In [1]:

```
# import datetime
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datlib.FRED import *
from datlib.plots import *
import pandas_datareader.data as web


#FRED.py
#. . .
def bil_to_mil(series):
    return series* 10**3
# . . .
#fedProject.py
# . . .
data_codes  = {# Assets
               "Total Assets": "WALCL",
               # Liabilities
               "Total Liabilities" : "WLTLECL",
               # Interest Rates
#                "CPI":"CPIAUCSL",
               "Core PCE":"PCEPILFE",
               "Currency in Circulation": "WCURCIR",
#                "VIX": "VIXCLS",
               }

rate_codes = {"Effective Federal Funds Rate (%)": "DFF",
#               "Federal Funds Target Rate (Pre-crisis)":"DFEDTAR",
#                "Federal Funds Upper Target":"DFEDTARU",
#                "Federal Funds Lower Target":"DFEDTARL",
#                "Interest on Reserves (%)": "IOER",
#                 "5 Year Forward Rate": "T5YIFR",
               "Unemployment Rate": "UNRATE",
}


inflation_target = 2

unemployment_target = 4.5
# Select start and end dates
start = datetime.datetime(2000, 1, 1)
end = datetime.datetime.today()

## year variable automatically adjusts the numper of periods
#   per year in light of data frequency
annual_div = {"Q":4,
             "W":52,
             "M":12}
### choose frequency
freq = "M"
### set periods per year
year = annual_div[freq]
```

In [2]:

```
rate_keys = list(rate_codes.keys())
diffs = ["Diff", "Diff-in-Diff"]

# freq refers to data frequency. Choose "D", "W", "M", "Q", "A"
# a number may also be place in front of a letter. "2D" indicates
#       alternating days

if "data_gathered" not in locals():
    data = gather_data(data_codes, start,
          end = end, freq = freq)
    rate_data = gather_data(rate_codes, start,
          end = end, freq = freq)
    # transform bil to mil
    data["Currency in Circulation"] = data["Currency in Circulation"].mul(1000)
    data.fillna(0, inplace=True)
    log_data = np.log(data)
    log_diff_data = log_data.diff(year)
    log_diff_data[rate_keys] = rate_data[rate_keys]

    # calculate monthly rates as well. This data will be used for ADF and KPSS te
    monthly_log_diff_data = log_data.diff()
    monthly_log_diff_data[rate_keys] = rate_data[rate_keys]

    data = log_diff_data
    monthly_data = monthly_log_diff_data
    data_gathered = True
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py:402:
RuntimeWarning: divide by zero encountered in log
  result = func(self.values, **kwargs)
```

In [3]:

```
for df in [data,monthly_data]:
    df["Currency in Circulation / Total Assets"] = df["Currency in Circulation"].
    df["Inflation Loss"]= df["Core PCE"].sub(inflation_target)
    df["Unemployment Loss"]= df["Unemployment Rate"].sub(unemployment_target)
    df["Inflation Loss Sq"]= df["Inflation Loss"].pow(2)
    df["Inflation Loss Sq"][df["Inflation Loss"] < 0] = df["Inflation Loss Sq"][d
    df["Unemployment Loss Sq"]= df["Unemployment Loss"].pow(2)
    df["Unemployment Loss Sq"][df["Unemployment Loss"] < 0] = df["Unemployment Lo
    df["Loss Function"] = df["Inflation Loss Sq"].sub(df["Unemployment Loss Sq"])
```

In [4]:

```
data={"Data":data,
     "Diff": data.diff(year),
     "Diff-in-Diff": data.diff(year).diff(year),
     }
monthly_data={"Data":monthly_data,
     "Diff": monthly_data.diff(),
     "Diff-in-Diff": monthly_data.diff().diff(),
     }
diffs = list(data.keys())
for key, val in data.items():
    data[key]["Date"] = val.index.astype(str)
```

In [5]:

```
# pd.set_option('display.max_rows', None)
# pd.set_option('display.max_columns', None)
# data.to_parquet("FedDAGData.parquet.gzip", compression = "gzip",
#                      engine = "pyarrow")
# data.to_csv("FedDAGData.csv")
```

In [6]:

```
# import pandas as pd
# data = pd.read_parquet("FedDAGData.parquet.gzip")
```

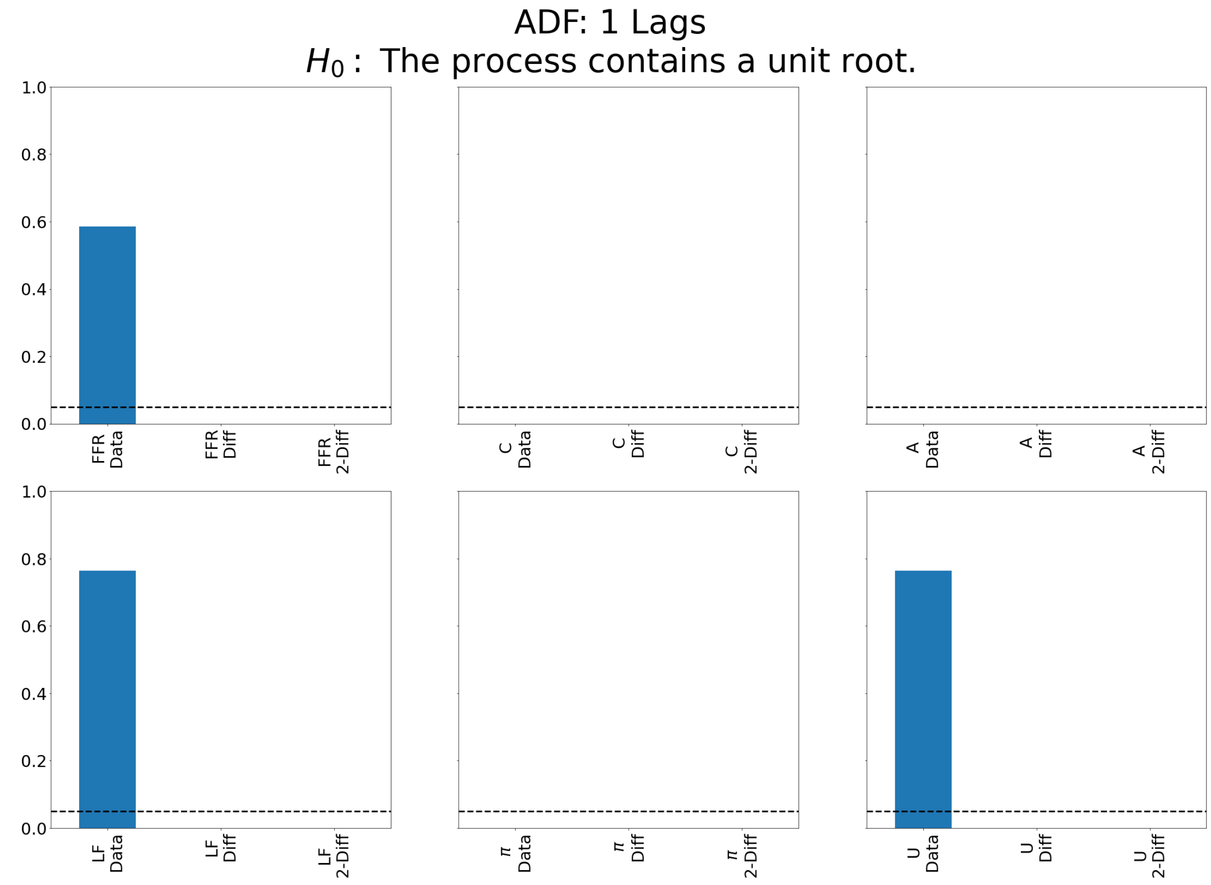
In [7]:

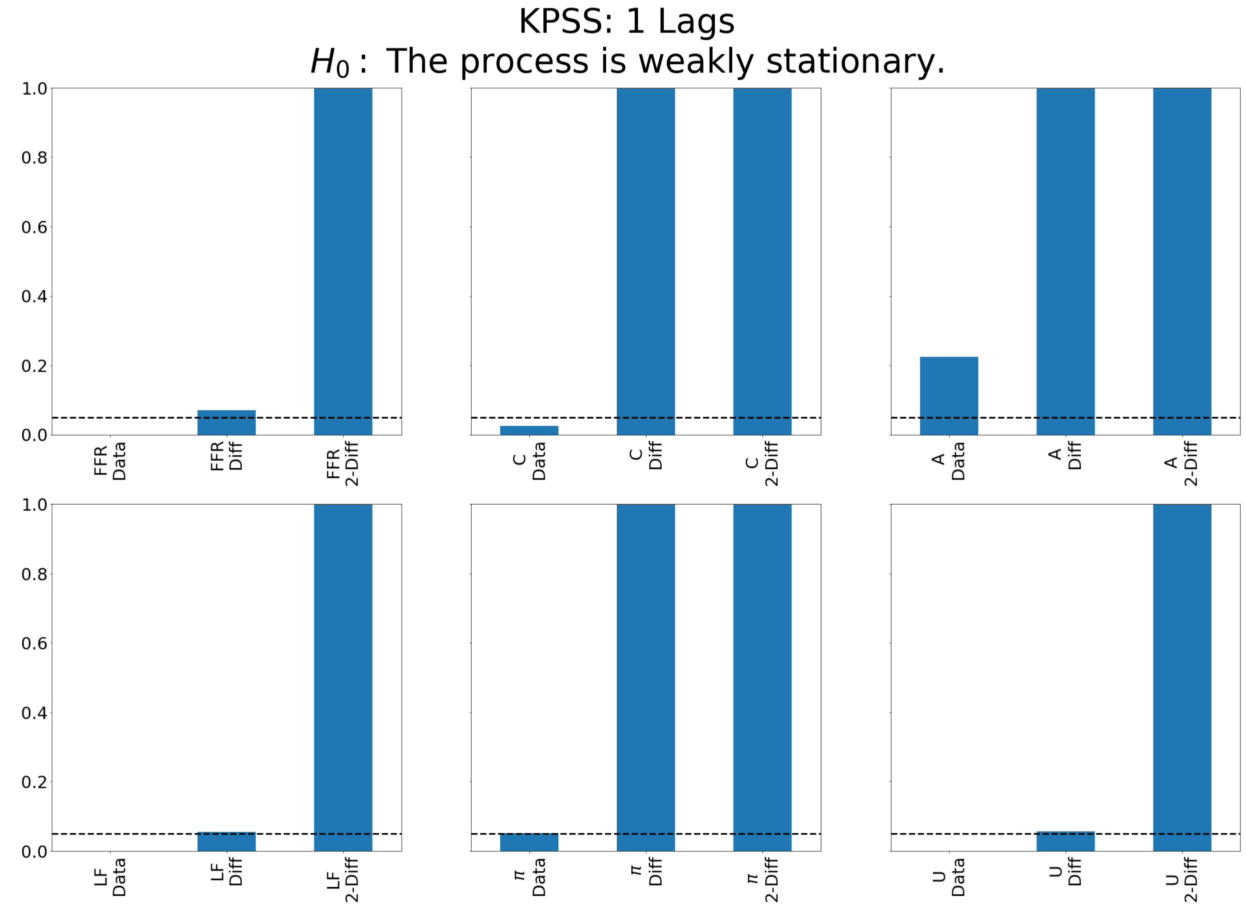
```
from statsmodels.tsa.vector_ar.vecm import coint_johansen
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.api import VAR
import statsmodels.api as sm
from arch.unitroot import ADF, KPSS
import copy
import pingouin
from scipy.stats import pearsonr
from datlib.ts_tests import *
from statsmodels.tsa.adfvalues import *
import warnings
warnings.simplefilter("ignore")
import statsmodels
from statsmodels.tools.validation import (
    array_like,
    bool_like,
    dict_like,
    float_like,
    int_like,
    string_like,
)

from statsmodels.tools.sm_exceptions import (
    CollinearityWarning,
    InfeasibleTestError,
    InterpolationWarning,
    MissingDataError,
)
```

