ow7NLhBxvq+E3tNzNbymM/LT5iLZPGA3GUw4fwyZ1K
pvO61bjpVbRtw7AHG/Ib7RydAeBYbwWnvdTdPZMuAhrR6Jbb9eAQNFTxEBAxX0CC45XT
kQTHzJzlh+Ed9RVNMhP2GF8/3KdSb3TfKn7T5MDgxsiy88thw6aAhiEYsBZgoRaHAtMAVq
7CZd5MKSp3qiYmqy4J69LeLHx2Yu8/bbKG0/pHpW7LF+53JxZU2DcCNfkAHn8qxpbKCgTOvNpcr2s
qD3XOoPDgo+QSDPcyq2Klnj/nu6bbdnN6dpWpXXdZ+NG/Yb0HPTEFsbUrBv5tYbDy01AzMG4x9F
eSWqOV2Rf5ZKnb5/KVm6Dg8kZ1hUtoh8h8dG30ERMCVV7TpFu0Hpdnq61yYRWgX88q8IHb/u6bc
1ju6NSRvaYRyxO1L6W5XlqqH5V//Vyz2W4AmS5Hij0imQ/ukq4jOdjxQQWgEKVpY6Ce2W4W2tD
8bq9UbaLxSIbfsDw+n2nDqXmgWNc5LIVcvw5+37EK9SN7zyxgaGfPdLrvvWZTYY7eFbXtuaFhz
+rZG1mtQ7nF+EWKIyjbh9k2LbbhXx0UrFWiJ/NT7k+4ST8uMQyNP6NJWM0nzIhVQmQ2rnzdv7G
/Y1fIyyyAgWrB4Md6iFoa34ffRLKJzUTOmDsp1WaF03stjvBpw5xOlAaQyDTCpLKy/W1/B4E
+OBiCmXL8WiAeuI2d6JnMUWm/ZXo/5Dh5j/cwPzVNzFm3O4ovPHUxKbuAFq0SF5vzAXB6y6ZOgHZV96Z
5/R8o0/eGt8HJTlFzmcOiFHW8Dv3YOdz47cV5e4TbBEEM6cZzR0iEB8uCB7bg6cXfPPaVO2BlbcsRfsxf
WzSD0N6d6IELjgcWnL/2DCPcUMxFgQpK5y5Ojq9hfnBKjM1HfKGIUwxhXYVzZ40DEPXoJvv9fz2y
fVD
bq1oasFEqnmd+6iOzrDtoZmQfLtKORtr/kCI64g63IuYA+J6sRV04urE2cLQ2+ZtfUwXXm4
Y0AjU4MbKsVaVkyoC3+6f6BD6tW1EZF/L6Umnxf5mFWOBwPhLyBoLW1ywRyl37uMJ6qjhsJI
I0NLvWodRQlaQfw2xFo3rWc3rsLCrjzFnEatVSc70i/dfRQR7jZDgoXG+3dHoQ2dq6HHlFf6lM/
WJr/dl2Qb6kzgF9Dmh0IQeRWLkLq
```

+qX3/9VTVr1tRPP/2k3r17Kz09XTfddJOuvfZa7dq1SwsWLNBnn33m1sYDBw6oV69eysjIUMe
OHVWzZk2tX79evXv3ltPpVPXq1Yu0r7j3s2DBAk2YMEG1atXSbbfdpsDAQO3cuVOfffaZNm3a
pNWrV7s+L+lsIOzdu7euuOIK3XPPPdq2bZtWr16ttLQ0ZWVlKTc3V4mJifr++++1fv16paam6
v3335fdzt+RQKkygHLul19+MWJiYoykpKQiy2JiYoVK1e6lufl5Rndu3c3YmJijIMHD1607v
fee8+IiYkxRowYYeTl5bmWL1++3IiJiTEmTJhgGIZhrFy50oiJiTFGjhxp5Ofnu7WjZcuWRtu
2bV3lN23aZMTExBgNGjQwtm/f7to2MzPTSEhIMOLi4owzZ864lnfo0MG47rrr3No1btw4IyYm
xli4cKHb8s8++8yIiYkxhg8f7lr2+OOPGzExMcbDDz980fdrGIaxd0OlSV/v+/e9/u5b//PPPR
suWLY3rr7/eOHnypNu2bdu2NbKystzqSUpKMmJiYoylS5e6LZ8zZ06R9gwcONCIiYkxPv74Y9
ey3Nxco1evXkU+2/O9n9zcXKNFixbGrbfe6tZ/hmEY48ePL9Jfhe176KGHdi6v5CQfx4AF1CzZk117drV9TowMNA1QnLgwIGLlv/4
449ls9k0evRoBQYGupbffvvvtGjx4sFq0aCFJev/99yVJTzzzxhBwOh2u7q666Svfee6Svfee6+OHj2qD
Rs2uNUdHx+v5s2bu15XXlRJzZs3V05Ojo4cOXXLeNuXn52v58WuWuEaBzdejQQS1atNAnn3yi06
dPu6279dZbL/p+z9WtWzczc1a9bM9bpmzZrq06ePMjMz9fnnn7tt27ZtW4ZtzW4W4WGhrpeHz58WJs3b1b
Lli115513um07YMAA1alTR2vxf8WGRobS0tL05ZdfqkWLFurWrZtru6gIIk0cOfK87fvj+Gy
KNCzzz6riRMnKiwszG1dQKkKCJBV7euuvf/2ra5QqICBATZo0kXT2NFRQUJBru6ZNm0qSDh06d
N42AfAMp5mAC6hdu3aRZZ3UqZVJ09v4tF/Pdd9/piiuucDs1IUk2m02PPI6/WuXbsUHBysd9
55p0gd+/Fvly/Hbv/v29Dn3wDL1Hbfv/99/O2af/+crKylJ+fr5efvnvnlIutzc3NVUFCg7777tt
dd51r+VVXXXXeOovTsmXLIssKf9Hv3btt9++3nr3rt3ryS57b+Q3W5X8+bNtX//fu3bt0+5
ublyOp1uwalQ06ZNFRBQ/I+5FYYb279+4+4zbZDt7r2/+HH3/8UT///LP27z7dunr7/+WtLZ04Z/dPXVV
7u9LgxCf6w/ODhY6cIcwAF3DuX9aF/jhXZNmyZUX+GmAF9oU/Gw1DbVPF/WLVuqq6PSrsm+yko
CpUqFDk9EtZqFChgiTpz3/+s1544Tpz3/+s1544YUy209OTk6RZadad0nZIkValS5YJlC9uYmppa7PrCABMREa
GCggJJRQNSofP1/x8dOnZdR2OAwcOVFBQkMaPH6+WLVuqdu3tWr16tT7555JMS1QPA9wgzgJc
WLFhw3nUxMTHatm2bjh8TJh8/7hqbVKNSjbl8V/vOf/6hr/Zm7GxL+Mlr/NpKRhSFzR8ZnOHr
166KjY31ur1169ZVUFUFCQdu/eLcMwu/uCKw8lEwyeLsb2G+Os/6+++zdj1+mpUmf+QlKrKvKvXr0UERHh8X6++++Zbde7c2W3Zv//9b
0lSXFzcBcsWvs/t37r1q0KDAxdq1FR0dLZvNpm+++k++qtrMnFiYqIMw9Czzzz7
rNhcnNTVV48aN057c57OuNiAgIClJ7+f73276odfBSk7t276/vvv9f8fPdtv3666/1wgsv6P33
31flypU5U92l+h9957Tz/++KPr9Pr/+++KPr9cGDBzV//nWFyxbo0YNtWzZvdHSJJUpvnXbfhWZkczWWX1Xkp
Oj7779Xhw4dFB4erujicJbAuLVaLVtVVvWlwmjSlnbW3ja6bfffnNbvnvXlc/nd
uPAC4djMwAZeiuu+7SJ598r+u7777SJ598oqL0ouXXL1+u7777ZSJ7+2/hcoPZGPHjmMnPv30U61evVr
fffedrrpJuXn52vVqrQZtQOKAgBiDIMw9nHnUxMTH33/bpYJJ3Um2gxmtZOhaNdatTJcLa0o1
fffedbrrpJuXn52vVqlXKyMjQiBEjikw0Lano6GdPHhQo0aNUTqaNUStqtWrXTHHHXdo1khGIY++eQT5TkaNasWUKZ
s8884x69+6tsWPHavXq1a77zGevFk1atTQ2LFjXduOGNG99xxzjsstio6O1r/+9S
+lp6e72nIxHTp00Isvqg5c+bop59+eCxpIcHY0AUKYYmQHKkmMPh0Ku
vvqoRI0YoJydH77zzjjjZt2qTbb79dCxcudD2rGU0bvbHk2cxaIzmNbrFPNWfs1TiSIiWLFmiVatW6ZprrtHL
L7+sBx980OM2JCcn69Pp69prr9XXnDs1Ssbv14Ycf1SIyIPLSsv14cfSpIiYP13nvv6rvr6xEHHtDRo0e1YMECix
YuL3KDPE4MHD1afPn20fv16rrKsO6fAtVe+v++16rVm1Rk2bNtXf//+53tyuyuLLqRzPTnTpaunSpevbsqe++vbsqe++
jwYT3wwANatmyZ2wTjunXrauhXrp2++uorLVmyRFdffbVrKUk4Sk6OlpvvvmmbsqrWvvGmbnhnwOlpvvvmbmzbrxxrRm3
atEnvvPOO9u/frz59/mjVqlWUqWKvvzzyS041AZcgm8GRCAUL888YSeOIJ9e3bt8z3
53Q69csvv+jKK690u5ePJP3yyy65ZZePPP3yyy+65ZZb1KtXL40bN67M2wLHk3
j8pKSmSpBtuuMEftQPgI8yZAWBpNptN9e7996Hr+bNm6fbbn79dbdu79
p06eLvZgIoQ4QZQQ4QZQAJaXnJysysrOMiDZdu3nGxRDLJJJNLgqrEnG2+Un5+vq666So899+SoF
LY84MAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACw
NMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwN
MIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwNM
IMAACwNMIMAACwNMIMAACwNMIMAACwNMIMAACwtAB/NwDWtn79ei1btky7du1SamqqKlSooCZ
Nmqh37966+eabXdu9/PLLLmjVrVonqfPjhhzV06FjhhzV06FCNHj1aH3zzQH1NptNFStWVI0aNdS5c2cN
GDBAQUFBrvV9+vTR5s2s2b9emnn+qqq67y/k0CMKW4Y9dms6ly5cqX7+++vTpo06dOrnWFR6z5
3I4HAoPD1dcXJwGDhyo+Ph4n7Qd1kSYgUdOnz6z6tJ598UkA9uWjWL1
+uBx98UA888IAef/xxSVKnTp1Uq1YtYWBHi/Dbjc/ev7/Pr7ZjM5ZwzDBlQLu3ruu1YRg
6cuSIPvzwQ82YM8YMUM//vijXnNzxxTJ6pwA8de6xq/8/xxm5+fr7S0NK1UfmsTUnmacJZDVF7ZtZ4F4Prt/8QT

TygiIkKSlJeXp99++00ffPCB7r//fs2dO1etWrXy+XuARRiAB4YPH27ExMQYc+fOLbIuNzfXS
EpKMmJiYoz33nvvvHV06NDBiImJOe/6xx9/3IiJiTE2bdpU7PqcnByja9euRkxMjLFnzx7X8s
J9//LLLybeEYDScqFjNysry2jbtq3RqlUrw+l0GoZx4WM2NTXViIuLM+66664ybzesizkzMO1
f//qXVq1apc6dO6t///5F1gcFBWnSpElyOBx6++23y6wdwCHB+vOf/yxJ2rJlS5ntB0DpCQ0N
VYsWLfTbb78pLS3totHHRUpfv362rdvnw9aB6viNBNMW758uaSz57nPp2bNmvroo4909dVXl
2lbwsLCyrR+AKXv119/VZUqVSlSpWLbltQUKAjR46oTp06Zd8wWBZhBqZ98803CggIUNOmTS
+4Xb169cq8LevWrZMkNW7cuMz3BcCcU6dOTPT9f777+ubb77RM888I4fD4bZ9Zma
ma/uCggL99ttvmjt3rk6cOKEJEyb4vP2wDsIMTEtNTVVKlXcriAqS+f+QJTOTiQ8evSo3n33
XW3atEm33HKLmjdv7pO2ACi5hx56qNjlnTp1cp0iPldiYmMKx2yclJal169a2jZcXgzMZMHc
Cg/P99n+zvfD8QqVaqof//+GjZsmM/aDkHn/8ccXGxko6OzKTmZmpbdu2afHixuhH/58Ri2r9
58RUZGurafMmWK/vSnP0k6OzKTkZGhDRs26J133+GtG+ffs0d+5cBQcH++W94NJGmIFp0dHR2r9
/v/Ly8nwyOlP4A9EwDB0+fJ+Fjz5s1TamqxowZo9tvv73M9w/AM40aNeO+nU9
nnnmGb366qsaM2aMa12LFi2K3K3Buq40eioiI0Lx587Rw4UL17dvXF02HxXA1E0yLj4+X
u3bL7jdE088oZEjRyo1NdWr/VS/TVS69atdffdd2vJkiWqWVXVnb4HpjRzZs0avfnmm0XWnzzpSS
OH
D1d+fr4efvjMjkVVatWLT311FPKz8/X448/rjNnzpT6Pg5PgCUjcJHHZT0jr5mt0f5w5wZeGTSp
Ek6efKknnvuOX300Ue69dZbFRkZQMHDuiDDz5QWlqa7rvvvPt13331l1obE6eNm3apfWrVunKK68sctP
TJk4tcujl9+nRVqFChSLnQ0FA98cQTZdYuAGd99dVXOnr0qXXFvq0q0D4VQ1d0TzdnaaxA6tZ69YsI+kprmhVOcyZtBWrLpM5B8
NdevWuR5nIEnZ2dlav369Pv/8czVs2LDI4w+AQoQZeCQ8PFwpGfwfVNfYKlauLVk4UdNxG4rvQvH9rrVz/F3E/6v7g6AzDMPzdCA
AAAE8xZwYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFjaZXHTvIICp9L
Syu529na7TZGRFZSWdkOJ7fluRj6y5zy0F9RUZVKpZ6yPtal8vF5lCb6y5zy0F+ldbybwchM
CdjtNtlsNtntNn83xRLoL3Por0sLn4c59Jc59FfZIMwAAABLI8wAAABLI8wAAABLI8wAAABLI
8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8
wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8w
AAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wA
AABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8wAA
ABLI8wAAABLI8wAAABLI8wAAABLI8wAAABL8yjMLF++++rVq1ftrVuVuz549VOu+RFFqsmDz3/fA1p4ofIJSlfgpRb6RU
pKUrNmzdS+fXulpKS4U6RnZ5o5c6c6bbatGmjk2b2b6oEHKDcMh1mPvzwQz355DUJO
65557tGLFCnXr1k2PvqoOcJQZ9VXVmG7kcJXnXHd3PpUqQm0s/dQpWLQQ+iB/l6JVXQDVnVaHZRgsNWV
acKECVq8eLFsNpGDhyovLy8g1jAACgfAgws7FhGJoxY4uv/9+3X/9fmz
Zu1efNmBQUFAdy4cQoICFc9evV08OBBBvfHGG+rZs6f6FfmqyOxVbrQR2NPz48MCg3hpWdefHQ0rRCeJ2dWblUl0ftatWdkpf91ZISEBO3
q7dq26d+9e+u8QAABc1kyNzPz86dOiQevTo4woQbXi8XZ0aryWQfX0fHaj/ybuUDYFeX+0FQSyfbb+ue7B1VAAACJ3pQS6Ay0Mh0Tkp+B4bL6J7Zpv
fv18nTpz+rpZV3VpXVXQWfDnKpDRzGvkN06q0bJLzKu87dxHhrDqtgNpUasBtekVAbVuPzCnQdc57xjJsU
Rt1999367LPPPJElHjx5V9erV3CpUq1ZZNNnT48GEdPXpUknTFVFcU2ebIkSMevEAFC+mTrNd
Pr0aUnS448/rocffffljR47UmjVrNGTIELL355pvKyclRUFCW5ng4GBJUm5urrrKzsyWp2p2G1Onj
zp8ZuQpICAsrswy+Gwu/0fF0Z/mUN/mVOWx7rE52EW/WUO/VU2TIWZwMBASVL//2v2VmJgoSWr
QoIF2796tN998UyEhIUUm8/btzv/btwm9DQUI/fhN1uU0REBY/Ll1R4
uOdtLI/oL3Por4vz1bEu8XmYRX+ZQ3+VLlNhpvAUUkxMjHJ//t6+9//f//d++SH/fsb8lYvVbEUzr7a5uQK33t7OqH2tKZ9rKCrQa9FjabT7ttfpy/TYpWmqCmcm
6Ojo5Wfn+9aVqtWLbdtYmNjzbn+v5xOQ5mzWZWR6Xvxi8hEzFqc7bNTzBCZ6gIcTWSMNBgzGSQKHbn55l90ytD56VaJJH
oMDpk/1cLugvc+ivkvFVH/F5mEN/mVMe+ysvL0/Z2dmdqXLmyJOmXX37R9u3bR9u3qkqF7XxcCNurr3uHUIMNEMN
UE1a9b0qG5TYSykJEQDBgzQK6+8oujAxGxyimaMxFMbEqtgw2/9NLpmbhJM0uRLmytL+ho2orlQ+ZzRq7xT1eOyAWvp7t/uahPHHU49AGYMQEP
DffPON5s+fr/Hjx0s6O1cmKSL2IJU6dOvoXGRWpGrUgKEpU6aoevXq6tSpk0dvAXKRXYkmpNr5Vb7Gaiq2tnkxLbAY0+MMk9+M9r/iyzoKaiwR5dPArtZp91mcz1c2+Yv1Q7tb6XP5fZZ
T
/X2PHjlXnzp01YcIELV26wZmbzzkMKrJVpSU9FpdPAAeZAJMOVfPMnNw0qzDhrJxA01QBqMEWgsGezxUgDbBddhtRgxtlk0AAp2k+5HSnOe
p5dfflk33HCDJGnu3LmaOHGHgiEhMTFRUVpVGjRrnm10jSsGHD1J+fr7FjxnfSURSUKsmfz38FCKGNhx21mJqNfTlaqKR+iZvjABC9FvaZlw6PZK
ikYAABcHl566SU1bNhQbdu21fTp0/XRRx0NgZwZ1/7m23Pwz9ajvOlqmLUjbtEPAwhbtdaRyDCjJc+Mgtkx4Ywxd4R1sSkBfCqRqctT3Y/lSnSpn2Wd
hT9dsMwzDKqO0+U1DgVFramTKrPyDAroiICkpPP1PPuhgU9QX+ZUx76KyqqUqnUU9bHulQ+Po/
SRH+ZUx76q7jj/cYbb9TkyZPVvn173Xr5/69+9fZLuUlBS9/fbb+uKLL0ztk2vDAABA
mcrOzlZYWJiks7dF7bPTD2ySzj4Z4JNPPi2Pil2uxUrVqhy
5am6zc9ZwYAAMCMXr16afLkLkycrMFTjxo01e/eebX122+/+/acWKFdq8ebeee

YZ0/UTZgAAQJnq3bu3MjIy9Pbbb7vu+L9+/XqtX7++yLajR4/WHXfcYap+wgwAAChzQ4YM0ZA
hQ5SZmamsrKwil2V7gzADAAB8Jjw8XOHh4ZKkM2fO6PTp06pcubLbY47MIswAAACf2rBhg6ZN
m6bdu3fLMAzZ7XbFxcXpscceU3x8vOn6uJoJAAD4zKZNmzRo0CBFRUWpb9++stlsGjBggLKzs
9WvXz9t3brVdJ2EGQAA4DOvvPKKunTpotdee0233XabDMPQ8OHD9eGHH6p169aaMWWOG6ToJMw
AAwGe+/fZb9ejRo9h1vXxXv31s6dO03XSZgBAAA+ExgYqIKCgmLXZWDkKDg42HSdhBkAAOAzjRo
10rx585Sbm+taZhiGDh48qBkzZqhDw48qBkzZqhDhw6m6yTMAAAAAnxk2bJh27typ4cOHS5JsNpsSExPrVs3
VahQQaNGjTJdJ5dmAwAAn2nevLkWLFigXbt2KSQkRHFxcYqKilKvXr101113KSgoyHIlkyBxKj
OBTcXFxiouLkyQtXrzY6/oIMwAAwGc++OCi26TmJhoqk7CDAAA8Jknn3yy2OWGYchms8l8lutx
NmAADApevTTz8tsiw7O1s7duzQrFmz9NJLL5mukzADAAB85sorryx2eb169ZSVlaVJkyZp0aJF
Fpurk0mwAAHBJuPbaa7V7927T5QgzAADA7/Ly8j8ivcdvUnYhvwSZGeWEJKIjZiUyx4nFXB
MJSWlqbc3FyNHDnSdJ2EGQAA4DM33z9ZGo+4eyAxbtpqGeMZSodJr3+oL5SSFjBggugL4aRnS
ZgBAAA+t3HjRm3atenp6ekDw/X9ddfr3bt2slms5muizADAAB8Jjc3Vw899JA2bNggh80hiI
gIZWRka07cuWrSpIlSUlIUHh5uqk6uzgIAD4zZcoU7dy5ZgIAAD4zZcoU7dy5UzNnztTOnTv15ZdfaufOnXr99dd
1+PBhTZ482XSdhBkAAOAzK1eu1IgRI9SpUye3U0pt27bXbvVY489ps8++8x0nYQZAADgM7m5uee9
C3BkZKRHdRJmAACAz3Tr1k0LFizQ77//7r/7rY8QzbQdzxnhuG++27TdTIBGAAA+ExYWJi2bt2qz
p07q3Xrv1qpSpYqPHz+uL5lJVRWSR4Qa7TLmk1999xzF62TMAMAAHxBx3NZTcQjWNdWXXR2dV
XVV67lYWFhrqBT6I831zsfwgwAAPCZTZz/9tNTrJMwAAACfczqd+umnn5SZZmakVaqbt26
FmAEAAD61ZMkSvfTSRmgWfXq1fXXYY4+pR48epuvjaiYAAAzq1ev1rh49ubNhx49eh6aHxZQv+
NptN48aNU/369fd6bSsHvCjRAZmPoRrPqEhAJMSHfk0bQc49mfX2NpQ0/qvmhTOeSSdh3fkB4
XVV67lYWFhrqBT6I83lzsfwgwAAPCZTz/9tNTrJMwAAACfczqd+umnn5SZZmakVaqbt26Htd
FmAEAAD61ZMKSvfTSSzpx4oRrWfXq1fXXYY4+pR48epuvjaiYAAAOAzq1ev1rhx49SlSxeNHj1a
NptN48aNU/369ZWcnKx169aZrpMwAwAAfGbu3Lnq1auZKcDN1N9992aM2eO/vznP
Nl8MOIHBN.8

cuRI2Ww2tW7dWjabTT179tSDDz5ouk7CDACUcyHBAR7Pe4mtHakurepIDNCghKpXr67ly5dr7
969+tOf/qT+/furWrVqat26terVq+dRnYQZAIAkz0Z1qkWGlVFrcDmrWLGirr/+eknSyJEjva
6PMAMAAHzmiSeeuOg2kydPNlUnYQYAAPjMli1bZBjuV82dPn1amZmZqlChgpo1a2a6TsIMAAD
wmXXr1hW7/MCBAxo+fLg6d+5suk6emg0AAPyudu3aGjFihF5//XXTZQkzAADgkpCdna3U1FTT
5TjNBAAAfGbWrFnFLj9+/LhWrlyphg0bmq6TMAMA8CueuF2+zJ49u8gE4EI1a9Y0fSWTRJgBg
EuGNw+K9DQQ+JO3dx7midvWtHv37iLLcnNztWPHDj311FPav3+/ateubapOwgwAXAK8fVCkFX
lz52GeuH15CQ4OVsuWLTV8+HBNmzZNHTp0MFWeMAMAlwBvHhQpSfVrR6prqzqy2az3XAGeuI1
Cdrtdv/zyi+lyhBkAuIR4+os9KoKHRMIaPvjgg2KXZ2RkKCUlRTExMabrJMwAAACfefLJJ8+7
7pprrtEzzzxjuk7CDAAA8JlPP/202OXh4eGqWLGiR3USZgAAgM9ceeWVkqS0tDRFRESUyjyv8
jNtHgAA+N13332njh07qlWrVuratat+/vlnSdLzzz+vd955x6CTMAAMBnJk6cqICAAI0ZM0
bBwcH6v//7P0lSYGCgJk6cqCVLlpiuk9NMAADAZ3bt2qXJkyfr++++++sl0nYQZ
RRxUWFqZFixbp7rvvNlUnIzMAAMBngoKCFB4eLklq0KCBAgMDdeDAAUnSdddddp59++sl0nYQZ
AADgM9dcc42+/PJLSWdvkle3bl1XmMnIyJDT6TRdJ6eZAACAzwwePFiDBw9+sl0nYQZ
18bN25U3bp19cYbb6hOnTqm6yTMAAAAn5k0aZLy8/M1ZcoUvfDCC65Ls5cuXaqAgABBNmTLFdJ
2EGQCAZXn6tHCn0+ABlX4SExOj3r17KzQ0VPZzHqwaFBSkFi1aqHr16qbrJMwAACynUlignE5
D4eGePZOqwOlURnoWgcYPpk+fXup1mgozGRkZmjZtmj7//HOdPn1a9evvWCAy
zx5NnDhR3377rapUqaI+ffqof//+vrJOp1OzZs3SkiVLkyfr1ltvVYsWLXXXXfp9OnTevT
RRxUWFqZFixbp7rvvNlUnIzMAAMBngoKCFB4eLklq0KCBAgMDdeDAAUnSdddddp59++sl0nYQZ
AADgM9dcc42+/PJLSWdvkle3bl1XmMnIyJDT6TRdJ6eZAACAzwwePFiDBw9+sl0nYQZ
18bN25U3bp19cYbb6hOnTqm6yTMAAAAn5k0aZLy8/M1ZcoUvfDCC65Ls5cuXaqAgABBNmTLFdJ
2EGQCAZXn6tHCn0+ABlX4SExOj3r17KzQ0VPZzHqwaFBSkFi1aqHr16qbrJMwAACynUlignE5
D4eGePZOqwOlURnoWgcYPpk+fXup1mgozGRkZmjZtmj7//HOdPn1a9evvWCAy
zx5NnDhR3377rapUqaI+ffqof//+vrJOp1OzZs3SkiVLkyfr1ltvVYsWLXXXXfp9OnTevT
ITJYfdsEK3A6VRGepZfAo3D4VmbCWAAADMIM5c4u90mh92ud1fvUWpalqmy1SLD1LtLA9ntNp

+GA5vt7P7Cw0M9Ku/PAAYAsB7CjEWkpmXp0PHTPt2np6e3AgLsstttWrhmr46dOGOqrL8CWCF
O6QGA9RBmygFPTvfYbDZVCg/x+PSW5J8A5o3SOKV3KjNHhmEu0Hh6Og4X5mkw9TTMAvAfwsxl
rFJYoFeneyR5dHortnakurSqI1nsd4I3p/RqXxmu29teoypVwjzat9NpKCCAOUalxdtgCsBa/
BJmnE6nZs2apSVLligzM1PXXXednn76aV199dX+aM5lKyQ4wOPTPfVrR6prqzo6np5tenSlWq
Rnv9DP5Y/RisJ9ejKiFBUR6nFf16lRWT3a1FOlSswxKi3eBFOrhnGgPPNLmJk9e7YWLVqkyZM
nKzo6WlOmTNHAgQO1YsUKBQUF+aNJlzVPfzn7Q2mMJjmdht9OFXjS19Uiw7yeYxQY6FBBgdNU
2dJwqY8Kefp5ALAWn4eZvLw8zZs3T8nJyWrXrp0kafr06WrTpo3Wrl2r7t27+7pJJebNaMGl/
kP/UuHNaJL0vxElb0ajbDY/BaF08794vQ1/3gY/RoUAXAp8Hmb27t2rM2fOKCEhwbUsPDxcDR
s21JYtW8o0zHh+dY7D69ECJoea4+nk4cIRJSuNRnmjNE4lehoc/X3lGQAUshlmf7t66ZNPPPtH
QoUO1Y8cOhYSEuJYPHz5cOTk5mjNnjuk6Dakox52u82rv7qzc35XgQfd5bDbFRLk8Grfp7Py
VGDyF0ZggF1hIYHlpqw/913eykqSw25TxbAgOZ1OXeywKK1QXtJj3WaT7Ha7pfqU7741yvpz3
/4qa+ZYl/zzR7jPR2ays7MlqcjcmODgYJ08edKjOm02mxyOsj81EBoSWOb7OJ+KYZ7PJSpvZf
257/JWVjobGnzF7LFuxT7lu2+Nsv7ct7/K+vJYN8vnLSscjcnLy3Nbnpubq9BQ6w3zAwAA//J
5mLniiiskSampqW7LU1NTVb16dV83BwAAWJzPw0xsbKwqVqyor7/+2rUsMzNTu3fv1vXXX+/r
5gAAAIvz+ZyZoKAgJSUlaerUqYqMjFSNGjU0ZcoUVa9eXZ06dfJ1cwAAgMX55aZ5w4YNU35+v
saOHaucnBzFx8crJSWFG+YBAADTfH5pNgAAQGm6dK+zAgAAKAHCDAAAsDTCDAAAsDTCDAAAsD
TCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDT
CDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsDTCDAAAsLQAfzegNBQUOJWWdqbM
6rfbbYqMrKC0tDNyOo0y28/lgv4ypzz0V1RUpVKpp6yPdal8fB6lif4ypzz0V2kd72YwMlMCd
rtNNptNdrvN302xBPrLHPrr0sLnYQ79ZQ79VTYIMwAAwNIIMwAAwNIIMwAAwNIIMwAAwNIIMw
AAwNIIMwAAwNIIMwAAwNIIMwAAwNIuizsAAwC8Y7d7diM3h4O/ieF/hBkAKOfsdpuqRITJYfc
smDidhmw27mgL/yHMAEA5Z7fb5LDb9e7qPUpNyzqBFXqHMvt+eFXhBkAgCQpNS1Lh46f
NlWGERlcIjZCQAALI0wAwAALI0wAwAALI0wAwAALI0wAwAALI0wAwAALI0wAwAALI0wAwAALI0wAwAAL
I0wAwAALI0wAwAALI0wAwAALI1nMwHAZcJut3n0t0wEeHw/u/a+12mwICzNfjdBpyOg2v94/yjT
ADAJcBu92mKhVDZbcRHGYQa3Mj8CEwKt3HC/b9pQ2CCCzyAGCWw8fvkM3duXqg5ySxVlVNmHryIfPw
MlW2fu1IdW1Vx6MnYIcEB8hut2nhmr06duKMqbLVIsPUu0sD2e02wgy8QpgBgMtIalqWDh0/
bapMVESo9/tNN79foLQwARgAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAAFgaYQYAA
FgaYQYAAFgaN80DgEuEp89WUAqAVRFmAOAS4K9nK3Kl4FwP/eLS3gHXhN4vm8hGGbGDhCfJrkhJuHGVo52bMmvv+lsaG+DkJJRvNHQG8FhhEtanGmOQKfbTGsEN5rUs3L4Uvm7AXVn
AlxJNnK0ml83wlwKoYzwQAAJZGmAEAAJZGmAEAAJZGmAEAAJZGmAEAAJbmUZhZvny5unXrpiZ
Nmqh79+5atWqVa92hQ4f0yCOPKFeuXOp08aqaLXvqaXdCvtXFHd0ZfXi3sv2apgqlx3ugVLWg89Hk5T8uGA6QHz9ZX1+TZc3Xq3Lz5iM2bxK8uVeGZjLMq8fL3TUe6Ai/4VDPadBpeDqoMyoyPWwMK
LgAAQLllOsx8+OGHevLJJJ3XPPfdoxYoV6tatmx599j5HF9+9//TUp6Vhpm6uvxr166bsyTGnFbjNFNfBmwQYIbnNoosfkkzCdJOpAd7nn0tYdWtH8i7H7YuC3pT/5+dHn3zagUX6uOYaBoS/U7+VCxX/vK+oR/OUVp6LgNa+BX3L6
GbMmKGlS5e6ys+ePVuFi3ShAkTtHjxYtlsNg0cOFB5eXml+sYtEHb9xvZkjs/dT5X0DKtvEW7k+vDV+zONUaYFKG05wlsBGALe9KgxCFd5FXeHHgqsqT0a
t1//33S5IeughbFd++XZs3b9bmzsVFBSkcpcaGKSAgQPX/XNFPOqfVYkmqe8hUEJLa/+gknzUsm8KPnL0xLQGWkr++99YP9rUf
UnJysdu3aSZKmT5+uNm3aaO3atereZ+pU4XFUcMCFWfSQAx2Wg9ggF5p3R66kb6aKG+hQbevCDk2B2hmk7FTZFEKjaDdh6QawKe4mIzBexSj7Uz8LBP5jJfb8JjuK7PGfPpC84j5CYsgSCPi8i
W7duVVx8vAIC/peREhMxIFdXxvfvflmCV19KuHCcQNvY+/Fcc0zmvMMbnMThh2thJ1sNPLg4cGFpsbSmgmnO6ZI0/+bfQAAB6ZC
jMHDhyQJGVlZal///fh9/0e4dDP/5dV0cRJZ0enpLmzp2y+8ehsb7yYa2vVjMvJ01gkdcXqdiTN/dqzc0S2yeDEZsKj0ZqAOKQpyW8Ydg2r
FkmyNHjnj0BgAAQPlm6jT6dnb7H9+OOP6+GHbIkSOl1Zs0aDRKyRG+aZycnIUFUF
4OFiSlJubq+zsbEkqdpuTTu096/CYkKSCg7C7MKnwaLU+lLRn6yxz6y5yyPNYl/30ehfuz2Wwe
PV/JVcYm0+W9KatzNvd0v+Xpu8/xXjZMhZnAwEBJUv/+/ZWYmChJhatq0333xTISEhR
Sby5ubmSpLCwsIUEhIiScrLy3P9u3Cb0FDPnytit9uv5EVHB4/IlFR7Os0/MoL/Mob8uzlfHuu
S/z8PhsCsgwGG+3H+ftu2wmy/vVVmbF2X/+wu9PH73y+N7LkumwkzhKaSYmBi35ddcc40+//xyx
z1ahRQ6mpqW7rC19HR0cpbPz/ftaxWrVpu28TGxppv/yxnwNKqmoesvzT1GrZ5jmC1z//yxnwNKqmCqyHOdZzZtp71z1m4p5mdN1XmTh1vXqtDkRfeiyRYtSu
Z5nt53JBf5lTHvqrtAJIWR/rkv8+j8L9FhQ4lZ9fYLp8gdPp+r/Z8l6VNbwo+9/+vzy/+3/E8
V42TIWZhg0bqkKFCtqxY4Y4euv/561/J9+/apVq1aaatGihRYtIuQoqzeNTh1VrnTh1VrV
pVlSpVUsWKFfX111K111+7wkxmZqZQ2Z2796tpKQr95tpKQkr95Ifn7ZfyomO/pC5PL98ZYH+Mof+Khlf9ZG/Pg/
DMGQYhkflzv5Dpst7U1bnbO7pfsvjd788vueyZCrMhISEaMCAAXrllVcUFRWlGGVMALd8jBl4fxx4
YYPeeustXXPNNZo7d67GjBmjAQAQMG6Jtv vtH8+fM1fvx4SWfnyiQLWnq1KmKjKjIxUjRo1NGXKFAHT6bc/1N6
FWvXl2dOnUqzkcIAAAub6bCjCQNGTJoahGhmj59uo4d2erp5Zdf1rg33erp5Zdf1g033CBJmjt3riZNmqSnS
jExERFRUVp1KhRrvk2dl5+dr7NixysnJUXz8/UjfFJSUoopr+166d+/f/b
Vuy4uLk6LFy8+b1mHw6FDD2nm59GK1y9OzxKcx1 DataflhisRlockIzvRLMsVDlDyjgvowv3Y7UeWQ
AAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTADAAAsjTAD
AAAsjTADAAAszaNLswEAAMzIy8dna2tdna2KleuLEn65ZdftH37dh37Kv
AMAAMrU3r17df/992v3N/df/992vs2LHq3LmzKnn31WAQHm4glhBgAAlKmXXn

nZ2tsLAwSdLp06cVGxtb7HaxsbHKyMgwXT9hBgAAlKmaNWtq27ZtkqSEhAR98sknxW63YsUKt
WzZ0nT9zJkBAABlqlevXpo8ebIyMzPVuHFjzZ49W8ePH1fnzp1VtWpV/fbbb1qxYoU2b96sZ5
55xnT9hBkAgF85HJ6dJHA6DTmdRim3BmWhd+/eysjI0Ntvv62TJ09KktavX6/169cX2Xb06NG
64447TNVPmAEA+EWlsEA5nYbCw0M9Kl/gdCojPYtAYxFDhgzRkCFDlJmZqaysrCKXZXuDMAMA
8IuQ4ADZ7TYtXLNXx06cMVW2WmSYendpILvdRpixmPDwcIWHh0uSzpw5o9OnT6ty5coKCQnxu
E7CDADAr1LTsnTo+Gl/NwM+tGHDBk2bNk27d++WJC1evFj16tWTJMnTkGtwqeKCQnxu
2bNmnQoEGKiopS3759ZbPZNGDAAGVnZ6tfv37aunWr6ToJMwAAwGdeeeUVdenSRa+99ppuu+0
2GYah4cOH68MPP1Tr1q01Y8YM03USZgAAgM98++++236tGjR7HrevfurZ07d5qukzADADAB8JjAw
UAUFBcWuy8jIUHBwsOk6CTMAAMBnGjVqVqpHnz5ik3N9e1zDAMHTx4UDNmzFCHDh1M10mYAQAAP
jNs2DDt3LlTw4cPlyTZbDYlJiaqW7NBgAAPtO8eXMtWLBAu3btUkhIiO
Li4hQVFaVevXrprrvuUlBQkOk6CTMAUIrsdpvsdpvpcp7e0h+wori4MXFxUmSFi9e7HV9hBk
AKCV2u01VIsLkH+OcSIRSjTpGyfLnBPYP2+ZVnrJOiiYhO8Xj4BHSgEEi8NqkyU1fAmJh
ZjM/qgNYyZNPPlnscSMxLIPjtd4cGfVtiU3VyYhcAAFwSrr32Wu3evdt0OcIMAA
Dwu7y8PL3//nqWrWq6bKcZgIAAD7TsWNHGYbhlFGhLsBZmjbMmpOtW0epbPfGKhmjkvu3lcX86S3DsBgAA+MwNN9x
QJMzY7XaFhoSnT0t7ctqbeckwAwAAfCY3wMaGGaP8B1N1/fXX
q127dh7da4wAAfCY3N1cPPfSSJzDoYiJCGVkbDWFY3aEaLzMj7oD1BPSAwVCUPjvJEhQ4cO6cyZM0pa
AA+M2XKFO3cuVaWMzVMzZ87Uzp079eWXX2rnzp16/fXjwYU2ePN10nYPqmYCA
Mnt1NKbdu21WOPPabPPvvMdJ2cZgIAWJanTxt3Og05nQYTjhzAAMqzkzADALC
cSmGBcjoNhYd79rvdr79yrAqdTwP7O1tvvPGGdJmDBnJk2bADALC
AJYTEhwgu92mqm2/ymhiJM6bKVosMU+8uDWS32wgzFnN0oZLM6GAq2eQLF5ZJh+eAS8aWdVagdKMGwACUJFNwUAH
AAA+M2XKFO3cuVaWMzVMzZ87Uzp079eWXX2rnzp16/fXjwYU2ePNl0nYQZAADgMytXrtSIESPDzKYtd5EMk8qqm7UNEsPZkHwVdjL2VoIJXsLPpeqc7Sn4eJvZ
Mnt1NKbdu21WOPPabPPvvMdJ2cZgIAWJanTxt3Og05ncbFN0Spy83NPe9dgCMjIz2kzADALC
cSmGBcjoNhYd79kyrAqdTGelZBBo/6NatmxYsWKBWrVopMDDtTw7O1tvvPGG7r77btN1EmYA
AJYTEhwgu92mhWv26t26tiJM6bKVosMU+8uDWS32wgzfhAWFqatW7eqc+fOat26dZAhwTtuVqgRjW6PqKcFwSCqmc+fOat26tapUqaKLLbboRmXpR4NGxkXFUnVi6SX99hR6l4Wl7ljJ36bcSMqHqwB32ZnxR1lXYCaBlGmnqcIBDqbK8YtdAC1TyPWq3rf0O6n
LL72kEydOuJZVr15dgcoscVe+rq8WYccQ/AOodoLN0pHasjPcC76VkDi7nEb6JQAW71q9Ti4gmGPWFyA/GoTyQdX2PUzrfHRAyqtaGC8iNQmnMktaBF4pmjDomLXpu9FH0TfzC6Ne0Dp1Us4fyGOYY+OtxOVcVUdgCp8CHoIUMxGzJoO9u6ri+VdpnlM8qcoDLPuuhx0VM4q+Wx56Tm9Ks
Dq1aungwcP6o033lDPnj2Vl5enefPmKTk5We3atZMkTZ8+XW3atNHatWvVvXt3T5oCAAAs5Np
rr9WyZcu0a9cuhYeHq2vXroqKilZDdDfeV6727du7lm/lX/8fEK
CPhflQkJCZozZ45OnDihQ4cO6cyZM0pISHCtDw8PV8OGDbVlyxbCDAAAsLb69dJpoS
dO8rs9UmMnMzNSoUaM0duxYXXHFW7rjh49qpiYGLdl0apVKyqhhabLRn8l/5FF69PeJ8U8faN
6YbjgAALCeWbNmXXSbgICAuDVs2VJdddZVCCZzcVZeZYHAvzyWLyzU+Jnwh3jMzybTXZnIY
x25xvVrMZAQFld2GWw2F3+z8ujP4yh/4ypyyPdcm7z6OwjM1mk81mM1Xb1Npst6W96rfZ+z
uS/3Wxpl/XHMcbyfnT97vgdIOhwOXXnllWUX6YvX66tW7fqo48KnZ9SEhIkYm8ubm5ks5eh
hUSEiJJysvLc/27cQrQkCK/QKrSgs7BZYlWzXZ9fdcKVXk4zJm8H2xGY8rWpmWvW/
AwV8Zud/3fbFlvy3tV1uan/XpT9r9Bwp/HXHk+3nfv3l1kWpk6+6wk6Sxx6r+3efhcNgVEO
mli5dqhMnTrjNk5Gp59WikpKbryiuVmprqtjg1NVXR0dEX3S4lJXPKpKVfqrfs+H7Y2Sbw3k
lJmZdfENPeRw2BUeqrMzGwVFDjLjHf8ZTjeQSr6DdOUcr8SU3/w9IsdE5T+UFrWPbrw8SHpm6DgrRM5L
fq/2bLelveqrOGn/XpT9r+fqz+OOY734dEkVGRiSJo3SiBEj9vmzXr/jXqG4WcvgOrc6sdB5K0SfIR
gtu/XWWzVs2D169DB9i+eDAtDbpE/69fTrnu9uCgQ3Gd9saNG1WnTh1pVrp9LfQq6Z5pGz
mZqZ2796tpKQkU4uuTn5+2X8pv6v6BaujygV9/NX/Jjj2y1eRJ+H/A/Hgw/zjR2mWY55Dpst6W
92rf52zuy/2WRll/HnMV7cNpFNb7dvpFsmBxtxDiMmiJvMrt1ZhqT6V9ezt6T2tDzmAYnrbGEn
MAAffPNN5o/f77Gjx8v6excmaQVFEo/GpSqkjRnZRkQRo3G2zdkT0mDy3FdubCpa1bvUgU8Ro
vWrU2X8+g+M8WpWrWqzq5drGJO8KpzXqMMVFVqdMEVFsbNJTZNMVSUVB2JZbSU1Jn5bzj3Uxw
UlJQik4IBAED5VKYjM8X57XsZ4yjKvAAUXr0kbH9YZGyczeduTldlEMYuVAAQVaSfeGwYAAC4L
hBkAAGBphBk4AXx4f16xZs1fXmD8dqhtE2dlkyHd0Di9t1ZGGf9iq/43q3DFdDADKJ
cIMAACwNMIMAACwNG5nAcBUzOBcm8N9yrcecnYWMAKNUXXNPf9AUZMBRLs9ku+LokCD
MAgHKlUlignE5D4eZtcW4Ww8AFkFpbSjDhdqGw=jZKF3axcZF95vUj5/D2T8oqb
HyG63aeGavTp24ozp8tUiw9S7SwPZ7TbCjJcqV66sO+64w/U6IiLC7XVJEWYAAOVSalqDh0/
7e9moBQwARgAAFgaYQYAAFgaYQYAAFgaE4ABAPvvss8tRPvvsss4tuQ5gBA
AA+c8MMN1w0zPz/S3Ujw0zPz666/avn27CgoKSlQnOHMYQYAAPjMm2++6dazcfIa8Pqnj2/7ouLk3bD62O5edNhjhRMQfGKQqe5mWvLAwHD

jQoUR1EmYAAIBfpKamavXq1Vq5cqV27Nih0NBQdejQQYMGDVLbtm0VFBRUonoIMwAAwGfS0tK
0evVqrVq1Stu3b1dwcLDat2+v/v37q127diUOMOcizAAAAJ9p27atJKlz586aPn262rVrp+Dg
YK/q5NJsAADgMx06dJDD4dC3336rXbt2af/+/V7XycgMAADwmZdfflmnT5/W2rVrtWLFCqWkp
Ojqq69W165d1a1bN9WtW9d0nYQZAADgUxUrVlRiYqISExOVnp6u1atXa8WKFZo9e7bq16+vrl
27qnv37qpRo0aJ6uM0EwAA8JuIiAj16tVL77zzjj799x5PHddtttWrVqlTp27Ki//OUvJaqDkRk
AAOAzDRo0uOhN82w2m2w2m7755psS1UmYAQAAPvPwww+bepxBSRBmAACAzzz00EOlXidhBgA
Dzgc5qedelLmcvTBBx/om2++UYsWLdSjRw9J0t69e1W7dm2FhISYro8wAwCACZXCAuV0GgoPD
/WovNNpyGazlXKrrGPWrFl65ZVXKlSJS1cuFC//fab+vXrpwkTJujw4cNasGBBia9iKkSYAQ
DAhJDgANntNi1cs1fHXTpa6agX16hwru738hpmlS5eqb9++evzxx5WSkqLXX39dffv21XP
PPadHHHnlEM2bM0AsvvvGCqTsIMAAAAeSE3L0qHjp02VKc8jMoXS09NdT8O+8847XJXqVP3666+q
WbOmBg8erHHjxpmuk5N3AADAZ6Kjo3Xs2DFJZ+8xQQQAAAHzmmmuu0YIFC9S1Sxddd911On
r0qBYYtWqTNmzfrn//8p2677TbTdRJmAACAz/zf//2fpJewMDAQO3cuVPS2blEBw4cUM+ePfX
oo4+arpMwAwAAfGbFihWqV69eV69eqdZJmAGAYnBDNKBslHaQkpQjzgzAODGZrN5dUM0ABfWsWNHU89m
+uyzzy66DWEGAM5ht9s8viFAOALuqdvSAS7ghhtu4EGTAOALntwwEGTAOALntwQLSqC0RyUjN1uU0CA+
dOSTqchp7N0g4CvTZo0qdTrJMwAAOAjhc91qR/MMUJanA6VRGepblA41hGPriiy+0efNmT
p1SpUrV9aNN96o1q1be1QfYQYAAB/x5rlO1SLD1tLtNkuHmZychA0cOFBbt25VWFiY25OrK5x
5I7vdrrlz5+qmm27S7NmzFRQUKUZpOwgwAAAHzmxIkT573T6CWWD5x24t5XTL169X2dLBFw9udUtDDmxlAHJluMEKbEJ
9qOJcMbmQmPDzcNaJSaNVfr555910O6tIFVFrGMDbmxlB5X/
q6TMAMAAHzmxIkT573T6CWWD5x24t5Xmm
bdMXX3yhsWPH6qWXXtKoUaNM1cl1hAAAwGdWr16tIUOGqHHjxq6JGxRXlBQZYuGDRumlStXm
q6TMAMAAHzmxIkT573XTL169XTixAnTdRJmAACAz0RHR2vHjh3Frlu7dq3O5TYSYjI0
P/93//p7Zt26pFixbq1auXtm7d6lq/Z88VvRr8SUlqVmzZmrfvr1SUlLLcyjudTs2cOVNt2rRR06Z
N9cADD7ge+w0AAC5/3bt316xZszR79mzl5+e7yjMyMvTSSy+peXPmHt5H9+4v+NQnM
06bp/fffV6NNGjdS/f3/9+PP79+/VS7dm0tH0ubPPz7/OHdMkm4utb8POiwtb9x8o4s
GGCFi9eLJvNpoEoDyovL890otbvn+eseJsPtDGTTbCeDT2jAJOBi5uYo
k6S3w108GDB7VhwwYtLVq0kCSNG/VPLsZ7Tf/u3ADiSjQL7Zh7A93fz6BAvBcJeUw3
qjTfeUM+ePZWXl6d58+YpOTlZ7dq1Vub7S1f58H7NWm20N7rzZc0qUtlOXRHDe/+ea6sL
GQoICNBrr72matWqYmKSYSpSjfo7YL31/uEoAaP6NSHFLa2azpHmDaPkCejA51cNmK6z5y
f/LRwkJCdq/f79OnDihvXv36syZ8Zh8xVd/f36syZM0pISHCtDw8PV8VLyxaPGg8AAKypSpUqUqhi
qxMGGDcJTPBWD5fkS6i6+crQq63EcbPzQNX61F2Kw+Y/Vt1E8O3a6Xt+WHi8pHGYTgr7QGJS
vY2FiPg4xkYmQmPDzcNaJSaNWqVfr555910O3uzgNT2rRzD2ehlTqpOoSCmIUhlf62ZSNtrPi2QDRAb0
cUWSbI0eOeNT4c3lyW+iSKnwSLk/ELRn6yxz6y5yyPNals7eZlyTZZPoZS67tfVzWr/s+Z3Or
9Jc/+7o0+oufFUV5fNO8bdu26ckn1THjh1l11803a/LkyUXs3NVXZ2tiQVu83Jk
yc9bYaksz98IiIqeFVHSfAUXXPoL3Por4vz1bEuSQ67XQEBDtNl/FHWn/t22Py0X6v2tTf99d
8Qw8+KojwKM+vWrdPIkSPVtGlTTZTZs2TZIUEhJSZCJvbm6uJCksLEWEwIWefQG5GXl+f6d+E2oaH
efTBOp6HMzCyv6rQgh8Ou5mDBQZWZmq6DAWb7uV3zQX+aUh/4qrQBS1se6JAUGG0lSxYogKnE71
5xeYKlvgdLr+78uy/tx3geGn/Vq1r73pr//+fLjUF1b46g+Oc5k0OM3//+981ceJEderUSVOnT
nWNtFSvu2paqalu2xa+jo6Vj5Lk/vmtZrVq13LaJjY313PP7nKC5mtBL7+Tq+gPX9wsZcD1dnrcvRqIAvT0PGuPxqrIHvpxTnQLMa8Gy9I4vt1zj8PnD2E
n32pokyb5L6BdhY3dPc2Ll6x1+gHU0tJ1jpFxG4NDLVt2ZYUfaB/jpSTCWBtB8D21qA8gDWVeekaA+2Pt3PV/2//y+I0rXBFHwpx/G1xn3WTAyYTAS5mpX2v18vrGdwh0TVmWtUJ1Z
pJcaAkwJWAkn2lSAQvZyASV0BN2EqiyAAXu1cTrKGSmP0gAVLLe7Kt5G2jyeaGmGj6tOkA0GZ6bYTTQiNdOl/Oy2QrG8Pnb4SrF0KlKGYFhdl/32FW/ZE0bWQ79G0zv
vR6cvKZXi/LT6hp7ypq72+wPr