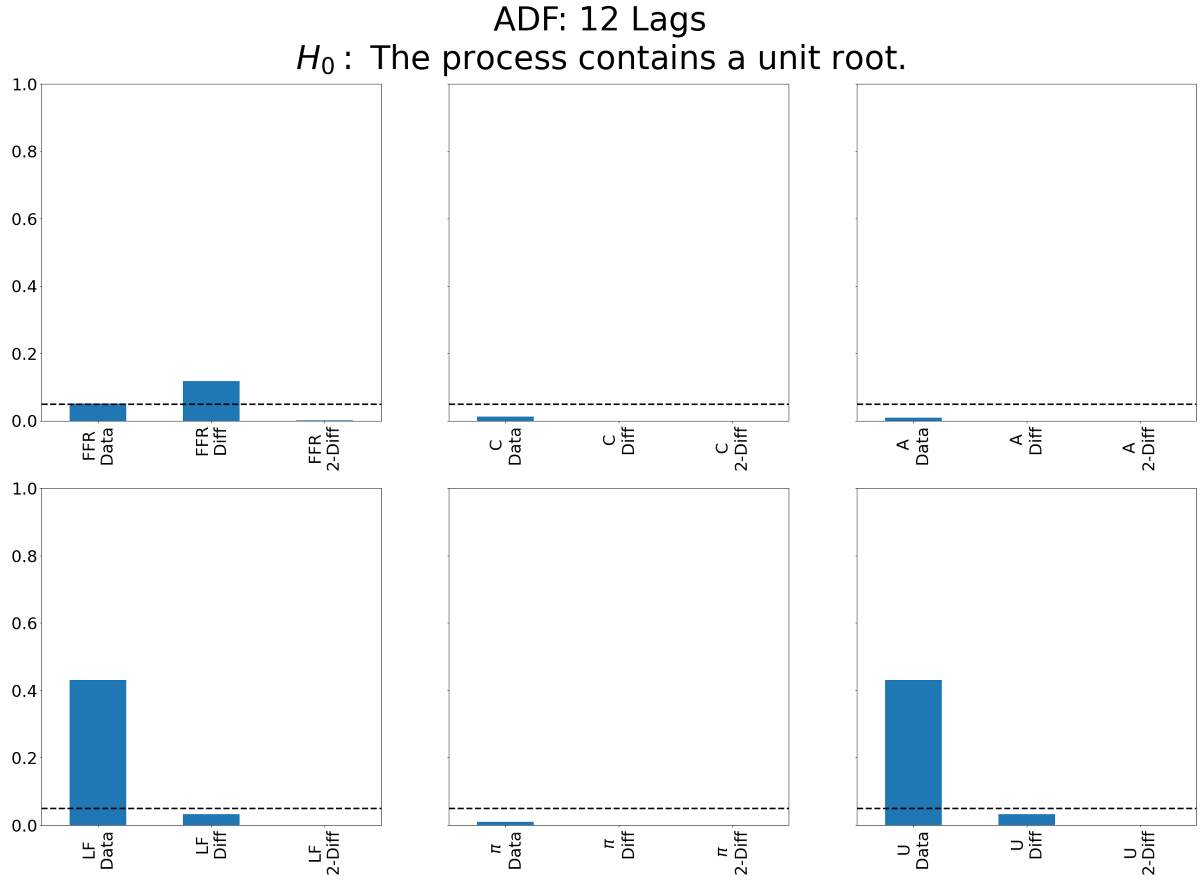
**For the unit root tests, we use monthly data since data frequency must match lag frequency for these tests. Since the annually differenced data covers data of 12 months, we simply test if there is a unit root 12 months out. We infer from the tests using 12 monthly lags that we can use the annual lags and annual rates for our DAG estimates.**

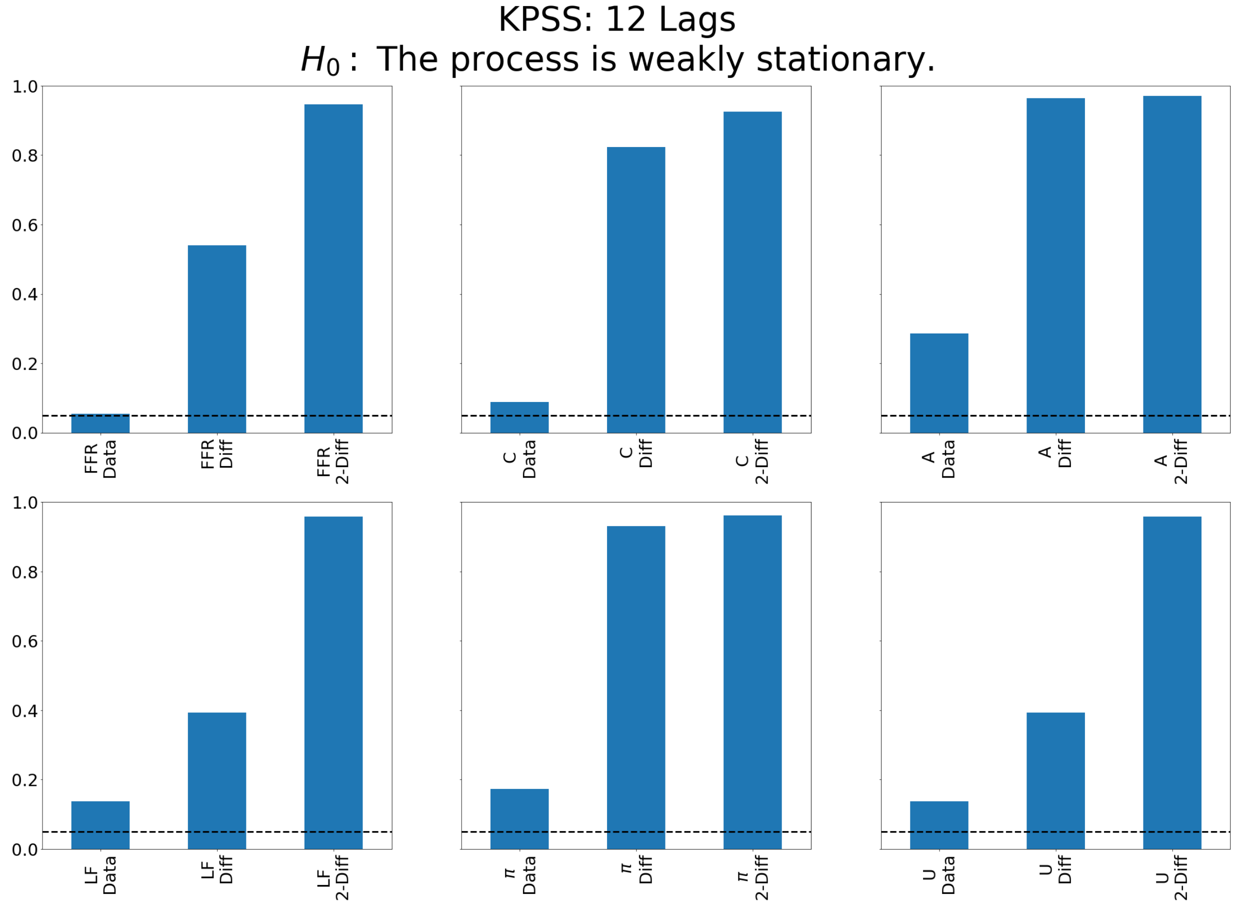
In [8]:

```
def run_ts_tests(data_dct, lags, adf_dct = {}, kpss_dct = {}):
    def build_result_dcts():
        for key in keys:
            adf_dct[key] = {}
            kpss_dct[key] = {}
            for diff in diffs:
                adf_dct[key][diff] = {}
                kpss_dct[key][diff] = {}
        return adf_dct, kpss_dct

    diffs = list(data_dct.keys())
    keys = list(data_dct[diffs[0]].keys())
    build_result_dcts()
    for diff in diffs:
        test_data = data_dct[diff][list(rename_dct.values())]
        test_data.dropna().to_csv("TestData"+diff+".csv")
        test_data = test_data.loc[:"2020-02-29"].dropna()
        for key, val in test_data.items():
            adf_dct[key][diff] = ADF(val,
                                     lags = lags,
                                     trend= "c").pvalue
            kpss_dct[key][diff] = KPSS(val,
                                     lags = lags,
                                     trend= "c").pvalue
    return pd.DataFrame(adf_dct), pd.DataFrame(kpss_dct)

def bar_plots(dct, width = 2, length = 3,title = "", title_y = 1):

    fig, ax = plt.subplots(width,
                           length,
                           figsize = (38,25))
    i = 0
    j = 0
    for key, df in dct.items():
        df.plot.bar(ax = ax[j][i],
                    legend = False)
        xtick_labels = ax[j][i].get_xticks()
        ax[j][i].axhline(.05, ls = "--", color = "k", linewidth = 3)
        ax[j][i].set_xticklabels([key + "\n" + diff.replace("Diff-in-Diff","2-Di
                                  fontsize = 30)
        ax[j][i].set_ylim(0,1)

        if i == 0:
            ax[j][i].set_yticklabels([round(y,2) for y in ax[j][i].get_yticks()],
                                  fontsize = 30)
        else:
            ax[j][i].set_yticklabels(["" for y in ax[j][i].get_yticks()])

        i+=1

        if i == length:
            i = 0
            j += 1
#         if i == 2 and j == 1:
#             ax[j][i].set_axis_off()
    fig.suptitle(title, y = title_y, fontsize = 60)
    fig.savefig(title.replace(":", "-").replace("$","").replace("\n","") + ".png'



rename_dct = {"Effective":"FFR",
#               "Circulation / Total":"C/A",
              "Circulation":"C",
             "Assets":"A",
              "Function":"LF",
             "Inflation":"$\pi$",
             "Unemployment":"U"}
abbrev_keys = list(rename_dct.values())
lags = 1
# lags = year




for diff in diffs:
    # test all variables, include loss fucntion components
    test_vars = ["Effective Federal Funds Rate (%)",
        "Currency in Circulation",
        "Total Assets",
        "Loss Function",
        "Unemployment Loss Sq" ,
        "Inflation Loss Sq"]
#     adf_data = monthly_data[diff][test_vars]
    for key in test_vars:
        for rename_key in rename_dct:
            if rename_key in key:
                monthly_data[diff].rename(columns={key:rename_dct[rename_key] for
                                inplace = True)

test_data = copy.copy(monthly_data)
for diff in test_data.keys():
    test_data[diff] = test_data[diff][abbrev_keys].dropna().loc[:"2020-02-29"]
for lags in [1,12]:
    adf_df, kpss_df = run_ts_tests(test_data, lags)
    # plot_ADF tests
    # create filler val to get null hypothesis
    val = [i for i in range(10)]
    title = "ADF: " + str(lags) + " Lags\n$H_0:$ "+ ADF(val,
                       lags = lags,
                       trend = "c").null_hypothesis
    title_y = .96
    bar_plots(adf_df, title = title, title_y = title_y)

    # plot KPSS tests
    title = "KPSS: " + str(lags) + " Lags\n$H_0:$ "+  KPSS(val,
                           lags = lags,
                           trend = "c").null_hypothesis
    bar_plots(kpss_df, title, title_y = title_y)
```


ADF: 1 Lags
$H_0:$ The process contains a unit root.
1.0
0.8
0.6
0.4
0.2
0.0
FFR Data
FFR Diff
FFR 2-Diff
C Data
C Diff
C 2-Diff
A Data
A Diff
A 2-Diff
LF Data
LF Diff
LF 2-Diff
π Data
π Diff
π 2-Diff
U Data
U Diff
U 2-Diff

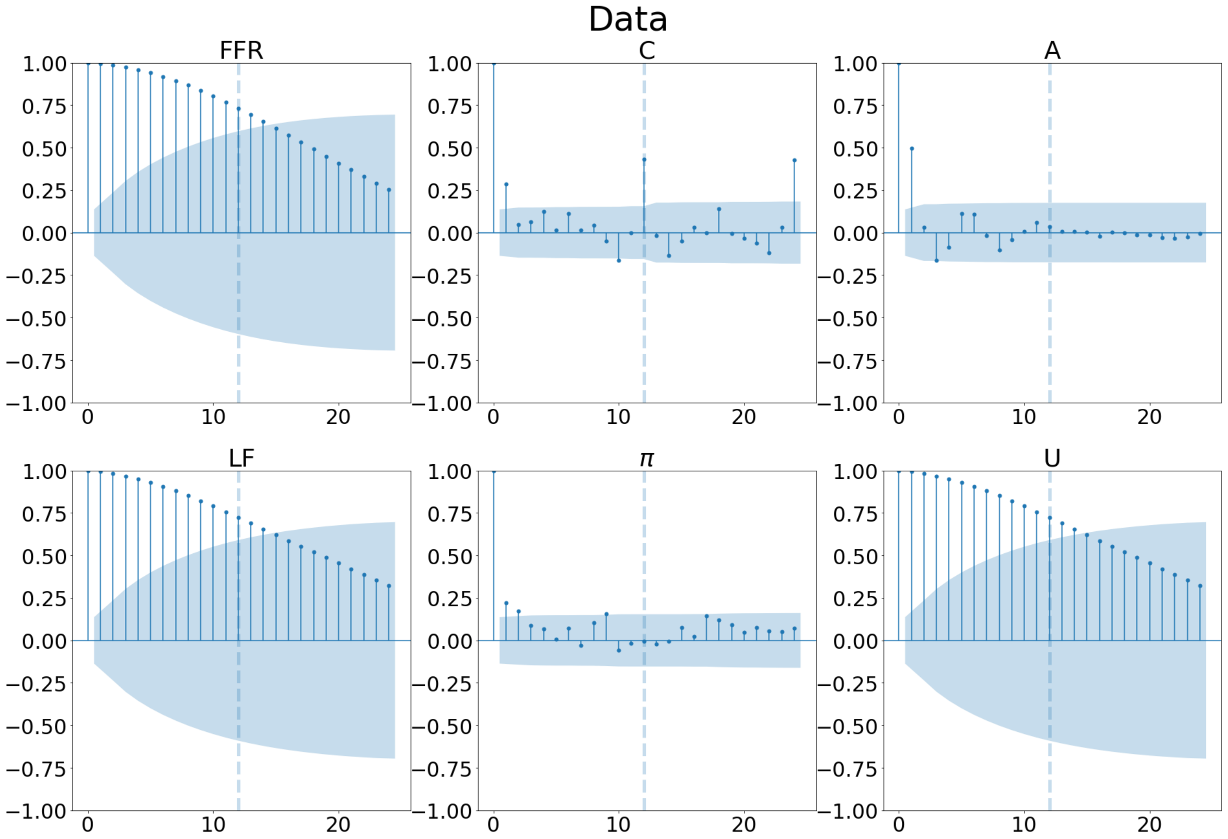

KPSS: 1 Lags
$H_0:$ The process is weakly stationary.
1.0
0.8
0.6
0.4
0.2
0.0
FFR Data
FFR Diff
FFR 2-Diff
C Data
C Diff
C 2-Diff
A Data
A Diff
A 2-Diff
LF Data
LF Diff
LF 2-Diff
π Data
π Diff
π 2-Diff
U Data
U Diff
U 2-Diff

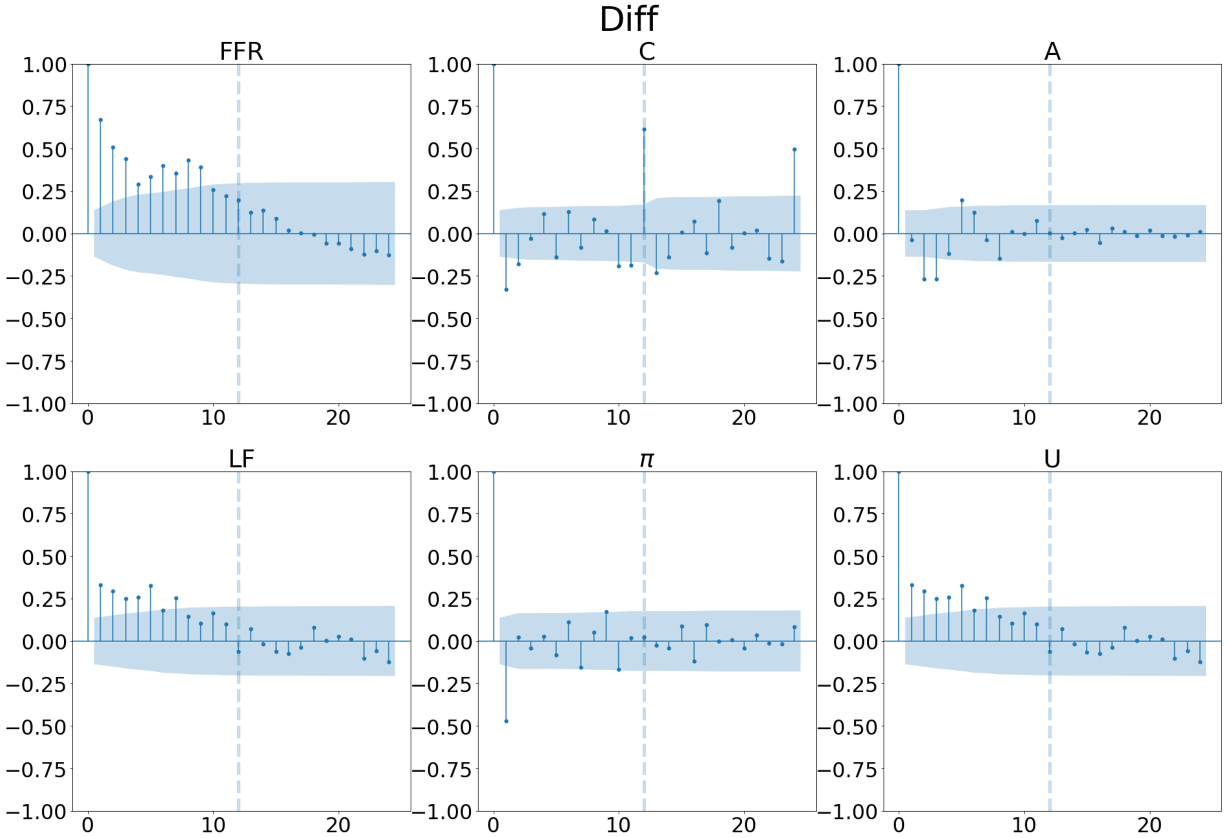

ADF: 12 Lags
$H_0:$ The process contains a unit root.
1.0
0.8
0.6
0.4
0.2
0.0
FFR Data
FFR Diff
FFR 2-Diff
C Data
C Diff
C 2-Diff
A Data
A Diff
A 2-Diff
LF Data
LF Diff
LF 2-Diff
π Data
π Diff
π 2-Diff
U Data
U Diff
U 2-Diff