```
K4SZEvD2WUPS2XuQnMrMkWGYCzTeBChYiz8+a0I2gMsBYaYEvHnWkCTVvjJct7e9RlWqMLHwc
udJIAkIcHgVlp1Ow+N7o3gass/dN2EIgL8RZkzwZlKqp2GIy4WtoTQuKffm++FJ2dII2dz1GM
ClwC9hxul0atasWVqyZIkyMzN13XXX6emnn9bVV1/tj+b4THm5V0x5VBp3TE5N9/z74el3y5s
RR+56DOBS4ZcwM3v2bC1atEiTJ09WdHS0pkyZooEDB2rFihUKCgryR5OAUmHF+5pwx2MAVufz
GYd5eXmaN2+ehg4dqnbt2ik2NlbTp0/XsWPHtHbtWl83BwAAWJzPw8zevXt15tjZSQkJaFw
4erYcOG2rJlS5nu2263KSDAbvo/HjwHAMCly2Z4ehmDhz755BMNHTpUO3bsOHDlZ
OTozlz5piu0zBKdkWF3W7zaiLt6aw8FXgwNyAwwK6wkECPylPWHCu224plJclht6liWJCcTqc
u9lOktC47L+mxbrNJdrvdUn3Kd98aZf25b3+VNXOsS/65zYTP58xkZ2dLUpG5McHBwTp58qRH
ddpsNjkcZT96UjHMu/k83pSnrDX2Xd7KSmdDg6+YPdat2Kd8961R1p/79ldZXx7rZvm8ZYWjM
Xl5eW7Lc3NzFRrKlTsAAMAcn4eZK664QpKUmprqtjw1NVXVq1f3dXMAAIDF+TzMxMbGqmLFiv
r6669dyzIzM7V7925df/31/31vm4OAACwOJ/PmQkKClJSUpKmTp2qyMhI1ahRtQ0OOmTFH16tXVqVM
nXzcHAABYnF9umjds2DDl5+dr7NixysnJUXx8vFJSUrhhgAAM3nl2YDAACUpkv3OisAAIAS
IMwAAABLI8wAAABLI8wAAABLI8wAAABLI8wAAABLI8xcxKFDh1S/fv0i/y1/ZssTfTbukzJ49W
3369HFbtmfPHiUlJalZs2q3769UlJS/NS6S09x/fXEE08U+Z61bWTy0sfzjWS47j3RyO97
Lnl5vmWcl3332n4OBgrVu3Tjbb/57WW6lSJT+26tLy1bdmMmYqPj3ctS09PV79+/XTLLbd
o/Pjx+s9//qPx48erSpUq6tmzpx9mtZPx9b63xOX8sFSVJ1A9YqdFDn0zf8gY3BWkdP5fDAzQ5/LQ8EtBWeLYXsZ6/eNHEYbz0zBzfXu3r3OLDhcE2jxAspcyyI1MjpWQ0M3ZkzW+5f4nF/H9T/3XU/vx7N4/Y7Xn1YZIxgsw/lJ+6WIOw8MjGMEt5fFlwGsIDXs4HvJKjo5gFGpZRkHvmWG7
z8ZaEHM3y+eAgdm41NnzGD2XP8DJvscNn3sqfEmQAOLwcyrA5/DjJnUdrmzI11/3Zx8p/QLj8
h8bAqFcVRSoCtjfCRfxPf5UZ/d5t8y2/iVgDJTIVkndDS+AnhNWFW3EH8sV08wVs4P7L4i4
v4nrXQj6XfA0OnaWzmEu3n/Q3bCRfrHr2emJHMqw4WCmMeg89rQhe9TQm9sA4C+0VHT28Sv
K5hchO9/JHKm/Lc2HptTG/WeqDyHFmjhWfRp3VKpVZCZQGLs/jP6uAIW4sqeoV2euAIZpjgu
x74nTGX2qyb83kiNW+Fns5E8=
```

```
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "df_init_2_time_points['trt_grp'] =
pd.Categorical(df_init_2_time_points['trt_grp'], \n",
    "                          categories=['<=30 days apart',  '>=90 days
apart'],\n",
    "                          ordered=True)\n",
    "df_init_2_time_points['age_years'] =
round(df_init_2_time_points['age_at']/365.25)\n",
    "df_init_2_time_points['gender'] =
pd.Categorical(df_init_2_time_points['gender'], \n",
    "                          categories=['Female',  'Male'],\n",
    "                          ordered=False)\n",
    "sns.set_theme(style=\"darkgrid\")\n",
    "\n",
    "g = sns.displot(\n",
    "    df_init_2_time_points, x=\"age_years\", col=\"trt_grp\",
row=\"gender\",\n",
    "    binwidth=1, height=3, facet_kws=dict(margin_titles=True),\n",
    ")\n",
    "custom_colnames = [\"CTRL\", \"BB\"]\n",
    "for i, ax in enumerate(g.axes[0]):\n",
    "    ax.set_title(custom_colnames[i], fontsize=14)  # Adjust the
fontsize\n",
    "\n",
    "g.set(ylabel='')\n",
    "g.set(xlabel='age')\n",
    "g.fig.suptitle(f'In-center program', y=1.06)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "TtestResult(statistic=4.024969228864507,
pvalue=6.724888995398134e-05, df=433.498231695994)"
      ]
     },
     "execution_count": 25,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
```

```
   "source": [
    "demo_init =
df_init_2_time_points[df_init_2_time_points['time_point']==0]\n",
    "demo_ttest(demo_init, 'age_at', '<=30 days apart', '>=90 days
apart')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 26,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "count    8326.000000\n",
      "mean        9.996774\n",
      "std         3.094034\n",
      "min         4.010951\n",
      "25%         7.600274\n",
      "50%         9.629021\n",
      "75%        12.119781\n",
      "max        17.987680\n",
      "Name: age_at, dtype: float64 \n",
      " count    746.000000\n",
      "mean      10.436555\n",
      "std        3.243900\n",
      "min        4.087611\n",
      "25%        7.911020\n",
      "50%       10.087611\n",
      "75%       12.800821\n",
      "max       17.911020\n",
      "Name: age_at, dtype: float64\n",
      "count    620.000000\n",
      "mean      10.327182\n",
      "std        3.147210\n",
      "min        4.030116\n",
      "25%        8.035592\n",
      "50%        9.980835\n",
      "75%       12.496235\n",
      "max       17.549624\n",
      "Name: age_at, dtype: float64 \n",
      " count    386.000000\n",
      "mean       9.991552\n",
      "std        3.120711\n",
      "min        4.191650\n",
      "25%        7.563313\n",
      "50%        9.824778\n",
      "75%       12.566735\n",
      "max       17.911020\n",
      "Name: age_at, dtype: float64\n"
```

```
     ]
    }
   ],
   "source": [
    "age_ctrl =
df_init_2_time_points.loc[df_init_2_time_points['trt_grp']=='<=30 days
apart', 'age_at']/365.25\n",
    "age_bb =
df_init_2_time_points.loc[df_init_2_time_points['trt_grp']=='>=90 days
apart', 'age_at']/365.25\n",
    "print(age_bb.describe(), \n",
    "      '\\n',\n",
    "    age_ctrl.describe()\n",
    ")\n",
    "\n",
    "age_ctrl =
df_virtual_2_time_points.loc[df_virtual_2_time_points['trt_grp']=='<=30
days apart', 'age_at']/365.25\n",
    "age_bb =
df_virtual_2_time_points.loc[df_virtual_2_time_points['trt_grp']=='>=90
days apart', 'age_at']/365.25\n",
    "print(age_bb.describe(), \n",
    "      '\\n',\n",
    "    age_ctrl.describe()\n",
    ")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 27,
   "metadata": {
    "metadata": {}
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "trt_grp\n",
       "Initial    4166\n",
       "Virtual     310\n",
       "Name: user_id, dtype: int64"
      ]
     },
     "execution_count": 27,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# compare the virtual program to the in-center program (initial)\n",
    "# only keeping subjects who stayed for >= 90 days\n",
    "# note the difference between df_comb and df_comp\n",
    "\n",
    "cols2drop = ['birthdate', 'gender', 'Status', 'Hours', 'Center']\n",
```

```
    "df_1 = df_init_2_time_points[df_init_2_time_points['trt_grp'] ==
'>=90 days apart'].drop(columns=cols2drop)\n",
    "df_1['enroll type'] = 'Initial'\n",
    "df_2 = df_virtual_2_time_points[df_virtual_2_time_points['trt_grp']
== '>=90 days apart'].drop(columns=cols2drop)\n",
    "df_2['enroll type'] = 'Virtual'\n",
    "df_2programs = pd.concat([df_1,
df_2]).drop(columns='trt_grp').rename(columns={'enroll
type':'trt_grp'})\n",
    "df_2programs.groupby('trt_grp')['user_id'].nunique()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 29,
   "metadata": {
    "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>user_id</th>\n",
       "      <th>trt_grp</th>\n",
       "      <th>test_name</th>\n",
       "      <th>pre_score</th>\n",
       "      <th>post_score</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>2462</th>\n",
       "      <td>708978</td>\n",
       "      <td>&gt;=90 days apart</td>\n",
       "      <td>ML_max_score</td>\n",
       "      <td>NaN</td>\n",
```

```
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2069</th>\n",
"      <td>614364</td>\n",
"      <td>&lt;=30 days apart</td>\n",
"      <td>TS_max_score</td>\n",
"      <td>5.0</td>\n",
"      <td>7.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2108</th>\n",
"      <td>619539</td>\n",
"      <td>&lt;=30 days apart</td>\n",
"      <td>ML_max_score</td>\n",
"      <td>6.0</td>\n",
"      <td>8.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2646</th>\n",
"      <td>753667</td>\n",
"      <td>&lt;=30 days apart</td>\n",
"      <td>DT_final_score</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2686</th>\n",
"      <td>764189</td>\n",
"      <td>&lt;=30 days apart</td>\n",
"      <td>SS_max_score</td>\n",
"      <td>0.0</td>\n",
"      <td>2.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2977</th>\n",
"      <td>846125</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>FM_final_score</td>\n",
"      <td>81.0</td>\n",
"      <td>64.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>280</th>\n",
"      <td>261518</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>SS_max_score</td>\n",
"      <td>3.0</td>\n",
"      <td>4.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2749</th>\n",
"      <td>773417</td>\n",
"      <td>&gt;=90 days apart</td>\n",
```

```
       "      <td>FM_final_score</td>\n",
       "      <td>132.0</td>\n",
       "      <td>132.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1414</th>\n",
       "      <td>491731</td>\n",
       "      <td>&gt;=90 days apart</td>\n",
       "      <td>SS_max_score</td>\n",
       "      <td>6.0</td>\n",
       "      <td>6.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>2526</th>\n",
       "      <td>720536</td>\n",
       "      <td>&lt;=30 days apart</td>\n",
       "      <td>DT_final_score</td>\n",
       "      <td>0.0</td>\n",
       "      <td>10.0</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "     user_id          trt_grp       test_name  pre_score
post_score\n",
       "2462  708978  >=90 days apart    ML_max_score        NaN
NaN\n",
       "2069  614364  <=30 days apart    TS_max_score        5.0
7.0\n",
       "2108  619539  <=30 days apart    ML_max_score        6.0
8.0\n",
       "2646  753667  <=30 days apart  DT_final_score        NaN
NaN\n",
       "2686  764189  <=30 days apart    SS_max_score        0.0
2.0\n",
       "2977  846125  >=90 days apart  FM_final_score       81.0
64.0\n",
       "280   261518  >=90 days apart    SS_max_score        3.0
4.0\n",
       "2749  773417  >=90 days apart  FM_final_score      132.0
132.0\n",
       "1414  491731  >=90 days apart    SS_max_score        6.0
6.0\n",
       "2526  720536  <=30 days apart  DT_final_score        0.0
10.0"
      ]
     },
     "execution_count": 29,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
```

```
   "source": [
    "df_rmANOVA_virtual =
df_virtual_2_time_points[['user_id','DT_final_score', 'ML_max_score',
'RT_final_score', 'FM_final_score', 'TS_max_score',
'SS_max_score','time_point', 'trt_grp']].melt(id_vars
=['user_id','time_point', 'trt_grp'],var_name='test_name',
value_name='score')\n",
    "df_rmANOVA_virtual =
df_rmANOVA_virtual.pivot(index=['user_id','trt_grp',
'test_name'],columns=['time_point'], values=['score']).reset_index()\n",
    "df_rmANOVA_virtual.columns = ['user_id',
'trt_grp','test_name','pre_score', 'post_score']\n",
    "df_rmANOVA_virtual.to_csv('/Users/rye/Documents/Documents -
Mac/BrainBalance/data/df_rmANOVA_virtual.csv', index=False) \n",
    "df_rmANOVA_virtual.sample(10)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 31,
   "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>user_id</th>\n",
       "      <th>trt_grp</th>\n",
       "      <th>test_name</th>\n",
       "      <th>pre_score</th>\n",
       "      <th>post_score</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>4070</th>\n",
       "      <td>209574</td>\n",
```

```
"      <td>&gt;=90 days apart</td>\n",
"      <td>ML_max_score</td>\n",
"      <td>5.0</td>\n",
"      <td>7.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7169</th>\n",
"      <td>249406</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>TS_max_score</td>\n",
"      <td>4.0</td>\n",
"      <td>3.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>15623</th>\n",
"      <td>418060</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>TS_max_score</td>\n",
"      <td>4.0</td>\n",
"      <td>7.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>20468</th>\n",
"      <td>555349</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>ML_max_score</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3013</th>\n",
"      <td>197521</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>FM_final_score</td>\n",
"      <td>51.0</td>\n",
"      <td>70.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>17591</th>\n",
"      <td>474405</td>\n",
"      <td>&gt;=90 days apart</td>\n",
"      <td>TS_max_score</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>304</th>\n",
"      <td>174070</td>\n",
"      <td>&lt;=30 days apart</td>\n",
"      <td>SS_max_score</td>\n",
"      <td>NaN</td>\n",
"      <td>4.0</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
       "      <th>16411</th>\n",
       "      <td>445744</td>\n",
       "      <td>&gt;=90 days apart</td>\n",
       "      <td>FM_final_score</td>\n",
       "      <td>100.0</td>\n",
       "      <td>90.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>22362</th>\n",
       "      <td>617945</td>\n",
       "      <td>&gt;=90 days apart</td>\n",
       "      <td>DT_final_score</td>\n",
       "      <td>4.0</td>\n",
       "      <td>6.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1593</th>\n",
       "      <td>183211</td>\n",
       "      <td>&gt;=90 days apart</td>\n",
       "      <td>RT_final_score</td>\n",
       "      <td>NaN</td>\n",
       "      <td>86.0</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "      user_id          trt_grp       test_name  pre_score
post_score\n",
       "4070   209574  >=90 days apart    ML_max_score        5.0
7.0\n",
       "7169   249406  >=90 days apart    TS_max_score        4.0
3.0\n",
       "15623  418060  >=90 days apart    TS_max_score        4.0
7.0\n",
       "20468  555349  >=90 days apart    ML_max_score        NaN
NaN\n",
       "3013   197521  >=90 days apart  FM_final_score       51.0
70.0\n",
       "17591  474405  >=90 days apart    TS_max_score        NaN
NaN\n",
       "304    174070  <=30 days apart    SS_max_score        NaN
4.0\n",
       "16411  445744  >=90 days apart  FM_final_score      100.0
90.0\n",
       "22362  617945  >=90 days apart  DT_final_score        4.0
6.0\n",
       "1593   183211  >=90 days apart  RT_final_score        NaN
86.0"
      ]
     },
     "execution_count": 31,
     "metadata": {},
```

```
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df_rmANOVA_init = df_init_2_time_points[['user_id','DT_final_score',
'ML_max_score', 'RT_final_score', 'FM_final_score', 'TS_max_score',
'SS_max_score', 'time_point', 'trt_grp']].melt(id_vars
=['user_id','time_point', 'trt_grp'],var_name='test_name',
value_name='score')\n",
    "df_rmANOVA_init = df_rmANOVA_init.pivot(index=['user_id','trt_grp',
'test_name'],columns=['time_point'], values=['score']).reset_index()\n",
    "df_rmANOVA_init.columns = ['user_id',
'trt_grp','test_name','pre_score', 'post_score']\n",
    "df_rmANOVA_init.to_csv('/Users/rye/Documents/Documents -
Mac/BrainBalance/data/df_rmANOVA_init.csv', index=False)\n",
    "df_rmANOVA_init.sample(10)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 34,
   "metadata": {},
   "outputs": [],
   "source": [
    "df_2programs_RM = df_2programs.drop(columns=['SS_avg_score',
'test_date', 'age_at', 'age_years',
'cum_interval_int']).set_index(['user_id', 'trt_grp',
'time_point']).stack().reset_index().rename(columns={'level_3':'test_name
', 0:'score'})\n",
    "df_2programs_RM.to_csv('2programs_rmANOVA.csv')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [],
   "source": [
    "n_times = 10_000\n",
    "change_score_dict = {}\n",
    "res_dict = {}\n",
    "i = 0\n",
    "creyos_tasks = {'SS':'SS_avg_score', 'DT':'DT_final_score',
'ML':'ML_max_score', 'RT':'RT_final_score', 'FM':'FM_final_score',
'TS':'TS_max_score'}\n",
    "for t, score_name in creyos_tasks.items():\n",
    "    diff_score_1test = get_test_df(df_2programs, score_name)\n",
    "    res_init =
stats.bootstrap((diff_score_1test.loc[(diff_score_1test['trt_grp']=='Init
ial'), 'diff_score'].dropna(), ), \n",
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
```

```
    "    res_virtual =
stats.bootstrap((diff_score_1test.loc[(diff_score_1test['trt_grp']=='Virt
ual'), 'diff_score'].dropna(), ), \n",
    "            np.nanmean, confidence_level=0.95, \n",
    "            n_resamples=n_times, method='BCa')\n",
    "\n",
    "\n",
    "    change_score_dict[t] =
res_init.bootstrap_distribution/res_virtual.bootstrap_distribution\n",
    "    res_dict[i] = {'task name':t, 'program type':'at-home',
'ci':'low', 'value':res_virtual.confidence_interval[0]}\n",
    "    res_dict[i+1] = {'task name':t, 'program type':'at-home',
'ci':'high', 'value':res_virtual.confidence_interval[1]}\n",
    "    res_dict[i+2] = {'task name':t, 'program type':'in-center',
'ci':'low', 'value':res_init.confidence_interval[0]}\n",
    "    res_dict[i+3] = {'task name':t, 'program type':'in-center',
'ci':'high', 'value':res_init.confidence_interval[1]}\n",
    "    i += 4"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 40,
   "metadata": {},
   "outputs": [],
   "source": [
    "## bootstrap for testing treatment effect of the at-home program and
the in-center program\n",
    "creyos_tasks = {'SS':'SS_max_score', 'DT':'DT_final_score',
'ML':'ML_max_score', 'RT':'RT_final_score', 'FM':'FM_final_score',
'TS':'TS_max_score'}\n",
    "res_dict = {}\n",
    "dist_dict = {}\n",
    "n_times = 10_000\n",
    "i = 0\n",
    "for t, score_name in creyos_tasks.items():\n",
    "    res_init_1 =
stats.bootstrap((df_init_2_time_points.loc[(df_init_2_time_points['time_p
oint']==1) & (df_init_2_time_points['trt_grp']=='>=90 days apart') ,
score_name], ), \n",
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
    "    res_virtual_1 =
stats.bootstrap((df_virtual_2_time_points.loc[(df_virtual_2_time_points['
time_point']==1) & (df_virtual_2_time_points['trt_grp']=='>=90 days
apart'), score_name], ), \n",
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
    "    res_init_0 =
stats.bootstrap((df_init_2_time_points.loc[(df_init_2_time_points['time_p
oint']==0) & (df_init_2_time_points['trt_grp']=='>=90 days apart'),
score_name], ), \n",
```

```
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
    "    res_virtual_0 =
stats.bootstrap((df_virtual_2_time_points.loc[(df_virtual_2_time_points['
time_point']==0) & (df_virtual_2_time_points['trt_grp']=='>=90 days
apart'), score_name], ), \n",
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
    "    dist_dict[t] =
{'pre_test':res_init_0.bootstrap_distribution/res_virtual_0.bootstrap_dis
tribution,
'post_test':res_init_1.bootstrap_distribution/res_virtual_1.bootstrap_dis
tribution}\n",
    "    #res_dict[t] = {'init_0':res_init_0.confidence_interval,
'init_1':res_init_1.confidence_interval,
'virtual_0':res_virtual_0.confidence_interval,
'virtual_1':res_virtual_1.confidence_interval}\n",
    "    #res_dict[t] = {'init':{'pre':res_init_0.confidence_interval,
'post':res_init_1.confidence_interval},
'virtual':{'pre':res_virtual_0.confidence_interval,
'post':res_virtual_1.confidence_interval}}\n",
    "    res_dict[i] = {'task name':t, 'program type':'init', 'time
point':'pre', 'ci_low':res_init_0.confidence_interval[0],
'ci_high':res_init_0.confidence_interval[1]}\n",
    "    res_dict[i+1] = {'task name':t, 'program type':'init', 'time
point':'post', 'ci_low':res_init_1.confidence_interval[0],
'ci_high':res_init_1.confidence_interval[1]}\n",
    "    res_dict[i+2] = {'task name':t, 'program type':'virtual', 'time
point':'pre', 'ci_low':res_virtual_0.confidence_interval[0],
'ci_high':res_virtual_0.confidence_interval[1]}\n",
    "    res_dict[i+3] = {'task name':t, 'program type':'virtual', 'time
point':'post', 'ci_low':res_virtual_1.confidence_interval[0],
'ci_high':res_virtual_1.confidence_interval[1]}\n",
    "    i += 4"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 41,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "SS 0.935 0.9555\n",
      "DT 0.7745 0.5913\n",
      "ML 0.3312 0.372\n",
      "RT 0.8317 0.4935\n",
      "FM 0.8626 0.8282\n",
      "TS 0.7793 0.9023\n"
     ]
```