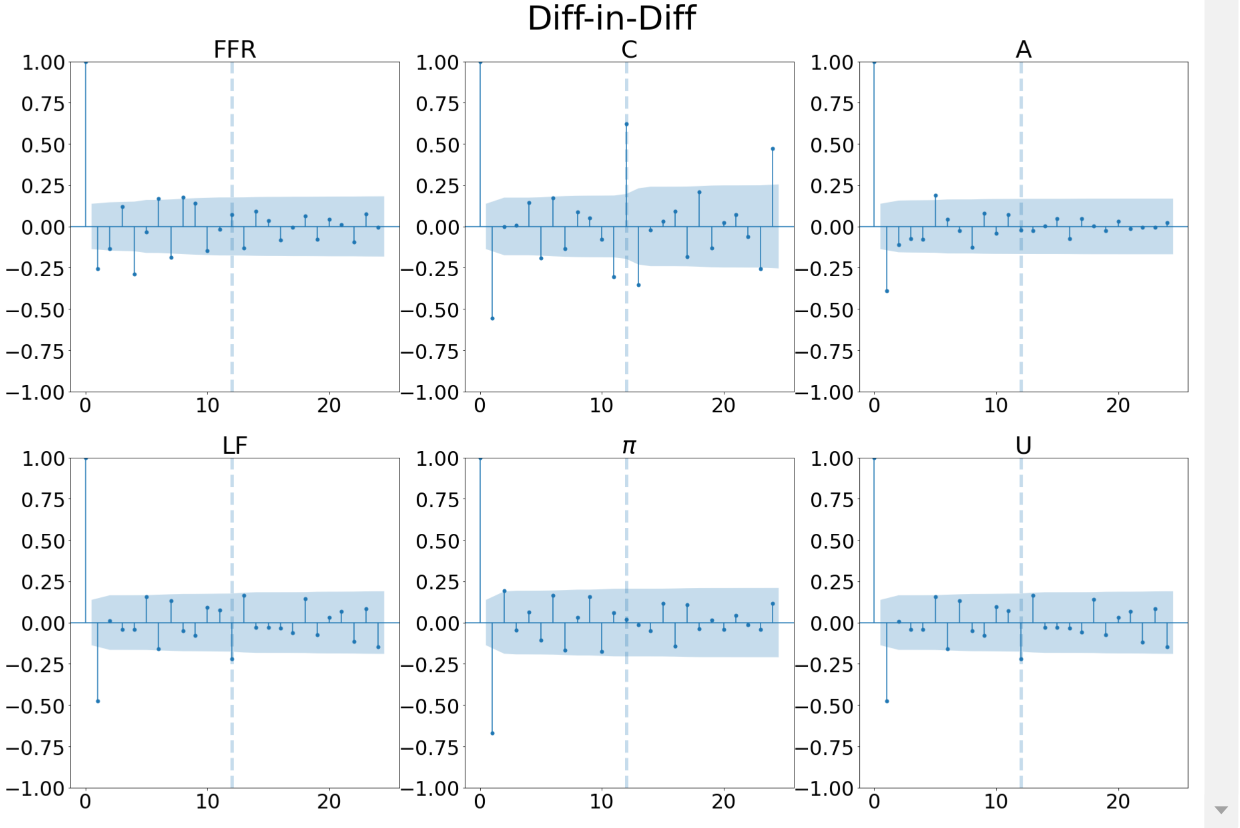

KPSS: 12 Lags
$H_0:$ The process is weakly stationary.
1.0
0.8
0.6
0.4
0.2
0.0
FFR Data
FFR Diff
FFR 2-Diff
C Data
C Diff
C 2-Diff
A Data
A Diff
A 2-Diff
LF Data
LF Diff
LF 2-Diff
π Data
π Diff
π 2-Diff
U Data
U Diff
U 2-Diff

## ACF plots by difference

In [9]:


Data
FFR
C
A
LF
π
U
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20


Diff
FFR
C
A
LF
π
U
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20

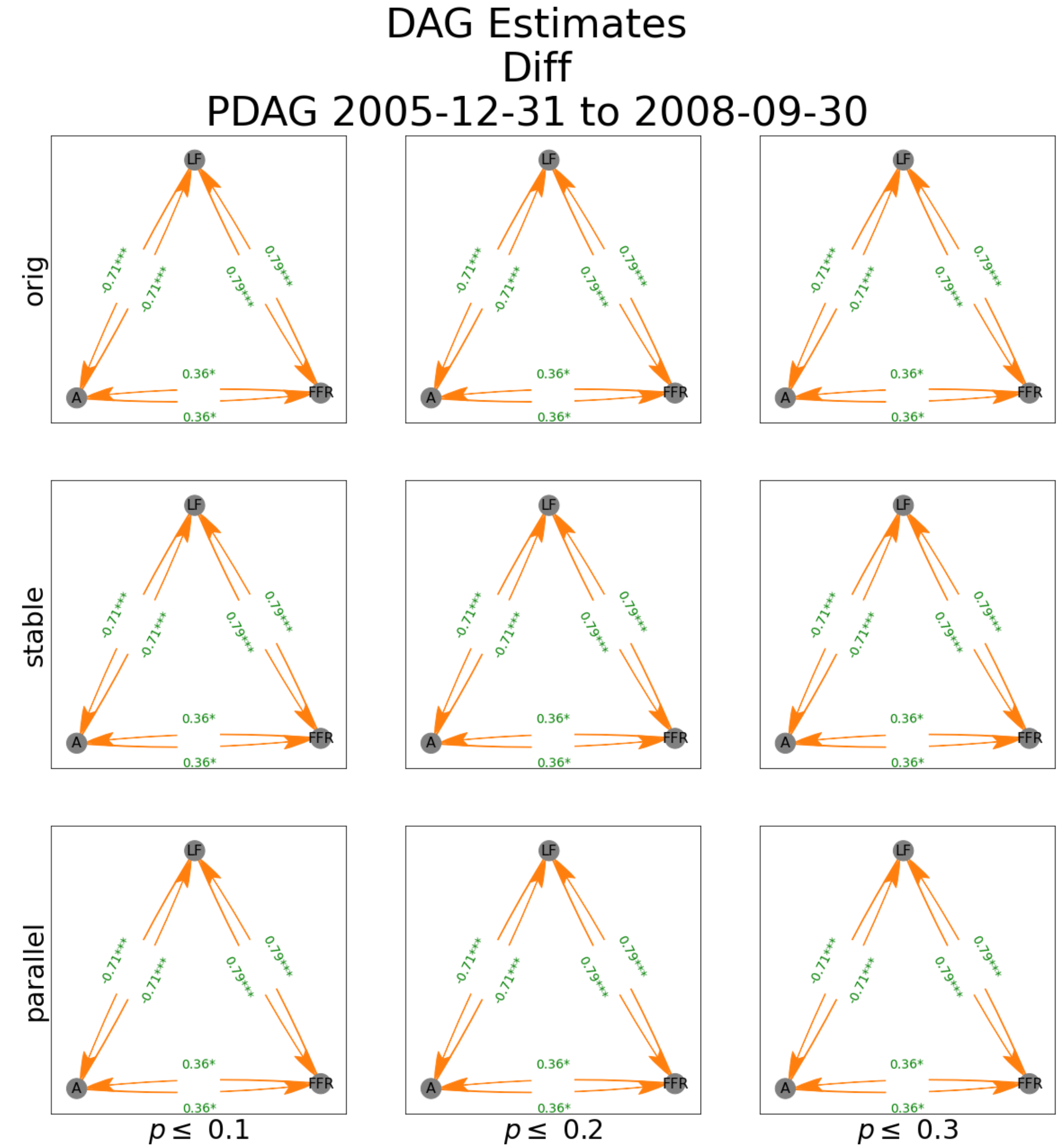

Diff-in-Diff
FFR
C
A
LF
π
U
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20