```
    }
   ],
   "source": [
    "def find_nearest(array, value):\n",
    "    array = np.asarray(array)\n",
    "    idx = (np.abs(array - value)).argmin()\n",
    "    return idx\n",
    "\n",
    "for t, score_name in creyos_tasks.items():\n",
    "    pre_hist = np.sort(dist_dict[t]['pre_test'])\n",
    "    post_hist = np.sort(dist_dict[t]['post_test'])\n",
    "    pre_idx = find_nearest(pre_hist, 1)/len(pre_hist)\n",
    "    post_idx = find_nearest(post_hist, 1)/len(post_hist)\n",
    "    print(t, pre_idx, post_idx)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 42,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "SS 0.9119\n",
      "DT 0.201\n",
      "ML 0.365\n",
      "RT 0.042\n",
      "FM 0.2744\n",
      "TS 0.4721\n"
     ]
    }
   ],
   "source": [
    "for t, score_name in creyos_tasks.items():\n",
    "    hist = np.sort(change_score_dict[t])\n",
    "\n",
    "    idx = find_nearest(hist, 1)/n_times\n",
    "\n",
    "    print(t, idx)\n",
    "    "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 43,
   "metadata": {},
   "outputs": [],
   "source": [
    "df_ci = pd.DataFrame.from_dict(res_dict, orient='index')\n",
    "df_ci_plot = df_ci.set_index(['task name', 'program type', 'time
point']).stack().reset_index().rename(columns={'level_3':'ci',
0:'value'})\n",
```

"df_ci_plot['program type'].replace({'init':'in-center', 'virtual':'at-home'}, inplace=True)"
]
},
{
"cell_type": "code",
"execution_count": 44,
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"/Users/rye/opt/miniconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:718: UserWarning: Using the pointplot function without specifying `order` is likely to produce an incorrect plot.\n",
" warnings.warn(warning)\n"
]
},
{
"data": {
"text/plain": [
"<seaborn.axisgrid.FacetGrid at 0x167ddc550>"
]
},
"execution_count": 44,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAA8wAAAIzCAYAAADVm5AWAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIW
XMAAA9hAAAPYQGoP6dpAADBPUlEQVR4nOzdeVxU1fsH8M/MsAwgICCbIqCgiAuKIagtopYmaq
UtlmZG4oZoZiQQulVaalpop4k6imWXu5fLVn5pWhuK+byCoIALKvjMz9/cHODoy7DADw+f9evn
ycufcO8/IeGaee859jkgQBAFERERERERERERERERERERERERpEKs7QCIiIiIiIiI6iMmzERERERERERERERERERERERqMGEmIiIi
IiIiUoMJMxERERERERERERERERERERERRREREREZEaTJiJiIiIiIiI1GDCTERERERERERERERERERERKQGE2YiIiIiIiIiNZgwExERERERE
anBhLmREgRB2yEQUSPEvoeItIF9DxFVFxPmRujw4cMICQmp9fOOHDkSI0eOr
hR8PLyQqdOnfDKK69g7ty5ePjwYam2Z86cwfjx4+Hj44OOHTvC19cXM2bMwN27d7UQOVH1se/
RLjc3N5U/7du3h4+Hh4+PD/z9/XHs2DGVtn369Cqp/tk/06daIaoa9j3a9bjP+OGHH9Qr+rlAo
8OKLL8LNzQ07duwAAMTHx6v8TKRNetoOgDQvIiJC2yE0ajt37sT06dMxbPjhhzAyMkJ0d
DTWrFmDv/76C9u3b0fTpk0BAJGRkQQICEDfvn0xd+5cmJJmZ4e7du/jpp5/w9ttvY+vWBvXBA0dN
TuCyKqJPY92vfWW2/h7bffBgAUFRUhJSUF27Ztw9ixY/ixY/HFF1/g/fffxAsX74chYWFyuOCgoL
Qvn17BAYGKvdZWlpqNwFFKPnTp1CsnJyVqIiqhiymDATaVhYWBgGQ
DRqEr7/+Wrmve/fu8PPzfu8PLywuuuv45t27YhICAAALBq1Sp06D98Morr2D9+
vWYPXu2xl8DETVMdnZ26NKli8o+Pz8/jLt/zc+hD8Oce0NSqh
qja9euOH36NK5cuzYIOHTqoPLZ37164u7vj2rVrWoqOqHyckt3IjBw5EhP5ElFRUYiKioKioKZ
NjwJALh+/TqCgoLQvXt3dOjQAS+++CIAL/lh/TqCgoLQvXt3dOjQAS++++CLmzp2L/Px85bH//fc
zz//oGPHjpgpgxY0Nwpo1a+Dr64tOnTrh3XffxfhqfxYlwFt1TaHp0CMOHD4enpyc6d
uyIV199FZs2bVI+fvLkSb+uSEyMhIjR46Eh4cHfH19cHfH1sXXrViqVQnnN6e
np+PLLL9GzZ0896tQJ77zzDiIjI8v9d5d5w+XqZxNHlbb
q2NjY4PPPP8fzzz9fbpxE9QX7btlX5eKL6jH1P/eh7vL290axZ

M+zfv19lv0wmw8GDBzFw4MByjyfSJpHAKgiNSnR0ND777DMAwOzZs+Hq6orc3FwMGDAAXbp0w
ciRI2FgYICjR49iw4YN+OSTTzB+/Hjcu3cPgwYNwptvvol+/fohIyMDS5YsUXZ0YrFYeR/Pzz
//jFOnTiEgIACvvvoq5s+fD7FY/bWZkSNH4tq1a3BxccGYMWMgCAK+++47FBUV4ciRI5BIJDh
69CjGjRuHDz74AH369EF+fj42bdqEf//9F7/++iu6du2KkydP4oMPPPoClpSXGjh0LNzc3rFmz
BlFRUXB0dISfnx88PT3x66+/4vDhw9i6dSs8PDxQUFCAd955Bw8fPsSUKVNgY2OD7du34/Dhw
1i3bh169OihNu67d+8iNTW1zH9nV1dXNGnSRO1jU6ZMwf79+/Hyyy9jwIAB6NatG2xtbdW2Xb
hwIdatWwdvb2+8/vrr8PHxQcuWLct8XqL6in2P9vseNzc3BAUFYdKkSWof9/X1RcuWLfHzzz+
XeqxPnz7w9vbGgULynuovqIfU/96XvS0tLw999/49ChQ8rH/vnnH0yePBlbt27FwIEDMX/+
fAwdOhTx8fHo27ev8mcibeKU7Ebm6Q7t8dS68+fPw93dHUuXLU+1rNnT0RGRuLUqVMYP348L
l68iPz8fIwbN06Z3Nnb2+Pw4cPIzc1V6SQvXryIcePGoV+/fuV+aDwmk88kQHh6uPEdOTg5CQk
Jw7do1dOzYEdHR0XjjjTcwa9Ys5TGenp7w8fHBqVOn0LVrV+X+N998E/7/gAAY2NjDBs2DB4
eHpg8eTIAoGPHjjh8+DDOnj0LDw8P7N69G9evX8fvv/+Ozp07Awb99evBeeukljBw5EosWLcL27dvV
xuzo6Fjte4e4e/+eYbKBQKHDx4UPmh4ejoiD59+sDf3x92dnbKth9//ZGTysrKwfft2REVFAQBsb
W3h6+uLUaNGwcXFpVoxEGka+x7t9z0VadasfPzwy+//ILLLy8rZ9Pt27
cPffv2hVQqrdG5ieoSE2bCCy+8gBdeeAFFRUWIjY1FXFwcbty4gdTUVGXxKqbGbd6gdO8PQ0BBvvfU
W/Pz80KtXL3h5ecHDw0PlXf31deMZ09e3aFHxpA6auSjz+Y8vLyAEBWVhaA4qJa1tbW6NChNChA2QymbJd79698f333
wPm5ualYlYoFFAoFGW+JolEApFIpPYYXYoFF+ofFGW+JolEApFIpPYxU1NTLFu2DPHx8Th27BhOnjyjykydPIIiAr///
OVH4YGBgb4+uuvMWnSJBw7dgwnTpzAyZMPnsWXLFuzYsQOLFy9G//79y4yDqDqPZvqcyanI
sUUPBvkc7fc9zzz0HW1tb7N+/HxxNqYzYZSEOHTqEhQsXlnscckbYxYSYoFAr88MMP+OWX5Cb
mwt7e3t4eHjA0NBQ2cbBwQGbNm3CmjVr8P78PvvvvcMiIgJmZYPnw4Pv74Y+UHRHx8PF544QWcP
HkSoaGhmDFjRoXb2RkpPLz43M97phTU1Mxe/phTU1Mxe/ZsHDp0CCKRCE5OTnjuecAlF5XUd10oGfP/7
T09HSkpKSUKkDxWEpKitoPjpkzZ2Lnzp1llnnnfjxo3w8fEp83IgRGDFiBBQKBBQKBQ4dOoQ
ZM2Zg7ty5pZZRsLa2xvtvvYW33noqt6/wyiuvVOOoDmqi+Yd+jnb6nLEl8JSWjT
pk21jiVqSNj3aKfvEYlEePXVG/G///0xx+IxWI8//zzzSEpKKvdYIm1iwYkz2YNI
iIiMGfOHPTv3x+mpqbYAoEzGFaxwDbVLmbzZlMlatqgpubG/z8/AAAbdq0werVqgEaGo
q1a9di4MCBpa7di4MCBpa7GVlVwcDBiYmKwfv16LwljzaFYQYGBsjLy8PFUtNTS8aeLTXH2dkrfmKwKyjMyJtYv4g4OT
D2v1BQUEYMWJEmedt6aqV2v0HDhzA7Nmz8euNevH06zdXKff3/8dAHDhwgVMmDAB
CxcuLFXcy8fHUvkdzfU9FYmLi8MPP/yAkycWBDgdT0vQsZjs0Kzf/FDdh1freKKGhH2P9
voePz8/bNiwARMmTKCxZ7mVfdzq+iwzg6Omd33jkgF1b7qd5gJLugY+bB3b2Bd
dvKq92RkREoE+fPCKiAUMJVIV48+u1Da2cfiaHLNmzSo1Bg2YPKUzRbgwEOdLgI6
qzpw5g/79+6N79+9+4wMDAAAPz9998AUO70oMrw9vZGYmIrKys0KlTJ+IcCDhKJRO1x
Dg4OKu2f/VNW4Ys2bdogPT0dGZzsUPt4XFwc2rZGcUPt4e3wWLZVZcBwdnZGXl4eNm7cqPZ1hcwD
oPBvucJTfc9FVm2bBmkUimGDBlSk5dFc0bKH1mKSEuKJFC/z55584cuQq2NTg8AR5k
bIzMwM586dQ2RkJNq3bw8PDw+sWLECa9asZUXDDNzEAahYwFvwpunfvjwTyZDg0WnXZ9
4//33ZEI8Ntvv8HAwAC9e/cu9RxRSRffPEFxo4di7Vr1yIwMLFPZ1xKy+fqmAlE
h3PnzmH16tUQiUTK+Kpr6ArnCh2LRpE/z9/TF+/HjY29vjv//+o+/+w9q1a/H+++X+lXP1q1bY+zYs
Vi9ejXu37+P1157DXZ+2dnj06BDli1bBi5t6ryf7G0rkIVM7NJh8x8M7ve6fDKu8bYAnYVdk6ZGuxtbXv
srC//3f/+HnTt3YtGiRbznkBoM9j9gOFBs7FDO8FJBkO2OB0bIpH+tXI0SPJpdU5JCcRk8f3q
FB4l0BfueJ7TV9zzt1VdfxcaNG8ejVKy13aeezMNXPdG2Qx8sT3wu0aLbTZQGqwRLtbXj3XfyE2NcVMxY
M2YM5s+fj3HjxHjLxiEtLQ0bNMkH16tXXrrfyn2tXcpDxfFmuRZIvm0hTpX5nQFhGYIx+LBpMhq
+Xo2LEjJfvrpJ7Ru3Vrt8/Tq1Qv9+/fHypr0b9//2pPedFF6wYAG++eYnYgHbe5VdZ2dnfPYPXVV/jjj
9w+vTpav87AMUVJX/55RcsxRwcYCxcuRFZlq0aIFPP/0UH3330UHUUY3OX3ZapU6fC3d0dW7duxdy
5c5GdnQ0Q0zMzN4eXlh27h27Zgg2bBLtu++GrVkKoSFhWq10fXAXZq2eNbEpdS7+sAbyC+L+cX/uXaCaxx
aQP7nie00fcAwLLZt25RrLevr68PeGPGb4tH4Sb8r8yNPe0Fbf8zQ/Pz+Eh4djw
IABFdZg2bNnD/dPqSvu0xCOvXFs0dpKqzhLsnnF2ZDr2NTkWkC+OuRREWIL16ERERERkCTMRERERGRGk
yYiYiIiIiNRgwkCkWEREEREEESkBhNmIiIiIg3btWsX/Pz80aP98/aN5U5BsB9rmBx/sb1L9
MF198EV26dMH777+Pa9euaTBaIGiwkzEREREN3MH8Hm5OUx9fs+bMwOHhhO0z779+2rHvhtCEn
nNr2c+bMwbZt2/DNN99+bNaNq0KcaMGCwMv/f+Ne4Bvx6tk+J+fsV+DexGDD3+/V+ef4//6/YknD
hKi3D969GH4e3tj3Lhx3j6Hp4kjJ2fvwKj8ePKaBVGr4w8Ho0OyfuwEpJijbH5XpM5fhbnqfpu+CfyPxAcwE
/bAdQHcrkCqak52g6DiCpgbW2q7RBHgSfooaohtvqf27dvvIyEhAYMVDYsAYMHD1bZHXx4EerrBv9v3//+/+/CzM
zM7z00kvKvKFbWjy5EiN4mDfQ9Q6xJ+bDJ+NOy2AxaUe7ubRFSjRGAAhCJJZsbJN
y0yA4+Li0LJlSxw8eBBDhw7lSxwPp6A2PaqrbyP3ZzEOBdjkB1yWKDMGLqXNRaKY2Z9lVI5wFCbe4rNmqHwrfmzF2+fqwYbNKvxZu0xAW7XDK6TKhXsa+Ex0ZGXTNX3xX6ZK1yktmF9YW2YyRzIe+F6ad6D2mY4wF87fnHKIIsyFfdfX+gYRS1

ETIqKKMGEmIiIi0hB9fX0AxfcsDxkyBADg7u6Oq1evqk2Y9fX1kZWVhSVLlsDFxQUAsGTJEvT
q1Qs7d+5EQEBAteIQi0WwsDCpwSshImocmDATERERaYidnR0AoG3btir7XV1dcfToUbXt9fT0
lMkyAEilUrRs2RLx8fHVjkOhEJCZmVvt44lIM3hhS/vqVcIcGxuLoUOH4osvvsDQoUPVtklJS
cH8+fNx/PhxAED37t0xY8YM5QcQERERUX3Vvn17mJiY4MKFC/Dy8lLuv3nzJhwdHUu19/Lygk
wmw6VLl9CpUycAQH5+Pu7du4eBAwfWKBaZTFGj44mIGoN6c/NKUVERgoODkZtb/tXOTz75BIm
JiVi/fj3Wr1+PBw8eIDAwUENREhEREVWfVCpFQEAAwsLCsGfPHty9excrV67E8ePH4e/vD7lc
jpSUFOTn5wMoTph79uyJkJAQnD59GtHR0Zg2bRokEglef/11Lb8aIiLdV28S5tDQUJiYlD/lI
DMzE6dOncKYMWPQvn17tG/fHmPHjsWVK1eQlpamoUiJiIiIqi8wMBCTJk3CkiVL4Ofnh//973
8IDQ2Fj48PEhMT8cILL2Dfvn3K9qGhofD29kZQUBDeeustZGdnY+PGjbC0tNTiqyAiahzqxZT
sU6dOYcuWLdi1axd8fX3LbGdoaAhjY2Ps2rUL3t7eAIDdu3fD2dkZ5ubmGoqWqH5QpCci/79f
AADSniMgbmqv5YiIqLFg/1Nz/v7+8Pf3L7XfwcEBN27cUNnXpEkTzJkzB3PmzNFQdET1F/sf0
jStJ8yZmZmYNm0aPv/8c9jbl/+GNzQ0xLx58/D111/Dy8sLIpEI1tbW2LRpE8Timg2Wc2kFam
iyTvwKefxlAEDBiV9hOihYyxERUWORH7lZ2f/kR/4K4wFTtRwRETUW7H9I07SeMM+ZMwddunT
B4MGDK2wrCAJu3LgBT09PBAQEQC6XY8mSJZg4cSJ+/fVXNGnSpFoxcGkFaoiyMhKf/JCRyPcw
EWmMIu3+U9sJWoyEiBob9j+kaVpNmHft2oXTp0/jzz//rT7vXv3YvPmzfjrr7+UyfGqVavQu
3dvbN++HaNGjapWHFxagRoihUJQ2U5Ly9FiNJrBiwJERERERpElaTZi3b9+OR48elbpvefbs2Q
gPD8fevXtV9p85cwatWrVSGUk2NzdHq1atEBcXV6NYuLQCNTSCIKhs8z1MRERERFS7tJowL1q
0SLlswmP9+vXD5MmT4efnV6q9vb099u3bh4KCAhgaGgIA8vLyEB8fX6kp3URERERERESVpdVK
V7a2tnByclL5AwBWVlZo0aJFqbUI33jjDQDAlClTcP36dVy/fh2fzzzzh48CB+/vlnWFlZaS
kyIiIiIqpLTJiJiCoglUrh5OSkss/c3Bx6enql9hMRERGR7mBZSyIiIiIiIiI1OMJMRKTGggU
Lyn180qRJmDRpkoaiISIiIiJt4AgzERERkYbt2rULfn5+6NSpEwYOHIj+/eX/eX2Xbnzp1wc3Mr
9efOnTsajJiIqHHiCDMRERGRBu3evRszZ85ESEgIfH19sqVKbnzp6Ymvvvo0Kr
732WrXPL5MpKm5EVI8IgqCyzfcwERHpKibMREQV8PDwwMGDB1X2/fzzzzh48CB+/vlnWFlZaS
kyIiIiIqpLTJiJiCoglUrh5OSkss/c3Bx6enql9hMRERGR7mBZSyIiIiIiIiI1OMJMRKTGggU
Lyn180qRJmDRpkoaiISIiIiJt4AgzERERkYbt2rULfn5+6NSpEwYOHIj+/eX/eX2Xbnzp1wc3Mr
9efOnTsajJiIqHHiCDMRERGRBu3evRszZ8JCEBgwcPVtk
fHh5e5jE3btxA//796zo0nZMfuRny+Msl27/CeMBULUdERA0RE2YiIiIiDYmLiwMA5ObmYvTo
0bh69SocHBwwYcIE9ONdoCLPP/+M9PR0dO7cGcHBwWjVqlWNYtHT0+1SN
kJ64lPb93X+9TYWIpEIwlPb/L1SXWPCTERERKQh2dnZAICQkBAEBQUhODgYBw4cQGBgINavX4
8ePXqotL958yYAQCKR4LvvvkNubi5BhBJmogqrpchvEbX/LDg7ROkZ6I/P9+AQBIe46AuKm9liMiIl0l

Eokg7RVQ7oVlAICBMaS9AtR+RopEouIk08AIgHWln1tQyJ4k0s+OWD+7T6Oj2XkQCvMgZKVU/
piKRrMNTUpGsp+ZVl7Ho9mcYUfawoSZqIGo6nIZipTbkNi41HFUROXLj9wMefzlku1fYTxgqp
YjIiJdJrFsAePXZiH/2Dq1n5l1lViJxHoQGZsDxuaVPqbc0ezHo9fP/txIR7M5w460iQkzUQN
R1eUyZHFnmTCT1inS7j+1naDFSKi2cEok1XcSyxYwfuNL5Gya8mQ0Ut8IxgODIbZuXW/epxzN
Rhmj2SYqybZgaILCU9s4w460hgkzUQMgT0uA7N7lKh1To+U1iIjU4JRIaihEIhEgefI1V2Ror
DMXkWs0mv30Pdj5ZSTXDWE0u6zTcYYd1QEmzET1lCL7EWQxJ1EUfQKKR3erfLzGl9cgIp3GKZ
FEDZfKaLZpHYxmP5VkCwXZgKKOR7PLwRl2VNuYMBPVI0J+NopiT0MWHQl54k0AQrXPpefctfY
CI6JGTSGXI/+v1ZwSSdTINMTRbM6wo9rGhJlIywRZAWR3zqPoViTk8Zdq5aqs2LoVxNatayE6
ItIVT77E5hT/Kcwt/qJamFs8WlSYU/zltjDnSZuSPyis/BdZTokkatxqczQb+TkouvUf5InXK
//8nGFHtYwJM5EWCAoZ5PFXURQdCVncWUBWUG57sWVL6Ll2h6SZE/IOraj2chlE1PAJ8iLVZL
YgB0JBbsm0yKf/Lt2mzov+lOCUSCKqqrJGs8UWzZG7e26lz8MZdlTbmDATaYggCFAkRaMo+gR
kt6Mg5GeV217UxAr6rj2KE2VLB+V+bSyXQUS1S1AogMLcUiO5qn9yi+8RLBkBFkpGgCEv1Hb4
FeKUSCKqLWIbF4itW1VqaU3OsKO6wISZqI7JUxMgi45EUcwJCFkPy20rkppCr3U36Ln2gMTWV
e0ocUNZLoNI1ymnOKtMbS5rirPqCHBVpjhrjZ5B8dIuhsbFceekVvpQTokkotoiEokg7RVQbt
FBAJxhR3WmXiXMsbGxGDp0KL744gsMHTpUbZuioiIsW7YMu3btQlZWFjp27IhZs2bB3d1dw9E
SlU2R/QhF0Schi46EIvVe+Y31DKHn3BX6rj0gcWgPkbji/5a6vFwG6Y6Gsl6v6hTnklHdejbF
udpEkuI1TQ2MAWkTiAyMITI0UfsHyu2SNhJ95WnkSdGcElnLdu3ahTVr1uDevXtwdHREUFAQB
gwYUOFxf/75J4KDg3H48GE4ODhU2J5IF0gsW3CGHWlNvUmYi4qKEBwcjNzc8itwzpkzB0eOHM
H8+fPRsmVLLFmyBGPGjMH+/fthamqqoWiJShPys1F0Owqy6BOQP7hZfmORBJKWnaDv2h16Tp4
Q6RtqJkgiDdH0er26PsUZahNdY+UIMMpKgvUMa+XiBKdE1q7du3dj5syZCAkJga+vL/bs2YOp
U6fCzs4Onp6eZR6XkJCAr776SoOREtUfnGFH2lJvEubQ0FCYmJQ/hevevXvYtm0bq9eDV9fX
wDAt99+izfeeAOXL19Gjx49NBAp0RNCUQFkd86hKDoS8nuXKxxtkti7Qc+l/Rbd4NI2kRDUR
JpVnXX663UFOenkt+GOcX5mYTWwEQ5Aix6PAL8zEgwDIwhEou1GjqnRNYeQRCwdOlSjBo1CqN
GjQIATJw4EWfPnkVUVFSZCbNcocBnCoNnn32TUKW7Zswa5d
u5SJsDr//vsvzMzM8NJLLyn3mZmZ4qciRIxqIkqhYcYXrK8XFuypT4dqqZXHxLhcfiJtYaShKI
u0QBAH5x9ZVar3e3D/mQmRuD5FyZLiBTXEuY1S3MlOcGyJOiawdt2/fRkJCAgYPHqyyPzw8vN
zjVq1ahaKiIgQFBTFhriSxRXPIsx+VbPN9SUTVo/WEOTMzE9OmTcPnn38Oxcu
WOHjwINasWYOkpCS0b98e06dPxaka4u6elp9+o91W+CoID8QTQKb0WiMLriCtdiM2sYtOkB
gzaqFa5rk0gkgvDUNt/DVB8okmMqNW0XAFCYByHltvJ9rFGlpjgbl5Hs1s0U54aKUyJrLi4uD
gCQm5uL0aNH4+rVq3BwcGRP/30E7Zt24akpKRai0XXPzdMXngfuf/+DA
AwfmEEJDr+ehsLfv8hTdN6wjxnzhx06/Kdl1JVWdbKzs3H37l2sl9qG2Zm8GZ88MxnBVUaxWGtfnkHSJMbZxzHGBEJtDvFbrJNP8ZGNbAZf/6/L78PjZZh+8g3Wb2nDDjE5lMnYtJ8uRBjtMk0Zh3Ma9lpkq7fhtJnVmm1GVU69+r7rpDyLn6T5uO8khfgvA
j3759sLKq3uidWCyChQUrelJphcl3kX3lb2Rf+ReyjJRy24qNzdCk/fNo0uFF+dfHLPE
IigePzffw1RPyO6c09yTVTTFuayktx5McW7IOCWyZrKzswEAISEhCAoKQnBwMA4cOIDAwECsX
7++1O1lubm5CA4ORnBwMJydnWstYW4UnxsWrkBr3vOta/j9hzRNqwnzrl27cPr0/afz555+Vaq
+vr4+srCwsWbJEOaK0ZMkSPq8OqgGdQSkI42gdH9ZW+Twd9HjK/1fJCTRtPmmo1skYTVNXS28V
J4jkG6VUCSJ9FSVB92TljmSM/PeMP/Mi4Jb1H9pD0c5vb8hHTnhs4xn3UDhvPyJkVMa9Iq+lTlHEZ/6lEFXrPlPDCS+FehW2RvbbG7TfIR6C
1a9q/v9V8ZHyoq6zwW0W
a+EgSu1m6Z1F0ZnFa4KY7LpqT2Uxe5BRjKkLSKBaWfv8Y3dlwLQq2fV3MlVFzWi/DZ9TzhMxxt97K2b7W+AfH
Hx/n
9J8yPcQsPXncYt3H08R
Iw6P4L5F5K/c3i7Q
KdpcjPguz2qcpVuBZLoNfSA3quPaDn1BkiveIK13IFAIXm3keCIKhs8z1M9UFV19/V7zwAUp9
36igaovrHzs4OANC2bVuV/a6urh ...

VhypQp6N+/P+7cuYMvv/wS6enpmD9/vhajJSIiIqK6woSZ6jVlhetbkSiKPQUUlF+gRGRmU5w
ku/pA0pQVrqlqkpKSMGvWLJw5cwatWrVSeey3335D9+7dMXbsWACAk5MTPvnkE8ycORNfffUV
DAw4vZ+IiIhI1zBhpnpHEAQoHt0tHkmOOQkhJ7Xc9iIjM+i5+EDftQfE1q1YTIiq7cqVKzA3N
8cff/yBsLAwJCQ8KRz30UcfQSwWlzpGJpMhOzubS9sRERER6SAmzFRvKDKTS5LkE1Ck3S+/sb
4Ueq28Sipct2OFa6oVffr0QZ8+fdQ+1r59e5WfCwsLsX79enTo0IHJMhEREZGOYsJMWqXIy4Q
sJgpFMZWtcN25pMJ1Z1a4Jq2RyWSYNm0aoqOj8csfHp6pUeudYXEsgVkJetlSixb6PRr
bUxEIhGEp7b5eyUiIl3FhLmRUKQnIv+/4i/20p4jIG5qr7VYhMI8yOLOoig6EvKEqxVXuG7hD
n2X7tBr9RwrXJPWZWdnY8qUKTh58iRWrp0+h8YrEIFha6+7428QvAo4PhAACrfqNhoM
OvtTHJEovwuOfW9fcwERE1bkyYG4n8yM3Fyy8ByI/8FcYDpmr0+QW5DLJ7FyGLPgHZnXOAvKj
c9mLrVsVJsos3K1xTvZGcnIwxY8YgPj4ea9euRffu3Wt8ToVCQGZmbi1EV0+Jm0L66qcAgBwA
OWnlF+6jhkGhEFS20xrB75UXBYiIGicmzI3E0/cEK9ISymlZewRBAXniDciiT6Ao9nTFa7Nb
aHv0r24eFdTO43ESFRZGRkZGDVqFLKzs7F582a4ubnV2rllsvJmWRDVP4IgqGzzPUxERLqKCT
PVqicVriNLKlyndteZNy0pMJ1d4ibObPCNdVb8+fPx71797Bu3TpYWloiJSVF+ZilhSUkEha
eIyIiItI1TJipVigrXEdHQpGeWH5jfSShUGDfvn0oKirCqFGj
Sj1++PBhODg4aCEyIiIiqpLTJip2hS5GZDdjkJRdCQUybfLbyzRg55jl+IK1y09WOGa6r0FC
xYot8ViMS5evKjFaIiIiIhIG5gwU5UUV7g+g6LoE5AnXAGeuo+tFJEIkubtoeR9aUuHawxdX
IREREEdUQE2aqkCAveqrC9fflKVLhuXZwku3hDbNxUCYiIiIiooaPCXMjIAgcJc9+bkgF/KkaIhtXFSqUhdX
uL6DolslFa5z08s9r7LCdZseEFs5scI1ERERHVNibMpKAorHDr4cMb39SErEREEdUpuUKA+KltorrGhFnHyVMTkH9sHYS8rHhbkCA8I
/KQu3suxNatIO0VAJFE70mF64D4ZZSQ7NWwwZRzvPxbcSeEEqwS94Pgp4PgrhuvVpvTY90EAXA2KuqqFO7hS
0mT01A7h/zypxOrUiJRe72LwBBUf6JJPrQc3qqwrVEvw6iJSIiIqn1whAKLH2xV8hyWqBdV
KmFNTUxEeHo7//vsPKSkpWLduHQ4dOoR27drhj5Zdfru0YqRoEQUD+sXUV3ntcZrIsEkHSokNx
8S7n5yAyMKr9IImIiIiIOqxKifM9+7dw3vvvvvYeCggI899899xuH79OuRyOWWLEv
r6+dRAqVYUiOQaKlNgqHye2aQ191x7Qa+x7Qa+0NsTEnuRARERUeNV5YT5u+++WcYgXx7++WcYGx
ujY8eOAIDFixcD8junKtSe7dwFBHFnWSNj8+YsvwjtXIf9o4TTaRtldgXYTrNpzgG2UHvoToMz+V8E
4OjoiKCgIAwYUMv28uXMcHBDQ0P069ozKeFJQweDSfI3x9w71pdSkBRqG0ZmCRDIjz/
ZmalqiIPGzYMt27dqrXgqpEgpwqtZdYNGeyTEREpAG7d+/GzJkzMWOzZswd+fn6YOnUqsAWQkYNlEknFGE8hPUFF4yxf
p0rfbE7OTkZ/v/v7+cHR0xM6dO7FixWWoI0sKPnZFUtikiTsgjB66q8qwdfh1xtYEvvhwmZaJwQP3rZdo
XanoiIiKpOEAQsXboUo0aNwhRhRo+Dk5ISJEyeKZ8+eIqKKtU+ISEBL774ubng1nZ2d07do
Vb7/9NiIjI7UQPRFR41PlKdleXl5Ys2YNvbsCUNDQwCASCr7/+ig5kFR1ek6e
KDy/t/Ltnfl7IyIiqmu3b99GQKICBg8erLPRFRFRFRmnfhvx/PPP12mcR
ERUrMoJ86effor+JlvEVFo33V9sDOeIDDMRYEKKM3NAUYKAfxKKwIYLg0AjFwQuthiF4HqbMBe6QDdKQgYmRjHgBTp5wl2ZoY9H8Kdlb
RSKRCNNJeAeWuwwwAMDCGtFcAp9ITERFpQHZ2NgAgJCQEJCQ4kQ4kQ4RSOOUQMBYd4xRB3cjBkgmMQgXNxeRJ8eZ
R67aNEi5OfnY9GiRfjggw+we/dumjhUJ75YqsVgECwcvejkUNz7NDwfU12rcsK8a9cudO/eYN
sXL66LeKGgWSSxbwPi1Wcg/tk7tSLPYuhwSkvQIgsWyhheiIiIgaH319fQDA6NGVQ2oCTCI6RIGc8
7rl69WmHC3KlTJBwBAaGgoevXqhf/7DG2+8Ua4FAoBmZnlXFAnaiDS6Lah4QUB7aty
wjx//nzwqmMCcPdnZ2dRC5dRFEp1TKJZQsYv/ElcjZNgZCxXMMhtm7NkeUGTGzRHXbscbkkbQlRUI
OhBRNQQP7+1LZtW5X9rq6u6aOHr0aKn2MTc2TYEGxsYG5ujqSkpBrpN8QgrFQX8MXxSzhF
3A9zHVtSpP+reyskJmZmZdxEJ1RCQSAZIn10ZEhsaQ2LgwWW7gpD2GQ+LQERKHjpD2eE/b4RA
RUSW0b98eJiYmHDhgsr+HDhgsr+mzdvwtHRsVT/75l2LaZmBhY//75Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9//75+Bx9
jzC/8847+Prrr3Hy5Em0adMzq1h+Dq1X9nMj8ZfZ2NW23K1Dey5CIrGAYE53kmr2D37Eg
1t4eHHg796OH78OCIiICXy5GamgpTU1NIpVKKK8/vrrCA8Px2effeYapU6xUicIfIyBKDWjFeH3
r37q3tl0NEpPOqnDAvWLAALB79261j4EkZERLixZgqSkJLi4uCA0NBQ
+Pj6Ij49H3759H3759MX/+fAwdOhQWEhttItP1SiIh0XpUT
5sOHD9dFHEREHERERESNhr+/Pz9Uvtd3BwwI0bN1T2tWrVapavr9ZUaET12kPBAk1FxbcopAgWa
KnleEj3VTlhbtHiSXGhvLw2nTpsqqj0SkGYmPcrD50C0AwPCX28DXs9WX4ceSR
RAkVwGADgi6oGuWo6HdF+VE2YAOH36NBYuXHlLy5BEAQAgIeHBz755755vBN07969VgMkVV+
LV2JTAQC/HY7GJ+901nJERERz9+nJERERERHXrEZpiVdYrAAArM6mWo6HGoMoJ89ZmpWCAwM/HOhx+5uWCWAM
RLNmzZCcnIy9e/kiCAAP//8MZw9PesiVqoBLkGkexIfPlk/8/5D3V6DkIiIiIhIG6qcMP
4/w8vJCEeHi4SrGJoKAgjB49SrSZ8Uq6kgtnyf3ncTs5aaz0cbZ5hJERERERERFanyOsyXLl
3CBx98UKoyvgsxvvvv4+LFy+w8+8jR4tZaqhisstSxxNnM+XfGb97+YvrQUtoSE0KTcGY2ToZvL9uctURERLqryiP
MJiYmkMlmyk0kMlkkMakah8rKpipS3tNhNMRYlGrqlbm8tZFVxzEtTXqvf2zxxMT5U5c3OzsaaNWg

5eVVqwESERFR/fJ41hIREZGuq3LC/Omnn2Lo0KF4+eWX4evrC2tra6SkpODo0aMoKCjAt99+W
xdxEhEREREREWlUlRNmJycnbNmyBcuXL8fff/+NjIwMmJubw8fHB0FBQXB1da2LOImIiIiIiI
g0qlrrMLu6umLWrFmwtrYGAKSnp+PBgwdMlomIiIiIiEhnVLlKdmZmJvz9/TFy5EjlvosXL+K
NN95AYGAg8vLyajVAIiIiIiIiIm2ocsK8aNEi3Lp1C1C1OnTlXu6969O1asWIHLLy9j2bJltRog
EREREREREkTZUOWE+cuQIQkJC0fP+U+AwMD9OnTB1OnTsX+/ftrNUAiIiIiIiIiIibajyPcw5O
TkwMzNT+5iVlRXS0tJqHBQRERERUU0kPsrB5kO3AADDX24DeyuTCo4gIiqtyiPMHHTp0wPbt29
U+tmPHDri5uUVU7mNjYWWHh6emLHjh2Vav/nn3/nn3/Czc0N8fHx1X5OiIiI1tI9vx6+hSuxqbgSm4r
fDkdrOxwiwiaqCqPMI8YcIEj8BkzBkOHDsUrr7wCKysrpKam4mjVhw7vDhw7hy5QpWrVpVrUCKiooQHByM
3NzcSrVPSEjAV199Va3nIiIiiiLdlvjwyXfK+w9ztBgJETVkVU6YTV1/+eaxcuRLLli3Dsm
AgCRCIR3N3dsWLFCrz00kvVCiQ0NBQmJpWbKbNQKPDpPDZZ5+hQ4cKOOHHiRLWej4iIiKg8sOHnH4RRVB2uwRLWej4iIiIiKg81V
qH+bnnnkNYWBgsLCyQnp6OrVu3Ij09HcbGxtUK4/tSpU9iyZQt27/fX1/fCtuvWrUKUKRUVFCAo
KYsJMREREREREdaLKCfPFixcREBCAd955B8HBwVixYgV+//13jMqaYvmPmzQfvb7fXfzvn0ybXfvvn0rfb7M
zExMmzYNn3/+Oezt7Sv1/D99D2bduGpKSkqoZfJDj29Kt/OTaRVIpHqNt/DRERES1q8oJ8
5IlS9C6dWsMGzYM+fn5+/Dee+/hy/yy+/xJdffolVq165dOH36NP78889P+/p9r165dOH36NP78
HBwQgODoazs3OtJcxisQgWFqycSA2LWCxS2eZ7mIiIiodlU5b5w4QKbYDyA2jcjIAjIyIcQKGyLgn4qBZnhb2f+yy9Tmf4ntfZ2pVi//797PX
+fj5ef/11AICfnx/++OP9Sp9r165dOH36NP78889KtZ87dy6rvvvVjXscikUAjIzK1ds
jKi+UCgEle20NN0vaMKLAkT1A5frISKixqLKCbNYIxYzAzM4OHhwcAQCgQAGFyLIaBgQEA4NixDR
NLn2r59Ox49lqTqvuXZs2cjPwce/fuLdXewEwAnp6eAAC5GDRoEFA4EF577TX577TV8fvnNXp0SjRsTAB+x
KZotrHEmmDIKhu8z1MRJryeLkeAPjtcDQ+eaeziItIiZKqvlSGZBKpVi+1x1rmVaQ7hlyP
1hUgkwqNHj7B27Vm69VcPp1FsXfb169XsR169DsI89vcPkn2BXXVPn5woUL+0yzzz7Bm
zRq4uLhU37sWEsX1Qb3v37sH8sHR0RFBQUEYMCA2ra3bt3CwoUL+ceHCBYjFYYjF
nTr1g3Tp09H8+bNWw1EVdq0aDQgICMDevXuvvXs6KPWgQgICMDevXt0y0aQgICMDevXtWxtbd
Xut7KyQosWLSCXy5GamnKLqMjqNIpsS2Q==

T0lBbWTfDFKC9MXX4cGTnFt9IYGUgw9d0uaG1vVqujsYb6EthZGsPOUv2tOYIgILdAhkcZqkl
0alY+Ukv2pWUVQK6o/tzvgiI5Eh/lIvFR2bf3NDHSV6n6bWX21BRwUymamhpAIub91ETqMGEm
IqoEKysrrFu3DvPmzcOQIUNgbW2NadOmYciQIdoOjYiIniESiVTWUjaW6sOlublW4jCR6sNEq
g9HW1O1bRQKARk5hcpp3sqk+qmp348T/+rKzitCdl4R7iZllxGn6lJaKgm1GZfSosaNCTMRkR
oLFiwotc/DwwNbtmzRQjRERKSrxCX3KVuYGsKlufo2RTJFcYGyjHzVxLpk1PpRZgHyyrlvuyK
VWUpLX08MC1M1Vb81uJSWIAiQyZ/UTMjNL0JMQgZaN6/dmQNET2PCTERERERUj+nriWHT1Ag2
TY3KbJNXICuVRKeV/Py48vfTyWZVFckqt5TWs0n0kyngUlg0Maz2UloJKdkI33tNZbQ9r1COe
T+fgbOdKUYPrNy96URVxYSZiIiIiKiBMzLUQwvrJmUmjYIgICu3SDlCXVdLacWnyBBfzlJa5i
YGT91PrXpPdVlLaSWkZJdb/TzuQRbmbzqLGe93ZdJMtY4JMxERERGRjhOJRDAzMYCZiYFWl9L
KyClERmWW0ipJpi2aGCLqWlK5S4UBQG6BDD/tu4bPP/Di9GyqVUyYiYiIiIioWktpPXs/de0u
pVU1sYlZuJ2YqZUCb6S7mDATERERERVGl1IeltMpz7uZDJsxUq5gwEzVArBJJRERE9ZG2l9LKz
a/ZlHGiZzFhJmpgWCWSiIiIGrKqLKW189htnLyWVOlzG0v1aylKomLVq+tORFrxuEpk3AP1hT
IeV4lMSMnWcGRERERERtefxUlovezlU6TjPts3qKCJqrJgwEzUQgiAgfO+1SleJFGqyLgQRERF
RPdC6uRmc7dRP7X5WK3tTtLY3q+OIqLFhwkzUQNy+n1nmyPKzHleJJCIiaqzsmz0pStW8mYkW
I6GaEIlEGD3QHcaG5d9Jamyoh4/83FnLhWodE2aiBuLElQdVan/u5sM6ioSIiKj+e69vG3RoZ
YkOrSzxbl9XbYdDNdDCuglmvN+1zJHmVvammPF+V9ZwoTrBol9E9VhWbiHO3ExB1NUkXL+bXq
VjWSWSiIgaM3srE3w6rIu2w6Ba0sK6Cb4Y5YWpy48rC58aGUgw9d0uaG3PVUKo7jBhJqpncvO
LcPbmQ0RdT8LV2DQoqnkvMqtEEhERkS4RiUTQkzyZIGss1eeay1TnmDAT1QP5hTKcj36IqKvJ
uBz7CDJ5zQt2sUokERERERVHNMGEm0pLCIjku3X6Ek9eScTH6IQplinLbW5oaQK6AyvrLZWGVS
CKi+m3Xrl1Ys2YN7t27B0dHRwQFBWHAgAHlHqNQKDBmzBh06dIFkyZN0lCkRESNGxNmIg2SyR
W4HJuKU9eScPbWQxQUysttb97EAN3a2cDb3RYuzc1w/2EO5m86W+7SUqwSSURUv+3evRszZ85
ESEgIfH19sWfPHkydOhV2dnbw9PRUe0x+fj5mzZqFf//9F126dNFswEREjRgTZqI6JlcocP1O
Ok5eS8K5mynIyS9/HeUmRvrwamcDH3cbtHFoCrH4SeL7uEpk+N5rapeYamVvio/83Fklkoion
hIEAUuXLsWoUaMwatQoAMDEiRNx9uxZREVFqU2Yz549i1mzZqGoqAhmZpw9RESkSUyYieqAQh
Bw6146oq4l4/SNZGTlll+x2thQD13drOHtbgN3JwtIxGWv+MYqkUREDdft27eRkJCAwYMHq+w
PDw8v85h//vkHr7zyCsaOHYvXXnutrkMkIqKnMGEmqiWCIOD2/UxEXUvGqetJSM8u/15jQwMJ
PNs0g3c7W3RoZQl9vcovi84qkUSkLYIgQCZ/UnMhN78IMQkZaN2cF+wqIy4uDgCQm5uL0aNH4
+rVq3BwcMCECRPQp08ftcd8/PHHdRKLXhU+d4jqi6e7GZGI72Oqe0yYiWpAEATcTcpG1LUknL
qejIcZ+eW219cTo7OLFbzdbeHhYgUDfYmGIiUiqrmElGyE772mUnwwr1COeT+fgbOdKUYP5C0
hFcnOzgYAhISEICgoCMHBwThw4AACAwOxfv169OjRQyNxiMUiWFiYaOS5iGrT07eq8X1MmsCE
magaEh7mIOpqEqKuJSEpLa/cthKxCJ1aW8Hb3QadXZvByJD/7Yio4UlIyS636GDcgyzM33QWM
97vyqS5HPr6+gCA0aNHY8iQIQAAd3d3XL16VaMJs0IhIDMzVyPPRVSbFApBZTstLUeL0dQ9Xh
DQPn5zJ6qkpLRcRF1LRtS1JCSklN85i0UitHe2gLe7Lbq2bQZjqb6GoiQiqn2CICB877VyK/Q
DQG6BDD/tu4bPP/Di9Owy2NnZAQDatm2rst/V1RVHjx7VaCyyCpYzJKqPBEF1m+9jqmtMmInK
8TAjD6euJyPqWjLuqKlK/TQRADfHpsVJsps1zIwNNBMkEVEdys4rwunryWor86sTm5iF24mZr
KtQhvbt28PExAQXLlyAl5eXcv/Nmzfh6OioxciIiEgdJsxEz0jPLihJkpMQk5BZYXvXFubo5m
4DLzcbWJgaaiBCIqLaJQgCUjMLkPgoB4mPcpH4KAf3S/6uqMq/OuduPmTCXAapVIqAgACEhYX
B1tYWHh4e2Lt3L44fP46IiAjI5XKkpqbC1NQUUqlU2+ESETV6TJiJAGTmFuLMjRScupaEG3fT
IVTQ3snOFD7utujWzgZW5vxCQ0QNg0yuQHJankpCnPgoFw8e5aKgSF5rz5ObX/UkuzEJDAyEk
ZERlixZgqSkJLi4uCA0NBQ+Pj6Ij49H3759wwMX/+fAwdOlTboRIRNXpMmKnRys0vwpmbKTh1LR
lX49KgEMpPk1tYm8Db3Rbe7jawtTDWUJRERFWXVyDDg9Rc3H+Yo/w78VEuUtLzIFdUdEmw5li
3oWL+/v7w9/cvtd/BwQE3btwo87gjR47UZVhERPQMJszUqOQVyHAh+iGiriXjcuwjyOTlf3G0
tTSGdzsbeLvbsOorEdUrgiAgM7cIiQ9zSo0Yp2UV1MpzWJgaormVMYwM9XD6Rkqlj/Ns26xWn
p+IiEjbmDCTzisskuNizCNEXUvChZhHKKqgmqKVmRTe7W3g3c4WjrZNWOmViLRKoRDwMCMP90
umTt9/VJwgJz7MrbBqdWWIRSLYWBjB3soYzZuZwN7KGPZWJrCzNFYugycIAr7ZcLpShb9a2Zu
itb1ZjeMiIiKqD5gwk04qkilwJTYVUdeScC76IQoKy783r2kTA3RrZwvv9jZobW/GJJmINK5I
JseD1JL7ix8+Lr6Vi6S03Aov9FWGob4EdlbGaG5lDDsrEzQvSYxtLIygJxGXe6xIJMLoge7lr
sMMAMaGevjIz519KBER6QwmzKQz5AoFrt1JQ9TVZJy9mVLhyIupsT682tnAu50N2rRsCjG/4B
HVusRHOdh86BYAYPjLbWBvZaLliLQvJ78IiQ+fTJ9+PGL8MD2/woKDlWFqrA/7pxLixyPGFma
GNernWlg3wYz3uyJ87zW1I82t7E3xkZ87b18hIiKdwoSZGjSFQsDNe+mIup6M09eTkZ1XfmVW
Y0M9POdmDe/2tmjn2BQScfmjKkRUM78evoUrsakAgN8OR+OTdzprOSLNEAQBaVkFTyXEuXhQc
p9xZk5hjc8vAmBlLkXzZsVTp5+eSt3EqO4KbrWwboIvRnlh6vLjyCh5HUYGEkx9twtn5xARkU
5iwkwNjiAIiLmfiairSTh1IxkZ2eV/+ZQaSODZphm83W3RoZVlhVMPiaj2JD7MVW7ff5ijxUj
qhkyuQEp6Hu4/zMWD1BzcfzxynJpb4a0glaEnEcHW0rjUiLGtpTEM9SW18AqqTiQSqfSjxlJ9

rrlMREQ6iwkzNQiCIOBOUhairiXj1LUkPMosvwKsgZ4YnV2bwdvdBp1aW8FAS18siUg35BcWL
9OU+PDJiHHioxwkp9XOMk1GhnqlplDbNzOGtbkRxGKO2hIREWlLvUqYY2NjMXToUHzxxRcYOn
So2ja3bt3CwoULceHCBYjFYnTr1g3Tp09H8+bNNRwtaUJ8SjairiUj6loSktPyym2rJxGhU2s
rdHO3QRfXZpAa1Ku3NxHVc4IgICu36Jl7i4sT49QKLtJVVtMmBiWjxcUJsb2lMeybmcDcxIDT
mYmIiOqhepNRFBUVITg4GLm5uWW2SUtLg7+/P7p164ZNmzahoKAA3333HQICArBz504YGhpqM
OKGpSEV3nmQmouoa0k4dS0ZCRVM4ZSIRXB3toCPuy082zSDsbTu7t0jIt2gEAQ8ysgvqUb9ZA
p14sMc5OTXzjJN1hZGJdWojYuT45KR48fLNBEREVHDUG8+uUNDQ2FiUn4Sd+jQIeTl5WHBggX
K5HjhwoXo1asXzp49ix49emgi1AapvhfeeZieh1PXk3HyWhLuJmWX21YEoJ2TBbq52+C5ttYw
NTbQTJBE1KAUyRRISi0eKX7wdPGt1NpZpslAT/xUQvxkOrWNhTH09VgrgYiISBfUi4T51KlT2
LJlC3bt2gVfX98y2/Xo0QNhYWFqR5IzMjLqMMKGrz4W3knLKsDp68XTrWPuZ1bY3tXBHD7utv
Bys4Z5E84mIKJiuflFqlOoHxaPGKek50GohXWamhjpq65dXFKR2tJMyuXoiIiIdJzWE+bMzEx
MmzYNn3/+Oezt7ctt6+DgAAcHB5V9q1evhqGhIbp161ajOPR0fDTg6e90IpH2Xm9mTmHxSPKV
B7hxN73CNUdb2ZuhewdbeLvbwspcqpEYG4r68jslKosgCJDJn4zk5uYXISYhA62bV335IUEQk
J5dqDpa/LA4Qc6ohWWaAMDKTAr7ZqVHjDmLhYiIqPHSesI8Z84cdOnSBYMHD67ysRs3bsTmzZ
sxY8YMWFlZVTsGe09vbXi6yqqmX292biEiLyXi7/MJuBj9EIoKKso625vhxS4t8GK
XFrBvptu/l5rQ5u+UqCIJKdkI33tNJZnNK5Rj3s9n4GxnitED3dHCukmp4+QKBVLS85H4MOep
olvFSzblFdR8mSaJWAQ7S2PYPz1ibGUCOyvtLdNERERE9ZdWE+Zdu3bh9OnT+PPPP6t0nCAIW
Lp0KVauXIlx48bhww8/rFJy8pTVIVCQFpa3U7LziuQ4ezNFJy8moRLMY8qXH
bF3soYPu1t4dPBDi2eSpLrOs6GTNO/0/qAFwUahoSUbMzfdBa5BeoLaMU9yMK3m87iw1fbQaZ
QKCtRJz7KRVJqbq0s0yQ1kDxZu7iZibIatXVTKSRizsYgIiKiytFqwrx9+Y8evSo2H3Ls2fP
Rnh4OPbu3VvqmKWXO9Dqs6fv4xOEunm9BUVyXIx5kunKhrS
bgY86jCojrNzKXwdreFt7sNWto0UU7R1PXfRW3RxO+UqKoEQUD43mtlJsuP5RXIsHL35Ro/n3
kTAzQvGSF+eip10yZcpomIiIhqTqsJ86JFi5Cfn6+yr/+/PzUHjNt2jT83T83Y//yzsHT
vXoyBAwdqIkwqR5FMgcuxBlLRnnbz1EQVH5UyYtTA3RrZ0NvN1t0crelF9oiXTM7fuZiHuQ
VavnFIkA66ZGpe4ttrcy5lJyREVKe0mjDb2tqq3W9lZYUWLVpALpcjPTUVpqamMFEbq2LFjB
/bt24dp06bB29sbKSkpkpymetMett6G6J5MrcO1OGqKuJeHszYfIq2AkycxYH14lSbKrgzmryhLpsH
O3Hlb7WAM9cfH9xU9Noba3MoYtl2kiIiIiLdF60a/yJCYmom/fvz6GDh2fVg4QZHXBB6ggEqAADDOOFPrFbD00AAC+//5
7fP/99yptH7ehuqFQCLhxLx1kYKsvOKym1vItXDc27W8Ha3hzc8zSNRI5OaX3zc8
y6W5GQY/3wr2VsawMucyTURERFS/1LuE+9GPPqn8cXPHz6Fg54XBQRoNN9KP2cJAERERVJfqXcJMta8qa6EKo
C4B1k4dS0ZUdeTkJpZUO65DF9DDN3a2cLDxRL6eelyWhagx8+iYDiqU2oaWxy1Hsx0Ue44hCgBOsJvqmmjd7ZDhgHBO
jI4KCgjBgwAC1bdPS0jB37b3lz8/fffAIZq5Pff6bg4zMMD5oj97ys7iQNrVpksGk6P6jf7GmtEs6m
iopphwqzjKrMWavNmJkhIycHJa0k4dS0Zyel55Z5TTyJGp9aW8Glvi84uzWBowCSZiIq1bm4G
ZzvTShX+amVvitb2ZhqIioiIiKh6mDDrsMqshzxy1yyjGmYNzHAw4x8tW0ek4hF6NDKEt3a2cCzj
TWMpXzrEFFpIpEIowe6l9v3AICxoR4+8nNnpXwiIKq15j16KjKroVaJFeUmLREA7Rwt4u4
vgOTcbNDHi8i1EVLEW1k0w4/2uCN97Te1Icyt7U3zk544W1k20EB1R/bBr1y6sWbMG9+7dg6O
jI4KCgjBgwAC1bdPS0jB37lz8/fffAIBXX3OVM2bMgLGxsSZDJiqarIXaxsEc3u62
8GpnA3MTg1qOjKhhqvwvLLy7F7925kZGTA3d0dwcHBh4Nq1q7ZDq1daWDfBF6O8MHX5ceUtI
UYGEkx9twta25eun0DUmOzevRszZ85ESEa9PQs1X7y5MkoChARE
QEMjMzMWvWLHz11Vf47rvvvtBA9EVHjwoRZR1V1LVRzEwO86uOIbu1sYGnG9ayJyJy5Ups374
dCxYsQMuWLbF27VrF27VqMGT+/btK3NcZm5FiMi0j5BELB06VKMGjUKa0aN
AgBMnDgRZ8+exbu3kZe+8eRVRUVKmE+dy5c4iKisK+ffi/vg4uICAMHjw4cxcOBAzR
VTPNs3Q39uRyTJRBQ4fPoxBgwbhhRdegJOTE6ZPn47s7GycP39e26ERUQw+/ZtJCQkYPDgWs
r7w8PMW7cuFLtT58+DWtra2WyDABdu3ZFUgpjbevl2bZxzwbk+aWGP6RKHCwFuKb0t4OZ7owjzMlvNxowD0pWDoz5fr8I0ONJ8N+2TtVq8x3fiFGqNa8
//voL77//Puzt7ZGYmIjw8HAMGTKkzPaXLl3CjBkzsRNAngKSOrJhxVS3n4DSuAeSQ2W7uuHT
9klJSbC3t1fZZ2NjAwBIS0urlXiI6oJcIeDu/cc4+yAFCQ9zSrUzMzLR1/a2cDTxq2kKpDc3
+jbty8kEgnEYjGWWLl0KR0fHfk4TkmNe0HhA5kw/+5+7/zhi+c4Y3DcvAlIMR0uFy+EhcbjqnXPWfL
9fjx49eqi0z0kLJBQyqS0dO5XqVVNkb4sidGtdqYi+SyT1ozgKdNmmYKg8Q1WXmvWSPUalNBpZkNsq
iYmJgZmaGL7/8EnVq1cLLL7+M5OTksqjfTwlNUoo2QU0mtYUv+c/4hdjPDlLGZ7WYVdoJKYdPdn8M/8PfnlqPHdY93UKX7VU5s2b
NhvdIeQdxEP/5TL/SGDx2wOYY+/w8aJnxNH6sahesIsiLfsPYHoooPXd7mbshuqXNzMq7VPQ42WEpJSRjTLKXLN2R+/wiYp
M7pGjx6NIUOGYOjQoejQoQNCQ0O1kBGoE4gQDjCLKD7vVbmrICRfrJcnw0QYKZjH5nXpnx25+3oLSCX9JJzNP9T4M2SeJqmnEvsQUxZsMu3T+lmEKgomxwlUS
9NGp6Jjj7KLcGPJG5dMjrl1wK22dwd7vdNZ6oLPZnWOiO0+HmDCS8uo3w7
4oLCzug97p7aLzX2wai8b2hRWovS+tdnZ2AIC2bduq7Hd1dcXu3bvVdvDjb+zuMn7GN6bWV5vXtndi5JKesGSdb7eiS
4J6nKEU1f6ulpMeA6JIqS+1KVpOnKbJ8zUA1xy0FIe7FuJWUyqhwYjI6NeCg8IuqRt1BjvOWe6Fh5uF
EpBBBBw9c0cYh4+XPNH3wPHNw1YD4tvK4ICuH1H5hxvHp9sxCqL3o43bY
RIAuJwdEZEG8KYjHfd4LdSn11M2MpBg1gfP4fMPvJgsE1XB48I7N27cUNl/8+ZNODs7q72Nt7c3Hj58iNjYWOW+sLAwAECnTp3qLlirC47G5eJ
3N1zKY1gtwdG9KBKvZzn9MlSvN7V9lWcdKJJw0T9pdOcXkUAq5xgy1h/CUjDZ5jKb7Xfe4Ma6hqO3R9nGRkPvAVpT8lMvAqn8o0tRgSLrW6LJ
tJPbGcHqnDEDe2ciBhTrXV0fNSJVMnWIKFBQYD4Bq8mnDFg1r7dtLvEYK9XsQSZaMzDg2uVmvUe09NSObBLq
8QrugCZpXKoI1CAFtDeD2IkkQktFUOwOCeWDsZDQgy4HNabv/1GOR9rJQWkHJvszbi0B1lW0V+wOO5bL6iFl
ZlPaQQVJr3IxRsNvHsrPPKb1UFPw9hDp5ZsSMKS6BA6T6CLTmCPQe93ORGybNafdZ4Ku3KjCKtUcjdkYnpTPoKiM8SIXJvVVtYJB9tO
zFUXv6WFqkYbmzLRQ7ZFw69lxxdD6xq7hNWrWdPPRiwbiDZJIfzx54RWxB4jM5kwRfpYnYPQOzxbwBUX4JY5x6pIn9eWSaz+Lgd7XQn5uKA9tPSpfVbG5dYf+2IfbDH51ZGGEW7Gv9AjG8s6Kc
WEaEPwdeoVm0yDvXK7C0TFV8OFtROE7rdycFNXsGUx5NrFTiEXcmhfGJPHS8fZdBJ9c37kfDaSPmgMrggZu6zUtkqbyYuHo6UUCB58rKn/fA8Oun
ab3jY3SfG2EhpZjCrZSLyCs8XH2MPoKRcK0aGsJfuwnUGnn0MCIGfrxLLiSDyrmWlRqKx9kIXlMLcvl4Ay6eK7XpGV6EvjOdA6BkHdSDGYhCqHh9Z7
jdbFW4o4Flt4wEZtyChIepaw2lwRLKo6PiPdW+u0bpq5q7V5gQ9FEodPGSHm4SDFsTdJU7NIEv5thYgfhUE7V9FrOpcHVqUXJBxgwZmhhJbEYGpDgAw
0ExkqPM+OvzPCzZfELOIJyyF3eIhV0Sk3nNnR4QJtOyWWb6rFtwsQzyF2zfmYyPzm5VsxG1nALEVbQgD5bf5g1mKuCSBW3l4T2U0fRjR1rlUrvP0

VSBAQEICwsDHv27MHdu3excuVKHD9+HP7+/pDL5UhJSUF+fvGtDJ07d0bXrl3xySef4OLFizh
x4gRmz56NN954g0tKERFpAEeYGwGuhUpUOzw8PODl5YWQkBDMnj0bdnZ22LVrFyIjI7F582Zt
h0dEDURgYCCMjIywZMkSJCUlwcXFBaGhofDx8UF8fDz69u2L+fPnY+jQoRCJRFi+fDm++uorj
Bo1CoaGhnj11VcxY8YMbb8MIqJGgQkzEVElicVirFixAj/++CNmzJiBjIwMtG3bFhEREejSpY
u2wyOiBsTf3x/+/v6l9js4OJSaxWJlZYVly5ZpKjQiInoKE2YioiowNzfH7NmzMXv2bG2HQkR
ERER1jPcwEzVQ9s2eLCfSvBmXHCAiIiIiqm0cYSZqoN7r2wabhVsAgHf7umo5GiIiIiIiUoMJcyPBAlFERFRb+JlR
NrC/oc0jfcwNxIsEERLWFnylEpC3sf0jTRIIgCNoOQtvkcgVSU3O0HQYRVcDa2lTbIdQ69j
9EDYOu9T/se4gaBl3rexoiTskmIiIiIiIiUoMJMxEREREREkY7gBGJVNf33YMJMRERERERESkAw4
fPoyQkBDlzydPnoSbmxtOnjypxahqLj4+Hm5ciXCw8Nr9NxMmImIiIiIiHRA
REQEEhMTlT936NABW7ZsQYcOHbQYVc3Z2Nhgy5Yt8PX1rdJxP/74I/Ly8MmrO3KySTUREREREp
IOaNGmCLl26aDuMGjMwMNDa6d7AGBf37doVEyd
OxL1798qNafr7+Oc+fO4fz583j77bfh4eGBQYMGITyUuX8N2/exLhx49C1a1d07doVEyd
OxL1798qNafr06Rg5ciS2bduG3r17w9PTEx988AGuXr2q0i4lJQXDhw+HjQ2PZ5x8a1b8vp7+Aa1a1d07doVEyd
c+aM8vFnp2Tv2LED7du3x4ULFzBs2DB06tQJvr6+WLt2rfIYNzc3AMDy5cV29XBhJmIiIiIi
KiBmz17Ntq3b4/27duXOw37wYMHmD9/PsaPH48ff/wRGRkZmDx5MqZOnp33nkHP/zwAxQKBT
755BPk5+cDAGJjY/Huu+/i0a0WHNBgAebNm4d79+7hvffwNHj8qN69q1aiyZAmCgoKwcOF
CpKenY+TIkUhKSgIAREdhY+jQobh37x4/x4+//LFq0CCKRCKNGjUJUVFSZ51UoFJgyZQr8/Pyw
Zs0aPPfcc1i0aBH++ecfAMC4WLVsAAG+99ZZyuzo4JRuAWCyCpaWJtsMgokaI/Q8RaQP7HiLd4
+rqiiZNmgBAudOX8/LyMHv2bLz5ZzsMzt2eqp+1j5SxxYmNBvx+ZsmBjDW2+9BQCQy+WYPHkyYmNj4e7uju
XLl0MqlSIiKL5HD169MDLL7MdevWqRQae1ZWVhZWrlyJbt26AQA8PDzw8ssXsvIyIiAiEhIVi
+fDn09fWxceNGmJoWrzvt6+uLQYMGDAQLz99tsAg0eeew7/93//h6NH
j+LFF19U/hvY2dnVaDo3E2YAIpEIolI22EQUSPE/oeItIF9D1Hj1rVrV+V2s2bNAKfm2U2bNAgm2U2bN
gUAZGZmAgBOnDgBHx8fSKVSySUGyAMX3R3t5eeG///4r97eyinXUVFR6N
27tzJZBgA9PT0MHDgQP/30E3Jzc3G3vuMCzPAwDGQYWFhyMnJKfPcANZ2kFRI/J4lPh
pUqm0zPbp6enYt28f9u3bV9oavS/oxScSvwcp/LxZq4Lxsa1D4rKytcuXIFAJCRkaFM2FU2p/WrFkzCIKAjMzM0s
Ms/9bMxisbjW16FmwkxERERERERElMjU1RsBeXNePVwd1X22bt8ud/qE3PnwIKysrAIC5uu
TkePnxYqk1KSgoAwMgCAsnJydWdWInunYwYSYiIiItIiItIBYrEYCoWi/r7e2N6OhouLu7k7QxhtuNPvT8mVzkFaPmzCKQOR
AQHBwMJycnuLu7l3ns3bt8ER0dDVXdWBAU1ISzp8/jzFjxgAAunbHrxhr/++gtZWn8ubwatoxKadlyuRx
79+5Fp06dYGUGuO24xekaWwiIiIiIiIdYGZmhtjYWERGEiJjI6PWzhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmKhsYGIi7d+9+ZnmK
OHToeP755x9MmjQJe/fuRbt27co99nFxcnfQnf379eY/Emx0oinvj8/eAgQMICAiAmZkZRo4cCQAIcgpCYWEhPvjgA
+zfvx+HDx9GQEAA7t27h61Tp9YobjMzM5w7Dw6nTp2q9l1tRtJsxERERERERREQ6YMSIEdWMTYMW
Pw999/19p527Vrh19++QUikQjTpk3D5MmTkZKSgrCwPMPTr16/cY5s3bw5/f398++23mDlzJpy
dnfHbb78pC4uadMGmzdvRvNmzTBz5kx89tlpnEAQBGzduRM+ePWsU9/jx4jxF0ZH0oFeQGMSGTMeGiYmJ
1TqHSKjtu6KJiIiIiIiIio0Zs+fTqioqJw5MGbYdSbRxhJiIiIiIiIlKDCTMRERERERGRGpySOT
URERERERKQGR5iJiIiIiIiIiIi1GDCTERERERERERERKQGE2YiIiIiIiINZgwExERERERERHanBhJmIiI
iIiIhIDSbMREREREREREREGroaTsAIiIiIiIi0jxBEHDjbhpOxn6A7g5WkAkEmk
7vHqB6zA3UoIg8D8BEWkU+x4i0gb2PUSkDRXpe4mG9hv6n7yE76zhJIQuh1d7RI5UoIgD8BEWkU+x4i0i0ib2PUSkDRXpe4mG9hv6n7yE76zhJIQuh
G6HDhw/jwIED+P7772v1vCV
+PRt2/fCs83f/58DB06tC5CJao+WPHjsHOzk7ZtbtCgQVi8eLHK8eLHatu+88w
4uXLiAoKAgJo0qa5CJqpV7IO00ozrfZY4cOYINYINGzbypUrYNmzR1
yo3HnQSZClv+LnLyiMtuYGOnju6AXGn3SzCnZjVBERIS2Q2j03nrrLbz99tvKnwsLC1i2s
WrUK/v7+2L9P2xsbLBlyxLm5SUFAQWBWHChnw9fVV7nd0dNRk6ETVwn5H+3r16oXAwEC1j
1laWiq3xhw5goKCAhgaGqq0i0ePHP4ULF+o0TK6WiXbnhnzp2YPn06hg0bhg8//BC8bhg8//B
BGRkaIjo7GmjVr8Ndff2H79UL2rSphl+F7hEEAT/+dq7cZBkAcvKKs1Mj3iUCZ2kk6Nf3xHjx9G8eXMLFuHTp
Nc+bMwR9//IErV67Ayt0ZMUV5kCQ0NxfHjx9G8eXMcPXoUr7/+OmbPo2zZ89i8Le2zZ89i8eLFuHTp
EiwtLdG7d298+umnaNKkCQDHIbjStKozs7dOnSRWWfs7c3HB0dERAQgAMHDmDEiBEiqbeljf
wEUf6g8eywRUUUsLS0r1Xd07doVp0+fxrFfjxx9CvXz+Vx/bt2Wd3d3dcu3atjjqIkIl1iYGBQpe
8yYWFhGDRoEL7++++mvlvu7du8PLywuvuv46tm3bhoCAGJT87C2t2XE2+UVan2RTIF0rI
KKtX21r10jPrqAPTlyq8V7WBrijGvd4SDjWmlzvvY999/j+DgYMybNw87duxAUFFAQfvnlF3T3t
2hUAcO7cOXTq1Am7d++GXC7H9evX8eGH2L8+PGYN28eOe+jYp58iO+//rVHH+//v4fd7A6W89nOAl2ZjNIKsOO7PTkXtKsOjVr09NFTgrP4
zIjR45EVFQUoqKiANe+bmhpMnTwIiArl+/jqCgaGe4NaBBUy2FMnrYJcXnE+HdfJZ+1sY+d+N+ySPtkNSqAcx6qc53AI8
bB09MT3bp1Q2BgIG7fvl3mc/o2zzz/o2LEjZsyYgbmgNcZ+h/SghCOnjVrZB3NkHt09udOUd98
tNYJx6NAhDB8+HJ6enujWxTsn4yNSmXDdm7hsSHIIwcOR1eH7w9fXfF1q1bkZyc

jKCgIHh6eqJXr16lrjKnp6fjyy+/RM+ePdGpUye88847iIyMLPffcfr06XBzcyvzz+N/16oyN
a1aJ0PUELDfqd/9zrNatmyJjh07Yv/+/aUe27dvHwYOHFgrz0OkKeyDGk4f9PDhQ7X/bu3atc
OMGTPQsWPHGj+HLlqz8xLOXk9Gclpepf5UNll+LC2roMJznr2ejDU7L1U59jfffBMjRoxA69a
tERwcDA8PD5X3OQBMnjwZLVu2hLOzM8LDw9GjRw8EBgbC2dkZXl5eWLx4MS5cuICoqKgqP39l
cIS5kZk9ezY+++wz5barqyuSk5OVo5kLFiyAgYEBjh49ig0bNqBZs2YYP3487t27hwkTJuDNN
9/EJ598goyMDCxZsgRjx47FwYMHIRarXns5deoUgoKCMHDgQMybN6/cqz0HDhyAi4sLPv/8cw
iCgO+++w6TJ0/GkSNHIJFIcPToUUycOBEffPABJk2ahPz8fGzatAnffPMN2rdvr7wCBQBTp07
F2LFjMWHCBKxZswazZ8+Go6Mj/Pz8MGzYMPz666+YP38+unbtCg8PDxQUFGDUqFF4+PAhPvnk
E9jY2GD79u0ICAjAunXr0KNHD7UxBwYG4t133y3zNbm6upb7e1AoFJDJZMqfCwsLER0djW+++
QampqaVut+HqKFgv1M/+h1BEFT6ncf09Ep/FfDz88Py5cuRn58PqVQKALh9+zauX7+OsLAwLF
q0qNznIqpP2AfVjz6oMnx9fbF3714UFBRgwIAB6NatG2xtbQEAH374YY3PT/WPt7e3ys+dO3f
Gf//9p/zZyspKZUDp6tWruHPnDjw9PUudKyYmBj4+PrUeIxPmRsbV1VU5v//xVJjz58/D3d0d
S5cuVT7Ws2dPREZG4tSpUxg/fjwuXryI/Px8jBs3Ttlx2dvb4/Dhw8jNzVW5Z+DixYsYN24c+
vXrh/nz55f6QHmWTCZDeHi48hw5OTkICQnBtWvX0LFjR0RHR+ONN97ArFmzlMd4enrCx8cHp0
6dUvnQePPNN+Hv7w8AMDY2xrBhw+Dh4YHJkycDADp27IjDhw/j7Nmz8PDwwO3H9+nX8/vv
v6Ny5MwDgpZdewsiRI7Fo0SJs375dbcyOjo41und4xYoVWLFihco+AwMDeHl54eeff4adnV21
z01U37DfqR/9zq5du7Br165S+3/55Rd4eXmp7BswYAAWLlyIY8eOoX///3sGR5c9PT3RokWLa
sdApA3sg+pHH1QZ33zzDRQKBQ4ePIhDhw4pn7dPnz7w9/fn96MyjB3SCet2X8a9Sk7Jzs4rQm
5+6QuoZTGW6qGJkX65bVramiLg9arPAHj2oq1CoVD5//P4ou3Tjw8ePBjjx48vda6n63HUJib
MhBdeeAEvvPACioqKEBsbi7i4ONy4cpQOpqanKwgqdO3eGoaEh3nrrLfj5+aFXr17w8vKCh4eH
yrnu37+PMWPGQBAEzJ49u8IPDED1gwyA8kMpLy8PAJT3quTm5uLu3buIjY3FpUvFUz6KilSLF
Tx9telxJcXHHwYAYGFhAQDIyiruUCIjI2FtbY0OHTqojLz07t0b33//PTIyMmBubl4qZoVCAY
VCUeZrkkgk5V5Zfuedd/DOO+9AEARcvXoYpFixbVWcECovqE/Y7m+53evXtj4sS
Jpfa3bt261L7mzZujS5cu2L9/v0rCPGLEiDLPT9SQsA/SfB9UGaampli2bBni4+Nx7NgxnDx
EidPnkRERAR+//13hIeHh4eHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR
EidPnkRERAR+//13hIeHq1wsoGIONqaYM0b9zAB1rt9JxWfL/ql0+6/H9oCbU90ko5cuXUKfP
n2UP58/fx4dOnQs32bNm1w69YtODk5Kffdvn0b33//PTIyMmBubl4qZoVCAYVCUeZrkkgk5V5Zfuedd/DOO+9AEARcvXoYpFixbVWcECovqE/Y7m+53evXtj4sS
Jpfa3bt261L7mzZujS5cu2L9/v0rCPGLEiDLPT9SQsA/SfB9UGaampli2bBni4+Nx7NgxnDx
EidPnkRERAR+//13hIeHh4eHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR0dHR
ERAjRinZBMRERERERGpwYSZiIiIiIiISA0mzERERERERERqMGEmIiIiIiIiUoMJMxERERERZ

EaTJiJiIiIiIiI1OA6zERERERERI2QIAgouH8LuTejIM/LgcTIBMZtvWHYvA1EIpG2w6sXuA4
zALlcgdTUHG2HQUQVsLY21XYItY79D1HDoGv9D/seooahLvuewpS7SPlzOQoSY0o9ZmjvAuvB
QTCwdqyz59eEM2fOQBAEeHl5VfscnJJNRERERETUiBSm3MX9jZ+rTZYBoCAxBvc3fo7ClLsaj
qx2DR8+HHfv1uw1MGEmIiIiIiJqJARBQMqfy6HIL3+WiSI/Byl/hqGxT0jmlGxwWhJRQ6FrUy
IB9j9EDYWu9T/se4gahsr0PYWPEvDo4HoUPYqv1DkFWRHkOemVjkFi0hQiPf1y2+hbOcCqnz8
MrFpU+rxubm6YM2cO/vjjD1y5cgVOTk6YMmUK+vbtq2xz9OhRrFixArdu3YKJiQkGDRqETz75
BIaGhgCAY8eOYenSpYiJiYGxsTF69eqFGTNmwNzcHG5ubsrzDBkyBAsWLKh0bE9jwgx+aBA1F
Lr2hRVg/0PUUOha/8O+h6hhqEzfk/jr18i7fUED0ZTPqHVn2L/3ZaXbu7m5wdjYGMHBwejRow
d27NiB8PBw/PLLL+jatSsOHTqESZMmISgoCAMGDMCdO3cwZ84ceHh4IDQ0FKmpqejVqxemT58
OX19fPHjwANOmTUP37t0xb948pKSk4IUXXsDMmTMxdOhQmJpWrx9nlWwiIiIiIiLSuDfffBBMj
RowAAAQHB+PUqVPYtGkTunbtitWrV+OVV17BxIkTAQCtW7eGIAiYMGECYmJiUFhYiMLCQjRv3
hwtWrRAixYtsGrVKsjlcgCAtbU1AMDU1LTayTLAhJmIiIiIiKjBsuo3Go/+bz2KHlZuSrY8Px
tCQV6lzy8yNIJE2qTcNvrNHGD1in+lz/mYt7e3e3ys+dO3fGf//ef//9BwC4efMMmBg4cqPJ4t27dAAA
3btyAn58fBg0ahPHjx8Pe3h4e9e/aEr68v+vTpU+U4ysOEmYiIiIiIqIEysGoB+3c/r3T7I7/ISb
uB8xo9Lt7d/7EtIWbasTWoX09FTTUYVCAbG4uC61IAil1oJ+PHr8+LjFixdj4sSJ+Pvvv/Hff
/9h6tSp6Nq1KzZu3FhrhUPRO6+RcjdtwiK9ERth0NEjQj7HyLSFvY/NWfYvA0M7V
0q19beFYbN29RZLJcuXVL5+fz58+jQoQMAoG3btjjhz5ozK46dPnwYAuLi44Pz58/+/j222/RunV
rfPjhh1izZg2+/fZbnDx5Eo8ePaq1GJkwEZVQ+ZGbIY+/DHn8ZeRH/qrtcIIoEWH/U3O7du2C
n58fOnXxqhIEDB2L//v1Lv/v1lti0qKsLixYvokokuXLnj//fdx7do1dUH/w/6k5kUgE68FBE
EtNym0nlprAevDEUqO8tWnDhg3848S88/ERsbi+w+w7Xr1/HqFGjAGjYJ4/GwYWHMHERYWhtjYWP
z111/45ptv0L0Lt3b7i4uKBKByvbVykzhYbGvCt37uDGjRvNvzvDGYZJ2RFy8YvowsvokuXLnj//fdx7do1DUZLVLH
Sqh0jE2aiBkqRdv+p7Qn5+7p7e3s6ciWHDhmHPnj3w8w8/PD1KlTce7cObOb0Xt58yZ
g23btuGbb77B9u3b0bRpU4wZxMwZZWVkajpxI+9j/1A4Da0c0/2BumSPNhvauaP7BXBhYO9ZpH
MOGDcP69evx2muv4fTp0wgPD0e7du0AAAAMGDMCiRYvwv//9D4MHD4bs/2bMxcOBA/PjjjjjwAAV1
dXhIaG4sSJE3jjjjTcwfPhw6OnpYe3atcpp3R999BE2bdqEmEmTNVjtG3sNMRERERepCGCIGDp0qU
YNWqUchRl4sSJOHv2LKKiouVHAdp6DL/b8g6anS/t69e9e9i2bBMC33X19N954A5mA5cvX0aPj00
/RKISEcYWDuiuf93KLh27+bAv93KLh/C7k3oyDPy4HEyATGbb1h2LxNnY4sP9amTRuEuNGyUMn1XX9g4eHwatAQr3bsd4fy
4tPdZdv8fLygkwmw6VL2VI6iQRSOn+PrlCgOMVMwaNCJCb7B/p06fDxaVyRXvKoqfHO/Oo4RGJRBCe2ub7
AKGhoVi0aBGCgoKQn5+PrJXnVfxZxXPo7uuYPGjbC0tNTyKyEi4S4S3EbSzuzgRZBgxJiZqRD8xpZEcK3vKIHo3nfzl3Pnn
nmzJmDOXPmaCg6IiJ6I2cSl/ExpCGrOepBRqVAXBdMWLLyJbZv3465dN26dN26NcaMGYOkpCKpaXB398fzs7
O2L59OyZNmoSlSSdi+/bt2g6biIiIWwJoEoR5iJiEocPnwYgwYmwYGNwsvvAAAmD59OrZu3Yz588
Li4OBgYGmDNnjrJi7Z07d7B27Vq8eabWo6ciIIiIiIIoOeCR5iJiEo0bdoUf/31F+Lj4yGxy7Fly
xYYGBjA3d0dp0fdmdcc9tCYfRtau3HhaCn9+Q6Y/fu3REbG4tHjxx5pMWoiIiIqscYSYiKjFr1x
u3LyQSCcRiMZYuXQpHR0c8ePCg1LqpNjY2NjY2AID79+5+/Dysqq2s+rp8drl9SwiEQiCE9t8z1MRES
6igkzEVGJmJGJgYmZKZ1Kz72bp1K0JCQrBp0bybk5+fDwFhUKRaLYGFhUqO4
iTQtSyyComSb72EiIrJ1TJiJiaAkcEQVDZ5nuYiIh0FW86IiICYG9vDwC
ONE2djYuNrPq1AIyMzMrX7gRFqgUAgq22lpOVqMRjN4UYCIqHFiwkxEBODixYsoKipCp06dVP
Z37twZf//9N5o3b47k5GSVxx7/bGtri86dO+PKtV9oxLz3xYn8rTysoypl6arQRbk15WQOc1pNMBkInCHLxYbb
l1j+9efMmnJyc0K1bN5w5cwZyuVz5WGRkJe1sJYb+5hJFfiImnaV2pkE2sWkbFwUQVWdEEiiIqpS5nSwkxEBwMDDwwNeXl4ICQnB
iRMnEBcXhx9//BGRkZEYO3Ys3nzzTRnzTWRnZ2Ijo7Gjhj07sGHDBowbN07Rd7boRMRERERFRHeGUb
CIiAGKxGCtWrMCPP/6IGTNmICMjA23btsVKEi3bh3mzZuHUIUOCwNraGaGtOmTcOQIU
O0GzgRERER1RkmzEREJczNzTF79mmzMnj1b7eMeHh7m7YsmWLhqMiIiIiIiIm3hlGwiIiIiIiINzg
wExEREREREEanBhJmIiIiIiIhIDSbMRERERERERERERGowYSYiIiIiIiJSgwkzERERERERERERkRpMmImI

iIiIiIjUYMJMRERERERERepIaetgN4VkJCAvr06VNq/9y5c/H2229jxowZ2LFjh8pjtra2+Pvvv
zUVIhERERERETUC9S5hvnHjBgwNDXHo0CGIRCLlflNTU+Xj48ePx/vvv698TCKRaDxOIiIiIi
Ii0m31LmG+efMmWrVqBRsbm1KPyeVyREdHIzAwENbW1lqIjoiIiIiIiBqLencP840bN+Dq6qr
2sbi4OBQUFMDFxUXDURERERERERERVFjUy9HmK2trTF8+HDExcXByckJgYGBePHFF3Hz5k2IRCJs
2LABf//9N8RiMXr16oUpU6Yop2xXl55evbt2QFQukUgE4altvoeJiIiIiIiGpXvUqYCwsLERcXB
yMjI0ybNg3Gxsb4448/MGbMGKxfvx63btsGrVKty5cwffffcdbt68iQ0bNkAsrl
7CIBaLYGFhUsuvhqhuZYlFUJRs8z1MRERERFT76lXCbGBggFOnTkFPTw8GBgYAgI4dOyImJgb
h4eFYu3YtPvzwQ5iZmQEA2rZtC2trawwbNgyXLl1C586dq/W8CoWAzMzsWnsdRJqgUAgq22lp
OVqMRjN4UYCIiIiINKleJcwAYGxsXGpf27Zt8e+e+//0IkEimT5acfA4AHDx5UO2EGAJlMUXjo
npEEASVbb6HiYiIiIhqV7266fH69evw9PTE6dOnVfZfvnwZzrq6u+PTTTzF69GiVx0JkSt
YKIyIiIiIiIqqOepUwt23bFm3atF06dPIyYmBvPnz8f58+cxfvx4DBo0CMePH8fKlSt
x9+5dHDt2DDNnzsSgQYNYOZuIiIiIiIhqVb2aki0Wi7Fq1SosWrQIU6dORWrQWZmJtq3b4/169fD
zc0Nbm5uWLp0KVatWoVVq1bB1NQUgwcPxfr3h
78/+vfvr8GIiIiIiIiIqDGqV1OyiYiIiIiOoLJsxERERERERERERaajBhJiIiokoTBAGQy578XJ
ALeVK0ylJ3REREuqLe3cNMRERE9ZM8NQH5x9sMt4srMoD7m750Js3QrSXgGQWLbQXoBERES
1jCPMREREVCF5agJy/5gHRUqs2scVKbhI/WMe5KkJGo6MiIio7jBhJiIionIJgoD8Y+uAwzy
GxbmFo9Ac3o2ERHpCE7JbiQU6YnI/+8XAIC05wiIm9prOSIIoFMkxZY4sl2qbEgtFym1Ib
FzqOCoiIqK6xxHmRiI/cjPk8Zchj7+M/MhftR0OERHVc4JCAXnqRReO4r8fzZU6VhZ3Nk6ik
p37Nq1C35+fujUqRMGDhyI/fv3V+q4P//8E25uboiPj6/jCInqHxYdJG3gCHMjoUi7/7/
jIiJVirxMKJJjIE+KgTw5BvKKArnEgpyajk63bJ7927MnDkTISEh8PX1xZ49ezB1ezB16lT
2dnB09OzzOMSEhEhLw1VdfaTBSovqDRQdJW5gwExERERNTKCXAbFo7vFiXFJgixkpdTa+UWGJrV2L
l0jCAKWLl2KUaNGYdSYzs4uzZs4iKiozYVYoFPjss3HnhmE2nd46KDZd
VReFx00Pi1WUyaqdYxYSYiItJhgiBAyEl9MnKcCHAPFwziVaY21Tc+5a52du6G7ffs2EhISMHj
wYJX94eHhZR4jlUoRFBRU84SZiIiIiKiIiKiIiqHBIUcitT4J/cdJ8P/cdJ8VAkfGg6icyMCp
OjpV/WkMkbVKtmCSWLWD8xpfI2TTlSaVafSMYDwyG2Lo1R5Yrwc7ODgDQtm1blf2urq6IjI
qfbbt2+HgYGBshiYXC4HAwaNAivvvYavv7EZFN4+Ddn9a
4BCXu1zFcEAaWm6U6mfyb/2MWEmIiKqBxS56ZAnI1A3G0hRt+I
hUe1NvRSIRIHny9UFkaMz7Baugffv2MDExwxYULI1y4gM8
++wxr1qyBi0vN/t1lsipO3SeqRYMcEAamctZbGTJ9/bVKuYMBMREWmYICuE
4uEdyJNvK0eQhexVTVT6PyMgMEltXiG1aFyfJ1q0g0pfWQcRUW6RSKQICAhAWFgZbW1t7
969OH78OCIiIiCXy5GamVamgpTU1NIpVI4OTI4OTmpHP/gUDbWNCQ6dEtUzgiB
DFnYUi9V7FBxmaQOLYBYqkWxAykytszqKDVBeYMBM1QIIgqCwJxNnOCC6dEEtUzgiB
AyEpRLcz16G7VpxyK9SBu5qRy77GoiRX/zzdAgYGBMDIywbpH4x6mFnnR
6Nu3L+bPn4+hQ4dqO1SiGhEJMSiKOwNZ7JlKrfuGWnb3NXOz6Dk/B4m9G0RiSYRMANg0
UGqMyJBEGpn/kMDJpcrkJqqO/c6qJO9+VPl6IWoiRWaDF+s5YiouuSpCY2226I61tam2Q6h1ut
7/KNITkf/fLwAAac8RED113JEdU8ozCte1ikpGvLk21Akx0DIz6ryeUSm8lk8lx60htnKESKJ
fBxFXXWP8TNG1/kfX+x7SHkEhg/z+dchiz0B251yllrUTmdtC3/k56LXygtjaWe1tJI31+4+u
9T0NEUeYiRqQiq6wKlJikfvHXZiD3lpHxHLdvvjJkaIJo67MZDhFGhOd/OkS7V9hPGCqliOqXYKggCItEfLk6
OJ7j5NuQ5GWgCrfi6dnCMnjadU2LhDbtIbY2i4xOyiYiqm2CraCye5chizsD2Z3l3zlasHX9vaZ0i7f5T2wl
6r56Dn/BzEFs0rHBlm0UHSFibMjQCn7+oGQRCQf2xdR9ChbnIP7Ymi
ajKR2KPIyoXjqvmN5cixQlFfl84gtmpckxiUJskULiMRcE5eIGg6hIAeyO+chizsL2b1LgLyi
IoUiSOzalCTJXSE2ta7yc7Loi5ydnr5SHwiCAChkgFwwGQV5Uf
L+ivAiCXKa6v+RnQS4D5EVPbcsARZFyW5GVonYakjqKlFgoUm7zQ4SoBgS5DIrUe0+teRxTq
IzpRiaqNx3LLFuBZEhlwkhooZHkZtenCDHnoH8/nVAqKAWg1gCSYsOxfckO3ly5gw1SEyYdVh
jmb6rkpiW/F2TxLTsNk/tf/a5lNuP9xfVaA3B2iCLO8uEmaiSBEGAkJOqWpjr4Z3i/8tVA
bNVStTCXmQ1nexBRg1Xl5Z/0DKHXshP0WnlBz9EDIgNjCWsDS
EzrK6GAxVyIyiLICiBPiSu577hk9LgShWmeJTKxVN57LLZ1gaSZE0R6hrUfMBGRhiiXfyqpbF

3Z5Z/0nDyh7/wcJA4dINIzqPtAiTSECbOOUiTHVGn6bu72L4vvCSkzMS1JiqnB4JRPomKCIED
ISHpy33FSTPEXQEFRtRNJ9CGxbvVkzWMbF4ibWNZN0EREGlRbyz8R6SImzDpKdudcldpX6uoh
PSESAWJ9QCIpXuZFrAdI9CGSSEr260Ek0SvZr1fcpuRnkaS47ZNtPUCsr9wWlRwDif5T23pQZ
CSh4J+ISoeo59y17l4/UT0mFORAnny75E9xkoxqzLgQmduWJMatIbFxhdjKofj/JBGRDqir5Z
+IdA0/+XWU7kzHFSkTxieJ6eNktIzE9Ok2En1ALHkqYS1Jap9JRkUl53qSsOo/k7w+c24tXEU
V7Nuh6PqxSs0cEFu3gti6tQaiItIuQSGHIi3hyX3HyTFQpCdW/UT6RsWJse1TyzpJufYlEekW
TSz/RKRrmDDrqGpNx1VJEitIRiuTVKpJWEUlo7JPEtaSUdoyE1ZO73lMJBJB2iug3EJuAAADY
0h7BfADrYpOnjyJDz74QO1jDg4OOHz4MGbMmIEdO3aoPGZra4u///5bEyESiiu0FifGt4uT5J
RYQFZQtZOIRBBbOChHj8W2LhA3tedICRHpJKEgB7K7F4pHkquy/JPzc9BrVb3ln4h0CRNmHaX
n5InC83sr3d7o9c+hZ+tahxFRbZBYtoDxa7OQf2yd2pFmLhVWfZ6envj3339V9t28eRNjx47F
+PHjAQA3btzA+PHj8f777yvbSCS8qFOWmq4BL8iLoHh4R6VytZD9qMpxiIzMnqx5bOsCSTNni
AyMqnweIqKGonrLP7UvTpK5/BORCibMOkps4wKxdatKT9/l8kMNh8SyBYzf+BI5m6Y8WV9b3w
jGA4Mhtm7NkeVqMjAwgLX1k6voRUVFmD9/Pvr164e3334bcrkc0dHRCAwMVGlH6lV1DXhBECB
kPXxSmCs5BoqHd6tebFAsgbiZk7Iol8TGBSLTZvx/QUQ6j8s/EdUNJsw6itN3dZvo/9u797io
63x/4K8ZcLhfhnG4CILcxjuG4a12V9PSzcvTPVDLpLbkroa2Za0mmvlz8xMvJCxmW2dz
dSyPe3J3fJUZt5STJDbgIKiXB1gYICZz+8PZGICZYCBufB6Ph4+Gr7f73h27Rreecd AEB+fj50Oh0iI/l7bo95NeDXw2XcPAitpqm0041ciFp
Nh68l8VSY1DyWKkJZzoSIegWWfyLqGUyYHRiH7xJ1jk6nQ0pKChYtWgR/f38ATcOzJRIJ0tLS
cOzYMUilUowfPx6JiYnw8ura4lDOzo4zd1YIAa1ZNeBrofv6nY6d3NkFzv4RcA6IhFNgJJwDI
iF19+10rNR5EonE2HclkUgc6j1MZMuEMMBwIw8N6lNozD8DobnR7nMkHnI4h42EczjLPxF1Bh
NmB8fhu0Qdd+jQIeh0OixcuNC4LTs7G1KpFMHBwUhJSUFBQQE2bdqErKwspKWlQSrtXMIglUo
glztOzey6oixUmlkDvj19FMFwCVbBNVgFl2AVZMr+/KJnI6qkEjRXXsXa09zCRrela+ae7IVWG
c1FDoi5gwtwLcPguUcccPHgQkydPPhlwuN277/e9/j8WLF8Pb2xsAoFKpoFQMXfuXKSnp2PEi
BGdupbBIKDRgAWgDam3WWC5S6xGAQJo
8rKhyllOHt8aaA/TFUXkPdd+8DAFzveQRS3yArR2Q+ln8ish1MmImIWigvL8fZs2fx5JNPmmy
XSCTGZLmZsIU3VIpFOPw7vUNn/eFnLFvEuBIe6wNBiYKeuVZXfneopVn9BgmGBfi+ORghh
8tiR3sPkOOqO74e+8MKtxx/A/cEVVo7ozlj+icg2MWEmImrhzh2MwfNJvDFuZeKzKM9EUZLP9EZI
OYMBMRtZCZmYn+/fvDzc20Tu/06dOxbNkyy7eNixA9OmTYNarcbLLL7+M6dOnMonc+XsFjpaA955wMh
ujIaIyPZ0vPyTDM79Y5qGW4eOYPknoh7GhJmqMIqikn22qi5+x2fFfdh69atSEL4QUpKCry8
vDBjxjPyeIyyr2KM14KXKiB6IiojIelj+ici+MWEmImph3bp1t903COamQmOsA
U9ExPJPRI6ECTMREVkUa8ATUW9kLP90a04yyz8ROQHyNhPOAOJZ/Ir
IDTJiJiKhbsAY8ETmiLpV/GjASUm+WfyKyJ0yYiYiIiIjugOWfiHovJsxEdkpEjpvEJC2eExERES
WY9DcQGP+aTSoWf6JqDdjwtxLSOX9ozK8uu/WYC+04gpraBngZHzeC966JiIiaCCEAfePPPPPP+u0
0F/PgdQ/8rZzhoUMFQUNg21zj8NQ5m55Z/uQp8BcSz/ROSgLJYwl5eXExXIzU1Fd7wKkqqwZ
88efPnllxg0aBDuv/9+S12IlN7pKoZGu6GYMyqi3brYwEZXLPhNw6GWHmMcVpLfuBDoREvJPjXZrdAo4s2+Hao6R9zKK2PZRS4AcL5VbLN7Vo5LINpTsvlnw3LWmjNm84M4WhwVVL7OCBQG2YYPz4aAH8kZ7HcR/BcxEfdZyhYfKukBdAX
8vOAgADbXQHlrfapX+n8s/NSXJHS//pIJEyv4nIkdmkf/Dr1y5gnnz5kGn0+Huu+9Ez+Ho8b1uL+5YB95Qoob28Aa4jHkYhrLL
91Go13n77bbz99tuYMGCJS5FnST1DYL7l7FXWDoOIIio2+jLi+5YB95Qoob28Aa4jHkYhrLL
LP9ERO2ySMK8adMmKBQKvPfee3ee3B3d3eYcMAJs3b4ZOp0NKSorXV69esxZ
84cZGRkYMOGDbhw4QJ8fX2xcOFCxMfHW+xCAEXE6CB1fFNHE6r7e035pp3otdP/3brvnY/
knIgIslDAfP34cr7zyCry8vGAwGEDj6eanCPk5eWFiooKLFm
yBPfffz+Sk+vpg9e7YlXgoREVkp0p4MQUU8xN3MiFoUTdhTOw/BMRtWaxSRdO
Tk5tbq+vr+/QHbEDtTIZ4OWsdOP39+/1b60tDTIzKsW7cOzs7IyBnE6C
BH1lMaCsx1/Ess/EVE7LDIBy4uDrt27+/MQGIlEArPMAjPBgA8++AjAR440+1yXLL1CVFRUm/
tOnTqFUaNnXGdwn55zx/7NixUCr6s8AQmy0T
UJov0MK9cuRLz5s3D3ZmTMWbMGEgHEgkqEqSmpiI3NxcFBQXYv3+2efKysqCUqnE/PnzkZ+fj7Cw
MCQkJODODv/41iouLvVxvD/41iouLvVxvD/
NQuiyTMKpUKH3/8MbZu3YrExEQYjEZs2bIFKBZQ/V45pDQu2+ZPCa3IaWdFXkYAHqg6M0C+Xbd9yjR1GrO5+vOR/s66E
Hu7o7Dhw/j8ccfx969e3e1FXVweZLS+nYuLCwBcJF2QkeOkIRrIDKvNwfgN9FsLKsrR+6+rFOi6cwWUWR/EzmjYnQLkiU8
HxaL+xyPmHz/A/BGQRNR7WShPnjwHNhzMtMzmGCWg==

x1o4+bEuPBGZxyI9zA8//DBefvllnDhxAtHR0ejbt2+rYx566KF2z6NSqRAdHY3k5GS89NJLk
Mvl+Oijj/Djjz/i73//O/r27Ys9e/Zg7dq1WLp0Kc6fP4+0tDQkJydb4mUQERERkZ1z8guG+0
MvomZfIkTtzaaNfdzgPm0VpMoI9iwTUYdYJGH+y1/+AgA4dOhQm/slEolZCbNUKVKSgpef/1
1JCYmQqPRYMiQIdi7d69x4bA9e/Zgw4YNmDlzJpRKJVavXo2ZM2da4mUQERERkQOQSCSA089f
cyUu7nDyj7RiRERkryySMB89etQSpwEA+Pn54ZVXXrnt/piHHz44YcWux4YcWux4RERF1jFTeD/rqs
luPObS1Mw4ePIhdu3bhypUrCA0NxfLLy/Hggw+2eWx2djZee+01nDt3DlKpFKNGjcZzzz2Hfv
369XDURES9j0XMmAcHBxv/+fn5QSaTNf4dTwDpunrXDDsHgADT1Bj/77LMYO3aspS5DREREVib1DYL71FXWDsMuCSGwd
etWLFq0CIsWLQIAPP300zzz8hxgx5gx++OEHxMbGmhz/4nDDWIDEhMHzP7DTo/tTmWYxDYc
lzBuPGjevx10BkTXqDMPb46Q3CqrFQ72CRhPnMmTNYvHgx+vfvfvj4SEBPTt2xc3btzAkSNHsHT
pUrz33nutPgCIIiIepu8vDwUFRVhxowZwZJttTU1PbPH7cuHHYvn27MVlu6ebNm6tl5
gJfxcSN8rBoN9QYWSZjfeOMNxMXFITUSU1FU5OTsbty5cvR3x8PLZt24Z33nnHEpciIisk5+
fkAAK1Wi/j4eFy8eISFYtmwZZk6c2vR4kJAQhISEmGzuXMn6bxDKI5oKwZ42SKKOgRPxl5Nk
sikUC0eOzor7e30BsEIGl+bODflbqdRRLm9PR0bN682SRZvmvGpcXIBWfpWyPIYXd4Hj/9GUrkmuiflHbihyzCi1MfH2SwLyXNC5
q6moAwJo1a7B8+XKsWrUKX3zBRISErB37952h1j/9a9/xf79+5GUlASFQtHpOKRSCeRyj04
3x5USSUw3HrcG15vb6H+xc/8u1J3s0jC7OHhgcbGxjb3NTQ0QM0GOc0xEREfVmffr0AQDEx8cby
2IOHjwYFy9evGPC3DezeceOHiLsVhMAhoNNouncPWGVrMbzUYBCoqaqwYDXUXR/
+78oaA9VkkYR45ciRSUlJw7733wsPj5z9qgqoGa5Dt+zaP5z9qdXU1du3ahbi4H0/PBp6FtGYwaw//5
oBCn4hc5cgYGBAACVSmWyPSoql999VWbz2loPoaEBJpbHR0P5Bdqxl
h40QwuFfb2/Fvyt1N4skzCtXrsSsWbNw//Y33//33b8Y8KECVAqlgpKcFXX30FnU53x7rKRERERZF8+O
JqNn9TlAIC/Hc3Bsw+PsHJE9mPIkCHw9mPDAuXPnTDoUsrKZyEBoa2Ua2uZVq9/93em9ejf/93//F5s2bMW
3atJ4K1e6xkXgr6XRGWYJFEuawsBDs9DHGHcvTkOqSIIaIaihrhjzvLR6PcrLr6Ly+ebEr...

F+2mAEKz2tGCERERERkWNgDzORHSkqqcbGfWeQX1zV5v784ips3HcGRSXVPRwZERERERZHjYcJ
MZCeEEEg9kgGtrvGOx2l1jXjn8wwOzyYiIiIi6iImzER2Iu+q5rY9y7+kvlaFvGuabo6IiIiI
iMix2XTCrFarERsbiwMHDhi3JSUlYeDAgSb/fvOb31gxSqLup6mpx6ff5XfoOWezSrsnGCIiI
iKiXsJmF/1qaGjAqlWroNVqTbZfunQJTz31FBYsWGDc5uTk1NPhEXUrg0FAXaxBem4Z0vPKkH
+tCh0dYK2ta+iW2IiIiIiIegubTZi3bdsGDw8Pk216vR45OTlISEiAUqm0UmRE3aNKW48L6nK
k55XhQl45qmu7lvC6u/axUGRERERERL2TTSbMJ0+exIcffoiDBw9iwoQJxu35+fnQ6XSIjIy0
XnBEFmIQAgXFVUjPLcP5vDKor2o63It8J7GqvhY8GxERWdLBgwexa9cuXLlyBaGhoVi+fDkef
PDBNo+tqKjA+vXrcezYMQDAb3/7WyQlJcHd3b0nQyayOiEEGvUG48/augbkFt1ERD9vSCQSK0
ZGjszmEmaNRoPVq1fjhRdeQFBQkMm+rKwsSCQSpKWl4dixY5BKpRg/fjwSExPh5eXVpes6O9v
0dG5yENW1DbiQV4ZzOU1DrTUtainfSYjSA8MjFTibVYricm27x0f084aqvy8/PIiIbNChQ4fw
/PPPY82aNZgwYQI+++wzrFixAoGBgYiNjW11/DPPPAOdTod3330XGo0Ga9euRXJyMjZt2mSF6
Imso6ikGqlHMnCzxXen2no9Nrx3GgMCvRA/bTCClZ5WjJAclc0lzOvWrcNdd92FGTNmtNqXnZ
0NqVSK4OBgpKSkoKCgAJs2bUJWVhbS0tIglXYu6ZVKJZDLPdo/kKiDDAaBvKs3cTrzOk5n3MC
lgnIYzOhGdpU5YUS0EncPDsDdg/zhL2/qRSgo1mDNW9+g5g7DtT3c+mDFI3fDz48fGkREtkYI
ga1bt2LRokVYtGgRAODpp5/GmTNn8MMPP8MMP7RKmM+ePYsffvgBMP7RKmM+ePYsffvgBn3/+uXGE3csvv4ylS5dixYoVC
AgI6PHXQNTTikqqsXHfmduW1swvrsLGfWeQtGAkk2ayOJtKmA8ePIhTp07h008/bXP/73//ey
xevBje3t4AAJVKBaVSiblz5yI9PR0jRozo1HUNBgGNpv1eOyJz1NQ14Ke8cpzLLcX53DLcrDa
vF7lfXw/ERCowIqovVP190cc46kGgoqIGAODt4oS1j96N3YcvQt1G2aiIft5YOmMIvF2cjM9x
JLyxRWR9HBLZNXl5eSgqKmrVMZCamtrm8adOnYJSqTSZjjZ69GhIJBKcPn0aU6dO7dZ4iaxNC
IHUIxm3TZabaXWNeOfzDLzwaBzbIrIom0qYP/74Y5SVlZnMWwaAl156CampqThy5IgxWW6mUq
kAAMXFxZ1OmAGgsdHQ/kFEbRBC4MqNaqTnlSE9tww5RRoYRPvdyDJnKQaHyRETqcDwCAX6+rq
Z7L/dezJQ7o4XHr0bK9761jgsyU3mhBX/dRcigpq+sPL9TETdgUMiuy4/Px8AoNVqER8fj4sX
LyIkJATLli3DxIkTJATLli1//fr1Imciqcotuir+4yqxj1deqkHdNg8h+Pt0cFfUmNpUwv/7666irqzPZNnnyZDzzzDOYOnUqVq5cicrKSpO7sOnp6QCAqKioWVerdaXS
Mu5pfjfG4ZLqjLUVGlM+t5AX7uGB7hh5hIBQb290Uf586VRJNIJHB2+nkKgrtrH344EFG34pB
Iy6iurgYArFmzBsuXL8eqVavwxRdfCHAXv37sW4ceNMjq+trYVMJmt1HhcXF+h05n323A7X
byFbU1ffiMIbNbh8vQpXblTj8vUq5F1tPaLuTs7llGFgqLybIqTeyKYS5tvNw1EoFAgODsb06
dOxbNky7NixA9OmTYNarcbLL7+M6dOnM/M6dOnc+Vs6lZCCBSV1hh7kbMLb0JvxmTkPs5SDApt7kX2M8
5FJiKyJxwSaTl9+jSV/IuPj8fMmTMBAIMHD8bFixbXV1RX996ao9Op+vSKtlcv4WsSQi
B6+Va5F/TQH1Vg/xrN6G+qkFxWQ3MGKR3R42CU7jIsmwqYW7Pfffdh61btyIlJQUpKSnYsWMH
jBkzxkZiYaO3QyAHV1TciI78C5/OaVrQu15h3J1/p64qYiL4YHqnAoFBfyPp0rheZiMhWcEik5
QQGBgL4eUpZs6ioKHz11VdtHv/ll1+abKuvr0dlZWWXFvzi+i3UU3T1ely5UY0rN6pw+Xq18X
GtTt8t13OWwKHWcWHyb302nzBfunTJ5OcpU6ZgypQpVoqGHJkQAtfKtDif25QgZ12pNKsX2dl
JioGhvoiJUGB4pAIBcjf2rBCR3SutrEXG5QpkFlTgbHZph557NquUCfNtDBkyBB4eHjh37hzi
4uKM27OyshAaGtrq+FGjRuH1119HqUEBwsLCAAAnTpwAIwcObJEsXC9C7IkIQTKbtY1JcQlz
YlxNUoqatGVTmMfD5nJugntGRGl4HubLMrmE2ai7qSr1yPjcgXSbyXJpTfr2n8SgL4+rhh+a7
GuwaFyuMjYi0xE9q2iSofMyxXIKGhKks1tD9uirbt96bveztXVFUuXLsX27dsdREBCAmJgYHDl
yBN9++y3e2effdd6PV6lJeXw8vLC4uLi0xE9q2iSofMyxXIKGhKks1tD9uirbt96bveztXVFUuXLsX27dsREBCAmJgYHDl
yBN9++y3effdd6PV6lJeXw8vLC66urhgxYgRyBgZ599GwHvbLMrmE2ai7qSr1yPjcgXSbyXJpTfr2n8SgL4+rhh+a7
GuwaFyuMjYi0xE9q2iSofMyxXIKGhKks1tD9uirbt96bveztXVFUuXLsX27dsREBCAmJgYHDl
9XoUljYlxIW3EuPCkuou9Rq79HFCiNIDIf6e6O/viRBl0z83Fyf8Oe2UWaNcwoO8EBHk3e5xR
B3BhJl6FSEErlfUGnuRL12uNCmPcjtOUglU/X2NK1oHKdzZi0xEdk2jrcely5XILGhKkovLLT
c81921j8XO5YgSEhLg5uaGLVu24Pr164iMjMS2bdswZswZoFiUSCt95
6C8nJyVi0aBFcXFzw29/+FklJSdZ+GTbvWlkN9n+ZDQCYf380ghQc2tpRxl7jEtPk+EYXe437
+rgak+L+txJkpdwN0tt8t4qfNviOiw4CgLuLMx6bOpjfz8jiJEJ0dWq9/dPrDSgvd5y5DmSqv
kGPzMuVSM8tw/m8UpRUmtdr4uft0jTMOkKBQWFyuLnY1v2lP779Hco0Ta9F4e2K1xLusXJE3U
+p9LJ2CBbH9od6irauAZcuVxqHWReWmP++c3aSoFFv/teFtY/e7XBDsh2t/ekNbc//++hHXMg
rBwAMj1Dg2Yc7X360N2juNW5Oii3RayzrIzUmxS3/6+7a8e9UzWXt2uppDg/ywmNTHbOsnaO1
PfbItjIA6ja97S7rjYrmucjlyLxcgQYz5rI4SSWIDvHB8EgFYiIU6NfXg3cpichu1dU3Irvwp
nGIdcH1KrNXn3V2kiCynw8Gh8kxKEyOAYFe+Mv7ZzgkkuzKtdKfR01cLXXsmwMdIYRAmabOpM
f4SkkNbpRru9xr3LLHuL1e444KVnriT4visOKtb41zmt1kTljxX3chIsib39mo2zBh7iU+OJq
Nn9RNd1n/djTH4e6yNjTqcelKZVOSnFuG6xW1Zj3P11NmHGY9ZICfzfUiExGZq75Bj9yim7d6
kCuhvqYxa+FCAJBKJAjv54VBoXIMDpMjKtin1Qr/HBJJZH909XoUldbgyo0qFN5o+u+VkhrUt
lMi7k6ae41bJsed7TXuKIlEAmenn+uHu7v2cbjRLGR7mB30Eo54l7W0srap5FNuGTIuV6C+of
1eZKlEgqgQHwyP8ENMZF+EKNmLTET2qVFvQN5VDTILKpB5uQI5RRqz1mQAAAmA0EAvDA5t6kG
ODvFp94ZhsNITSQtG9sohkUS2rrnX2JgUW6jXWOF9a66xvydCm3uNfd0glfK7E/UeTJjJbjQ0

GpBdWGlcsOtamXkL1Ph4yDA8QoGYSAWGDJBzMRoiskt6gwEFxdXGlayzCyvNulHYLETpYexBV
oX6wqMTbSGHRBJZn65Bj6KSGhSWVOPK9aYSToU3qu84+qM9Mmcpgn8xnDpE6cHvTERgwkw2ru
xmHdLzmhLki/kV0DW0v/CERAJEBvs0JckRCvQP8LTY/Bkiop5iEAKFN6qNq1hnFVZ2aPGdAD/
3pjnIob4YFCqHt4fMInFxSCRRzxBCoFyja1XX2JK9xs3JsT97jYluiwkz2ZRGvQHZhTebkuTc
MhSZOXzc270Phhl7kf3g6cY7okRkX4QQuFamNS7SdelKJaprza9n3NfHFYPC5MZh1nIvl26Ml
ogsSdegx9XSGmNS3Lwgl2V6jT3Q39+LvcZEncSEmayuokpnTJB/yi9HXb0ZvcgAIvp5Y3iEAs
MjFQgL9GIvMhHZFSEESiprmxLkW/WQm4c5m8PXU2aSICt93boxWiKyBGOv8S/qGl+v0Jq9in1
bFN4u6O/vhZAWyTF7jYksgwkz9bhGvQG5RTeRnleO87llKCypNut5nm59MCzCDzERCgwN94OX
u2WGFxIR9ZRyTZ2xBznjcgXKNTqzn+vp1qcpQb41zDrQz51zholsWH1D8wrVlu419jCpa9zf3
5O9xkTdiAkz9YjK6pa9yBVmlzMID/Iy9iKHB3rzTikR2ZWbNfXGOciZlytww8ySd0BTiaaBob
7GXuR+Sg+OpCGyQUIIVFTpcPmG5XuNQ5Se6B/g2dR7rPRAgNyd34WIehgTZuoWekNTuZP0vDK
czy3D5evm9SJ7uDpjaLgfYiIVGBausNgiNUREPaG6tgGXLlcYh1l3pIyfSx8nqPr7YlCYLwaH
yRHq78UvxkSdJIQwKbOmrWtAbtFNRPTr2mruv+w1LrxRjcKSatTUdb3X2KSusb9np1ayJyLLY
8JMFqOpqTeuaP2TutzsD4+wAC8Mj2xasCsiiL3IRGQ/anWNuHSlaf5xZkEFrtyoNnv12j7OUk
QF+xiHWQ8I9DJZfZqIOqeopBqpRzJM1gSorddjw3unMSDQC/HT2q8X3txrbDKcuqQaxeVd6zX
283ZBf6XpCtXsNSaybUyYqdMMBgH1tZ97kfOLq8x6npuLM4aF+zUNtY7wg48nV3IlclTXymqw
/8tsAMD8+6MRpPCwckRdo6vXI7uoEpkFlcgoqEBBcRUMZn57dpJKENHPG4NvJcgR/bzRx9mpm
yMm6l2KSqqxcd+Z284Tzi+uwsZ9Z5C0YKQxaW7uNS78RXLclV7jPs5SBPf1MPYWh7LXmMhuMW
GmDqnS1uOCuhzpuWW4oC43u+RJqL8nhkcqMDxCgchgbzhJ2YtC1Bt8cDQbP6nLAQB/O5qDZx8
eYeWIOqah0YC8qzeRcWsect5VDfQG8xJkiQQYENiUIA8K80V0sC9cZEyQibqLEAKpRzLaXVRL
q2vE//voHKKCvVFYUmORXuOWw6nZa0zkWJgw0x0ZhEBBcRXO5zYNtVZf1Zg13NBV5oShxl5kB
euBEvVS10q1xscdmc9rLY16A/KLq4wrWecU3URDo6H9J6Kp3F1/f08MCmsq86QK8YW7Kz9miX
pK3lWN2aPdKqp0OJlZ0qHzN/caNw+nDvX3RLDSE55u7DUmcmT8JO8FOrrwRXVtA35SN5V8uqA
uQ5XWvF7kEKUHhkc0zUWODPbhXDwisnkGg8DlG80JciWyCiuhM6MWfLN+fT0wKLRpka6BoXJ+
cSayorPZpRY7l9zLxaTHuL+/J/zlbhwhR9QLMWF2cOYsfBHU1wNXrlfjfG4p0vPKkXv1pllDk
1xkThgSJkfMraHWft6u3fhKiIi6ziAErpbUIONyUw/ypcuVHaqJ6i93w6DQn2shcw0GItuhrT
PvBn9LfZyl6HdrrnF/f0/jgly8+UVEzZgwOzBzFr5Yt/ckXF2cUFNr3hfGfn09MDzCDzERCkT
392UvMhHZNCEEisu1yLzctEjXpcsVZo+aAZrmJg4OlRtXsuaNQSLb5d7BBbV+MyIIC6cMZK8x
Ed0RE2YHZe7CF3qDuGOyLOsjxZAwv1sLdvmhr4+bpUMlIrKo0srapkW6bvUiV1bXt/+kW7w9Z
Mbe48Fhcih93bpUs5WIek5sdF98/n2B2cf/ekQ/JstE1C4mzA6qIwtf/FKgn7txLrKqvw/Lnh
CRTauo0iGzRYJcerPO7Od6uDo3LdJ1a5h1kMKdCTKRnYro540BgV5mff8JD/JCRJB3D0RFRPa
OCbOD6ujCF0pfV0weFYrhkQr4+7IXmYhsl6amHpmXK4zDrK+Xa9t/0i1uLk5QhfjeKvUkR4i/
J6RMkIkcgkQiQfy0wXecjgYA7i7OeGzqYN4cIyKzMGF2UB1d+GLoAD9Mujukm6IhIuq8mroGZ
N1KjjMvV6CwxPzyVLI+UkQ3J8ihcoQFenIIJpEDC1Z6ImnBSKQeyWizpzk8yAuPTR2MYKWnFa
IjInvEhNlBdXThi44eT0TUno6WtGtWq2tEduFN4zDry8VVZtV/BwBnJwmign2Mw6wj+nlzcUK
iXiZY6Yk/LYrDire+NVYJcZM5YcV/3YWIoDu3P0REv8SE2UF1dOGLWFXfboyGiHobc0raNffw
1DfokVN009iDrL5aBYM5te0AOEklCA/yxqAwXwwOlSMy2AeyPlx3gai3k0gkJjfL3F37ILKfj
xUjIiJ7xYTZQXHhCyKyFnNK2q1/7zTuHRqIotIa5F69iUa9eQmyBEBYoJexzFN0iA9cZfwoIy
Iiou7BbxkOigtfEJE1mFvSTlevx7/OFpl1zhClR1OCHCqHKtQXHpxCQkRERD2ECbMD48IXRB1
38OBB7Nq1C1euXEFoaCiWL1+OBx98EACQlJSEAwcOmBwfEBCAY8eOWSNUm9SVknbNAv3cjatY
Dwz1hbe7zELREREREXUME2YHx4UviMx36NAhPP/881izZg0mTJiAzz77DCtWrEBgYCBiY2Nx6
dIlPPXUU1iwYIHxOU5OnC/bUkdL2gFAXx9XY4I8KFQOuZdLN0RGRERE1HFMmHsBLnxB1D4hBL
Zu3YpFixZh0aJFAICnn34aZ86cwQ8//ICYmBjk5OQgISEBSqXSytHaro6WtBszJABP/sfQboq
GiIiIqGuYMBMRAcjLy0NRURFmzJhhsj01NRUAkJubC51Oh8jISGuEZzc6WqJO4e3aTZEQERER
dR0TZiIiAPn5+QAArVaL+Ph4XLx4ESEhIVi2bBkmTpyIrKwsSCQSpKWl4dixY5BKpRg/fjwSE
xPh5eXVpWs7OztOneC4Qf4dKmk3arC/Q73+3qLlbB6JxLHew0RERC0xYSYiAlBdXQ0AWLNmDZ
YvX45Vq1bhiy++QEJCAvbu3Yvs7GxIpVIEBwcjJSUFBQF2LRpE7KyspCWlgaptHMJg1QqgVz
uYcmXYlVxvu6I6u+LnCuV7R4b3d8Xdw8N4loKdkgqlZg8dqT3MBERUUtMmImIAPTp0zSUOD4+
HjNnzgMADBERUUtMmImIAPTp0zSUOD4+dixYMaJT1zUYBDQar
WVehI14bOogrE87BW3dHUrauTpjydRBqKx0rNfeWxgMwuRxRUWNFaPpGbwpYH+C+rqjTFMHAO
jXl38/IuocJsxERAACAwMBNCXCLUVFReGrr76CRCIxJsvNmo8tLi7udMIMAI2Nhk4/1xYFyt2
R9Ej7Je0C5e4O99p7CyFMH/PvSLZo3qRo7BfZAID/mhRl5WiIyF4xYSYiAjBkyBB4eHjg3Llz
iIuLLM27PyspCaGgoVq5cicrKSuMiYACQnp4OoCmpJlMsaUdE1hak8MDKuXdZOwwisnNMmImIA
Li6umLp0qXYvn07AgICEBMTgyNHjuDbb7/Fu+++++7Wi2XLlmHHjh2YNm0aJUyfPp0SpkyZ
QgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnTJ0+fTgnT

0rZ98GS9oRERGRvWPCTER0S0JCAtzc3LBlyxZcv34dkZGR2LZtG8aMGQMA2Lp1K1JSUpCSkgI
vLy/MmDEDiYmJ1g2aiIiIiLoNE2YiohaWLFmCJUuWtLlvypQpmDJlSg9HRERERETWwsKJvURQ
X3fjY64USURERERE1D72MPcSXCmSiIiIiIioY5gw9xJcKZKIiMg2FBUVYeLEia22r1+/HnPmz
Gm1vaSkBBs3bsS3334LABg7diySkpKM5fCIiKj7MGEmIqJuE9TXHWWaOgCcDkLU7NKlS3Bxcc
GXX35pUmLNy8urzeOfffZZ6PV67N27FwCQnJyMhIQEHDhwoEfiJSLqzZgwExFRt+F0EKLWsrK
yEB4eDn9//3aP1Wg0OHnyJHbs2IEhQ4YAAJ544kgkJCSgoqICcrm8u8MlIurVmDATEVG34XQQ
otYuXbqEqjzbiC5uLjA3d0dBw8exOjRowEAhw4dwoABA+Djw7rmRERETdzaYTZrVajVmzzZuFPf
/oTZs2aBQDIyMjAhg0bcbOHCBfj6+mLhwoWIj4+3cqRLSAwJC
Qk4Ne//nWry11cXLXLBhwwa8/PLLLiIuLg0QigVKpxL59+yCVdq59+2BxcsHfvXgxAbG34XQQ
otYuXbqEqjzbiC5uLjA3d0dBw8exOjRowEAhw4dwoABA+Djw7rmRERETdzaYTZrVajVmzzZuFPf
/oTZs2aBQDIyMjAhg0bcOHCBfj6+mLhwoWIj4+3cqRERERE5snKyoJSqcT8+fORn5+PsLAwJC
Qk4Ne//nWry11cXLXLBhwwa8/PLLLiIuLg0QigVKpxL59+yCVdq3YibMzi6WQ/WkxiwESCd/H1P1
sNmFuaGjAqlWroNVqjdsqKiqwZMkS3H///////UhOTsaPP/6I5ORk+Pr6Yvbs2VaMloiIiKh99fX1
yM/Ph5ubG1avXg13d3ccPnwYjz/+OPbu3Ytx48aZHC+EwKxLlxAbG4ulS5dCr9djy5/bsRWjz5YtePrpp
/HBBx/A09OzU3FIpRLI5VxXxGoxyPVCoxecz3MXU3m02Yt23bBg8P0/8BPvroI8hkMqxbtbgeF1
6IjIxEQUEBdu/ezYSZiIiIibJ5MJsPJkyfh7OwmUwGABg2bbBhyc3ORmpraKmE2KmE+cuQI9u/fj3/
/+9/G5DglJQX33jcvxcjtwKhRgMAhqNtv0DiWyMwMBMHldU1Fgxmu7HGwLWZ5MJ88mT
J/Hhhx/i4MGDmDBhgnH7qVONmGrUKDg7/xz22LFjsXPnTpSSVlUPnc6cfHB+Hh4cjPz+9SHcjEQt
t3dvdU2lUqFb775ptX2O6dPIzw83KQn2cfHB+Hh4cjPz+9SHI2Nhi49n8gaAhXuKL3Z1P4EKT
z4PqZuZ3OD/jUaDVavXo0XXngBQUFBJvuKi4tb1RxsXmHy6tWrPRYjkS2YNykaQ8P9MDTcj6s
PE1GPYvvTeZmZmYiNjcWpU6dMtl+4cKHHNhcCCgoJQUFAAnU5n3FZbW4vCwkKEhYV1e7xEtobt
D/U0m+thXrduHe666y7MmDGj1b66urrj8KVmLi4uAGDyQdZNvfpsakqpzXDLg7BDyPsadJkT/AJQsVNpCqTLQIPIb+ZXHETEaOK9JoPc6qIjhW48E63/AC+seWSktcMgol6Iq
593nkqlQnR0NJKTk/HSSy9BLpfjo48++4wo8//oi///3v0Ov1KC8vh5eXF1xdXfHQQw8hNTUViY
mJ+MMf/gAAeOONNyCTyYwLohL1Jmx/qKfZVMJ88OBBnDp1Cp9++mmb+11dXaVfX2+yrTlbt
4k7m4YAARERH1BKlUipsUFLz++utITTyERqPBkCFDsHfvXgwcOBCFhYWWNGkSNm7ciFmzZsHf
3x/79+/Ha6+9hkWFkEqlSIuLg4ffPABvL29rf1yiIgcnkQIIdo/rGcsXLMGcZ86cMelF1F
MlkCA0NRb9+/eDr64vXXXnuNuP+7777DkiVL8N1333V6DrOXX2vmJoYW5mIqUrQIUVmlKrZSglP6Iw
desITIESiVXtYOwaLY9hDZB0dre+yRTfUwv/7666irqzPZNnnyZDzzzzDOYOnOqjhw5gr/97W/
Q6/VwcnhzcGIQ+Pu+fzhz4eFdXvCLCwYQERERERFRSzY1cTgAIBhYWEm/wBAMVAgeDgYs2fPRm1t
Ndau5XucnBwcOHAARWlpePLJJ3tVCtdaSiwOq1TkanOu0KLBnzx6o1WrIRvMnDkTfyN7uw1FlavX
o2ZM2daOzQiIiIiMHTxwNkYXt4sOm5KVEB6DMNEPc49KiVNgvlwb2tlV5frRe7xHnrsBzHnaudasZtTMuGaIS7RBdJxTt0d5mdgPn5HzaMAwYqqFwCwH8RNrBE+rIOe3XXjv4HhuPmj53hUwLR+t+aBjLl1LdSwo6Rn
Ak7xqiHdCDwu0MjhJcgpk3fNzm+r0BZCPQOH6DPZt7QWXfa2UFwBvD4ZGQX1ybuNZ9LaQXQIkuGfwMoi+UQQhjL8SUO7bW0mYCtBS6YUHcFd3+zoklNL6K2jL3dsWHDFWNR5KUkWQ8XwCKd8IyKwVUD6NhqRf8S/exK3x6EaQiB5x0+DpbhV

jIw02dfyur908eJF5ObmtjqnTqdDbm6uyWvpzfz8/DB//nx8/vnnyMzMREFBATIyMgAABoOh1
fHTpk3D1atXjT/v3r3b+Dg8PNz42NvbGwBQV1eHvLw8NDQ04O677zY5V1xcHPbu3WuyWEnLc7
i5uaFv375wc3MzbnNxcYFOpwPQ9Dc+f/48PvnkE+N+cWvIUm5uLkJCQjrwmyB7wfbHMbDtIXv
DtsdxsP0hR8WE2YbIZLLb7vvDH/6A+Ph4488+Pj44deoUAEAikbT5HIPBgBkzZuCpp55qtc/P
z++O1xW3WZjBYDBg7NixeOmll1rt8/LyMj52dXW9zSvpHUpLS/Hwww9DLpdj0qRJGDduHIYPH
47x48e3efyuXbvQ2Nho/DkgIADnzp0DAEilrZcaEE2jQwC0/vs3fyg5O//8v3fLx7c7Z8vnL1
26FDNnzmy1zx4Wa6HOYfvjGNj2kL1h2+M42P6Qo+KiX3ZCoVAgLCzM+M/X19d4dzQ9Pd14XGN
jIyZMmIAjR44gOjoa2dnZJs/T6/XYuHEjrl27ZtZ1f9kgRUdHIzc3F0FBQcZz+vj44JVXXkFW
VpblXrCd+/TTT1FZWYm//e1vSEhIwAMPPICbN28CaGrwf/l7DQ4ONvk7mfOhGxERAWdnZ+OXh
2anTp2CUqmEj49Pp2KPjo5GXl6eSTzXr1/Hq6++ipqamk6dk+wb2x/7wbaHHAnbHvvC9occFR
NmOxYeHo7JkycjOTkZx48fh1qtxosvvoj6+nqMGzcOjz32GDIyMvDiiy8iJycH586dw6pVq6B
Wq41Dmtrj7u4OAMjKykJVVRXmz5+PqqoqrFixAhkZGcjMzMTKlStx/vx5REdGYo
ra3FP/7xD1y9ehXffPMNVqxYAQCor683/l4zMzM73RB7eXnh4Ycfxptvvol/PP/0UBQUFeP/99
7F//3489thjt7373p7HH38c//nP0f7Ft2zao1WocP34cSUlJ0Gg0vMtKRmx/bBPbHnJ0bHtsF9
sfclQckm3nNm7ciFdffRXPPvssdDodRowYgXfeeQd+fn7w8/PDnj17sHXrVsyaNQtubm4YO3Y
s1qxZc8chUC3J5XLMn8M+bPnw8nJyfcddddSEtLg0Kh6OZX
az9++9vf4qeffsKmTZtQXV2N4OBgzJkzB0ePHsX58+cxe/ZsjB8/HomJiVixYgUee+yxTl1n7
dq1kMvl2Lx5M0pLSxEWFoYXX3wRDz/8cJdi37JlC3bu3Imd3fCx8cH9913H/74xz92+pzkmN
j+2B62PdQbsO2xTWx/yFFJxO0mbBARERERH1YhySTURERERNQGJsxEREREREbWDCTERERERNQGJsxERERERbWDCTERERERNQGJsxERERE
RERQEbWDCTERERERNQGJsxERERERQEbfj/CeFDEnEDD
8kAAAAASUVORK5CYII=",
       "text/plain": [
        "<Figure size 984.597x600 with 6 Axes>"
       ]
      },
      "metadata": {},
      "output_type": "display_data"
     }
    ],
    "source": [
     "#sns.pointplot(data=df_ci_plot, x=\"program type\", y=\"value\",
hue=\"program type\")\n",
     "g = sns.FacetGrid(df_ci_plot, col='task name', hue='time
point',col_wrap=3, sharey=False, margin_titles=True)\n",
     "g = g.map(sns.pointplot,'program type','value', dodge=True)\n",
     "g.set_axis_labels('', 'score')\n",
     "g.add_legend()"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 45,
    "metadata": {},
    "outputs": [],
    "source": [
     "n_times = 10_000\n",
     "change_score_dict = {}\n",
     "res_dict = {}\n",
     "i = 0\n",
     "creyos_tasks = {'SS':'SS_avg_score', 'DT':'DT_final_score',
'ML':'ML_max_score', 'RT':'RT_final_score', 'FM':'FM_final_score',
'TS':'TS_max_score'}\n",
     "for t, score_name in creyos_tasks.items():\n",

```
    "    diff_score_1test = get_test_df(df_2programs, score_name)\n",
    "    res_init =
stats.bootstrap((diff_score_1test.loc[(diff_score_1test['trt_grp']=='Init
ial'), 'diff_score'].dropna(), ), \n",
    "                np.nanmean, confidence_level=0.95, \n",
    "                n_resamples=n_times, method='BCa')\n",
    "    \n",
    "    res_virtual =
stats.bootstrap((diff_score_1test.loc[(diff_score_1test['trt_grp']=='Virt
ual'), 'diff_score'].dropna(), ), \n",
    "            np.nanmean, confidence_level=0.95, \n",
    "            n_resamples=n_times, method='BCa')\n",
    "\n",
    "\n",
    "    change_score_dict[t] =
res_init.bootstrap_distribution/res_virtual.bootstrap_distribution\n",
    "    res_dict[i] = {'task name':t, 'program type':'at-home',
'ci':'low', 'value':res_virtual.confidence_interval[0]}\n",
    "    res_dict[i+1] = {'task name':t, 'program type':'at-home',
'ci':'high', 'value':res_virtual.confidence_interval[1]}\n",
    "    res_dict[i+2] = {'task name':t, 'program type':'in-center',
'ci':'low', 'value':res_init.confidence_interval[0]}\n",
    "    res_dict[i+3] = {'task name':t, 'program type':'in-center',
'ci':'high', 'value':res_init.confidence_interval[1]}\n",
    "    i += 4"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 46,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "SS 0.9074\n",
      "DT 0.203\n",
      "ML 0.3582\n",
      "RT 0.0472\n",
      "FM 0.2717\n",
      "TS 0.4751\n"
     ]
    }
   ],
   "source": [
    "for t, score_name in creyos_tasks.items():\n",
    "    hist = np.sort(change_score_dict[t])\n",
    "\n",
    "    idx = find_nearest(hist, 1)/n_times\n",
    "\n",
    "    print(t, idx)"
   ]
  },
```