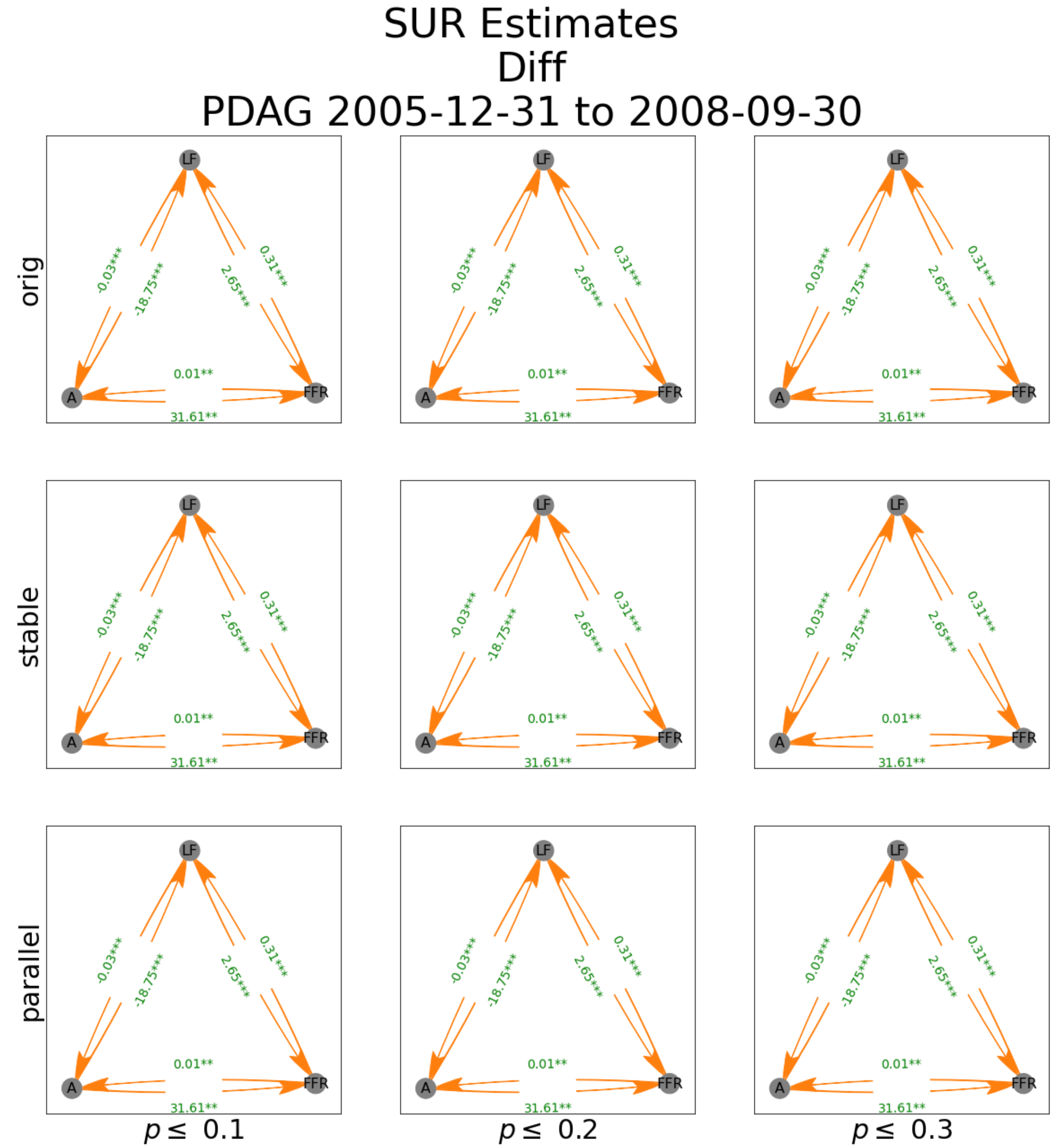
In [10]:

```
from datlib.DAG import *
from collections import OrderedDict
from linearmodels.system import SUR
from matplotlib.backends.backend_pdf import PdfPages
from scipy import stats


# plt.rcParams.update({"font.size":20})
# pd.to_datetime(data['Date'])
# data.set_index("Date")
# diff = "Diff"

sig_vals = [.05, .01, .001]


def add_lags(data, lags=12 / year):
    for key in data:
        for i in range(1, lags + 1):
            new_key = key +" Lag" * i
            data[new_key] = var_data[key].shift(year * i)

def rename_vars(rename_data, rename_dct):
    for key in rename_data:
        for rename_key in rename_dct:
            if rename_key in key:
                rename_data.rename(columns={key:rename_dct[rename_key]}, inplace



# only estimate twice differenced data since a some variables fail to reject the
diffs = ["Diff", "Diff-in-Diff"]
reg_dict = {}
lags = int(round(12 / year,0))

for diff in diffs:

    ## Use return_type = "pdag" to allow for endogeneity
    ##  "dag" disallows this sort ambiguity
    reg_dict[diff] = {}
    # only test the aggregated data, since the hypothesis is that:
        # 1) currency and total assets are indicate relative provision of liquidi
        # 2) loss function variables are targeted together

    plot_vars =  ["Effective Federal Funds Rate (%)",
                                        "Currency in Circulation",
                                        "Total Assets",
                                        "Loss Function"]
    sig = sigs = [0.1, 0.2, 0.3]
    variants = ["orig", "stable", "parallel"]
    ci_test = "pearsonr"


    # for plot_vars in plot_vars_dct:
    i = ""
    reg_dict[diff][i] = {}
    df = data[diff]
    var_data = df[plot_vars]
    rename_vars(var_data, rename_dct)
    add_lags(var_data, lags)
    var_data.dropna(inplace=True)
    start_end_list = [(str(var_data.index[0])[:10],"2008-09-30"),
                      ("2010-01-31", "2020-02-29")]
    #                       ("2008-10-31", "2020-02-29"),
    #                        (str(var_data.index[0])[:10],"2020-02-29")]

    # slice dfs by date range, house in dfs {}
    dfs = {}
    for start,end in start_end_list:
        dfs[start +" to " + end] =  var_data.loc[start:end].copy()

    # use dates (key) to track dates for which hypotheses are tested
    for dates, select_df in dfs.items():
        for return_type in ["pdag"]:
            edges = {}
            dag_df = {}
            fig, ax = plt.subplots(3,3,figsize = (20,20))
            fig.suptitle("DAG Estimates\n"+diff.replace(" ", "") + "\n" + return_
                        fontsize = 45)

            fig_sur, ax_sur = plt.subplots(3,3,figsize = (20,20))
            fig_sur.suptitle("SUR Estimates\n"+diff.replace(" ", "") + "\n" + ret
                        fontsize = 45)
            fig_var, ax_var = plt.subplots(3,3,figsize = (20,20))
            fig_var.suptitle("VAR Estimates\n"+diff.replace(" ", "") + "\n" + ret
                        fontsize = 45)

            for x in range(len(sigs)):
                sig = sigs[x]
                edges[sig] = {}
                dag_df[sig] = {}
                for y in range(len(variants)):
                    variant = variants[y]
                    a = ax[y][x]

                    constant = False if diff == "Diff-in-Diff" else True
                    keys = [k for k in select_df if "Lag" not in k]
                    dag_df[sig][variant] = select_df[keys].dropna()
                    # construct dag, save directed edges
                    edges[sig][variant] = DAG(dag_df[sig][variant], variant, ci_t
                    # construct graphs with PC labels
                    graph_DAG(edges[sig][variant],
                              dag_df[sig][variant],
                              title = "",
                              fig = fig,
                              ax = a)
                    if x == 0:
                        a.set_ylabel(variant, fontsize = 30)
                    if y == len(variants) - 1:
                        a.set_xlabel("$p \leq$ "+ str(sig), fontsize = 30)
                    a = ax_sur[y][x]
#                     identify sink nodes in directed dag edges, use info to dete
#                      additional graphs with marginal effects from SUR and VAR c
                    sink_source = identify_sink_nodes(edges[sig][variant])
                    filename = i + " " + diff + "DAGOLS " + dates + " " + variant
                    DAG_OLS(dag_df[sig][variant], sink_source, filename, a, diff,
                    if x == 0:
                        a.set_ylabel(variant, fontsize = 30)
                    if y == len(variants) - 1:
                        a.set_xlabel("$p \leq$ "+ str(sig), fontsize = 30)

                    a = ax_var[y][x]

    #                     filename = i + " " + diff + "DAGVAR " + dates
                    DAG_VAR(select_df.dropna(), sink_source, filename, a, diff, c
                    if x == 0:
                        a.set_ylabel(variant, fontsize = 30)
                    if y == len(variants) - 1:
                        a.set_xlabel("$p \leq$ "+ str(sig), fontsize = 30)
```

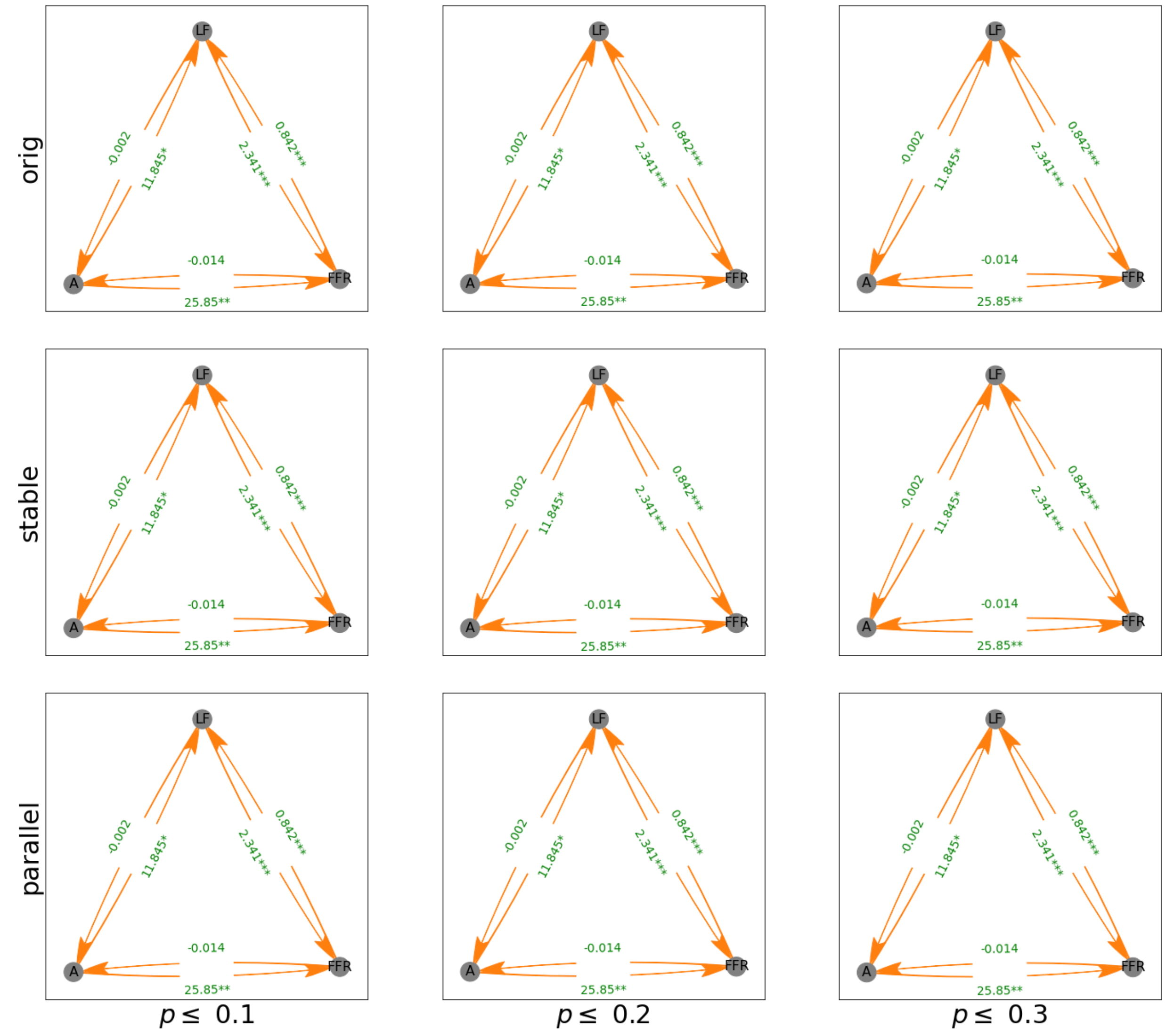
```
0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]

0%|          | 0/2 [00:00<?, ?it/s]
```

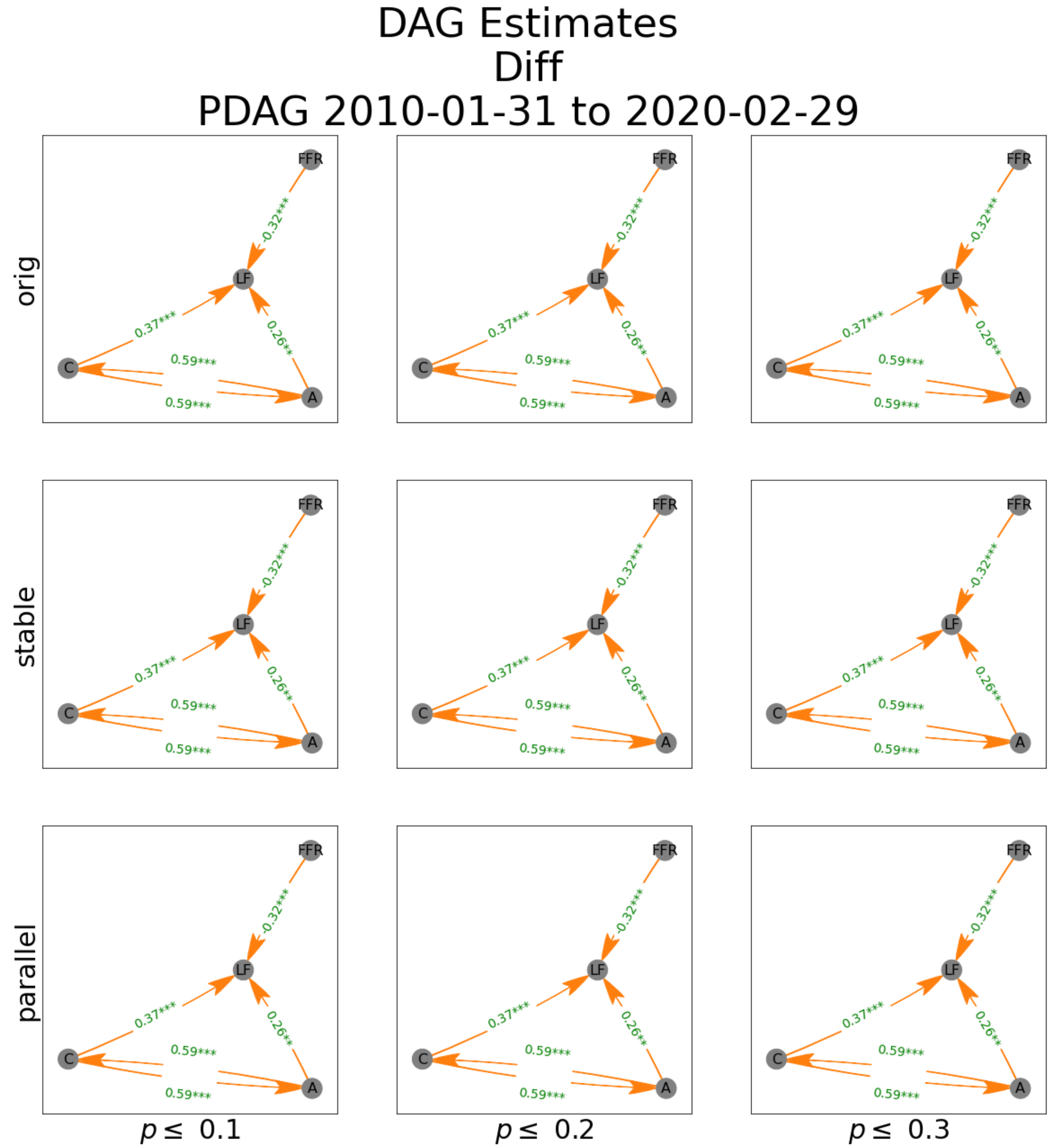
```
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/2 [00:00<?, ?it/s]
```


DAG Estimates
Diff
PDAG 2005-12-31 to 2008-09-30
orig
stable
parallel
LF
A
FFR
-0.71***
-0.71***
0.79***
0.79***
0.36*
0.36*
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3

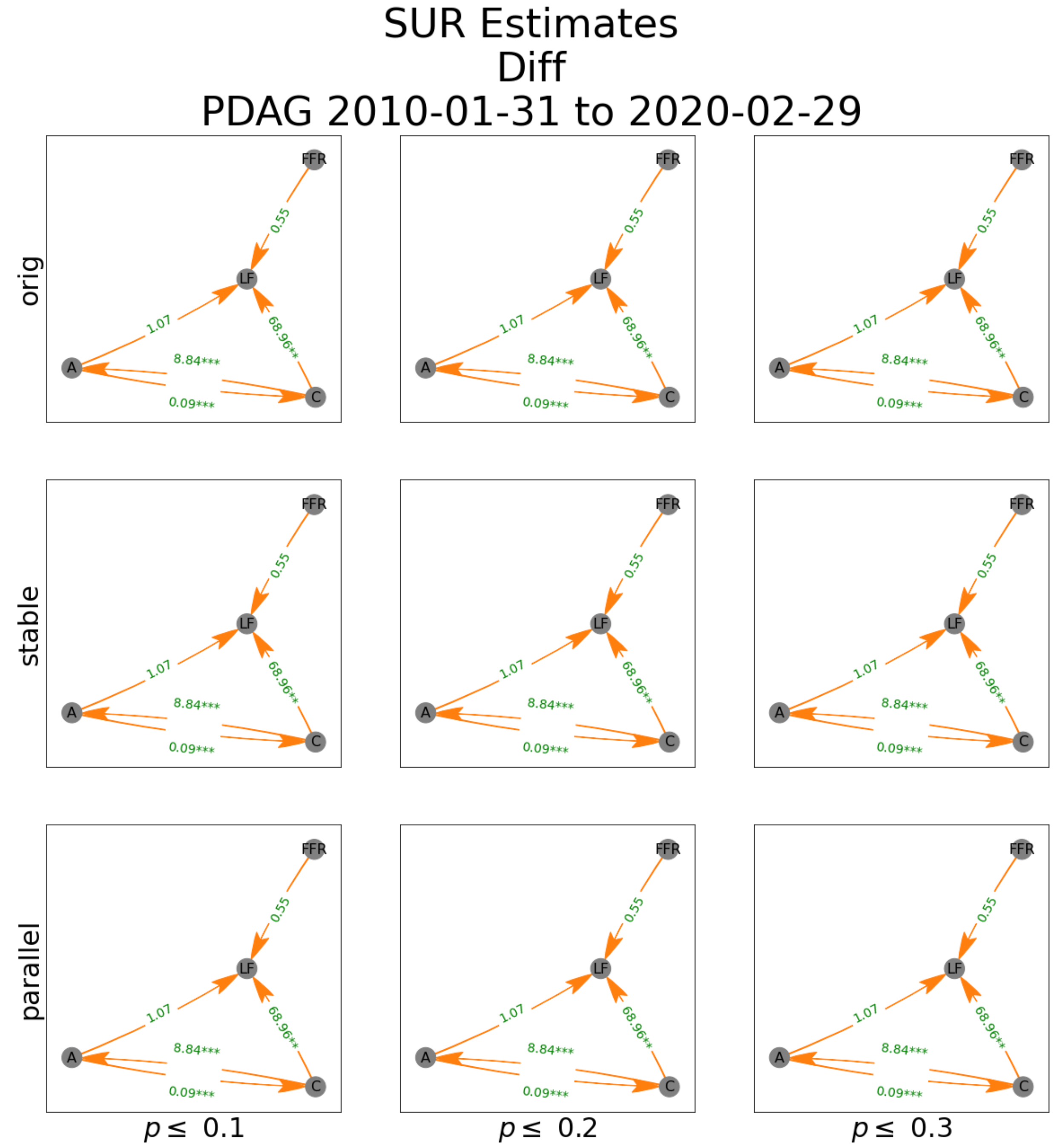

SUR Estimates
Diff
PDAG 2005-12-31 to 2008-09-30
orig
stable
parallel
LF
A
FFR
-0.03***
-18.75***
2.65***
0.31***
0.01**
31.61**
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3