```
  {
   "cell_type": "code",
   "execution_count": 47,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/Users/rye/opt/miniconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:718: UserWarning: Using the pointplot
function without specifying `order` is likely to produce an incorrect
plot.\n",
      "  warnings.warn(warning)\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "<seaborn.axisgrid.FacetGrid at 0x167df7fd0>"
      ]
     },
     "execution_count": 47,
     "metadata": {},
     "output_type": "execute_result"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAA4kAAAIzCAYAAACtCFzyAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIW
XMAAA9hAAAPYQGoP6dpAADJsUlEQVR4nOzdeVyU1f4H8M+wDIuAgoIghCgqCIKCIJgQLuGCWW
YuqbjgminYIio/vdFi6RXUmwtqCq6lZm430wqt1ApRxDUBkdxYBwUZ9m3m9wfx6FwWGQSH5fN
+vXw1nI3vI3Sc7zznOUckl8vlICIiIiIgKgpuoAiIiIiIqOlgkkhEREREQCJolERERE
REQkYJJIRERERERAiaJREREREJGCSSEREREREAImiUREREERCRgkkhEREREQCJokkk
Mvlqg6BiFo5zkNEpEqcg4gqMEkkAMDp06exZMmSBh93exZMmSBh93ypQpmDJlSoOP2xL98ssvmDZtGlxcXO
Dg4ABvb2+sWLECDx8+rNL20qVEd8evXph4MCBCAoKwv3791UQOVHHD4DykWjY2Ngp
/7Ozs4ObmBj8/P5w5c0ah7eDBg6u0/VdGCVENUP5yDVqpw71q5dW229TCaDp6cnbGxs
cPjwYQBAcnKywtfUcDRUHQA1DTt37lR1CK3akSNHsHTpUkyYMAHTp0+HtrY2EoaWIfTpUkyYMAHTp0+Hjo4Obt++ja+++gq//
vorDh06dq0hEJ4aKBhCkTBDCpU6ecWalmX7sHnLCFbGpHEdX2GX6wFy4+lmFSAqBsIhg
ne2LFjMW7cOABAaWkpMjMz8d1332HOnDn417/+BV9fXwDAxo0bt+ep+cKjfl0Ep05SQ4+b
lRkZGLzZ4oufEOUj11NTU8OOPP+KDDz6oUnfx4kkVIJBIVRNU6MUkkagI2bdqE1157
5u7uDhcXF7zxv47Zc/DyZdHg/DOgG+Npq9f6fJCNaT5U2v0Ptxr4PJMQtKbq9Z4w00tUP4/nAYQFdtmU1iEzI+HzTbZhn8/3EwYSZO
1LxpUHQJQG9Bb6GJbBy5MwQl6OFmcEN3AGKQgOBr2WjGOVo1vWzM9f+rotHJyW3lXTtmXLNm31LZRf37RTIBbTQBgJvNXPqIr/NlLT
8PGxgZRUVGYMmdsLtsPsdP9HmmmvXr0vnBcWlxBJ7UAALhDPIE+AjAaxMZZcW9OCKgG3lqKl
HvPjSrlq0emoMnPdH9AhtBvcVtEq2kRxwaRSMJOd3HfMKU2xpmgqOsP9IHAlLcZ0eCnj7T6cRi
0Xla70mfCrvS+rfWavUb7/8XsAMNZEcWxQlO8u1RtXR7gNsH6tRgpDFi1p8TxNpfyhiSqlFu9lYc5
OBmMTS4SE+pSAfAs+++dAAByEOmHa+/M+Z8Te3IcB7kSUL5ipZ5WvHy58lsQ9fN2rKJxgOeq
IWzLhDkFMGo0ofVn8HC+9OqAxNYnADAmSSqMoKo8CvvMUa+uOUFWb5OPn5kp1mTSnxUNUmvIGcp
wncW3IipWcRRwvNjRZhq/RzwSsp2+A2m+JvMmS4GwA35vTQAn1WfTWK13bO+hjQ1Jd0/C4eq8yV+
4FxOD4lrOjjH5PTh9jLeCIL+NJXHRpaB5SKGvVa/xHGtJkGAeAX5AF1Eq2Gn+IdMySQGOxfyKhlW
ZSM+t5AZUbipDSIBRfs/jFrg2XoU+j8Y0l8CMD46zFsFHx3TC3zaNHdwRbA9z9GaWcnvwqoDdZDyY
xg/8JHqcVNR/LvVG6pCtu+7TK7dEPbdPe+YMNSnt3uW+/JdzxUXhAn3UItvxj/lmtdcGhfnPRG
XmK0/xbBNdhVKJP9AtfDWcwBjlXPTKHPo4eJB2mVHGbggdaEHK88h8OzuAQvVGdC8l4vV4Q/u
bSt2ER+WSP7hoDx+gcFSa5fwb0xvj8uUd2Ed2vcZQ+B0v82YC7rN5KAHiqEgHwNzMp6Dq1
5o8i/cUX2fx4J+AUIy11jbHCsWUmpxi6GnFzaBSsaLaVOAs9Zk1sabbUWgGXUvJ2UGEUiJkrT
lSN+ho2EDkbpDL8pNtdPDXZVdvAjGfOFTJwhD/ahVP9tCz8T6TxrAnv97ZzfFR48YspjIO6kBiR4BHsRY
/FN8kB53sXU+MUhG8P0T7Y9/lv2RVTcP7DiaO6nnuqA0DzrS+FRf0Y4nzsGmhMrw8ODKIrIEi1
kDwHtXC7yEwemDlMwePVHS0hWxgW3UWqBQ2rwrpzd8u6VNQ9NnYFDmYnNTXdUIqYvUHjdmt2DLdBa
XTIPpNsx1t1QDHCc1kgxttMYl0UFQPgHAgV3lwCs4FTH8ddb0RQY/9CChSpA+tyQb5ilcaQRM5JRR
HdRT2hsQqk9sCm/KOFzf3PtlSVG7Y5x2KpgOxEeT+40kKQj/g69ZqnlHEcvWa9OCkUYdDWkA2UTh
Sq3BJnD0i+5X6RP4klsS6XE+SfSnrqHTsvM9YjO/0DDG8IKSdIfvvwmcSb/wp6+GmddsJY6jcHLI
cbmTVT2nn2VcfwakRfn44pLvMEC9vwVfLZEmQOH9EZ4JzmLCqWNWv8MagbiQ06hi2LAhfXUgfrXdVtz
UPgNWtqJkndk2x6zgeAlAbOYQJEdMmqNl9ayrG70gOlBSOYdWMjUJuwY1ozOCvTz4dZ+Zd4pMwOXt
cP3Uqqqc/NJsRNVCN3pc+gjBwSsbGzyCaptA80D8Qy2tRU8C9FKwbX8xbRJYhH1MuA8QP1pQgmzi4
aTLOuS3QbHOMkCCNcvtZf0NsyhLO+VUnEQTs0Ep2CDGGpDbGtFGhpjo7fCTnbjq4sDHsRGbuCxnnzM
wvMtoeBKSs3cL7dBGpnECNQmtHHuWnNdlUoxbdfa44nuo5D0aAu17Bbe6fJKrhWZk6GFZazAMIC1jV
YgWCk5VwsdDPVGvVaRHoVFycxG8M8tZ7ppYM7Db3qEsuSj1SM4cFRQ0bR2xOzAGgevWmgOKLrm
NCBvQuUOn5nGLRlWCMzC/ABZubfgPvVIGRqmz4zQHk9NPjzdKM8tOTR3OFcXW4rqV3ESXtD2Pgh
MJKeLoNJEMNbVpRm0kjl0qTAxeFb6+CpsqhHUVMADTGeyBFd1VltMwPjK0cIq7y91cDKU+/dS4RLIW
mOjxZoPXT6yc2nlw+N/vGL22XRkd3zeMKtCq/j/Za/nkXnhgY4Y4SFOyyVUOYnTo09+yHUPjDn8v
1R1kxnGIUeFomeMfw39CT4mohMtQKvHQiIsqGvTg8GX5dfuDxEzcTF2XMGrZYdJNTEIJT8Z+cp
HtTlT9aGe6f29NYNR5E4aNiW3gaY69Uey7EtyLeCG4JcUDdx1PgfiDFTDV3UEUF1PkAKpvk6bOG
AA5u5tJ8ZV36W9JJ8XIG2WWAMPqJgcRq1Kbn2rG8LqtnE87Lxn70/i2k8c+5E2o0+kDm4EKd2H0
J4MJnu+dbx4FB7vAQSvgvsrULG8l8IRrqmMzNUZ+rFQDaZmqNFMWKdo8I8sxXfV4xEXmNkN2Ie4bxHH
2R3T+a34WO6VyJvW0Qz88BsF+cqSVfZ57aVZEbepKaCyIEtUKJW0ZhQWcgDW0DnmhQCyPaM4tsu
jyZ6WnfLV1aDqOG0vpbE5fQUqo1iA+2d7Iyk26GG3CSGBEfy9+SX8bIMphOjkswm28r+0ywwh06WZf
2ayJoW/AkIBYgr6KbvpYcQ4p5rU6C8bSVX/OSbcQdCRd7+ZQzyfBEAXPTCoM6bGgE8zEsAvJ
Db5IlOHtZFjoONRtrBRyTlthORQtx67blq5Uu+xJP0vk4gCK9WYvaXV8Uy9ctcRNYu9lWRZFrX
Ry5GHFeS1jUtJxcgfzzTfE7SCi3TNhdVVyddgT+LhIDf3Q5xzfqO4lp8WWqwL8cXy8XTMzkVprY
J27wAEWrCwYLu/wyMF9GUAMmN8b23uWhSC2sbiN6ZTX47cYOUtSLzR9OdiYf4BDkzxNLZYW1yu
X13jmnCAS0edUqyzoo1c5ngnHw7v7njiTdbo9CzXnaz/IkwpSEVd7tsnvKYMdaRhtlIRsg3EaBZM
6+4lvGJY6J9A9Y8vPajlIkLpFUuV4+ZJC13Ok7Zqwj3zxkoysA4NL/DNw6oMT0iICRZhe2/9Pom
c9vzkhMMjGTzJTCNxpAvZTXg+U8+2rvjp2ANHsGpxCwa1aYcEgOGc2Mbl/WBh0ccOE8TyGaYF
5J46drSxaXiGw2kFkBtmzuUHqv+XkAhvcAT1pfmfA6D3nJqEvWE7QE+6QNrY1pPfiYM5oijDL4NBBx1
DcgtoHfUvTGHiFgvLncXq6H/fXp0NxEMa8JSIZP/lPQBmDG0H8d4l2Qgvy6uQ2Z9JBPq0I7Y2PNxG
RBrIb8ur74+LjRfkgyh7w/FGGfMkY5LNQ6k+sMcDF3Jy4b0jfAiejI+MWkYbeOiOwyHJf6n8Dh8E8bn
+uJfFXGPhOFgQOuGfQRVmZcoJuZFPmNySoZnzU5zaR9cuNPrfpP9UFPVGlSbXehyLpSpBT0aoyFdQP
+x0xrWEr2H+W/Y1KCYqwBVD87urcz0Y1g4fQyIKJkPzrJd2ATaFnEERzvdExOEyt1TRNuO2zfOejyL
oqm1UJ2gdaYjgaFkPXgxuHwCBZ4kLhxR3Kqpbcb23etBo2sFo5J1lXDMrqO0xfGHFwpGYpfyJJfj
8ZNuGhk8vgmNMFNiBrbd8d9pmnfT+YnpwmGSWHz5EPcGcfQjr2KfI4Sz1dKiOhyhR6/LkodBPE1yvJ
iwAcgIwuinwbBiM9KWcmV1UwjOLn9JJrL1zd8eKRsL2/+Db2xxHqbOsRJCZZd5eB21qwaGT1DDGrO
WlkCbzk2pDZSxUqG5tMH3r6FBw+Jo71/7ALWT0AahTH2z6A7oerfv+WAdGb4sO+q5w9BHb/6BPbdm
E9JUVeQPcHQJG2mAirO7MmnIHBYNQJ3kH1k6unHeQKr5RMr6gnGMi0XxbnlI9j/VYaIZN/lGbadiE
bbArmnaoP+tzDG8NZSTJMqyzqFXF2YDP2i6lHhv5NdfWA9yIwIHbVyKl+QcE7F1iZ/4wl6XTpdwWv
XS1hr6RumhbOm9rgTtpXNzrRgBxB3PJ17prfIrbVZ8wtSA6bwvzvUzUoaHKaPjYKDbGcE6cBmBN
4IG/1jn7E0ExkHJ9L3DUkBWE/n4p/GaF1rT4h9L1zBiiwZYjGT1C7fZRmgZoxnxl+JADDxRnehQL
+yiIjWgNoDKFHs8gEkS0MDjLVlwYnLZ1UG3mJ7i0tXnwDKgw3nPcS1/wZnNtxq2tBQvs4PqytJm2J
fEMmpFJDHfJH3ZbKPR2g9pFKbwcfjwjXrgzlZo/JqoFZ6GHijSJ9tvfXrBtDu4jA3AQcNdzrrGMmM
T/XiyjoSKhmsBTSWeVI0Kl7b8mM3EfC4PiddGklvfgbXQTsTSiyipcEJNHAGcBHgqUNbpUTbgXI
Wk3VF4e1xd98+LYOTmgbZntBJhMBNPPZzXSfHz7z7Y/s9Y3FVdcc55Eeyx8GHF+mDK/MFHg69aTD
pKzONN9f93YQLaHj4TtAGJifH2D4/nWXg+q3R/9U2ZaYZpjdi9OeoaWOMjuGFhjHW8d7HZHtFXbV
d9DA0xjwl9QMT4l5rOzNYc1tQS06njZcmbiUVLPCfFiGtmSYpMD+1ci4HbhqhsMBqYnPrvWkrLwfM
AvzZ3vumnDJAtQFBvvQb8OxFX/XxdvUZ5ck5uTPfQPBjxJ9Jj/r5R8y2/Rr/+k6xfLm8/ggk1TW6B
cdBOkKWSGHGgTz6pa1ekmzlBO6JvNU4SqUkkJlbU/3Jw/tEL/ZrXxupsDBrqXQB9HaOAiG1
W9TyM0AjKlIBDxVKFYaJbsngWL9SD4pG/Z+MrCgMG0JlW6C9M8v1X03yWrDnMsc/fNDRz1w
E+vvdC9bSIG/PeZP/ISJfNzzvOjP+DhZ4jXJjZ8GJw3j/dRwwxsmSIgKP7/LRL1edpewl/tAk
2CPd/4BK3obK9tOpGtP0y56ci0jI1A9HRn0HLBOIVZo2iwcS7OlIT7nHAN2ZK3f3AkVZnq0HIGZG8
vPxz/7CPvIF8gnrXYvmWmtLmGUJpW1YiURYeV2XcKbQ/F0+zMwY7ou2wXa5OkXjOgy3rEVVR83H
9YiaQ5DqIxhe/gSGnxK87Mmz2kZMvrHjOdQr/Ew6TQh6QF9fkF5nb4fiCLsSRWBkyp3dEJLS8vAqq
Nc2eOr5HP8oBt2pNqjMrK84OUmcRJGxQhJqm9gcgPbZ4aa+8WcyE8aKpdyxeHNPY3Pq2hOCdzVRS
Zq5vV/kzrnxxpfr/4xfwyLsAnM0v6eLkbfp+xlmXKbZ4Y06xtbp2yE1DTGE9Tm7LzhMP9J4c9hzv6ko
Nhl8HN4WSyTfN8xPSx1Xt9RGfSRsdFAo7/C+yymKYDkfS4Q+UB8vx6EHQ2yo4anvvp1tbrLNiDhTuIo
hIRhDeKDFNVNI/8Ns2RVFMCe7uLGwzkL0Ho9xH8vvLW4r6GQGdHPhugDTNRRy3sLSj9H0AIwpHMtFgt
xd4ewHWcbbqCcY5gMZRP64BL7mUlkWxyz6ELJqabI06fPqPBejqvuP2wKuhOpDQcyQLxgwKlqwzQ
vH5zFE4MohJyI6DbwOWuPzTbhuOmYx03J9cVLj5E9OpgoaH5PFfs9qoUhDnPdFCaoCKz6Xl81Lmx
Sj8IfBrYdsQi2vTnD46VEeO6YwApB0yJ0N1IYUzCXr7bGzMOUdIfwC73s9xzOK5Jr1AU48TmvSr
NMOSR3ZnO8Cfek9fyxTTa/wsaWR7WgOvlt+79Oe1RGNeMZGDsGq4g+iL4bSE8DaCiJbW1TpSJUNe
EX28aG+5j3DpR0Y6lHOPQOq6fUZr20wU70lxBU8lzmXCMlabN7XUYjqgW/R4S3b/sJ45OaaRzPP7
x+TPX0iYLr+wf9Y7a02XoDzBqpJ1YfbBgDlOpsfflwkyTQTenKRRUbIxtdFlUqTcxzZfb6RzyzfMCY
WXt4PAFJvWkt1XD7QqbN1vrqyXl6gtkr9JfWIJ4RbL0Dc2QfBUQWvhaCBhkxe8zpI3WkIK3OY4
W8ZOTB4Ju+ofQYT+GO7C7zvdvxzx7d9RAV7fnvymfeT1Bov8PPzPl3a75uDu9PeaEWnfQGWN2S
ZzSeyQgyyPZS2KuTmUOKN2ROkrAx2BMxeZp55onSL12V3Ouu5KFeHNfV+fsmyIKc+bJfXOMudWXk
OjKeEfbnIrDS+y86HEEfPu2Nrd2Sw6Tsf1LcJmRqM8fjlg8MveA/oHSS/+jn1h7HhWVTXbHCSO8gm
yjcO4tb/iQ0v7hvUDX+4N2mnLHdHSe/3uUdqhjqWCT2nL8A3rOb9FOy3lLTpcFtjdCYNfemmoFlb
K2X3kr1P+wOgaUK7Z0I6uBTHQD8JRVY7SgATAL6/iT6FDW3AuiDVh1jKHojn2lz8idNVp5Y
gEP/oZMNXYcUGd44v5KFI5xGjWjPfuINLRAYuE6hUJobgMaNn1BjhMSugDrFN2kmBEREREJ
IIyDz1gmMpxlVX3mNJoW2lDr5kPiBd1v4AZRsAIpnZ3R3QSRjq5o6ITs/ReGDi6EGrU1h1qFcVUSjC
OHRSyMNtRWMXWQnb2m8LiZlYJNt6y7P3fm6yqGwqa+2ZX5Mz6TQlojtvNXy+tsHEf9Kd5A8jyfLk
I4XvIRV5JrbtNfqZaE6UPIy3s+CkDXo5JqLDIyJ51cXfcvm1hKSfhf6YXW/MQXt1Hx2dG3vqEOJ
4k9J+HAh6LlYzpvnMbstt1ckOXZm51ZPzmWPBKXGYImnnZvxMpD5XovcGS2Z3A1vXAJtRhAtW
e8ZrJa7RY3Tb2aRhW2GHnL9lsCLSO4Q3ZMhfGcbswwaTBSXX0uT8+XD5S3OUrKP8dlHlhHnyfQSeu
8iBvMr6cbcfnt3Go5akPqGKIuajzbVpRnsJm4gXwvXnb+kQldxcF/YVc6YZF3j3DYbYkvZ+JZzR
UNWWXzqLQXgKbSOg0Vgt4aFAwMBzwrH4p+IGlMk8l3qJjCrXKhOT/7HpscybG+r06s55xF5YXpk3
h/C0tkJm8CCEqqbePkugtQqe1AVHk4IoNUJ3G5fOXr+Vqmi+bwkHQMg2TPgt9cwpQL2Jbs7uAk
hbBcJZbqBJtDALJ93uEpdPMhRUUBDVXYOB4crHFrBaHfCfyq0eoH0/1fPP7WGCBMwG6+jDdwB4o
ge5ztKa4CaJ0j2gsSR2qAwVl6TF+LRyqEXMpD3kDK41LbAHU0gmXGcWgXxFFSPRHYZGnfHbJqzG4S
/T+D+VKmUvMEAi9zRGQmiNuYczxwZVdLm3ECBUSPc4RJWUpLQLvWCC7tMRCcbJWeMm7M8mTL7P
KNjybnEPOvtWTdmeC8D//JY1oG1Tn5uqXBWXQ4i7XIgfz33VUpAkIK3sbdz6wjzs5nAHE0Ru
ZSqZLFJOumoUaMAjHDCsOGKF60kWvUJ3AoMpOtsrVkcW6RSe3OWSvV8vn3n9w+1B+x8iHLAtEsg
5IrU+0M5Ql96EhDa9kCc5PkdY6AdfJGnV6WyXKUyGJ5kGlZN8Cb0hMN3ZpVuHlxHGqdqBYEI0T9X
Y75LXCjRYbmZFtm1uzz3ZUSVCuDvMJ9DFvE1Ta1oWFPG7Xwuv3wEP1O4vJ96Qb6WRaqJrWa9Q
a9JTUi0LJxLF5AUbMn4I4kYWnIz4tYgTZbB5XhP9ZSrbOBN7Nk8T4T7dZ1RXLDQGDnZqmTQnLo
```