VAR Estimates
Diff
PDAG 2005-12-31 to 2008-09-30

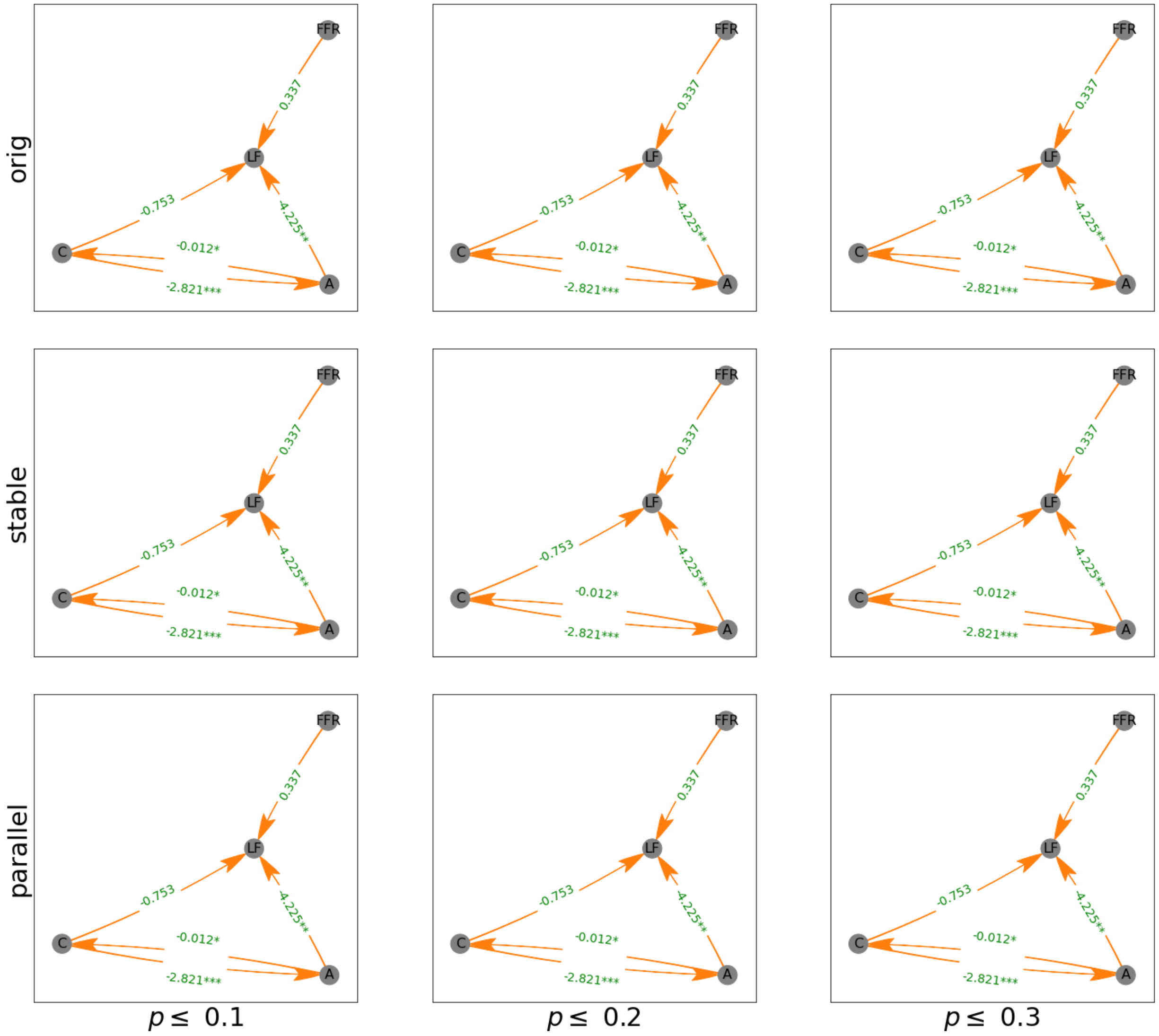


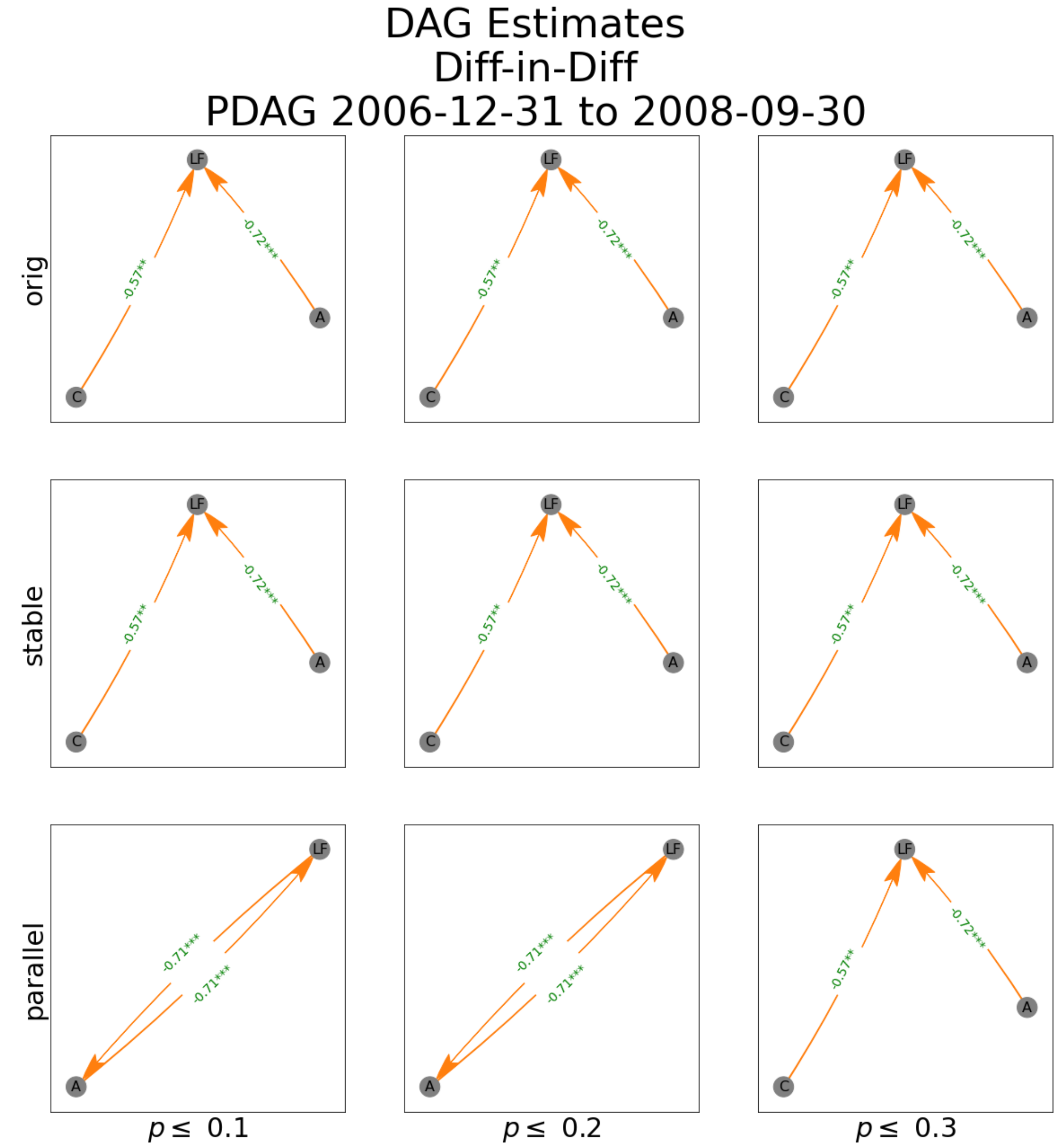


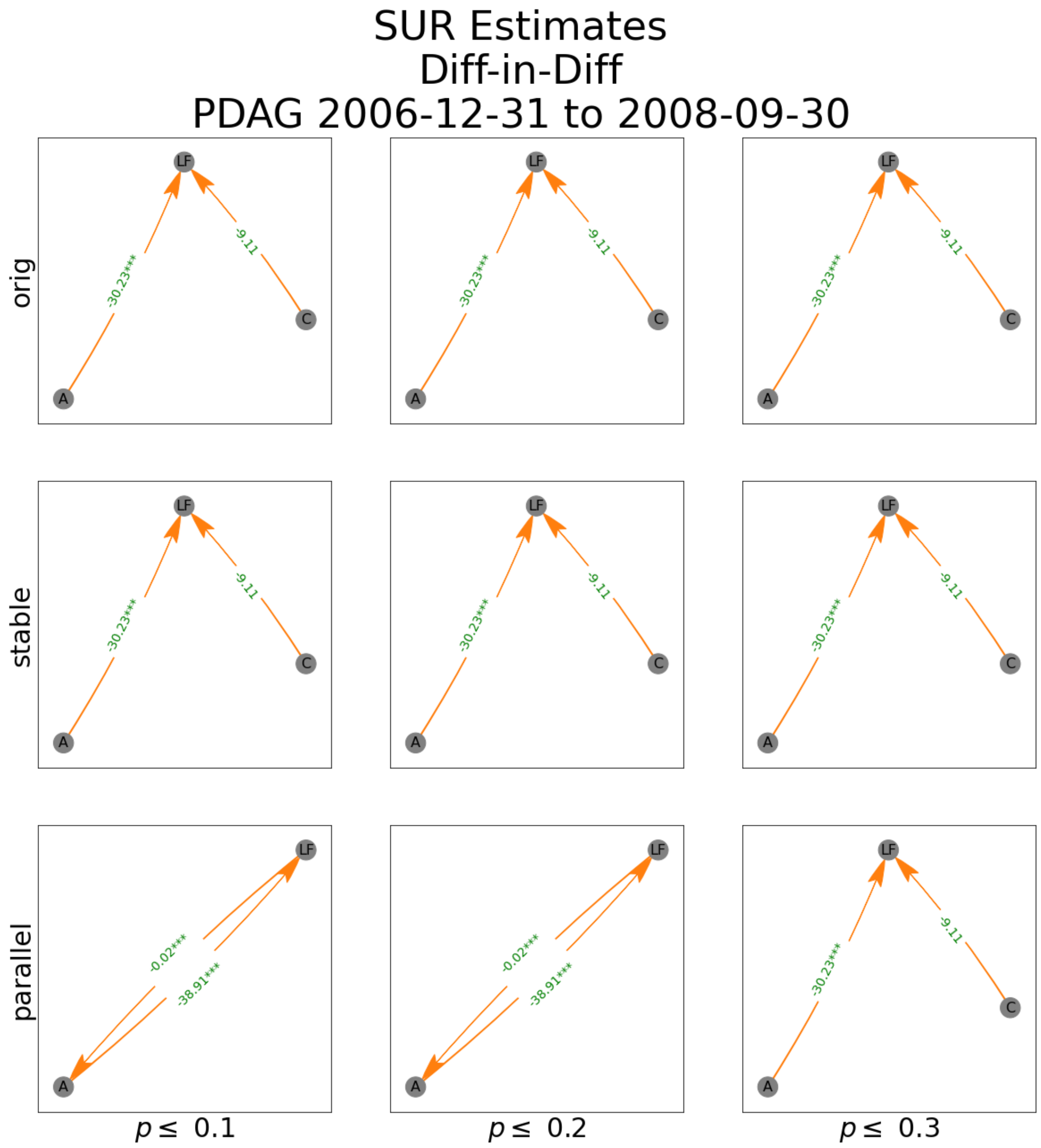
DAG Estimates
Diff
PDAG 2010-01-31 to 2020-02-29