```
"iVBORw0KGgoAAAANSUhEUgAAA4kAAAIzCAYAAACtCFzyAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIW
XMAAA9hAAAPYQGoP6dpAADJsUlEQVR4nOzdeVyU1f4H8M+wDIuAgoIghCgqCIKCIJgQLuGCWW
YuqbjgminYIio/vdFi6RXUmwtqCq6lZm430wqt1ApRxDUBkdxYBwUZ9m3m9wfx6FwWGQSH5fN
+vXw1nI3vI3Sc7zznOUckl8vlICIiIiIgKgpuoAiIiIiIqOlgkkhEREREQCJolERERE
REQkYJJIRERERERAiaJREREREJGCSSEREREREAImiUREREERCRgkkhEREREQCJokkk
Mvlqg6BiFo5zkNEpEqcg4gqMEkkAMDp06exZMmSBh93ypQpmDJlSoOP2xL98ssvmDZtGlxcXO
Dg4ABvb2+sWLECDx8+rNL20qVEd8evXph4MCBCAoKwv3791UQOVHHD4DykWjY2Ngp
/7Ozs4ObmBj8/P5w5c0ah7eDBg6u0/VdGCVENUP5yDVqpw71q5dW229TCaDp6cnbGxs
cPjwYQBAcnKywtfUcDRUHQA1DTt37lR1CK3akSNHsHTpUkyYMAHTp0+Hjo4Obt++ja+++gq//
vorDh06dq0hEJ4aKBhCkTBDCpU6ecWalmX7sHnLCFbGpHEdX2GX6wFy4+lmFSAqBsIhg
ne2LFjMW7cOABAaWkpMjMz8d1332HOnDn417/+BV9fXwDAxo0bt+ep+cKjfl0Ep05SQ4+b
lRkZGLzZ4oufEOUj11NTU8OOPP+KDDz6oUnfx4kkVIJBIVRNU6MUkkagI2bdqE1157
5u7uDhcXF7zxv47Zc/DyZdHg/DOgG+Npq9f6fJCNaT5U2v0Ptxr4PJMQtKbq9Z4w00tUP4/nAYQFdtmU1iEzI+HzTbZhn8/3EwYSZO
```

Cfv27cPp06dx8OBBODo6ori4GOPHj8fDhw/x3nvvwcTEBIcOHcLp06exfft29O/fv9q479+/j
6ysrBr/nrt16wY9Pb1q69577z2cPHkSr776KkaMGAFXV1d07Nix2rYhISHYvn07+vXrhzfeeA
Nubm546aWXavy+RM0B5yHVz0M2NjZYsGAB/P39q60fOHAgXnrpJezZs6dK3eDBg9GvXz+sWrW
qxu9N1JRxDmo6c1B2djbOnj2LU6dOCXXnzp1DQEAADh48iJEjR2LlypUYM2YMkpOTMWTIEOFr
ajhcbkoK/8NWLhW6cuUKevbsiS+//FKoe/nllxEVFYWLFy/inXfewbVr11BUVIS5c+cKCY2Zm
RlOnz6NgoIChUng2rVrmDt3LoYOHVrrpFiprKwM4eHhwhj5+flYsmQJ4uLi0KtXL9y+fRujR4
/GsmXLhD5OTk5wc3PDxYsX4ezsLJS/9dZb8PPzAwDo6upiwoQJcHR0REBAAACgV69eOH36NGJ
jY+Ho6Ihjx44hPj4e3377LXr37g0AeOWVVzBlyhSEhobi0KFD1cZsaWlZ72cBP/vsM8hkMvz8
88/CpGhpaYnBgwfDz88PpqamQtuFCxciNzcXhw4dwoULFwAAHTt2xMCBAzFt2jRYW1vXKwYiV
eI8pPp56Fk6dOhQ40oGouaOc1DTmYN8fHzw9ddf48aNG8JKqhMnTmDIkCHQ1tZ+rrGp7pgkUr
U8PDzg4eGB0tJS3LlzB3fv3kVCQgKysrKEDVR69+4NLS0tjB07Fj4+PvDy8oKLiwscHR0Vxkp
NTRU+BQsODn7mpAhU/aSpcuItLCwEAOH5vIKCAty/fx937tzB9evXAVRstvA0Jycn4XWHDh2E
2CsZGhoCAHJzcwFUbAxjbGwMe3t7lJWVCe0GDRqE1atXIycnB23btq0Ss0wmg0wmq/Ga1NXVI
RKJqq3T19fH+vXrkZycjDNnziA6OhrR0dHYuXMnvv32W4SHhwuTvVgsxqeffgp/f3+cOXMG58
+fR3R0NA4cOIDDhw9jzZo1GDZsWI1xEDUXnIde7DxUF8/Tl6i54Rykmjmob9++6NixI06ePIl
evXqhpKQEp06dQkhISK39qGExSaRqyWQyrF27Fl9//TUKCgpgZmYGR0dHaGlpCW0sLCywd+9e
fPXVV/j222/Z+xc+dOGBgYYNKkSVi4cKEwASYnJ8PDwwPR0dHYsGEDgoKCnvn9dXR0FL6uHKty4
snKykJwcDBOnToFkUiEzp07o2/fvgCqnnFU3bKG/x3/aY8fP0ZmZmaVB6YrZWZmVjsx/t///R
+OHDlS47i7d++Gm5tbjfVAxd/p5MmTMXnyZMhkMpw6dQpBQUFYsWJFle2djY2NMXbsWIwdOxZ
AxXMHixYtwieffAJvb+86/QNE1JRxHlLNPFSTjIwMdO/evV59iZojzkGqmYNEIhGGDx+OH3/8
EYGBgTh37hzUlNQwYMAAZGRk1NqXGg6TRKrWV199hZ07d+Ljjz/GsGHDvb29+/Xrx2zJjjz/
+fR3R0NA4cOIDDhw9jzZo1GDZsWI1xEDUXnIde7DxUF8/Tl6i54Rykmjmob9++6NixI06ePIl
evXqhpKQEp06dQkhISK39qGExSaRqyWQyrF27Fl9//TUKCgpgZmYGR0dHaGlpCW0sLCywd+9e
fPXVV/j222/Z+xc+dOGBgYYNKkSVi4cKEwASYnJ8PDwwPR0dHYsGEDgoKCnvn9dXR0FL6uHKty4
snKykJwcDBOnToFkUiEzp07o2/fvgCqnnFU3bKG/x3/aY8fP0ZmZmaVB6YrZWZmVjsx/t///R
+OHDlS47i7d++Gm5tbjfVAxd/p5MmTMXnyZMhkMpw6dQpBQUFYsWJFle2djY2NMXbsWIwdOxZ
AxXMHixYtwieffAJvb+86/QNE1JRxHlLNPFSTjIwMdO/evV59iZojzkGqmYNEIhGGDx+OH3/8
EYGBgTh37hzUlNQwYMAAZGRk1NqXGg6TRKrWV199hZ07d+Ljjz/GsGHDvb29+/Xrx2zJjjz/
Ar90qVLOHDgALZs2TiUQCS0tLoVwikcDW1rZK+0OHDkEsFsPJyQkAUF5eDgB47bXXbXX8
Prrr+PTTz+FpqYmNDU1FfvM16NEDBQUFyMnJgaGhodjxymRySKUFSvcjosbBD22IqCVjkkhIe5

SPb04lAgAmvdodZu35Dx+1Hra2ttDT00N0dLSQJEqlUty8eRO+vr5V2v/8888KX1+9ehWBgYH
46quvYG1tDZlMhldffRXjxo3DvHnzhHbXr19Hhw4d6pUgViork9W7b1PHeYiIVIlzEJEiJomE
facT8dedLADA/tO38f743iqOiOjFEYvF8PX1RWhoKIyMjGBubo6QkBCYmprC29sb5eXlyMrKg
r6+PrS1tdG5c2eF/pUb33Tq1Ant27cHAAwbNgzbt2+HlZUV7O3tERUVhe3bt2PZsmUv/PqaC8
5DRKRKnIOIFDFJJKQ9fLKELfVhvgojIVKNgIAAlJWVYfny5SgqKoKrqyvCw8MhFouRnJyMIUO
GYOXKlRgzZkydxvvwww9hYGCANWvWID09HRYWFli2bBnGjx/fyFfSfHEeIiJV4hxEpIhJIhG1
eurq6ggMDERgYGCVOgsLCyQkJNTY183NrUq9hoYG5s2bp7DclIiIiKi54H7qREREREJGCSS
ERERERERAImiURERERERCRgkkhEREREREQCJolEREREREQkYJJIREREREREAiaJRERERERJG
CSSERERERERAImiURERERERCRgkkhEREREREQCJolEREREREQkYJJIREREREREAiaJRERERER
EJGCSSERERERERAImiURERERERCRgkkhEREREREQCJolEREREREQkYJJIREREREREAiaJRERE
RComk8mwfv16eHp6onfv3pgxYwbu3btXY/sbN25g2rRpcHJygru7Oz766CNIpVKFNidPnoSPj
w8cHBwwatQonD17trEvg4haCCaJRERERCoWFhaG/fv3Y8WKFThw4ABEIhFmz56NkpKSKm0lEg
n8/PxgaWmJI0eOICwsDLGxsViyZInQ5vz58wgMDMSkSZNw9OhReHh4YP78+UhKSnqRl0VEzRS
TRCIiIiIVKikpQUREBPz9/eHl5QVbW1usW7cOGRkZiIyMrNI+JSUFnp6eCA4OhpWVFZydnTFu
3DhERUUJbbZt2LwZvb2/4+vrC2toaS5Ysgb29PXbt2vZvb2/4+vrC2toaS5Ysgb29PXbt2vUiL42ImikmiUREREQqFB8fj/z8fLi7u
wtlBgYGsLOzw8WLF6u0d3JywaGhoQEAuH37No4cOYIBAwYAqFi6GhsbqzAeALi5uSEmJq
YRr4SIWgoNVQdARERE1Jqlp6cDAMzMzBTKTUxMkJaWVmvfYcOG4e7duzA3N0dYWBgAQCqVoqC
gAKampkqP9ywUqIqIGkfYoH9+cSgQATHq1O8zat1FxRM1DYWEh
AEAsFiuUa2lpIScnp9a+oaGhKCoqQmhoKKZOnYpjx46hqKioxvGKi4vrHaeamgiGhi3zZ6qmJ
lJ43VKvk6iumCQSERFRg9h3OhF/3ckCAOw/fRvvj++t4oiaB21tbQAVzyZWvgaA4uJi6Ojo1N
rXwcEBALBhwwZ4eXkhMjISX15ewnhPq8t4tZHJ5JBKC+rd/3lqaqoGBkQVEaTsVPXfz4Z8DbcDeQOGt3PMQ0VXiI1gLzdzGLZ6wRqp
PEmUyGTZu3IiDBw9CKpWib9++oiaB21tbQAVzyZWvgaA4uJi6Ojo1N
rXwcEBALBhwwZ4eXkhMjISXl5ewnhPq8t4tZHJ5JBKC+rdvymTyeQKr7Oz81UYDVHdNOaHGSp
PEmUyGTZu3IiDBw9CKpWib9++CA4ORufOnattf+PGDYSEhODatWvQ0tLC0KFDsWjRIhgYGAht
Tp48iQ0bNuDBgwesrJCYGAgXnnllRd1SURERK1S2sMnCUTqQ77JrqvKZaYSiQSWlpZCuQig
a2tbZX2SUlJSE5OFpJBoGIpadu2bZGRkYF27dpBV1cXEolEoZ9EIqmyBFVZZWy5+rfVMnliq
9b6nUS1ZXKF1xzy2ciIiJqzWxtbaGnp4fo6GihTCqV4ubNm3BxcanS/ty5c5c1i4cCHy8vKEsvv
37yM7OxvW1tYQiURwdnbGhQsXFPpFR0ejb9++jXchRNRiqDRJ5JbPRERE1NqJxWL4+voiJDQUp0+fRnx8PFhYO0mP4IwcAA6uQkWg4ID
p0+fRnx8PN5//32YmprC29sb5eXlyMzMFJ41fOONN6Cvr4/AwEAkJiYi8ePEAkJiYiJiYGAQEBcHR0xKBBg
wAAfn5++OGHH7Bjxw4kJSVh9erViIuLw7Rp01R5qUTUTKg0SeSWz0RERERAQEAAxo4diyVLlmDix
PixIlQV1dHeHg4xGI0tLS40OHhgRMnTgAADA0NsXv3v3bshkMkycOBHz58+HnZ0dwsPDoa6uDgD
w8PDAF198gX379uHNN9/E+fPnsWXLFlhbW6vyUomomWRzkb+...

763iiJonpZNEqVSKqVOnIj4+Hp06dYKxsTHu3LmD48eP4+uvv8Y333wDfX39xoiViIiImpC/U
6W4m55bp7Z30nLxd5oU1p3aNnJURNSapT0sEF6nPsxXYSTNm9K7m65Zswbp6enYu3cvfvnlFx
w4cAC//PIL9u7di0ePHuHLL79sjDiJiIioiYm9lalU+8u3HjZSJERE1JCUThJPnz6N9957Dy4
uLgrlLi4uCAgIwM8//9xgwREREVHTk51bjP/+cQe/xCYr1a+gqLSRIiIiooak9HLT/Px8vPTS
S9XWvfTSS3j8+PHzkRERERNjEwux807WfjtSiquJD6ETC5Xegxdbc1GiIyIiBqa0kli165d8
euvv2LAgAFV6OnTs3SGBEREStekejn5Jfj9WirOXk1F5uOi5xrLqUeHBoqKiIgak9JJ4s
yZM/HBBx+gpKQEo0aNQocOHfDw4UN8//33d27d7FlyxYcPGQQMU/JmKxGPPn
z8eECRMaPEgiIiJqfHmFpfjjehrOXElElZBje3aaGvg5V5m8OrTCXK5vNFZzEgFAV0sDM3x68
vgLIqJmol7nJL777rvw9fYXFlStXkJOTg27du2DTPM9nsmWPPTYX9t6d+I/dPrxsJc6OG1Z8
hRgtTD0hqU2Lk+6QKVHUmEqLRS7jE0PXOmxKGdfl6mmF1o4mB/M4Fo+7Ps9tyI+Kcm9Wb2AiW+qsnX
DEyj4rHRIItDQgG34RYREYb1pJ7y+qWGCYGh4aL33ZWqPKtj8+4+kvppaS+VfbpQpcSuLsCwBbzR
fFTc4mvYlsIx+EvF12TqhZBNZgjcE7lT58DXHiJdg4tX2eN5ntrfqhMzz1/vvF+ZiMrO8WSyDx
YfPrwFoQPmtZxj0KZhY2o6mqX+dv3d4TCvdhl5qTuVPujx/UBBk5D11w29c+J3ho+9QOE25OMGXb
8/kDBA36oSGbGBdXRrT5b4snrR72u7FjtAC8GxFGB1YIxvyvtWWx9MhvW6HFhDXX9N1ulGQPCnBxn
Vn0cukJoCQJA1Qw1EYJa2YiYjYaAcQaHl+XC1OJ/+zgfdM3wvO/X3Sszg4Oc3sMqdc+aWZ+XKQk
TEO2hyLiuU3O+ckBB8EmEYyYlbMnKJZ69bKF9YL9BRoM0tEnIuvRqzHsLhi7oTBehsB5q1pjlKi
QMnIkKBs+gJ6jhYpZ5i1mkZvpX08nNFm5YXyfZoEl4Owd3zMN0vqRVIFFYYkIiIhIr1pInHWmxMnh
7G3d8kajTsWJZ+jZoMzqYe6fAxMZ2eU0TLmZIYVqJJpUXJ5sArFvvdqpd3lXpVLP2M1B9ZfaWsm
5Uey7ewMb7SUUL4y8RB6DIDbvkh2ZiXbBZqUrCBrr37kIoHCgKbIvtvuB+DUGtfPqsgJIIwHmzRX
KRgmNWR9g3hrvuC0Bq4SXxB2HWfAwr1+YoyI2Y7NHiopGZwT++BYCUHzX7FzdBiIpGqO2qmNG08b
sSAFqbqc3dAsWNZD8ZbmVWpgFM1wPYDIbmwKWlEBkaeHhE8i3uwrhnL7ZP3Cn2fJAF6JzF2g6TBlQGB3
iV2lAnMLH1FTcC8iRlEjaYS0D4Kh5H5FhWlYCfLiWtfDArLbHHA/mMTJ0RmpGiMsiG3a+GhCHPEsG
CjLwpoGRa0Ysn5Kqw0bd2YTXU5T1eyt8H07xPsoUpFUIhUaBPcvPaUgpmYvr3Nc4ULvtPfkZ7gvT
UPtrzPJ7xHvgmBVCHC4l0OUGgNZ9kyhijZ7xXtdhaZfvoaKZB1Y6mJ3D7Zz8lEN3rGqz2AnZ2oKmaQ
J6tZjfqbvPhr9gIgN1DL+8/Cx7KhkN83yDXx38LQXzbDSSwaRH8iDRK9vmDfCbrz1XmBFAqiUHCC
Pgx6LGcZe+EUNOu6rdyG4htxjIgwbcfXbemudhTQY3pnUM1MTPP8Bw7fHwRELwJc6oMw6fEUzl0D
1Am8fYTvuNiU6Yn2j9i+cvR9vlPTGX7c6cMtmLBsQ+dcLhZxGRQuO+WcxWQ2vmWBkeRBOIBIfZh4
WkqW4ZTSLnBKPVMx4nYCm+VFuMWEp2hdAyYCCdwc3mDQAIdG2A2IGtk3i2bRbDKGjIrBCs/fVgu9
w7vOlXGYzIR8Z3zODgCUyZpHr+mNFGMtlz5nhbrTOdyb1Bhzaw3eWc5h2djFqVRMmOj2fPrPXRQ
nIAXd+AHfWVgLeggo0JlOz0bPDDzcOeN8Qt4ZWhU5WRoKC9g2sdg2sPrDbPlK8U/bpAR6kJHxOUl
q1kMbjRSqi2dMxHNZHBbEp0nxCVoEeYpFNpZ0u2mldNxq7e2eNzmqrT5wYZ2J7nDs92TNFAWOCcT
l0PvcTbGi5IH6kBDW0Tf7J83eEhrVpl5jEyQjhmB6gOkwMBkAPvFW8ZOQKXRmfmHWRQSjTRsH5V8
e71nS2YoS3bdtxL8e9iE1a6j5rkcaR3CQVk9G/I7iAIqFjdfE8LRsSd/lRrlhRjb0pMC2L9GLFcy

uWYPPmzQCAvLw8pKamYuvWrbC3txf6Pn1INrUscrkcSalSnLmcggvxEpTWci5X104G8OrTCf1
6doSWJu8aEhE1FLlcjrLyJ/NvQVEpklJy0LWTAUQiniOrDCaJRNSqlZSUICIiAoGBgfDy8gJQ
sXze09MTkZGRVZbWp6SkwNPTE8HBwdDQ0ICVlRXGjRunsOT+1q1bkMvlcHZ2Vri7SC1PYXEZz
v+Vjl8vpyI5M6/GdlpidfS3r7hraNlR/wVGSETUOqRk5iH8hzjk5JcIZYUl5fh8zyVYmepj5s
ieXLWhBCaJRNSqxcfHIz8/H+7u7kKZgYEB7OzscPHixSpJopOTE5ycnISvb9++jSNHjgjnxQJ
AQkICjI2NmSC2YHfTpfjtciqib2aguLS8xnaWHfUw0Mkcbj07QkeL/+QSETWGlMw8rNwbi4Li
smrr76bnYuXeWAT5OjNRrKN6/4slk8lw69YtSCQSODs7o6ysDO3atWvA0IiIGl96ejoAwMzMT
KHcxMQEaWlptfYdNmwY7t69C3Nzc4SFhQnlt27dgq6uLvz9/XH58mUYGRlhzJgxmDp1KtSe4w
B0DY2WewzC06uARKKmea3FJeU4/1c6folNwZ00aY3txBpqcLc3xSBn81a3xKk5/ByJqGWRy+U
I/yGuxgSxUkFxGSJOxGH5VJdWNS/XV72SxGPHjmHNmjWQSCRQU1PDwYMHsWHDBmhqamLNmjVV
nuEhImqqCgsrjiP433lLS0sLOTk5tfYNDQ1FUVERQkNDMXXqVBw7dgxt2rRBYmIicnNz4ePjg
wULFiAmJgahoaHIycnBwoUL6xWnmpoIhoZt6tW3OVBTEym8bkrXejdNih+j7uLXSw9QUFTzmx
BLU32M6G+FgX1fgp6O5guMsOloyj9HImoZZDI5HkqLkJFVgIysAtx68Bh303Pr1PdOWi7+TpP
CulPbRo6y+VM6STxx4gSWLFmC119/HYMGDcL7778PABg6dCg++eQThIWF4b333mvoOImIGkXl
ZjIlJSUKG8sUFxdDR0en1r4ODg4AgA0bNsDLywuRkZEYPXo0duzYgeLiYujpVSxpsbGxQX5+P
jZv3gx/f/963U2UyeSQSmveKbO5kz11qLxMJkd2dr4KowFKSstxIS4Dv8amIDG55g8LNNXV4N
rTBIP7WqC7RVuIRCKUFpUgu6ikxj4tWVP7OTYmJsBEjUcml+NxbjHSswqQkV2IjKwCSLILkZ5
VgMzHhSiX1Xyk0LNcvvWQSWIdKJ0kbtmyBW+//TY+/vhjlJc/eQ5jzJgxePToEb799lulukkRl
t55PTExESEgIrl69CjU1Nbi6umLp0qXo1KmT0IZbzxNRXVUuM5VIJLC0tBTKJRIJbG1tq7RPS
kpCcnKysMkNULE0tW3btsjIyAAAaGpqQlNT8U5Sjx49UFBQgJycHBgaGtYr1rJadsxs7uRyxd
equta0R/n47XIq/ryRhvxa7hp2NNLFwD6dMMDBTLhrWF4uB1D/Ny4tgWl7XTzMKQIAmLVv06J
/Z4no+cjlckjzS54kgtkFkGQVIj27IiGsbZfo51FQVNoo47Y0SieJd+7cwZIlS6qt6927NzZs
2KDUeMpsPZ+dnQ0/Pz+4urpi7969KC4uxr///UlSKJVKcfPmTfj6+lZpf+7cOfznP//B77//
UKSKJVKcfPmTfj6+lZpf+7cOfznP//B77//LtwpvH//PrKzs2FtbQ2ZTIZXX30V48aNw7x584
R+169fR4cOHeqdIFLjKS2TIfZWJn67nIKEB49rbKeuJkJfG2N49TGHrWU7PtNSjYlDuuMbeSI
A4O0h3VQcDRE1BXmFpRWJ4D/JoCS7QEgMi0tq3virsehqt87HAZSldJLYvn17JCUlKezkVykp
KQnt27ev81jKbj1/6tQpFBYWYtWqVdDS0gIAhISEwMvLC7GxsejfvZ+3nicipYjFYvj6+iI0N
BRGRkYwNzdHSEgITE1N4e3tjfLycmRlZUFfXx/a2tp44403EB4ejsDAQHzwwQDXxQhqvGtIRK1c
cUl5RRL4z9LQyruBkqwCSAsabgmnpoYaTAx10NFQFx2N/vnvP4mhQRtxlVUccrkcZ66k1mnzm
i5m+uhqxhyhLpROEt977z3cunUL7733nrD2duro6AgMDERgYiJ3bb26dIFW7durXE8DQ0
NzJs3T2G5KanWw5xCnL2ahnNXUxUOWv5fhvpa8OrdCZ69O8FQv+oHkkRELVlpmQySxxWJX+Vz
gpXLRLNzixvs+6iriWDcTgcdDXXQ0Ui34s8/iWA7fS2lPpgTiUSYObJnreckAoCulgZm+PTko
wJ1pHSSKBaLsX37dvzxxx+IipCTk4O9PX10dgwMA+nWDXxQhqvGtIRK1c
baGFvPAy17K29uWU5ELZFMJse1pEf47UoKric9qnFLGREAB+v2GNjHHA7WRlB/zg8UiYiasrJ
yGR7lFP2TAP6zUcw/ieAjaZHCRmLPQySqWK5fkQDqKiSE7Q20GnSuNTfWQ5CvM8J/iKv2jmIX
M33M8OkJc2O9BvueLV29zkkEgAEDBlT7XKIy6rv1vFwux5dffonNmzdj5ulDXGFvPA
y17+/nWtGU5tQz8IINqk51bjHPXUnH2aiqypDV/8t22jRievTvhld5m6NC29uNOiIiaE5lMji
xp0VN3A5/cFXyYU/RcR0j8LyMDrX+Whv6TCP6zTNS4nQ401F/ch27mxnr41zQXfLDxD2HFiI5
YHR+83QddzQx4B1FJSieJQUFBNdapqalBV1cXVlZW8PHxeeYufspuPQ8ApaWlCAoKwvHjx7F4
8WLMnDlTob6xtp4HWu72801l63kiovqSyeW4eScLv11JxZXEh5DV8lG4vZUhBjqZo3e3Di/0D
QwRUUOSy+V4nFfyz3LQJ+cJVuwgWoiy8oZ7P9e2jbjK0tCORrowaacDsaZ6g32f5yUSiRTmdV
1tTZ6JWE9KJ4np6emIjY1FcXExzM3NYWxsjEePHiE5ORlqamro0KEDVlZW8PHxeeYufspuPQ8ApaWlCAoKwvHjx7F4
XahxL2a3nAWDx4sWIjIzEmjVrqmxsw63niYhal5z8Evz+z13DzMdFNbbT19WEh4MZXunTCR0N
dV9ghERE9SeXy5FbUFrlbmBlIlhc2nBHSOjpaCpsFFO5TNTEUAc6WvVefEjNlNI/8UGDBiExM
RG7du1Cnz59hPK4uDjMnz8fc+fDhwzF37lysXbsWG37lysXbsW69atq3EsZbeeP3z4ME6cOIHFixejX7
9+yMzMFMaqbMOt54mIWja5XI74+4/x2+UUxN7KrHXZlK1lO3j1MYdzD2NoclMuImqi8otKqyS
Blf8trGUzFmXpaKnDxFAXpgpLQyuWh7bh+YH0FKWTxJ07d+LDDz9SUSBABoGfPnli4xcG+/PJL

TJgwATNmzMDHH3/8zPGU2Xr++PHjAIDVq1dj9erVCuNUtuHW88qRy+UKyxEKikqRlJKDrp24d
puIXoy6zkN5haX443oafruSioysmp8Rb6OtgQEOZvDq0wlm7fn8KhHV7kW9FyoqKauaCP5zh7
C2s1qVJdZUg0k7XZga/bMk9J9dQzsa6kJfV5Pv76hOlE4Ss7OzYWRkVG1d27Zt8ejRIwCAkZE
RCgqevdGLMlvPR0REPHM8bj1fdymZeQj/IU5hO/jCknJ8vucSrEz1MXMkd4Eiosb1rHlohk9P
FJaU4bfLKbgYn1nrMzbdzNvCq08nuNqaNKlnZIio6Wro90IlpeWQPC6sNhnMyav5+B1laaiLh
MPkOxrqwsRIB6b/3BVsp1f1LEEiZSmdJNrZ2WH79u3o37+/wtEVJSUliIiIQM+ePQEAf/31V5
XzD6npSMnMq/U8mbvpuVi5NxZBvs5MFImoUdRlHgrecaHW7dh1tNTR394UA/uYw8KEcxUR1V1
93wuVlcuQ+bhQYUloRlYBJNkFyJIW13jcjrLURCJ0aKcN03/uBnZ8apmokYG2whFmRA1N6SRx
0aJF8PPzw+DBgzFw4EC0b98ejx49wpkzZ5CXl4ft27cjJiYGa9eu5d28JkoulyP8h7haDxwFg
ILiMkSciMPyqS78RIqIGlRd56GaEkQrU30MdDKHW8+O0BLzriERKUeZ90LrDl5Fn24d/jlkvh
APc4pq3UFZGSIARgbaMDXSgck/S0JN/9k8pn1bbe7ATCqjdJLo5OSEw4cPY8uWLTh37hyysrJ
gamoKT09PvPPOO7C0tERUVBQCAgKqHE9BTcPfqdJqDxqtzp20XPydJuX2wUTUoJSZhyppaarD
za4jBjp1gpWpQSNFRkStgTJzUJa0GL/EpjzX9zPU10JHQ52KJaL/LA01MdKFSTttaGrwgy5qe
uq1n23Xrl2rbBzztP79+6N///71Dooa1+XEh8q1v/WQSSIRNShl5yH7LoZ4d7QDt2Enogah7B
xUF/q6mgoHyj+9TJQrHqi5qde/trm5uTh//jwKCgogr+Z2++jRo583LmpEBUXK7aClbHsiomd
Rdl4xbstzuoio4dT3vY2uloZwZERHw6f+a6gLXW3OUdRyKP3bfObMGbz33nsoLCystl4kEjFJ
bOJ0lTwHh7sEElFDU3YeUrY9EVFtlJ1T+tub4u0h3aCnwyMkqHVQOklcu3YtunbtiqCgIHTs2
BFqanygtrlx6t4BJ87fq3P7qL/S0bOzIXp369CIURFRa6LsPOTUg/MPETUcZeegwX3Noa8rfn
ZDohZC6STx77//RlhYGFxcXBojHnoBunYygJWpfp0f2M4tKMWX312Du11HvP1qdxhwkiSi56T
MPNTFTB9dzbhRDRE1HM5BRLVT+jZgp06dkJeX1xix0AsiEokwc2RP6Cr5fM/5mxlYvi0a52+m
V/ssKhFRXdV1HtLV0sAMn55c3kVEDYpzEFHtlE4S586di02bNiE5Obkx4qEXxNxYD0G+zrAy1
a+2vouZPmb42KKjoY5CeV5hKb76702s/+4asqRFLyJUImqh6jIP/e8h1kREDYVzEFHNRHIlbw
n5+fnhxo0byMvLg5GREbS1tRUHFIlw6tSpBg2yKSgvlyErK1/VYTQ4uVyODzb+gZz8EgCAjlg
dH7zdB13NDCASiVBSWo5jf9zBT9EPqhwcq6OljnGDuuGV3p2gxk/Y6AUyNq7+H/SWrKXOQcCz
5yGipojzUMvBOahlCQz7E4/+uZHR3kAbIe++rOKIGk9jzkNKP5NoamoKU1PTxoiFVEAkEkFD/
ckNZV1tTYUzEcWa6hg3sBtcbU2w40Q8HkieLDUuLC7H7h8TcOFmBqaNsEVHQ90XGjsRtQzPmo
eIiBoT5yCiqpROEleuXNkYcVATZ2VqgH9Nc8GP0ffx3z/uoKz8yV3F+PuP8VH4BYz27IKhri9
BnTveEhERERE1W/V+N//w4UOkpaUhNTUVqampSE5ORmJiIvbt29eQ8VEToqGuhtdetsInM/qh
m7niJ2ylZTIc/DUJK3Zfwv2Muu2aSkRERBVkMhnWr18PT09P9O7dGzNmzMC9ezUf0ZCYmIg5c
+bAzc0N/fv3R0BAAFJTUxXanDx5Ej4+PnBwcMCoUaNw9uzZxr4MImohlE4S4+Pj4ePjA09PTw
wePBhDhgzBkCFD4O3tjddffx2fffZZY8RJTYhZ+zZY6uuMyd49oKWprlB3Lz0Xn+2KweGzf6O
0TKaiCImIiJqXsLAw7N+/HytWrMCBAwcgEokwe/ZslJSUVGmbnZ0NPz8/tGnTBnv37sejbRsx3h3dC5cSMrH35w
ThvCEAeCQtwtoDVzHAwRQTBneHno6mCiMlIiJqOgoLCwEAYrFYoVxLS6tO76t2796Nb775BkF
BQWjfvr2QdFY3XuUzi/WhpiaCoWGbevdvytTURAqvW+p1tgb8WTYMpZPEoUOHIjQ0FDo6Ohg+
fDi6du2KdevWYc6cOYiIiiMBLL73UGHFSM9LXxhi2ndvh219u49w1xU8s/7iejut/Z8HXuwdcb
E1UFCEREVHToa2tDaBitVblawAoLi6Gjo5Ojf3kcjm+/PJLbN68GXPJLbN68GXPnzS0sH/w4bj
xrvGeRyeSQSgvq3b8pk8nkCq+zs/NVGA09D1MjXUiyC4XXLfln2ZgJsNJJ4oIFC3Dv3j0cOnQ
Iw4cPR1BQEBYsWIATJ05AXV0da9eubYw4qZlpo60JP5+ecLPriJ0n4/Ewp0iok+aXIOzoDfTt
YYzJQ3ugnV7VZyqIiIhai8plphKJBJaWlkK5RCKBra1ttX1KS0sRFBSE48ePY/HixZg5c6Zz
65dO+jq6kIikSj0kUgkVZagKqushe5cLpcrvm6p19kaTBjcDeX/JP0TBnfjz7KelE4StbS0sH
79epSWlgIAPD098f333+Ovv/6Cvb29uRGZGdlhM9muuHIub8RGfNAYRK+dCsTcfey8faQ7hj
gYArR04vIiYiIWglbW1vo6ekhOjpaeB8llUpx8+ZN+Pr6+Pr6Zy5EiFOpFI
BGdnZ1y4cAHjxo0TyqOjo9G3b9/GuxCiJsCsfRt8OKGPqsNo9pROEitpaj55pszS0pLJIdVIS
6yOt4d0h2tPE+w4EY/Uh09u+xcUlyHiRByib6Zj6nBbGLer/zIYIiKi5kgsFsPX1xehoaEwMj

KCubk5QkJCYGpqCm9vb5SXlyMrKwv6+vrQ1tbG4cOHceLECSxevBj9+vVDZmamMFZlGz8/P8y
ZMwd2dnZ45ZVXcOjQIcTFxeHzzz9X4ZUSUXOhdJJYVFSE9evX4/z588jNzYVMpngLVyQS4dSp
Uw0WILUc1p3aIni6K36Iuosfou4JSwEA4K+72fhXeDTeesUaQ/paKDx0TERE1NIFBASgrKwMy
5cvR1FREVxdXREeHg6xWIzk5GQMGTIEK1euxJgxY3D8+HEAFWdXr169WmGcyjYeHh744osvEB
YWhnXr1qFbt27YsmULrK2tVXF5RNTMKJ0kfvHFF/j222/h7OyM7t27Q02N28pS3WlqqGG0Z1e
42Jhgx8k43EnLFepKSmXYdzoRF+IyMN2nJ8w7cDcqIiJqHdTV1REYGIjAwMAqdRYWFgpHjkVE
RNRpzNGjR2P06NENFSIRtSJKJ4k//fQT/P39MX/+/MaIh1oJCxM9LvigsiYBzhy9m+UPPVQc
VKqFJ/suIDXXraCj3tnaKjzgwgiIiIiohdF6XffpaWlcHFxaYxJqVRUxNhWD9LfDqzH3p2Nl
SoKyuX4+i5O/h0ZwzupElVFCG1FjKZDOvXr4enpyd69+6NGTNTwm4N69ezW2v3HjBqZNmwYnJye
4u7vjo48+glSq+Ht68uRJ+Pj4wMHBAaNGjcLZs2cb+zKIiIiGoTSSaKnpyd++23Rgi
8oZY0DFFvJ+fn6wtLTEkSNHEBYWhtjYWCxZskRoc/78eDQGBmLSpEk4evPDw8MH/+fCQlJb
3IyyIiIiKqlzotNz169Kjw2t7eHuvXr4dEIkHfvn2hq6tbpT3Xv5OyRCIRXundCQ5d22PPTwm
4cvuhUCeXAz9euI/YW5mYPsIWtv9z15HoeZSUlCAiIgKBgYHw8vICAKxbtV8
SkoKPD09ERwcDA0NDVhzZWWHcuHFYt26d0Gbbbtm3w9vYWtq5fsmQJLl++jF27duHTTz99cRdHR
EREVA91ShKXLl1apeyHH37ADz/8UKVcJBIx8bjYGL8HXkbeQW1Aq1EkeF2L1vs
vw6tMJ4wZ2g652vU9wIRLEx8cjPz8f7u7uQpmBgQHs7Oxw8eLFKkmik5MTnJychJ0e
OYMCAAQAqlq7GxsZWmTfd3NwQGRnZiFdCRERE1DDq9C779OnTjR0HkUAkEqfz46wszLCvlOJ
iPorXaH+zJVUXL39EFOH2aJP9w4qipJaivT0it8vMzMzhXITExOkpaXV2nfYsGG4e/cuzM3NE
RYWBqDiAOyCggKYmpoqPd6zaGi03E2cRCLF1y35WomIiJq6OiWJ5ubmCl8lWuemWcwcCBAw
EADx48wK+//orRo0fDwMCgwYOk1klPRxPTR9ihv60Zdh6/haz8EqEuI6cQq9b/qQoHr3xg6hn49TTDJuwc
MdMUqjJSas8LCQgAVh1k/TUtLCzk5ObX2DQ0NRVFREZF16lQcO3YMRU4tHkUAkEqFfz46wszLCvlOJ
oqYmgqFhyz0W5umzUVv6tRIRETV1Sq/Xu337NqZPnw6xWCw0bdhYN
HSc1Io5dG2PT2e64dCZJPwSm6JQdyFOgpt3szHx1e5wt+sI0dO3I4jqQFtbG0DFs4mVrwGgpLL
gYOjo6tfZ1cHAAAGzYsAFeXl6IjIwUnmv8301v6jJebWQyOaTSgnr3b+pkMrfBV/ER
s/CCDiFoypZPE1atXw9zcHBs3bhhTK3N3dcebMGbz77rsICQnBl19+2aBBEuloacB369
HSfjkZH15M1yXmEptn1/E9E3MzB1mA2MDLRrGYlIUeUyU4lEAktLS6FcIpHA1ta2SvukpCTkpCQkJ
ycLySBQsZS0bdu2kKdOdu2yMjIQLt27aCrqwuJRKLQTyKRFmCqqyyp84TbWnkcsXXLflaiYiImjqlH/
q4cuUK5s+fD2NjY4VyIyMjjzJO7F9HR0Q0WHNH/6vFSO3w6wxUj+3eG2v/cNbyW9AjLt0fj19h
kyJ5+x0lUC1tbW+jp6SnMXVKpFDdv3qz2TNhz585h4cKFyMvE8ru37+P7OxsWFtbQyQSwdnZ
GRcuXFDoFx0djb86pcBlZSUoLS0tNo6ooaiqaGOt7ys8a9pL
rDsqKdQV1RSjj0/38Lqr2ORntVyl+ZRwxGLxfD19UVoaChOnz6N6N+Ph4vP/++zA1NYW3tzfKy8
uRmZkpPGv4xhtvQGF9fH4GBgUhMTERUfHVoaUChOnz6N+Ph4vP/++zA1NYW3tzfKy8
uRmZkpPGv4xhtvQF9fH4GBgUhMTERMTAwCAgLg6OiIQYMGAQD8/Pzwww8/YMeOHUhKSSLq1as
RFxeHadOmqfJSiYiIiOpE6STRzc0NYWFhyMrKUijPkiOzS0NYWFhyMrKUijPysrCli1b4O
1tBQV/xVvpWcg4/CL+DE+Xsol3HZGtUuICAAY8eOxfLlyzFx4kSoq6sjPDwcmpqaeHDhQ
IkTJwAAhoaG2L17N2QyG2Sz+eHhUFdXBwB4eHjgiy++wL59+/Dmm2d/i/Pnz2L
JlC6ytrVV5mURERER1IpLLlVuX9+DBA4wdOxYlJSXo06cPjIyMkJ2djcuXL0NLSwt79+5Dly5
dGitelSkvlyErq2VupBAY9iceSSvukrQ30EbIuy+rOCLLlpT3Kx86T8UhMrrobZeeO+vDzsYVl
R30VREaNwdi49f0sW/IcBLSMeYhaF85DLQvnIGqOGnpOgvnMeUvpO4ksvvYTjx4/j7v+j/7bffFRkFBAW7cu
AGpVIoJEybg6NGjLTJBpPrnDNm6Ui093xuDQmSSUlpWrKEIiIiou
ZB6d1NAcDY2BhBQUEMHRM1ELPtV2BhLlixp6FiFInaoUbQPRJQUJJHGMYcHDAhe6XFRovwsoRwt4l6XiqBd0PmVW7EnKnkMcK6PuvRouvjBQFymEJT39h4dYt9Mws6vZmAoXahGd1pn1sV/fG5Vf3Zo
VBUpERERE1IQpfSeRqKlr31Yb741xxLjB3Y74xOxzX7KCno6lQl55VgFV7Y/H1z7dQWFymogiJiIiIiJou
JonUIolEIvTvZYoVs9zQr5qxHwUHo0bT91tJCIiIiIiJonUwhm0EeOdN3rB/y0Ht
NMTK9Q9khZj7bdXsf34TeQV8ugWIiIiIiKASSK1Ek7djbFiljte6lS1R7v6d2pSt2fN9KxfNt5XIyXQM
nNfomIiIiWhylk8SgoCBERkaioAHlIVPzoqutgekjbBH/dh8YtdUWyosk5di48y/W/XgN2fkltYxAzq0
LVRQhEREREZHqKb276e3bt+bt3bt3Hs2DFoaGgjA1dUVgwcPxsRvzCBA2Fubt4Y8RE1uJ5WIRvh0phuY
P198gKdvHl5Of5Ij4+48xYA3eDqaSQSQIiIiIiIVPUppO4sGDB/HHB/HHH39gxYovVMDQ0xMaNG
/Hqq6i/i9ddfx7p7163DlypVGCJJoMPC4O5YPtmtfRdYJClJpqmPC4O5YPtUFFsZtFOoKi8uw82xo
IiIiJSjXo9k2hoaIjXXhFhexgOVXoaXyfwX38dbatmDhxYKqXFhrupgz4b7Q
qaop3DePuZeOj7dH4+cJ9yGRN/1nFtEf5WHPgCtYcuIK0R/mqFIAXx4888/
8Z///AeTJ0/GrFmzjZfTo0QO+vr5KjSWTybB+/+/Xp4enqid+/emDFjBu7du1du+8TERMyYZM
wdubm7o378/AgICKJqaqtDm5MmT8PGxYODA0aNGoWk5zZ/W5zKpldBQV8PrA7A/rgYz9XdO1koF
BXUibD/l9u44u9l5CcmaeiCOtm3+lE/HUnC3/dycL+07dVHQ4RERERNVNKJ4lvv/
WLJw6dQ2NjYICQnBn3/+/+if/+979YtmyZUoeFUh//79WLFiBQ4cACRSITZs2dj+U2ejpKSktvs

7Gz4+fmhTZs22Lt3L7Zt24bs7GzMmjULxcUVm42cP38egYGBmDRpEo4ePQoPDw/Mnz8fSUlJy
l4qtTLmxnr4P9++eHtId4g1Ff/X+DtVik92XMSx3++grFymoghrl/bwyWZSqQ95J5GIiIiI6k
fpJDEhIQFlZWWwtbXFqFGj8MYbb8Db2xuGhoZKf/OSkhJERETA398fXl5esLW1xbp165CRkYH
IyMgq7U+dOoXCwkKsWrUK3bt3R69evRASEoKkpCTExsYCALZt2wZvb2/4+vrC2toaS5Ysgb29
PXbt2qV0fNT6qKmJMNT1JXw20w12Voq/0+UyOY79fgef7LyIv1OlKoqQiIiIiKhxKZ0kXrx4E
V9//TUGDx6Mc+fOYcqUKXB1dcWsWbOwbds2XLt2rc5jxcfHIz8/H+7u7u7kKZgYEB7OzscPHixS
rt+/fvj02bNkFLS6tKXU5ODmQyGWJjYxXGAwA3NzfExMQocZXU2hm308GHE/rAz8cWulqKmwC
nZObj8z0x2H86EcUl5SqKkIiIiiiocSh9BIaGhgb69OmDPowwUZvb2/4+vrC2toaS5Ysgb29
BiKRCHFxcXUaKz09HQBgZmamUG5iYoK0tLQq7S0sLGBhYaFQtnXrVmhpacHV1RVSqRQFBQUwN
TWt03jK0tCo1yOcTd7TpzyIRC33OutjkLMFnLobY9eP8biUkCmUy+XAzxcf4EriQ8wY2RN2XY
xUGGUF/hyJiIiIqCEonSRWevjwIf78809EN9ERUUhKioK6enp6NSpE7y8vOo8RmFhxfECYrFYoVx
LSws5OTnP7L9792588sSKoKQvv27YWks7rxKp9rZrC81NREMDds8u2EzpPbUrp4t+Trry9Cw
DT6e8zL+uJaKLYev4XHuk98lyeNCrPo6FkPdOsNvlD30dRVFid/jkRERETUEJROEr/44gtER
UXh9u3bUFNTg5OTEyZPnoyBAweie/bf/fuSo2lra0NoOoLZxMrXAFBcXAwdHZ0a+8nlnlcnz55ZfYvH
kz5s6di+nTpwOAsAz1fze9edZ4dSGTySGVFtSjHc2dc9KQ6di+1xRdz3PFN5nC3
8fk3xzvTP0fdw4a80TBthi742JiqJrzX9HJkAExERETUepZPE77//HsxgYGDw
7E41qFxmKpFIYGlpKZRLJBLBLJBLY2tpW26e0tBRBQUE4fvw4Fi9ejJkzp17dq1g66uLiQSiUIfi
URSZQlqfZSVNc1dLZ+XXK74uqVeZ0PQ1lTHDJ+e6Gdrgl0/JuCRtEioe5xGXgi8PXoOrrQkmef
dA2zbiWkZqePw5EhEREVFDUDpJ/Pbbb/F4eFisWbMGAGBKXSJGNMaBvaFDtb6R3b51nb+4i4P4uHn
NmzWet7h48WJERkYiJORgriGHKjYHW8mNnl0dHR6Nu3b51iIqqLXl3b47NZ
/XD4zN84fSkZT+VnuBgvwc27WZj4anf0tzet9f8XIiIiIqKmRukksbkS0lhCGHgsnmRkskbS0FDt37sSFCxdQWloK+
T+3L+RyOQoKCnD79R79m1cvXq1TTmWL4+voiNDQURkZGGMDo3Ce8hICExNTeHt7Y3y8nS/ZVBX1
8f2traOHz4ME6cOIHFixejxX79+yMx/Pz/MmTMHdnxZ2eOWVV3Do0CHExcXh888V/V/Z
SiWqlLdbAJO8e6NezI3acjEPaoyfLkfOLyrD9e9BzZ2ZY7Ez38zA1GE26ND2+KEpvf7h6
9WqsXbsWEokESULSUlJSElJQWFhIa5du4a4uDjMnTtXqfECAgIwdxYLF++HBMnToS6ujrCw8MhF
ouRlpYGDw8PnDhxAgBw/PhxWFhYMvPDVw8PnDhxAgBw/PhxIQYPDw+FP5VtPDw88MUXX2Dfvn148jc78e8eWzZsgXW1tbKXi
pRnXSzaIuP/Vzx2sudoa6meNfwxt9Z+Ff4BZy+lAzZ0+tBiYiIiIiaKKXvJP7888ooOn06Pn06
diq1bt+LmzZv48ssvkZGRAV9fX9fX8hkyj0Hpa6ujsDAQABCgvB1RERERcYPXo0
Ro8erVQcRM9DU0MdY16xhouEJXacisitaM8jFyhrrikHF9H3sKFuAxMH2ELs/bcdB/
LgRBw8ehFQqRd++fREcHIzOnTs/s/b9Pbm7K9QNHjwYKSkpCmpCmWjRo1CaGhog8dPRC
2L0ncSs7KyhGMubGxscP3dQBAx44dMWfOHOHlrFNlRH8un9cW4QdbQ/J9zChOTcxAccRE
/RN1FWTk3lSEioifCwsKwf/9+rFixAgcOHHIBJMLs2bOOr7Nj+tKKiIgQGBuL333333
1FRs3boVv//+u/AnODi4PpyM5MS+DiFoIpZNEfX19YcKysrsaRp3MPdM0zOqRk3TSRqtdZ3
z1eaqdQV1Yuw6Ezf2PFrhjcS8+tfgAiImpVKpKBERAX9/f3h5ecHWJY4IzCAlhYWEB
0TGxuLN998E1vd0EGD5KoOepopIvFk5wwdZgNtMXqCnX3JXn4bFcMvvvstCSWl5SqKkIiImoL4
+Hjk5+fD3d3d3k5+fD3d1dKDMwMICdnR0uXrxYZbz9v587B29sbR48exYwZM9Dc3BxlZWUICdm
bz8fvr6+mDt3Lvbs2VP1UC9NRkQ84DXvcAGIl0rIxdL6L/VgjqZdm4oeZwRTJPUHqiLCfoXeYCoRBi9kEpYGYH2RnDpECoLcX6nqsPkqT4GbuYAOUvs8AeRsLnlaGMoPbxDr5lLqMtdX6V6Lw9E2klK5nAMsekW2kzYJgkzN4CjUdXO3PtPd7vtIfnzCN01MGCPP/jVK+eKtXHnDYuESZKuFH9Qt
z8fvr6+mDt3Lvbs2VP1UC9NRkQ84DXvcAGIl0rIxdL6L/VgjqZdm4oeZwRTJPUHqiLCfoXeYCoRBi9kEpYGYH2RnDpECoLcX6nqsPkqT4GbuYAOUvs8AeRsLnlaGMoPbxDr5lLqMtdX6V6Lw9E2klK5nAMsekW2kzYJgkzN4CjUdXO3PtPd7vtIfnzCN01MGCPP/jVK+eKtXHnDYuESZKuFH9Qt

WWLVuG8ePHN/KVEFFLoHSS+MsvvzRGHEStnlhTHeMGdYOLrQl2nIhHcmaeUFdYXI5dPyYg+mY
Gpo2wRUdDXRVGSkREDU1dXR2BgYEIDAysUmdhYYGEhIQa+1b33kxDQwPz5s3DvHnzGjROImod
eAgMURPTxcwAH013wZuvdIWGuuJdw/j7jxEcfgE/Rt9HuYwP1TcUmUyG9evXw9PTE71798aMG
TNw7969GtsnJiZizpw5cHNzQ//+/REQEIDU1FSFNoMHD4aNjY3Cn0WLFjX2pRARERE9NyaJRE
2QhroaRr1shY/9+sHa3EChrqRMhm9/vY0v9lxCsqTibqNcLkdZ+ZOksaCoFEkpOdz0po7CwsK
wf/9+rFixAgcOHIBIJMLs2bNRUlJSpW12djb8/PzQpk0b7N27F9u2bUN2djZmzZqF4uKKnWrz
8vKQmpqKrVu34vfffxf+BAcHv+hLIyIiIlIaT+0masI6dWiDoMl98UtsMg6d+RvFpeVC3Z20X
Hyy8yI8Hc1wJy0XOflPEprCknJ8vucSrEz1MXNkT5gb66ki/GahpKQEERERCAwMhJeXFwBg3b
p18PT0RGRkJEaOHKnQ/tSpUygsLMSqVaugpVWxmVBISAi8vLwQGxuL/v3749PdL5XB2doa
BgUGV70lERETUlPFOIlETp6YmwqsuL+Gzmf1gb6W4bXm5TI7frqTiXkZutX3vpudi5d5YpDz1
fCMpio+PR35+Ptzd3YUyAwMD2NnZ4eLFi1Xa9+/fH5s2bRISxKfl5OQAABISEmBsbMwEkYiIi
Jol3kkkaiY6tNPBBxP64M8b6dh/OhH5RWV16ldQXIaIE3FYPtWFO6NWIz09HcCTw6wrmZiYIC
0trUp7CwsLWFhYKJRt3boVWlpacHV1BQDcunULurq68Pf3x+XLl2FkZIQxY8Zg6tSpUFOr/2d
zGhot93O9p381RaKWfa1ERERNHZNEomZEJBJhgIMZenUxwtb//oX4+4/r1O9OWi7+TpPCulPb
xg2wGSosrDibUiwWK5RraWkJdwZrs3v3bnzzzTcICgoSzo1NTExEbm4ufHx8sGDBAsTExCA0N
BQ5OTlYuHBhveJUUxPB0LBNvfo2B2pqIoXXLflaiYiImjomiUTNUFs9LXTt1LbOSSIAXL71kE
liNSoPpi4pKVE4pLq4uBg6Ojo19pPL5fjyyy+xefNmzJ07F9OnTxfqduzYgeLiYujpVTwLamN
jg/z8fGzevBn+/v71upsok8khlRYo3a+5kMnkCq+zs/NVGA3Rs/GDDCJqyZgkEjVTBUWljdq+
tahcZiqRSGBpaSmUSyQS2NraVtuntLQUQUFBOH78OBYvXoyZM2cq1GtqakJTU1EePHigoK
EBOTg4MDRWfLa2rsrKWe+zJ0xvxyuUt+1qJiIiaOj70QdRM6WprPrvRc7RvLWxtbaGnp4fo6G
ihTCqV4ubNm3Bxcam2z+LFi/Hjjz9izZo1VRJEmUyGwYMHY/PmzQrl169fR7cOHeqdIBIRERG
9KLyTSNRMOXXvgBPnaz7wvUr7Hh0aMZrmSywWW9fXF6GhoTAyMoK5uTnTlCQkJgamoKb29vlJeX
IysrC/r6+tDW1sbhw4dx4sCysuK9vT2ioqKwfft2L
Fu2TIVXSkRERFQ3TBKJmqmunQxgZaqPu+nVH3/xtC5m+uhqxuMYahIQEICysjIsXT6cRUVFcH
V1RXh4OMRiMZKTkzFkyBCsXLkSY8aMwfHjxwEAq1evxurVqxXGqWWzWwFxvHjxEAq1evxurVqxXGqWzz4YcfwsDAAGvWrEF6ejo
sLCywbNkyjB8/XhWXR0RERKQUJolEzZRIJMLkT2xcm8sCoprPg5DV0sDM3x68viLWqirqyMw
MBCBgYFV6iwsLJCQkCB8HRER8czxNDQ0MG/ePMybN69B4yQiIiJ6EfhMIlEzZm6shyBfZ1iZ6
ldb38VMH0G+zjA31nvBkRERERFRc8UkkaiZMzfWw7+muaBtmyfn/OmI1bFsal8sn+rCBJGIiI
iIlMLlpkQtgEgkgob6k898DERERREdUL7yQSERERGRgEkiERERERERCZgkEhERE
RkYBJIhERERERQmYJBIREREZGASSIREREREJmCQSEREREGRgEkiERERERERCZgkEhER
ERERkYBJIhERERERQmYJBIREREZGASSIREREREJmCQSEREREGRgEkiERERERERCZgkE
hERERERkYBJIhERERERQmYJBIREREZGASSIREREREJmCQSEREREGRgEkiERERkYrJZD
KsX78enp6e6N27N2bMmIF79Vqd/MmTOxYcOGKnUnT56Ej48PHBwcMGrUKJw9e7YxQieiFoh
JIsGsg67wulOHNiqMhIqHUKCwvD/v37sWLFChwD/Pnzk1S1JiXQUQtBJNEwsQh3WHfxQj2XYzw9pBuqg6H6
onJPjVn/P2l1qykpAQRERHw9/eHl5cXbG1tsW7dOmRkZCAyMrLLLTcWXo66ZXz2kfLSwx1cE9Zq9TUo3zoEHmZTJtjwkqoBZ0bpWg6u2EfYkHmKSSomM7XWMmZgzQ5ao8gO4KihNOdyEsUDQtnAkZbSd4L7IVPNnKyiHhCKwOOEyHyhMKrV1BTgsj
nftm0bvL294evrC2trayxZsgT29vbYtWtXY19Os8Q5iEiRypPExlheMXjwYGxubsb8wYNjY2Cj8WbRoUWO
E3yKYtW+DDyf0wYcT+sCsPSfG5orJPjVn/P2l1iw+Ph75+flwd3cXygwMDGBnZ4eLFy9W2+fc
uXPw9vbG0aNHoa+vr1Ank8kQkQGxurMB4AuLm5ISYmpuEvoAXgHESkSEPVAVQur1i5ciU6duyIk
JAQzJ49G8ePH4dYLK62T1FREZYtW4bff/x8bff0UajLy8Lte4Gx4bu7MS4+9wyWeHFSTY4gq2t0vzaNnZ7k8txEitkpPYY06Bz5uwhzXZFVlzrYaOu7MTGQlrUxAlaZQ4zq2XjJA2g7t8MsEzaNQFUh0/FwKzMQSgK0gnlvE44ra+xQmzxyA0ldp/RmqANhTVxbUO3NDYzDTS4kDyoMxQ8WSHsLGsaFsVUPZQ2t5Dl76ap9kqzCKkM3RfK8Gmq8kJbkVJQk3ncAeaAsKo8WLeVGqlXlHKdClGgeY5ODPm4PJo8gWn0tPn7MhTAu3AGWEnOyFxGSyCUZgpo6fsZX0/zGAnmJSS6K9t8AQKi0ImfakDNHWhrz6f5aP8IpIiGV5u8Wn0dDrfEgLQs0XHskC7IHAf8KJEs9qeOKfGIxE6Upl4sIg3GHANEH9qhp8AGyRSzWLKM9xp2IE+3OGwQ0GuUjXbaW64z7M1nFt3Gx0IrU7+t0NHA6Gtl1Ys1WvY08jz1/AcTvPI2PYXjy77l2r8VFlrA1c7LPS2bqJ6cYKGUbeHxyjxIrUUo/pHkaR4UeswlsQ==

XqGmpoZhw4Zh+/btsLKygr29PaKiorB9+3YsW7bsuWIViURQVxc91xhERPXFOYiIVI3zEFHro
dIkEWj45RUffvghDAwMsGbNGqSnp8PCwgLLli3D+PHjG/lKiIiIiIiImj+RXC6XqzoIIiIiIi
Iiahp48B8REREREREJmCQSERERERGRgEkiERERERERCZgkEhERERERkYBJIhEREREREQmYJBI
REREREZGASSIREREREREJmCQSERERERGRgEkiERERERERCZgkEhERERERkYBJIhEREREREQmY
JBIREREREZGASSIREREREREJmCQSERERERGRgEkiERERERERCZgkEhERERERkYBJIhEREREREE
QmYJBIREREREZGASSIREREREREJmCSSQC6XqzoEImrFOAcRUVPAuYgI0FB1ANQ0nD59Gj/99B
NWr17doONOmTIFALBnz54GHbelsbGxqVKmoaEBfX19Dg4YOHChejVqxeSk5MxZMiQZ463cuV
KjBkzpjFCJWoUnINUa/DgwUhJSamx/syZMzA1NRXavfbaa1izZk21bcePH4+rV69iwYIF8Pf3
b6yQiRoF5yLVqM/7m19++QW7du3CX3/9heLiYpiamsLLywvvvvPMOOnTo0Nght3hMEgkAsHPnT
lWH0OqNHTsW48aNE74uKSlBYmIitmzZAj8/P5w8eRImJiY4cOCA0CYzMxMLFizAvHnzMHDgQK
Hc0tLyRYZO9Nw4B6mel5cX3n333WrrjIyMhNdqamr45ZdfUFxcDC0tLYV2ycnJuHr1aqPGSdS
YOBephrLvb44cOYKlS5diwoQJmD59OnR0dHD79m189dVX+PXXX3Ho0CG0a9fuBV9Fy8IkkaiJ
MDU1RZ8+fRTK+vXrB0tLS8yaNQs//fQTJk+erNAmtkZQMWk+b99iYiUYWRkVKd5xNnZGTExM
Thz5gyGDh2qUHfixAn07NkTcXFxjRQlEbVEYrFYqfc3mzdvhpOTE3mzZzZtwmuvveLiwveeO
MNfPfdd5g1a1Zjh92i8ZlEwpQpU3DhwgVcuHABABNjYY2iI6OBgDEx8djYIFcHd3h729PTw9PbF
ixQoUFRUJff/8809MHV1xbvvvou917tw59OrVC0FBGxsEBUVhSlTp
r776CgMHDoSDgwPefvvtKp9Onzp1CpMmTYTRHYnGxsEBUVhSlTp
sDR0REDBw7EwYMHIZFIsGDBAjg5OcHLy6vKJ4ePHz/GRx99hJdffhkODg4YP348oqKiav17XL
p0KWxsbGr8U/n3qix9ff169SNqLjgHNe056H+99tqru
zUfOZix4+fFjt35utrS2CgoLQq1ev5/4erZ1IzqdzW73bt28jjMDAQABAcHIxu3bqhoKAAI0aM
QJ8+fTBlyhSIxWL89ttv2LVrF95//3288PDgAV577TW89dZbGDp0KHJycrBu3TqUlZXh5
59/hpqamsIa/IsXL2LWrFkYPnw4Vq5cCTW16j+jmDJlLlCuLi4mBtbY3Zs2dDLpfj3//+N0pLS/
HLL79AXV0dv/32G+bOnYupU6DKCoqwt69e/H7779j3759cHZ2RnR0NKZ0nQQojIyPMmTM
HNjY2+Oqrr3DhwgVYWlrCx8cHTk52LdvH06fPo2DBw/C0dERxcXFGD9+PB4+fIIj33sPJiYm
OHToEE6fPo3t27ejf//+1cZ9//+59ZGVl1fj3Z88XKpbqupiUOWHNrsD79vw5MnPFHZXjT//
m188sknuHfvHo4fPw5TU1r+PkMIjVnnINUPwcNHjhxdCQ6wl1pZkS5lqnP0o16ajyEYj/66upjVEnHvT/pF9ZKT2Bm19ewfNl0SSuHpjdYeffvDDSRbV9YFr8cBXAVo50MhzgZlSHWWWVlSE8PFZVYz8/H0uWL
EFcXBx69eqF27dvY/pOJRo0Vi2bJnQx8ubfn5H1dgS+rfeegt+fn4AAF1dXUyYMAGOjocpWzwcl/PWY6yWgJgZp4RHsx9FtsEvsNQsfi9yHeVvHncRFy/wTVNQnAJC0QXh3KYLCQm
4ICAgAAPTq1QunT59GbGwsHB0dcezYMcTHx+Pbb79F7969AQCvvPIKpkyZgtDQUBw6dKjamC0
tLZ/rWcCwsDCEhYVBLSvJzUNOYg44ePYqjR49WKf/666h4uKiUDZi
xAiEhITgzJkzGDZsGICKu4hODTk4wNzevdwxEqsS5qGnMRXx2WefQSaT4eeff8apU6E7zt48
GD4+fnxPVMDYJJI1fLw8ICHhwdKS0tx584ckJCcjKyhIeBO7duPduPJV+Pj4wM
vLCy4uLnB0dFQYKzU1VfFVfgELDg4+JkTIlD1U6bKSbewsBAAhHXmCuYPr168PrK8OkxaOTk5a
LSxXGcnJyEl5X7nZVOdkBgKGhIQAgNzcXABAVFQjY2PY29ujrKxMqM2m0CCsXr0aOTk5Nu2
bZWYZTIZZDJZjdekrq4OAJgmNOkjHT8HKyd6vrz4k4ePYqjR49WKf/666h4uKiUDZi
0GDBg1SWM1QqWvXrlXKOnXqhGqXHD59uDkyZMKSeLkyZcHJ+oOejJc9OLnorrQ19fH+vXrkyZycjD
NnziA6OhrR0dHYuXMnvv32W4SGhgq/D4ijBoqLCTnYW
FBfbu3YuvvvoK3377LbXbu3AkDAwNMMjQJCxcuFCa/5ORkeHh4Do6Ghs2bEBQUNAzv7
15VjVU46WVlZCA4OxqlTSidC5c2f07dsXQNXzjaapLsP53/Kc9fvwYmZmZsLe3r7Y+MzOz2
knx//7v/3DkyJEax929ezqzfc3NxqrDcxMYGDgwMAwNHREV26dMH06dPx3nvvYdu2bc89oRI1J5
yDXvwc1K5dO2EOqosRI0Zg7+bP//hU9L3vQaKeQ4xUuPHCpPy92r3wR26fhgcmTJ2P
y5MmQyWQ4deKgKCGLFChw+fLhBvkdrxSSRqvXhl3Fh586dGDduHOnTpvyNwBRGx3P0ogOpmqz8
dMTGjRtRUlKCS5cu4cCBGYWeh48+FztRLTaegsLi2rLFyxYUOun6F26dNVq+e7r3Xn7w
iQlJWHHjh1wdnaGWCxETE0NLTaegsLi2rLFyxYUOun6F26dFEqDj
c3N0yePBl79uzBt99+iwkTJVn6g54xyk+jnoWYYPH45Vq1bhzJkziIuLg7u7O9q3b9+g34ZIiLHi5duoRu3bph
7NixwoSYkZGBW7duCE9e7dy5E0VFRd54E4MHlCuAzNo8xyV0ZqpqDkNDtPzIQSGDAgNDQ3MmzcPf/75J5YvX16
Tp1wrJly6osgVDWpUuXMGzYMLi7i7u0MsGMAzp49Coi379+iEtLg0g9D8CcqKg
rbt2+Hurp6tf0sLCwU2v/vn/osGUxgX3vvffQz0UMHreegJ5rSHFSbh7h07om/
fvvj5559x8uRLQLnoiea6lzUvXt3PH78GLGGLt27aq2/7/gpH1xW5jGTHzx5rGmAHP4NHXH3zB2xHCqmQEMwXDlJdSoP//Lu7pBIJJg/fz
dxAVFYWcnBw4OjoiISFB2P3q4u7pBIJJg/fz7OnDmD33//HR1//HUFBQRCLx
Rg0aFCV76GtrY1/ZC8To6OuL777/HsWPHEB0djS1btmDp0qUQiURCfPU1Zs

wYdOrUCX5+fjhy5AjOnz+PtWvXYt26dTAxMYGmpuZzja8MPT09vP/++3j8+DHWrVv3wr4v0Yv
GOeiJpjQHPcuIESPw888/Iy0tDd7e3qoOh+i5cS56oqnORV27dsWcOXOwb98+zJkzB8ePH0dM
TAx++uknvPvuu4iKisLixYtVEltLwuWmBACYPHkybty4gdmzZ2PlypWYO3cusrOzsXv3bmzat
AlmZmZ44403IBKJsHXrVuTk5MDW1hZbtmzBpk2b8MEHH6C8vBy9evVCREREtRsdAICXlxeGDR
uGzZs3Y9iwYbC2tq5XvKtWrcJnn30mfFJnZWWFTz75BP/9738RExNT778HoGLHr6+//hpr1qx
BSEgIcnNzYW5ujg8//BAzZsx4rrHr46233sKBAwdw8OBBTJgwocZnA4iaM85BTzS1Oag2w4cP
x+eff46BAwfCwMBA1eEQPTfORU805bnogw8+QM+ePXHw4EGsWLECeXl5MDAwgIuLC7777jvY2
tqqNL6WgOckEhERERERkYDLTYmIiIiIiEjAJJGIiIiIiIgETBKJiIiIhIwCSRiIiIiIBE
wSiYiIiISMAkkYiIiIiARMEomIiIiEigoeoAmovychmysvJVHQYRATA21ld1CC8c5yC
ipoXzEBGpWmPOQ7yTSERERERERAImiURERERERCRgkkhERERERERQCJolERERERERQkYJJIRERE
RERERAiaJRERERERERJGCSSERERERERERAImiURERERERCTQUHUARNQw0h7l45tTiQCASa92h1n7N
iqOiKju+PtLRKrEOYhIEe8kErUQ+04n4q87WfjrThb2n76t6nCIlMLfXyJSJc5BRIqYJBK1EG
kPC4TXqQ/zVRgJkfL4+0tEqsQ5iEgRk0QiIiIiIiISMEkkIiIiIiAZNEIiIiIiIiEqg8SZT
JZFi/fj08PT3Ru3dvzJgxA/fu3aux/Y0bNzBt2jQ4OTnB3d0dH330EaRSqUKbkydPwsHBw4O
Dhg1ahTOnj3b2JdBRERERETUIqg8SQwLC8P+/fuxYsUKHDhwACUlvcjLIiIiIiIiap
ZUmiSWlJQgIiIC/v7+8PLygq2tLdatW4eMjAxERkZWaZ+SkgJPT08EBwfDysoKzs7OGDduHKK
iooQ227Ztg7e3N3x9fWFtbY0lS5bA3t4eu3btepGXRkRERERE1CypNEmMj49Hfn4+3N3dhTID
AwPY2dnh4sWLVdo7OTlh7dq10NDQAADcvn0bR44cwYABAwBAwBULF2NjY1VGA8A3NzcEBMT04hXQ
kRERERE1DJoqPKbp6enAwDMzMwUyk1MTJCWllZr32HDhuHu3bswNzdHWFgYAEAqlaKgoACmpq
ZKj1cXGhoqX51LVCORSPE1f1+JiIiIqD5UmiQWFhYCAMRisUK5lpYWcnJyau0bGhqKoqIihIa
GYurUqTh27BiKiopqHK+4uPi5YlVTE8HQsM1zjUHUmNTURAqv+ftKRERERPWh0iRRW1sbQMWz
iZWvAaC4uBg6Ojq19nVwcAAAbNiwAV5eXoiMjISXl5cswt6zyGRySKUFzzUGUWOSyeQKr
7Oz81UYTeNiAkxERETUeFSaJFYuM5VIJLC0tBTKJRIJbG1tq7RPSkpCcnKykAwCFUtJ27Zti4
yMDLRr1w66urqQSCQK/SQSSZUlqPVRViZ77jGIGotcrviav69ERERERVB8qfWjJ1tYWenp6iI6
OFsqkUilu3rwJFxeXKu3PnTuHhQsXi8vTyi7f/8+srOzYW1tDZFIBGdnZ1y4cEGhX3R0NPr2
7dt4F0JERERERNRCqDRJFIvF8PX1RWhoKE6fPo34+Hi8//77MDU1hbm5OczMzGBqagpTU1OYmZ
29FUbYtGlrawOoeDax8jUAFBcXQ0dHp9a+Dg4OAIANGzbAy8sLkZGR8PLyEsZ7Wl3Gq41MJodU
WlDv/k2dTCZXeJ2dna/CaIiejR9kEFFLxiSRqBlLyczDyr2xKCguq7b+bnouVu6NRZCvMxPFG
lQuM5VIJLC0tBTKJRIJbG1tq7RPSkpCcnKykAwCFUtJ27Zti4yMDLRr1w66urqQSCQK/SQSSZ
UlqMoqK5M9u1EzJZcrvm7J10pERNTU8aEPomZKLpcj/Ie4GhPESgXFZYg4EQf50+/CSWBraws
9PT1ER0cLZVKpFDdv3oSLi0uV9ufOncPChQnlN2/fx/Z2dmVxaGSCSCs7MzLly4oNAv
OjoaffV2bbwLISIiImogTBKJmqm/U6W4m55bp7Z30nLxd5q0kSNqnsRiMXx9fREaGorTp08jP
j4e77//PkxNTWFubg4zMzOYmpoiNDQUpqamMDU1hZmZKczMTGBqagpTU1OYmZ2dnR4eHjAD7T9/G++N7qzgiqouCotJGbd+aqKurIzAwEIGBgVXqLCwskJCQoFDWpUsXbN
26tdYxR48ejdGjRzdkmERERERQvBO8kEtIePtkxMfUh7yQ2F+pqyh1toaut2UiREBHR85LJZFi
/fj08PT3Ru3dvzJgxA/fu3aux/Y0bNzBt2jQ4OTnB3d0dH330EaRSxccKTp48CR8fHw4ODhg1ahTOnj3b2JdBRERERETUIqg8SQwLC8P+/fuxYsUKHDhwACUlvcjLIiIiIiIiap

```
qFE4e/ZsY18GEbUQTBKJmqHbKTk4H5ehVB+nHh0aKRoiInpeYWFh2L9/P1asWIEDBw5AJBJh9
uzZVY7TASp2S/bz84OlpSWOHDmCsLAwxMbGYsmSJUKb8+fPIzAwEJMmTcLRo0fh4eGB+fPnIy
kp6UVeFhE1U0wSiZoRuVyOX2OT8e+vY5FfWPuupk/rYqaPrmYGjRgZERHVV0lJCSIiIuDv7w8
vLy/Y2tpi3bp1yMjIQGRkZJX2KSkp8PT0RHBwMKysrODs7Ixx48YhKipKaLNt2zZ4e3vD19cX
1tbWWLJkCezt7bFr164XeWlE1EwxSSRqJkpKyxHxQxz2/HwL5bK6H2ehq6WBGT49IRIptzyVi
IhejPj4eOTn58Pd3V0oMzAwgJ2dHS5evFilvZOTE9auXQsNjYqtJW7fvo0jR45gwIABACqWrs
bGxiqMBwBubm6IiYlpxCshopaCG9cQNQMPHxdi45HruJ+Rp1Deto0Y4wZZ41RMcrXHYXQx08c
Mn54wN9Z7UaESEZGS0tPTAQBmZmYK5SYmJkhLS6u177Bhw3D37l2Ym5sjLCwMQMVZrwUFBTA1
NVV6vGfR0GiZ9xee/hxVJGq510lUV0wSiZq4v+5kYcuxG8gvUlxe2s2iLd4d3Qvt9LTQ394UH
2z8Azn5Fc+u6IjV8cHbfdDVzIB3EImImrjCwkIAFee2Pk1LSws5OTm19g0NDUVRURFCQ0Mxde
pUHDt2TDjXtbrxiouL6x2nmpoIhoZt6t2/KVN7ajO4lnydRHXFJJGoiZLL5Thx/h4On/0b8v9
ZXTqkrwUmDO4GDfWKTzpFIpHwGqjYydS6U9sXGS4REdWTtrY2gIpnEytfA0BxcTF0dHRq7evg
4AAA2LBhA7y8vBAZGQkvLy9hvKfVZbzayGRySKUFz27YDMmeeoxDJpMjO5u7vVPT15gfZjBJJ
GqCCovLEP5DHGJvZSqUa2qoYdpwG7zcy6yGnkRE1NxULjOVSCSwtLQUyiUSCWxtbau0T0pKQn
JyspAMAhVLSdu2bYuMjAy0a9cOurq6kEgkCv0kEkmVJajKKiuTPVf/purpD2Pl8pZ7nUR1xQX
XRE1M2qN8rNgdUyVB7NBWG//n25cJIhE1WWmP8rHmwBWsOXAFaY94J6aubG1toaenh+joaKFM
KpXi5s2bcHFxqdL+3LlzWLhwIfLynjynfv/+fWRnZ8Pa2hoikQjOzs64cOGCQr/o6Gj07du38
S6EiFoMJolETcilBAk+3RWDtEeKy3l6dTHCR9Nd0dlUX0WRERE9277TifjrThb+upOF/advqz
qcZkMsFsPX1xehoaE4ffo04uPj8f7778PU1BTe3t4oLy9HZmam8KzhG2+8AX19fQQGBiIxMRE
xMTEICAiAo6MjBg0aBADw8/PDDz/8gB07diApKQmrV69GXFwcpk2bpspLJaJmgkkiURMgk8nx
3W9J2HTkBopLyhXqXnu5M94b1xt6Opoqio6IqG7SHj75gCv1Ie8kKiMgIABjx47F8uXLMXHiR
KirqyM8PBxisRhpaWnw8PDAiRMnAACGhobYvXs3ZDIZJk6ciPnz58POzg7h4eFQV1cHAHh4eO
CLL77Avn378Oabb+L8+fPYsmULrK2tVXmZRNRMqPyZRJlMho0bN+LgwYOQSqXo27cvgoOD0bl
z52rbJyYmIiQkBFevXoWamhpcXV2xdOlSdOrUSWgzePBgpKSkKPQbNWoUQkNDG/VaiOojt6AE
X/33L/x1N1uhXFusjtmv2cGph7GKIiMiohdFXV0dgYGBCAwMrFJnYWGBhIQEhbIuXbpg69att
Y45evRojB49uiHDJKJWot5JYlJSEv744w9IJBJMmTIFDx48ENbUKyMsLAz79+/HypUr0bFjR4
SEhGD27Nk4fvx4la2bs7Oz4efnB1dXV+zduxfFxcX497//yMvzZuHIkSPQ0tJCXl4eUlNTsXX
rVtjb2wt9n94tjKipuJeei42Hr+ORtEih3Ky9LhaMcYBZe27BTURERERQvltJJYnl5OYKDg3Ho
0CHI5XKIRCKMGDECmzZtwoMHD7B3794675xVUlKCiIgIBAYGCjt0rVu3Dp6enoyenoiMjMTIkSMV2
p86dQqFhYVYtWoVtLS0AAAhISHw8vJCbGws+vfvj1u3bkEul2PZ2RkGBgbKh7RC/PH9TTs+j
EBZeWKO6i52BjDz6cndLRUfqOfiIiIiFohpZ9J3Lx5M77//nusWLECf/zxB+/fvm8ZMkSyGQ
yrFu3rs5jxcfHIz8/H+7+fvzB2bNgkJ4tMqD5tNSEiAsbExE0Rq
ssrKZdjzUwLCf4hTSBBFImDcIGvMG92LCSIRERERqYzS70QPHTqEgIAAvPXWWygICXl4eUlNTsXb
QICApR67i89PR3Ak/OBKpmYmCAtLa1KewsLC1hYWCiUbd26FVpaWnB1dQUA3Lp1C7q6uvD398
fly5dhZGSEMWPGYOrUqVBT4z49pFrZucUIO3odSSlShXI9HU2884Y97KyMVBQZERERERVEFpZP
Ehw8fomfPntXWdezYEVKptNq66hQWFgJAlWcPtbS0hDuDtdm9eze++eYeYbBAUFoX379gAqNrbJ
zc2Fj48PFixGzGiYOrUqVBT4z49pFrZucUIO3odSSlShXI9HU2884Y97KyMVBQZERERERVEFpZP
aaPgLcc0aGdznONz58jERERETUEpZPEzp0748yZM3j55Zer1F24cKHGXUmrU7mZTElJicLGMs
XFxdDRqfkNs1wux5dffonNmzdj7ty5d59ulC3Y8cOFBcXCxvo2NjYID8/H5s3b9eXs3b4a/v3+97ya
qqYlgaNgyNxFRUxMpvG6p16kqcrkc3//+NyL++xfKZXKFOu9+lnhnjCPEmurP/X34cyQiIiKi
hqB0kjht2jR89NFHHTxaBBgyASiXDv3j1ER0cjIiICS5curfNYlctMJRIJLC0thXKJRAJbW
9tq+5SWliIoKAjHjx/H4sWLmIoKAjIV6TU1NaGoqnifXo0cPFBQUICcnB4aGhnVFpaZP
GzGzZDsqcSF5lMjuxsnm3VUIpLyxFxPA5Rf6UrlKuriTB1uA0GOpkjP68IDfE33pp+jkyAiYi
IiBqP0kniuHHjkJWVhS1btmDfvn3Yvn07Ro0aRc9PFBQUICcnB4aGhnVFpaZPEhw8fomfPntXWdezYEVKptNq66hQWFgJAlWcPtbS0hDuDtdm9eze++eYeYbBAUFoX379gAqNrbJ
8yZ8fX2r7bN48WJERkZizZo1VXY/lclkePXVyY/To6dOhQ7wSxUlmZ7NmNm
iG5XPF1S73OF02SXYCNh28gOTNPodxQXwvvju4Fa/O2KC+XA5BXP4CS+HMkIiIiooagdJKYk5
ODuXPnYvkZWq5/6fZYsWQI7OzkJCSgsLAQ7uiI6OxvTp0x86VqLmqAUBKf/9Dd8cy290f/CpwB743
8c9YW7GgjJERERE1H7pdSfxYadtNmRmZoaoqChRUU12ubu7o6srCzCzN18nJyc32Z25ujoiICK
```

3ppkSPgrymFltSMnAut1irXWIhxkv/44MRvi4GioyIiIiIqOV0vqUxc+ZMxMbG4uTJkygtLW2
LmIiMTsHtSvxj+5lGCWLXLlZY+UIAE0QiIiIiMho630n85ptvcOPGDcyaNavJ7SKRCJcvX37o
wIiMxamMW9h2IBOK2jqt9sc8HTFvki+sLS3ucyQRERERUfujc5I4efLktoiDyOjUqdX46uc8f
H/qWqNtz4zqjUkjPSBuuMI9EREREZER0DlJXLhwYVvEQWRUZFVKbPzmIjJ/v6PVbiU1x9xJvv
Dv6/TIY3J1skaJrAYA0N2J6wgSERERkX50ThIBQKlUYs+ePUhNTYVMJoO9vT0CAgIwdepUSKX
S1o6RqF25ckOGj/deQFmFQqvdzbkTFoYNRDd7a4PE9XxwP+wScgAAzwX3NUgMRERERGT8dE4S
ZTIZzs6ciczMTh3h3Ozs7Iz89HSkoKPv/8c+zatQu2trZtESuRwR09dx2fH8qGk57th/
bti1v/0h1RiZqDIAFfHTljyrL/Bvj8RERERmQadk8T69Qd37tyJgIAATfuZM2ewbNkyfPjwsv
i5cmWrBklkaLUqNT4/lIVf0m9qtYtFIkwf1xchAe4e4Q8flDIiIiIjIBOi+BceTIEbz22mtaCSI
ABAQEYNGiRjfhx9aLTii9qBUVoMPPv+1UYJoZ22BqOf98VRgDyaIRERERGQydL6TWFDL
z/99BNGjhzZaNuRI0fQq1evVgmyJAEQcD3p65h98+5ELQfP8Tj/t3x/JNesDDX+UY8ERERER
G7p3OSOGfOHLzxxhtQKpWYNGkSnJycUFxejH4pxiIyIiiJqezoniaGhof0JnPkjENjNnQcPFiz
Ej0qhaVyfLznAq4XV2m1O9hJsWHz/ZGuqrCA8Px71z5eXo7On
TvD398fdnb8AE3G62zObWxJuYqZRZ1We/9e9pj/jB/srCUGioyIqP0TBAGqOrXma3lNLfKul6
NPdzsW9yIiMjJ6JYnffvstUlNT8f777wO4u/zFiy++iFdffRUhISGtGiBRW1OrBXxzzPB/7//t
bo20ThvfEX8f2gZmYzx8SEd3P9duV2PpdBsqrlJq2amUd3t/xKzxcbDFnYn+4OdsYMEIi6ihu
llRh1+EcAMCMJ/vB1bGTgSMyTjp/8t2zZw+WLl2K6upqTZujoyN69OiB1xYvxnYmxN/ymQvsL
dW1+Oir840SRKmFGSKmDMDD0J/oyQSQieoDrtysRszMNvxVWNLn9t8IKxOxMw/XblY84MiLqiP
51JAeX8ktxKb8UXxzJNXQ4RkvnT7/JycI4+eWXsb7969sW0sW7NGk1R7R0UtrUtGiBRW/n
9VgX+sf00Llwp0Wrv5mCNlS8GINCnq4EiIyIyDoIgYOOt3GZArVA/cT65QIflABoR7y0UTEbWy
m8Vyzesb99SYoJbTOUm8du0aRo0a1eS2UaNGIT8//GDImprJy4VYvWOX3H7To1W++B+Tvj7z
AC4OXFqAhFRc67ckN33DuK98m9W4MpNWRtHRERErUHnJnLJFr1644f0/58k9suX74/4EVGCIggCjqbbf0OmYs9nFbRERG1Jp0
/DU+ZMgUbNmxAp06d6d8OSTT8LBwFPgw1q9fj4UpaU4fPFOl9ZZMsTbQgxQ37LiK7oFyrvZOl
OeZP9sOAPo4GioyIyDjUqtTI+r0M53KbBKZQqfj5TW1bRQZERG1Jp2TxPnz5yyMvLw//+/
Mc/8N5772naBUHAhAkTEBkZ2aoBErWG3OvlVrtPvlSNp7AXcqlVrtPbva4NWwgejaxcpAkRERtW+yKi
XO55UgPbcYF38rhUJZ1/xB92FtadGKkZkWtVqN9evXY/fu3ZDJBg6dChWnVqFxZrxXBhOd
O4uLikJ6eDrFYjDJy5EF7vhcv35fX7DNu3Dhcv35hJkyYhYfzzkmiubk5PQ
g4iICPz666666+4c+cObGltMXToUj4+LRFjER6EwQBP5+9jl2Hc1Cn11i6YEOTngpkvCG1MDNd
ERE7Y8gCLheXKXGKcyy3GlesytFa5mcFeTq3Uk+lJSkrCF198198gZiYGHGrlg1xcXGGYO3cuU
JSIJFor9NbVlaGWbNmITAwEDt37oRcocHH36Il9+GXv37oVUKkVlZSVu3LiBiUe7sJXxVu3hBTZs2wc/PT3O
spaXloz41IjJCej981PP/r9a9fP/Tr1w8AcPbsWXTr1g03btxArgn8iuhUSe2ZHt29n3H9uFyc4tfP/
eC64H8YNceMCz0REuPu8MV0ci3d9t1Zsw/tdeZzNh/+kWhCfJEWoyGqd0OtdUlqaRCRCmAZIQ4s/cTewXTe1cfoi7wSzDsSCcy+8/24e9Sc/vo6
EolkpOTERUVhbFjxwqIAEhMTMRJ0MQBawIAEhMTMXr0aBw6AgTJO7U2v/w4cOorq7GBx98QCFjkZ
aWhqCgIGRnZ0MQBAwZMGr5dvy5E5FudeE4Sq6qq8N5778HHhe7f6Kmsdp4+8WCpaKXePEiAUm
ITk6Gq6trW8RK1dVBFRPIAl3W0PC1aXPS6GpFUYOE8dLaNNWwX4tRdMvcMwDiqVhwmkbCn7bXSo
0hwDPR3h39cJA3o7wNrrg5WbSWcmN2N8oDmMH2aV9xSIz6+Pvk9+7Ko7Ozs7+Pr64vTp0
42SxKCgIHz88ceaBLGh8vK7K7ZH1XXxHPd7Ijh8IZB6UXnjDE+Ph7ff/89Ro4cNsDEixpiEZ
Ii8M9//hPx8fFISEhocX9tMQf/4MGDWLduHa5duwPYDw9ERUVhzJgxup4gaNlL+aXY+M1FVNV
of2Dp694Zr04ZgC42jf9TJSIydYIg4EaJXDONNOpyWjTjSvV8DDyVPbPXBGLcDfXq7XQmD0
N2cbrAgfgq3fZTS5HEZvV1vMDu0PN2eb1joVk1NyeHfGy70X2bt27YqbN2822822t/d3R3u7u5ab
Zs2bYJUKkVgYCAADIDs7G9G9bW1oiMjMTZs2fh4OCAsLAwzJw5E2KxxNtNczN9T+2PWt4/UIkMt
3z7Aj4XrYOnZPEI0eOYPny5XfPPYnZXj66aeRkZGB69evIC+15+CfPHkSUVF
RWL58OYKCgvDVV1V19hwYIF2LdvHvHz5/hwhxA1UlPhAZB+wL4L4wN+NAQUQd
h6pOjexrfxjcHKKvGb0hiT8gEX4cJJxWqWdB8/MQBvrP8PyqvuFgizkpjhj
ef80cfVjncQm1FdXQ0AjT73SKVSZz3B/nss8+wa9cFixAoAtKiAqGhoVi4cCC
HOnDmD+Ph4lJeXY/HixXXrFKRaLYG9vmusIi8Uirdemp4dAdfL1qFzknjzh306dMzzz
D3Nxcc1exc+fOUChaXgGXaXg67Lebgb968GSHhEIQPcwALFu2DFnDsx7kurvv6nq6ZCSqFSok
f5eBX7Nva7VbmInw1tFBoLLqYZ6DZ6VXmIsxc7EfM9FdwzaqIzZZJCNqHhDfBwW
NjHEZ30qEIqEom0LsRZW1rAs3tnnfvpiOqLySiVSq3CMgqFAlZW96++LQgCPvroYsAHz58
/HSy+9pNm2bds2KBQK2NjcvYPrAtXGaKrXhtTVSrBchkcp2PMwbqBsXt1GoBZWV
VBoyGHkZHei/bMgHWOUl0c3NDVlYWAgIC8MMP8Df318zAB09erTR8HTZ00+Gq1GmlaVi+

fLnWtuHDh+PQoUO6nCYZkZslVVi/5wJulmj/x+XU2RILpg5ELxdbA0VGxoJVBcmYCYKAwlL53
buFOcXIaeE00m72Vpq7hX3dO3OmhQHVTzMtKipCz549Ne1FRUX3rRxfW1uLFStWICUlBUuXLs
WcOXO0tltYWMDCQjvZ9/LyglwuR3l5Oezt7fWKVaVS63Vce9fw34wgmO55dgR8L1uHzknijBk
z8MEHH2Dnzp3Iz8/HmjVrAACRkZE4fPgwVq5c2eK+WnsOvkwmg1wuh4uLS4v605Wpzmk25rnb
ZzKL8Mm3lxoVXBjQxwERUwbA1lpynyOJ7mJVQTJGqjo1cgrKNc8XFpVVN3uMSAT0c++ieb7Q1
ZFTsNoLHx8f2NjYIDU1VZMkymQyXL58WTMz6l5Lly7FoUOHkJCQ0GicUqvVePLJJzFt2jRERE
Ro2i9cuAAnJye9E0Qi6jh0ThJfeOEFODg44NSpU4iMjERoaOjdjszN8fbbb+PZZ59tcV+tPQe
/Pulsqj9dpsE2xZTnNBvj3O06tYCdBzPw1Y85jbZNC+6Hv03oDzMxn4Gh5rGqIBmLqpoG00iv
lD6wmmg9K6kZBvZxxKC+ThjYxxE2VlzMvj2SSCQIDw9HfHw8HBwc4Obmhri4OLi4uCAkJAR1d
XUoLS2Fra0tLC0tsWfPHk11+WHDhuH27T8ftajfZ/z48diyZQs8PDzg5+eHEydOYMuWLYiOjj
bgmRKRsdBrncSJEyc2+uCUmJiocz+tPQe//kObUqnU2r+5/lqC8/Dbjwq5Ehv2XsTF/FKtdku
JGeZN9kOAT1fIyk3zvaK7WvNCBqsKtg/GPKOhLd0sqcK5nGKczb6N7GvlULdgHmlXeysM7ueE
wf2c4dWzyyOdRsr3UX+LFi2CSqXCypUrUVNTg8DAQGzduhUSiQQFBQUIDg5GTEwMwsLCkJKSA
gCIjY1FbGysVj/1+yxZsgR2dnZISEhAYWEh3N3dER0djenTpxvi9IjIyOiVJLaW1p6D36VLF1
hbW6OoqEjrmKKkiokZTUPVhqnOajWnu9tXCCqzfcwElMu0Kfa6O1lgYNhCujp3adfzU/rCqYPt
gjDMa2kJdnRoZv5Xi1OVbOHXpJq7fbv6inVgEePdywDA/Fwzz7YYe3WwNVk2U76P+zMzMEBUV
haioqEbb3N3dkZWVpfk6OTm52f7Mzc0RERGhNd2UiKilDJoktvYcfJFIhCFDhuDUqVOYNm2ap
j01NRVDhw5tuxOhR+I/F27is++zUHtPEjjU2xmzQ/vDSmrQX2cyUqwq2D4Y24yG1iSvUeF8Xj
HO5dxdpuLeNV6bYikxw0BPRwzu54RBfZ20nr++c8dwvycd6X1kAkxEpsygn6rbYg7+rFmzMG/
ePPj6+mLMmDH4+uuvkZGRgffff9+AZ0oPQ1Wnxr8O5+Cns9rLCYhEwP+O9T25BhfpjVUF
2wdjmtHQGorK5DiXW4L03GJkX7uDOnXz00idOltqqpF63zONtL38vDra+0hEZKoMfuultefgj
xo1CqtXr0ZSUhISExPh5uaGDFPQ0qjmmhCTUBE5MvHQXXfv3Dp8fWkjRzGtRa5pq9x+0YuyIAe
wFqwrSo6BWC8i9/mc10nuX7GmKCEAfN7s/qpE6wc2pEy+IERHRI6F3kqhWq5GdnY2ioiIMGTI
EKpUKXbp00bmfTtDTpkZBlClTdI6F2pfsa3eQtO8iZFXahYh6udhDQBcOr8cMWIiABW
FaS2U61Q4WJ+Kc7lFOPClRJUVtc2e4xUYoYBHg4Y1NcJj3k6wq4Tl/EhIqJHT68k8ZtvkFCQ
gKKioogFouxe/durFu3DhYWFkhISIBJF4Ig4PCvBfj3j7mNpmCNGuiKF8KkF8Z7wcLczEDRkS
liVUFqLbfvVN9d1D63GFm/t2waqaOdtME0UntYsCIoEREZmM5J4oEDB/Dmm28iIiICQTQ2
//XUAwFNPPYV33nkHSUlJeO2111lo7TuogFLV12H4wEycv39JqNxOLMCPEC4/7d+d0K2p1rCpI
+lKrBVy5IdMkhteLmy/UIgLQu7udJjF0d+Y0UiIial90ThI3btyI5557Dm+//Tbq6uo07Wq9Ph7dbZQJEREf2pWqHCpfxSpOcWIz2vZd
NIJRZi+Hk4wL+vEx7r64TOnEZKRETtmM5JYn5+PpPSYtW9bktkGDBmHdunUPHRR1POU
gV2iXfvfq0QURUwbwAxURGVRxeTXSc0twLrcYWb+XDSuphmjSheBXT/DRSe1uppuhM/l5dOE2eiIiMhs5J
oqOjI/Ly8hAUFNRoW15eHhwdHQ0SGJkOlVrAv1MvI/Xiba12sViE/47xRrBvT04vJSIiIiMBT0I
CMtaMGi9gDQ29VWM420R1cbTiMlIiKjpHOSGBoairVr16Jr164YN24cgLtTA3p99hlRv1F69+XW
mnWz1IMk3yGhW2pFzGudxirXaJuRgvhfpghK+LgSIjoo6oRqnCpfwypOcW43xeMWTyFkwjNRf
D18MB/v3uViPtYiN9BJESERG1LZ2TxNdeew3Z2dl47bXXXIBbfvcPzwgsvQC6XIyAgAIsXL271
IMn0FNyuxMd7LuBWWbVu3MXSywMeww9utoYKDIi6khKZTU498fahZ1X70BV1/zi711sJA2mk
dpDYsFppERE7YEgCFrjuLymFnnXy9Gnux1nduhI5yRRIPfgy5Yt+M9//oMTJ06gvLwctra2
ZsGMaOHcs3gJp1KuMWth3IhKK2Tqv9MU9HzJ3ki06WFgaKjIhMnVoQ8NvCs000mtFlc0fhLv
rs/r/MY20ZzdOIyUiam+u367E1u8yUN5gfe1qZR3e3/ErPFxsMWdif7g58yzES+m1TiIAjBw5
ssnnEonup06txxlc/5+H7U9cabZs80OTR/WGmOB8R3j/WGmB+8iKiVKZR1uPxbKc7lFuN8Xon
7eXPQb1c8IgTyfY23IaKRFRe3X9diVidqY1KoBY77fCCsTsTMOK8CFMFFtI5yRxxYoV990mFo
thbW0NDw8PhIaGwt7+/+qGJi4zNDw8PhIaGwt7e/qGCI9Mhq1Ji4zcXkfn7Ha12K6k55k7yhX9fJ8MERkQmqaxCgfQ/ppF
mXC1Drar5aaSdbSQY5Hn3bmF/D3tIOY2UiKjdqp9aKleosKmJCvn3kitUSD6QgZUzAzgbpAV0
ThILCwuRlpYGhUIBNzc3ODs7o6SkBAUFBRCLxBycJJSQk2bNiAf/3rX+jRo0dbxE1G5MoNG
T7eewFlFQqtdjfnTlg4dSC6OVgbKDIiMhWCIODqrQcyylGem4Jrt6qaNFxPbvZwLuHB/XVvpOuSjRUt
kMRERtTFWnRrVChRplnebvGqUK1Qrtv7Xb61CtVKGmwbZqhQp16uaNjXQ4qXI2oo/2YFrtyUwbM7195
ujs5J4hNPPIGcnBxs374d/v7+mvaioiJER0dj5MiRWeyfmsjQ3ZGsPGHA0Gf4eM8afqGUmB8WhlNS11tUEemGRGZEtk4sQTFHMqnq0LBVTZ05SHtK4rnLNfuJuCXA7Xu4HNcpAVeUCA4d
/zkvUQQ+WmGSdBy9evT0FZ4fNM3VhTnNBm8MeXjsU49EQJyJ+eFXJIqd/5tXzt7OrVCk6hkRPwWbM8X9p/9DgkJP5HYPtzl16r7/cXQ+kpdajmhsyTsNJ5t6e1rLPoOpWhbyDT9e5wWJZTWMPlR+LoTmnsqdcDHBKmHbSwGbFUxMpuqEX+bt
EPsRQ5Xqx5bkWoMdJIR0bNZ4rZRpvjGYlyvpchZklHJ9lHI0RRn1D6NG5GMy9V2sCr0I0JbNxgRMFHKOKAyy4Q0lQoI/Zfx9a0IlUJL94Ui/JzMzmZ96O9lD/+jOO9Kj5qmLnd5oORnaVAlXjG

```
ia1N4qYqrVgvbUyz/uvtUo/pxuWa28J8G7J9Grb2/JRTRjIgJgKTWDpcQclhIzWEnv/l1aXoP
Ce5ZUexBrVtFvEZ2TRF9fX2zZsgVBQUGQSCSadqVSieTkZPTv3x8AcOnSJbi6urZepGQUMq6W
YeM3F1FxzyLUvV3tsGDqAE7tIiKNllSjW5V8Ci2ZmGRuJoJPT3sM6uuEQX0d4dTZqnWDpRZxd
bJGiawGANDdqZOBoyF6sNaqiHk3sWuQ1DWTvDV8tu7eZFBZa3qJnbRBQndvgmclMf8j8bu7zU
r6R5vEDJZS7X2lFmZN3tnNu16O93f82uKYBnuxWGJL6Jwkvvnmm5g1axbGjRuHxx9/HI6Ojig
pKcHRo0dRWVmJLVu24MyZM1izZg0iIiLaImZqhwRBwPenruGrn/OgFrQ/0o31744ZT3rBwlxs
oOiIqL0RBAFbv8tothrdgxJEW2sLzTRSXw8HWEn1XtWJWsnzwf2wS8gBADwX3NfA0RDdX0vHI
LlChdh/nYWfhz0UtQ0KrjS4w3fvus+mQCox+zOJuyfJs5Jq//2gdqnErM1ncvTpbgcPF9sWzT
Tp7WqLPq52bRqPqdD5f9TBgwdjz5492LhxI44dO4bS0lK4uLhg9OjReOWVV9CzZ0+cOHECixY
twpw5c9oiZmpnapQqbDuQidOZRVrt5mYihD/ljTGDuhsoMiJqr67ckLV46mhDbs6dNNVI+7ja
QSzmNNL2xNWxE5Y862/oMIiapcsYVCGvxcnLRc3vaGBSCzPtO3D3JG/1UzWtGiZ9De/o/dH+K
BK71iQSiTBnYv8H3hUGAGupOWaH9jfq50wfJb0uu/bp0wexsbH33R4UFISgoCC9gyLjcatUjv
V7LuB6cZVWu72tFAvDBqI3r9YQURPO5hTrtL+vhz1emuADpy6cRkpED0/XMaitSMzFWolaUwl
do2ROeu8dvruvO/JFMzdnG6wIH4Kt32U0mfz3dr37jPujfL7U2OmVJFZUVODkyZOQy+UQhMaT
gaZMmfKwcZEROJdTjM0pl1Ct0J5m4dOzC155ZgDsOknucyQRdXS6Vpfr2sWKCSIRtZqHqXBpY
S7W3KXTSuakf/7d5FTNJp7DMxPzUZzW4uZsg7+/GIA31v9HU4jISmKGN57zRx9X465Uawg6J4
lHjx7Fa6+9hurqpqsIiUQiJokmTq0W8M3xfOz/72+Ntk0Y3hN/HduHgx4RPZCu1eVYjY6IWpO
uY8oTg7tj6hhPWErMYG7GzzjtlUgk0np/rC0tuCainnROEtesWYM+ffpgxYoV6NatG8RMBjqU
qppafPLtZVy4UqLVLrUww+yJ/RHo09VAkRGRMRnczwkHTl5t+f6sRkdErUjXMegvA11hY8WLV
dRx6JwkXrlyBUlJSQgICGiLeKgd+/1WBT7eewG379RotXezt8LCsIGc501ELcZqdERkSByDiB
5M59uA3bt3R2VlZVvEQu3YiUuFWL3j10YJon9fJ/z9xUAmiESkk/pqdNbNLFvBanRE1BY4BhE
9mM5J4vz58/Hxxx+joKCgVQJQq9VYu3YtRo8ejUGDBmH27Nm4erX52/9qtRpz5szBunXrGm0b
N24cvL29tf68+eabrRJvR6OqU2PX4Wxs3n8ZStWfC7yKAEwd3RsL/zoQ1pZcm4yIdFdfjc7Dx
bbJ7b1dbZtdxJqISF8cg4juT+dP9/v378etW7cQEhICBwcHWFpaam0XiUQ4fPhwi/tLSkrCF1
98gZiYGHTr1g1xcXGYO3cuUlJSIJE0XR2zpqYG0dHROH78OPz9/bW2VVZW4saNG9i0aRP8/Pw
07ffGSc0rr1Rgw76LyC4o12rvZGmOuZP88Jino4EiIyJTwWp0RGRIHIOImqZzkuji4gIXF5dW
+eZKpRLJycmIiorC2LFjAQCJiYkYPXo0Dh06hIkTJzY6Ji0tDdHR0aitrYWdXeP54dnZ2RAEA
UOGDGlyO2kTBAGquj/vEMprapF3vRxqQcCGfRdxp1KptX+PrjZYEDYQXVmKnohaCavREZEhcQ
wiakznJDEmJqbVvnlmZiaqqqowYsQITZudnR18fX1x+vTpJpPEY8eOISQkBPPmzcPkyZMbbc/
KyoKzszMTxBa4frsSW7/L0Fw5A4BqZR3e3/Frk/uP8OuGFyf4QGph9qhCJCIiIiKiR0zvh8mK
i4tRW1sLQRAA3H1GsLq6GmfOnMHzzz/foj4KCwsBAK6urlrtXbt2xc2bN5s8ZvHixQ/t/Eyy+/jL1790
IqleLkyZOIiorC8uXLERgvn374OnpaaCzJCJjoXOSGBsbC5lMhmXLluGnn36
CRCLBE088gV9++QW//Eyy+/QW//PILPvvssxb3VV9MRqlUahWWUSgUsLLS75m3bdu2QaFQwMbmbiUqb29v
VFVVYcOGDYiMjNT7bqJaLUAmk+t1bHuTW1CO3Gt3WrRvnVpATbUSZWVVbRsUkQ5M5YINERGge
42Gw4cPo7q6Gh988AGkUikAIC4uDmPHjkVaWhqCgoKwefNmhISEIDw8HiEBUV1Wibu7u
e77777aE+QiIyOzklieno6li9fjmnTpsHa2hrffPMNZsyYgRkzZmDROkXYsWMHAgICWRX/TT
ToqIi9OzZU9NeVFQEHx8fXUMDAFhYWMDCwkKrzcvLC3K5HOXl5bC3t9erXwBQNVgCwpidySzS
af/TGUXo1a3p8tBERET0cHSt0RAUFISPP/5YSPP/5YkyA2VF5eDrVajbS0NCxfvlxr2/Dhw3Ho0KG2O
QkiMik6J4lKpRK9e/cGAPTp0wdZWVmabWFhYVi1alWL+/Lx8YGNjQ1SU1M1SaJMJsPly5c1V7
50oVar8eSTT2LatGmIiIjQtF+4cAFOTk4PlSCaEnlNbZvuT0RERC2na40Gd3d3uLu7a7Vt2rQ
JUqkUgYGBkMlkkMvljarRP6jmQ0uZUn2GhhqWXBCJTPc8OwK+l61D5ySxe/fuuHbtGgICAtCr
Vy9UVlaioKAA7u7uWibu7ukEgkOj1LKJFIEB4ejvj4eDg4OMDNzQ1xcXFBSEgI6urqUFpaCltb2
xatcygWizF+/Hhs2bIFHh4eaWrdHw2W
efYdeuXVixYgUcHR01SWdT/SkUCr3jNKX6DPcSi0Var031PDsCvpetQ+ck8amnnkJ8fDysrKw
wYcIE9OnTB4mJiZg3bx6eSk5PRo0cPnfpbtGgRVCoVVq5ciZqaGgQGBmLmLr1q2QSCQoKChAh
YmJiEBYW1qL+lixZAjs7OyAjs7OyQkJKCWsBDu7u6Ijo7Ijo7G9OnTdT1VkzW4nxMOnLza8v29WLSGiIior
ehbo0EQBHz00UfsYsGED5s+fj5deegkeegkANNNQlUrttY4fpuYDYFr1Ge6lVgtar1mLwXh1pPeyLR
NgnZPEhQsX4urVq/j6668xYcIEIEr1WibU7u
71nTWe/3444+N2szNzREREaE13ZS09eluBw8XW/xWWNHsvr1dbdHHlWtOEhERtRV9ajTU1tZi
```

xYoVSElJwdKlSzFnzhzNti5dusDa2hpFRdo1CIqKihpNQdWVqdRnuJcgaL821fPsCPhetg6dk
0SpVIq1a9eitvbuc2qjR4/G/v37cenSJfj5+WkNbtQ+iUQizJnYHzE70yBX3H8ZDGupOWaH9u
faiERERG1InxoNS5cuxaFDh5CQkNCosI1IJMKQIUNw6tQpTJs2TdOempqKoUOHtt2JEJHJ0Dl
JrNewgmjPnj2ZHBoZN2cbrAgfgq3fZTR5R7G3qy1mh/aHm7ONAaIjIiLqOHSt0bBnzx4cOHAA
S5cuxbBhw3D79m1NX/X7zJo1C/PmzYOvry/GjBmDr7/+GhkZGXj//fcNeKZEZECx0ThJramqwd
u1anDx5EhUVFVCrtW/hikQiHD58uNUCpLbj5myDv78YgDfDfW/wCSmOGN5/zRx9WOdx
CJiIgeEV1qNKSkpAC4u3Z1bGysVj/1+4waNQrV69GUlISEhMT0bdvX2zcuBGenp6GOD0iMjI
6J4mrV6/Gv//9bwwZMgT9+vXTe3F6ah9EIhHMzf58D60tLeDZvbMBIyIiIup4dKnRkJyc3KI+
p0yZgilTprRWiiETUgeicJH7//feIjIzEggUL2iIeIiIiIiMiCdbwPW1tYiICCgLWIhIiIiI
iIiA9M5SRw9ejR+/vnnNgiFiIiIiIDK1F00337dunee3n54e1a9eiqKgIQ4cOhbW1daP9Of
+diIiIijIOLUoSVy+fHmjtu++w7aRSMQkYiIiIHjlXJ2uUyGoAAN2dOhk4GuP
VoiTxyJEjbR0HERERERHRQ3k+uB92CTkAgOeC+xo4GuPVoiTRzc1N6+s7d+7+7g3LlzePzxxwEA
165dw08//YQpU6D6bAzs6u1YMkIiIiIiJqjqtjJyx51t/QYRg9nQvX5Obm4umnn8a7776rabt
Tri4uIQFhaGgoKCVg2QiIiIiIHh2dk8TY2Fi4ubnhyy+/1LSNGDECR48ehZOTE+Li4lo1QC
IiIiIinp0dE4Sz507hwULFsDZ2Vmr3cHBAfPnz0dqamamqrBUdERERERERESPls5JokqgkQlVVVZP
blEolamtrHzooIiIiIiMgydk8Thw4cjKSkJpaWlWu2lpaXYuHEjhg8f3mrBERERERERER0aPV
ouqmDUVFReF///d/ERwcDH9/fzg4OKCsrAxnz56FVCrFmjVr2iJOIqI2o1arsX79euzevRsym
QxDhw7FqlWr0tKtXryb3z8nJQVxxcHNLT0yEWixEYGIjly5eje/fumn0OHjyIdevW4dq1a/Dw8E
BUVBTGjBnzqE6JiIiISG8630ns0aMHUlzz0Eul+PixYuQWR49tlnsW/fPvTu3bst4iQ
iajNJSUn44osv8N577+HLL7+ESCTC3LlzoVQqm+1TX169bxMSmoVOnOnTti5cyc2b96MsrIyvGZqu
y1AoFACAkydPHyPIioqCjNmzMC+ffswatQAHl5eY/61IiIiIiIh0pvOdRABwdGsmXLWWjZGbWsWI
qJHTqlUIjk5GVFRUgAIDExEaGoqDQHLh4cOKlqMbWa6r7n7fAExTwmwgQrf7GiOp3aHuMhO4PH
07FmlpaQgKKCsLmzZsREhKC8PBwAMCyZyw3r+7YXGGsRs5Pqk+Xaq+gZ/KEBeGcsNvJlnd10/gkQERER
tTK97iQSKgSqRmaWK6bWKRqMaggAAILzMxoWA9c3D3ee4zz860/gkQERER
0uL+tOFubnpXtcTibRfm/K5EhERtXdMEomoQ6uurgYAAIxIuqOnDEWAWC9uAyXNpFahehYUH5NDp
ixAo6Ojpqks6n+6p9Z1IdYLIK9fSe9j2/vxGKR1mtTPlciIqL2zuBJoq5vCi0KrRGVPHYWuxPjT:
yMlJrG6sKElFLWVpaArj7bGL9awBQKBSwsrK673GCIOCjjz7Chg0bMH/+fLz00ksAoJmGem/R
m+b6a45aLUAmk+t9fHunVgtar8vKml6Pl6i94IUMIjJlOieJK1aswLhx4zBy5EhYW1s/dAD1V
QVjYmLQrVs3xMXFYe7cuUhJSWl0Jb5eTU0oNoqOjcfz4cfj7+2ttq68quHz5cgQFBeGrr77Cgg
ULsG/fPnh6ej50vERkWuqnmRYVFaFnz56a9qKiIvj4+Dr5TG1tLVVasWIGUlBQsXboUc+bM0Wz
r0qULrK2tUVRUpHVMUVFRoymoulKp1A91fLHsmCNqvTflfIciYiI2judH/rIrZ3fB5DL8cZJEZ
Mweff/45rl+/rtc3r68qGBkZibFixFjx8LHxweJiYm4desWDh061Fn9KtLT02FnZ9doe
8Oqgp6enli2bBn8/Pywft2vWIkItPm4+MDGxsbpKamatpkMhkuX76qGU/o9ZunQp/u///g/g
8JCQlaCSIAiEQiDBkyBKdOndJqTQ01NxdCbQ1v/BIiIiIhamc5J4u7du/Gf//wH//o3A4Lj
+/Xo8o8+eSTmDx5MhIE3Hu3LkW96VrVUEUAEAOHbsGEJCQrBv3z7Y2Y2tpqbWNVNSLSLUQiQXh4OOLj
43HkyBFkZmbi9ddfh4uL4uLC0JCQlBXV4fbt2+jpqYGALBbnzxx4cOHAAr7/+OoYNG4bbt29r/tTvM
2vWLHz33z33XfYtm0b8vLyB8bsbi4yMDLz44ouGPFUiUiFtHrmUR7e3tMnjwZkydPiAIOHXqiAIOHXqFD
766CNs2rQJn3zzzCTIyMlrUj65VBauqqnhNdxj96VBanLasKAqZbbY9VBamjW7RoeEVQqFVauXImamho
EBgZi69gattkEgkKCgoQHBwMGJiYhAWFoaUlBQBQAQGxsLGJjY7X6d+vn1hRs95h4ghRWL16nKZcpQY
ffti48aNnPJORERERkGvJLJgpmZaQZT6mDhwoWt8/79/lAqFe4DB4/o3Lmz1rzEN1I+5h2KIEK
9/DVBUETLvaHqsKUkdnZmaGqKgoREVFREfNdrm7u6ZKwszVKa4VIREREX9M
jonCQ+99xzuHjxItRfr06YPRHvH1aAPwd4cPh729vUtVtgnqyWzRAYNqVBVlkTwNL2Q
QERERtR2dk8SsrCyRUNqM

QATExMxOjRo3Ho0CFMnDix0TFpaWmIjo5GbW1tk4/+ZGdnQxAEDBkyhI8GEZHOdF4n8YcffsB
LL72Eb7/9Fi+88AIGDBiA3bt344cffoCbmxvUai7ETkRERNRSmZmZqKqqwogRIzRtdnZ28PX1
xenTp5s85tixYwgJCcG+ffuafMwnKysLzs7OTBCJSC8630ksLS3VXOXy9vbGl19+CQDo1q0b5
s2bh23btvFuIhEREVELFRYWAgBcXV212rt27YqbN282eczixYsf2Gd2djasra0RGRmJs2fPws
HBAWFhYZg5cybEYp3vEWiYm+t/bHvWsNyGSGS650nUUjoniba2tlAqlQAADw8P3Lx5E5WVlbC
xsdF8TUREREQtU11dDQCNnj2USqUoLy/Xq8+cnBxUVFQgNDQUCxcuxJkzZxAfH4/y8vJmE8z7
EYtFsLfvpNex7Z1YLNJ6barnSdRSOieJAQEB2LFjBwIDA+Hu7g4rKyscOnQIU6dOxdmzZ2FjY
9MWcRIRERGZpPpiMkqlUquwjEKhgJWVlV59btu2DQqFQvO5zNvbG1VVVdiwYQMiIyP1upuoVg
uQyeR6xdPeqdWC1uuysioDRkPUMm15MUPnJHHBggUIDw/H/PnzsWPHDsyYMQNvvvfUWduzYgay
sLDz//PNtEScRERGRSaqfZlpUVISePXtq2ouKiuDj46NXnxYWFrCwsNBq8/LygkqlQllZGTR
7fXqV6UyzdoTgqD92lTPk6ildE4SfXx8cPDgQWRnZwMAlixZAhsbG6SlpWHcuHGYN29eqwdJR
EREZKp8fHxgY2ODD1NRUTZIok8lw+fJlhIeH69yfWq3Gk08+iWnTpiEiIkLTfuHCBTg5OemdIB
JRx6Fzkvj222/jmWeewciRIwEAIpEIr7zySqsHRkRERNQRSCQShIeHIz4+Hg4ODnBzc0NcXBx
cXFwQEhKCuro6lJaWwtbWtkXrHIrFYowfPx5btmBh4cH/Pz8cOLECWzZsgXR0dGP4IyIyNjp
nCTu378f48ePb4tYiIiIiDqkRYsWQaVSYeXKlaipqUFgYCC2bt0KiUSCgoICBAcHIyYmBmFhY
S3qb8mSJbCzs0NCQgIKCwvh7u6O6OohoTJ8+vY3PhIhMgc5J4sCBA/HLL78gKCioLeIhIiIi6n
DMzMwQFRWFqKioRtvc3d2RlZV132N//HHHRm3m5uaIiIjQmm5KRNRSOieJ3t7e2LFjB77//
07dsXjo6OWttFIhFWr17dagESERERHRo6NzknjoCF07doVAJCbm4vc3Fyt7aKGq5ESERER
ERGRUdE5SWxqSgMRERERERGZBt1XUiUiIiIiIiKTxSSRiIiIiIiINJgkEhERERERkQaTRCIiI
iIiItJgkkhEREREREQaTBKJiIiIiIhIg0kiEREREREERaRg8SVSr1Vi7di1Gjx6NQYMGYfbs2b
h69ep99y8rK8OSJUsQGBiIwMBA/P3vf4dcLtfaZ9y4cfD29tb68+abb2+abb7b1qRAREREREERk9c0M
HkJSUhC+++AIxMTHo1q0b4uLiMHfuXXKSkpEAikTTaf9GiRVAoFPj0008hk8kQHR2Nd955Bx9+
+CEAoLKyEjdu3MCmTZvg5+enOc7S0vKRnRMREREREZGxMuidRKVSieTkZERGRmLs2LHw8fFBY
mIibt26hUOHDjXa/+zZsh16hRiYmLg5+eHoKAgvPvuu/jmm29w69YtEB2dyYEQcCQIUPg7O
ys+WNra/uoT4+IiIiIiIMjoGDRJzMzRFVVUFUaMGKFps7zg6+vL06fPzzs7DZ2Rmenp6
atmHDhkEkEuHXX38/38FAGRlZcHZ2CTZ2Rl2dnZtfwJEREREREQmxqDTTQsLCwEAHr16abV37d
oVN27caLT/rTRtc0F7Pv+HlUi+odJBJ06dJFs392djaJEs3sEmp95tRvtKJBJ06dJFs392djasra0RGRmJs2fPws
HBAWFhYZg5cybE4ofLic3NDf4IZ5sQi
bRfm+p5EhERERFR8wyaJFZXVwNAo2cPpVIpysvLm3zE7O/D4u3c9OFmqtoU1eyalWAR7+056H9+eicUirdemep5EREREREERNQ8gyaJ9cVklEqlVmEZhUIBKys3s1/veK6fU
ULcebMGcTHx6O8vLOL2K6JLXiV8mwJmVm8pC//UpxLw6VPvzBmTkfJUfKyurJvdXKpWN2hUKBaytrQEA27ZtgOKhgI2NDA29sbVVVV2LBhAyIjI/W+m6hWC5DJ
5M3vaITUakHrdVlZlQGjIWoeL2QQEREREtR2DJon1U0eLiorQs2dPTXtRURF8fHwa7e/i4oLDhw9rtSmVSty+fRvdunUDAJibm8Pc3FxrHy8vLwBAYWFho2PvJcJRTrw09oq7pGl4FRF8fHwa7e/i4oLDhw9r
w9rtSmVSty5cwfdunUDAFhYWMDCwkJrHy8vL8jlcpSX8Pe3l7veFUqtd7HtmeCoP3aVM+TiI
iIiIiaZ9CHz3x8fGBjY4PU1FRNm0wmw+XLl+Ls9mX0QNEfOZ0Rcm4Ra7+/7JAhQ6BWzFu3Dh
s2LBB67gLFy7I7AycnpoRJEIiIiIiKijsCgdxIlEgnCw8MRHx+sHuLm5IS4uDi4uLggJCUFdX
XR1KS0tha2vborUORSIRxo8fjy1btmBh4cH/Pz8cOLECWzZsgXR0dGP4IyIyNjpnCTu378f48ePb4tYiIiIiDqkRYsWQaVSYeXKlaipqUFgYCC2bt0KiUSCgoICBAcHIyYmBmFhY
jhx4gS2bNmC6Oho54qERERERGRUTBokggAixYtgkqlsqVwEBs3boVEEokFA+UFCA
4ORkxMDMLCwiASibB+/Xq88847ePHFFyGVSjFhwgSsWLFC09+SJUtgZ2eHhIQQEFBYWwt3dHdH
R0Zg+fboBz5KIiIiIiMg4GDxJNDMzQ1RUFKKiohpc3d3R1ZWllabo6Mj1q5de9/+zM3NERER
gYiIiFaPlYiIiIiIyNRxQTwiIiIiIiLSYJJIREREREREGkwSiYiIiIiISINJIhEREREREWkwS
SQiIiIiIiINJolEREREREESkwSSRiIiIiIiIiINJgkElGHp1arsXbtWowePRqDBg3C7NmzcfXq
YdN2fOHKxbt67RtnHjxsHb21vrz5tvvtvtkW4RMREREG1KiaJRNThJSUl4vvsvsB7772HL7G7
RCHPnzoVSqbzvMT1NYiKisLx8GD9HD9+XNn1apVbXkaRGTE2uJi1cGD
BxEaGoqBAWdi0ORmkJMaOHQsfU5QlJ+UyqM1RJ+OWXX9HYDV9oidCIyQUwSiahDUyqVSDUyqVSE5ORmRkJG53TWRKwOcoBwRS
JPHpKWlYerUqhPT4ednV2j7dnZ2RAEAaWlpW0WJxEZF66JSEQdWmZmJqqqqjBixAhNm52dHXx9
wZ2LdvH0aN0aUaysv++vv/4KDw8P2NjYNHn8unXr8MknnyA8PBxjx45FTk4O1q9fjxs3buh95HoGVUgQxOGDQsOrJiojEgLpjgAaZKSFjYwIlrw6kY
VMYGaMATeLv4q6h3OudUtRrq8O4qUAM0mJNjGQ+3oo+6nKrSJHJ+YDQEc3NTtQ
HEBISgn379F1zzSq3moe/qvq5R9wh4QTV4ahGe5AYq16cMoy9hndnY2rK2tERkZibNnz8LB
RGQCzA0dABGRITY2NwgMDAy8ndJEJNgO1zw3NkzQ9wKIUAL3VGAJ5aWDVeFgE8F49uTLaSTmCr
WGYOXMmxGL9r82Zm5vudT2RSPuulKZ8r0b2au1g1ceLERsfUX6yaN28eJk+rLVNrVYjLS0NyR2
cv12ofPnz4fZNOIqKGmCQSUYdWXzSmW7duAPDg4ICCggJtFSoR5nIKJ42nUeZnxX0ZTbIA11
i4+NRXl7ebtubH5XM5HC4uXbGfxtLb6DBPP8n1LVrZ9Lm9ULOkxfqHsqlIaZ8qrR9Y+VVF2QVmK1GD
aWOSSEQdmqWWlJYC70wqXwOAQqGAIX9qVmbd8pbvFN2Ak2bw9VVNWFDC7YqshOq4uZyO0SwoPU/YJ
lotQCaT6xWPMVCrBa3XZWVVBoyGqHm8kEFEREREtB2DJon1U0eLiorQs2dPTXtRURF8fHwa7e/i4oLDhw9r
7KfVFREXr27Klpl1yoqgo+Pj159WeXlsLe316tflUFPOQ5D
yuRLdq7UvVkmlUk1/DT3MxS/AtC9W8UIVGaO2vJjBJJGIOjqQfHx/Y2NggNTVwNTVTVkyTKZDZDJcvnw
Yd3gbmsbhQkPoMs3zc

4eHhOvenVqvx5JNPYtq0aYiIiNC0X7hwAU5OTnoniERkulr7YlWXLl1gbW2NoqIirfaioqJGU
1B1ZaoXcHihikgbJ1wTUYcmkUgQHh6O+Ph4HDlyBJmZmXj99dfh4uKCkJAQ1NXV4fbt25rpW8
0Ri8UYP348tmzZgoMHD+L333/Hl19+iS1btuj9PCIRmbaGF6vq1V+sCggI0Lk/kUiEIUOG4NS
pU1rtqampGDp06EPHS0Smj3cSCa5O1iiR3f0A3N2Jc/Cp41m0aBFUKhVWrlyJmpoaBAYGYuvW
rZBIJCgoKEBwcDBiYmIQFhbWov6WLFkCOzs7JCQkoLCwEO7u7oiOjsb06dPb+EyMF8ch6sgaX
qxycHCAm5sb4uLitC5WlZaWwtbWVms66oPMmjUL8+bNg6+vL8aMGYOvv/4aGRkZeP/999v4bI
wTxyAibSJBaHiDne6nrk6N0lLTnJ9+s6QKuw7nAABmPNkPro4cHKl9c3bueIvSm/IYBHAcIuP
T2uNQXV0d1qxZgz179muVr31lltwd3dv9mLVuHHjMHXqVERGRmq179u3D0lJSSgsLETfvn0R
FRWFoKCgh4jRdMchjkFkjNry8xCTxBYy5YGRyNgwSSQiQ+M4RESG1pbjEJ9JJCIiIiIiIg0mi
URERERERKTBJJGIiIiIiIg0mCQSERERERGRBpNEIiIiIiIi0mCSSERERERERBpcAqOFBEGAWs
0fFVF7YGbW8a5vcQwial84DhGRobXlOMQkkYiIiIiIiDQ63mUwIiIiIiIiui8miURERERERKT
BJJGIiIiIiIg0mCQSERERERGRBpNEIiIiIiIi0mCSSERERERERBpMEomIiIiIiEiDSSIRERER
ERFpMEkkIiIiIiIiDSaJRERERERREpMEkkYiIiIiIiDSYJBIRERERREZEGk0QjJggC9u7di5KSk
vvuk5qaCm9vbxQUFDzCyKiet7c39uzZY+gwmvXTTz8hNzfX0GGQEeI41P5xHCJTx3Go/eM4ZH
yYJBqx06dPY/ny5aiurjZ0KHQfx8fR2hoqKHDeKDr16/jlVdeeeB/rkT3w3Go/eM4RKaO41D
7x3HI+JgbOgDSnyAIhg6BmuHs7GzoEJrF3yN6GPz9af84DpGp4+9P+8dxyPjwTmI7l5OTg1df
fRXDhw/HgAEDEBISgu3btyM1NRUzZ84EAAQHBzd7C7C//Z8EAAQHBzd7C//o0aOYNGkSBgwYgIkTJ+Lnn3/WbKurq
8Onn36K8ePHY+DAgRg/7jzanpqbC19cXJ0+eRGhoKAYOHIhnn30W+fn52LBhA/7yl7
9g2LBh+Mc//qH1D+ynn35CWFgYHnvsMYSEhOCf//wnlEpl6/6A2rmG0yuWL1+OqKgofPjhhwg
KCsKgQYPw6quv4vbt2w/sQy6X2N47333sOoUaMwePBg/O1vf8P58+c129PSvC3v/0Njz32GB5/
/HG88847qKys1GwfN24cPvnkE0RGRmLw4MEYPnw4Vq9eZs6ciXXXr1gEA8
vLyMHfuXAwePBijRo3CkiVLtOJ84YUX8P/+3//DtGnTEBAQgH379rXWj4zaIYCjk
PGjeOQERKo3ZLL5cLIkSOFJUuWCLm5ucJvv/0mJCQkF5eXkJ6errw/fffa15XV1c32cfJkyc
FLy8v4amnnhJOnjwpXLlyRXj11VeFQYMGCZWVlYIgCMJ7770nBAYGCp/9uv7Ny5U/Dz
8xM+++wzrT6eeeYZ4ezZs8Lly5eF4OBgYdiwYcKbRo/nhtRNeXl7C119/LQiCIOq/MTli9fLuTm5gq//Ud/
GvXLuHq1avCsWPHhKeekpYtGjRo/nhtRNeXl7C119/LQiCICxbtkzw8/MTli9fLuTm5gq//P
KLMGzYMGHFihUP7GevHnCuHHjhKNHjwq//fabEB0dLQQEBAglJSQRkaGMHDgQOHjjz8W8vP
zhdOnTwvTpk0Tpk2bJqjVakEQBGH6+WPi9A0

```
oi4X4TcYmIiIiIiKjD4XRTIiIiIiIi0mCSSERERERERBpMEomIiIiIiEiDSSIRERERERFpMEk
kIiIiIiIiDSaJRERERERERpMEkkYiIiIiIiDSYJBIRERERERZEGk0QiIiIiIiLSYJJIRERERERE
GkwSiYiIiIiISINJIhERERERERWn8fygGGuzYsZ9CAAAAAElFTkSuQmCC",
      "text/plain": [
       "<Figure size 912.222x600 with 6 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "df_ci = pd.DataFrame.from_dict(res_dict, orient='index')\n",
    "df_ci.to_csv('ci_change_score.csv')\n",
    "\n",
    "g = sns.FacetGrid(df_ci, col='task name',  col_wrap=3, sharey=False,
margin_titles=True)\n",
    "g = g.map(sns.pointplot,'program type','value', dodge=True)\n",
    "g.set_axis_labels('', 'raw change score')\n",
    "g.add_legend()"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "base",
   "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": "3.9.7"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```