SUR Estimates
Diff
PDAG 2010-01-31 to 2020-02-29
orig
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
stable
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
parallel
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
FFR
0.55
LF
1.07
68.96**
A
8.84***
0.09***
C
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3

VAR Estimates
Diff
PDAG 2010-01-31 to 2020-02-29

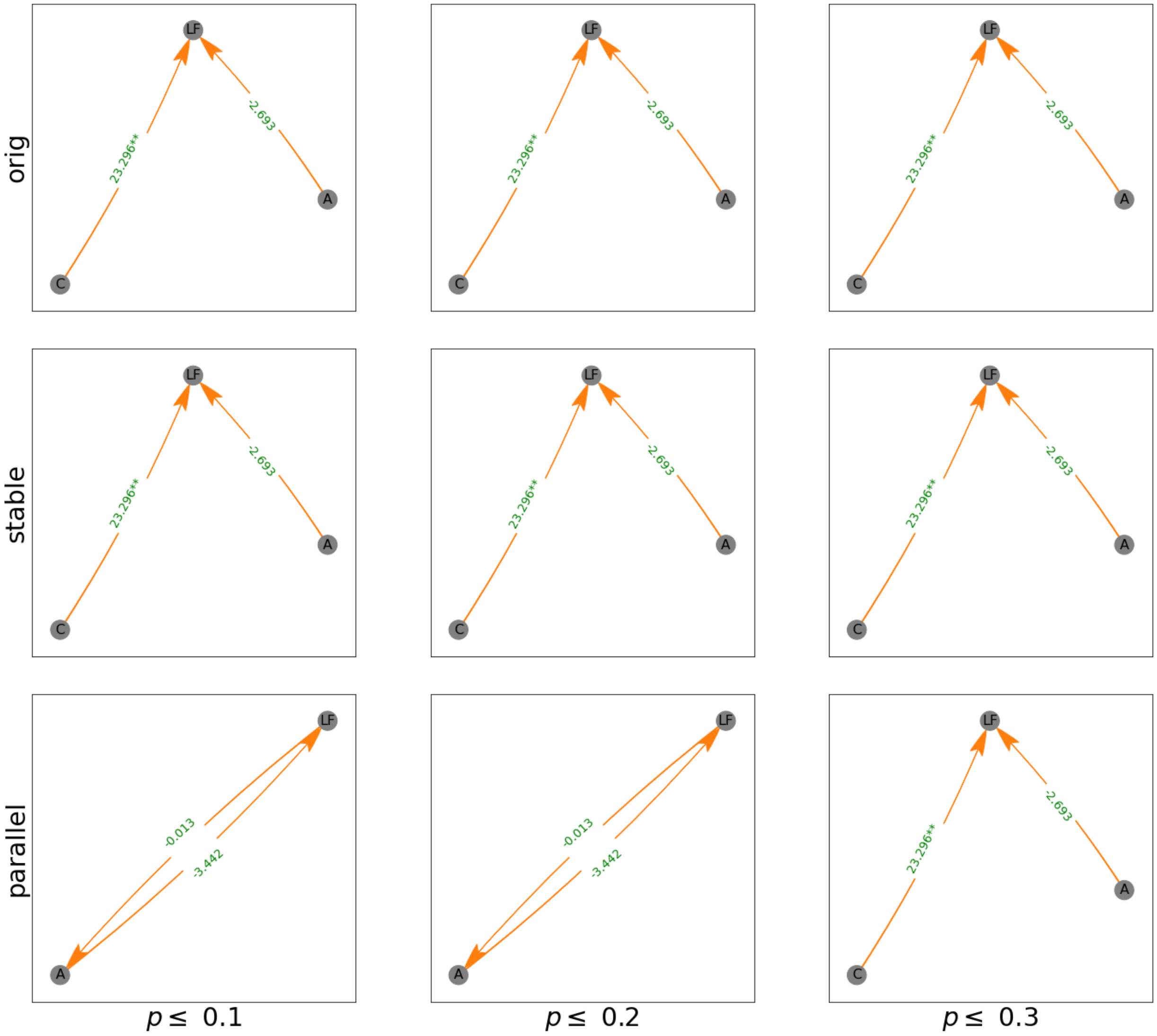


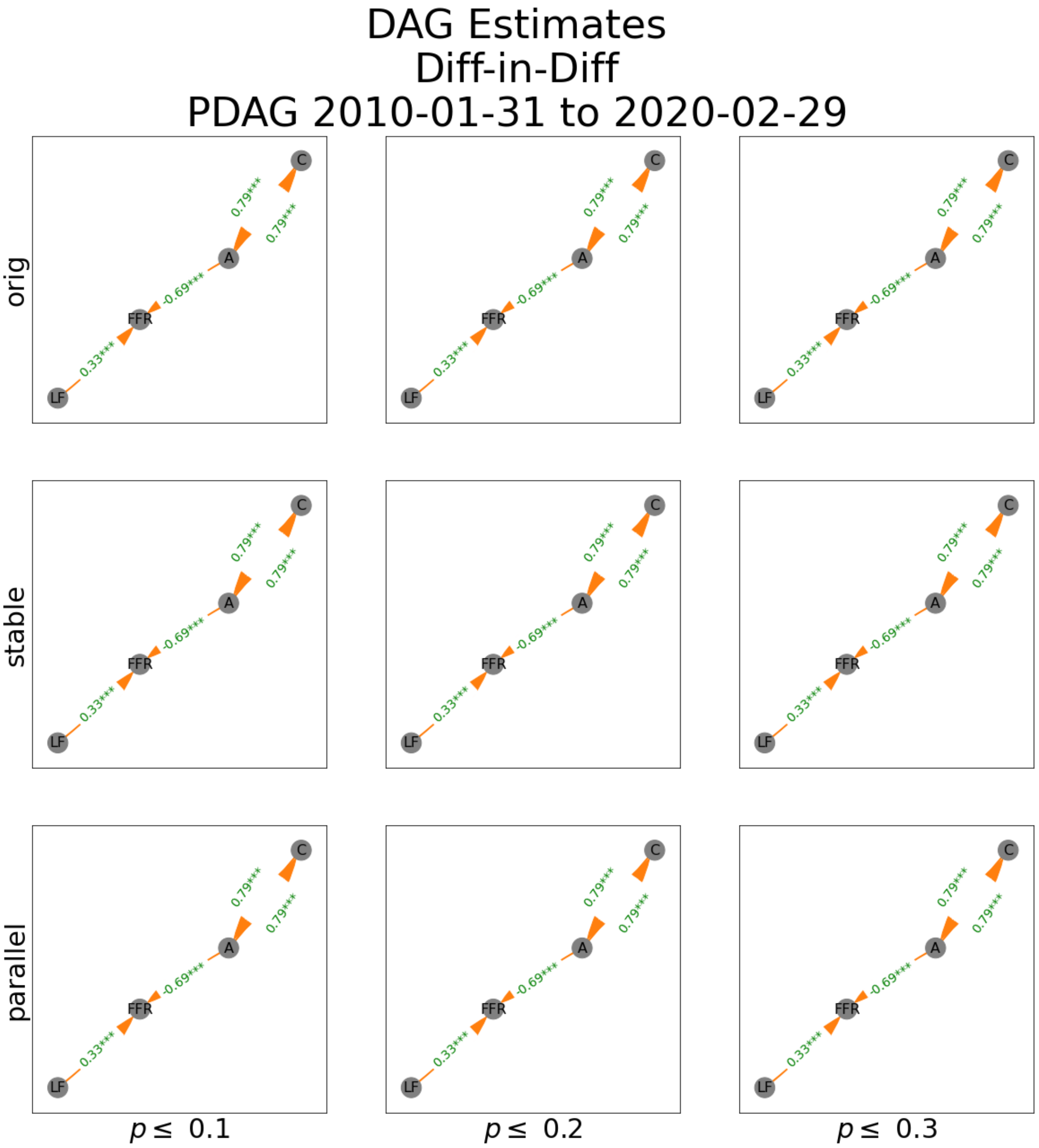

DAG Estimates
Diff-in-Diff
PDAG 2006-12-31 to 2008-09-30
orig
stable
parallel
LF
A
C
-0.57**
-0.72***
-0.71***
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3



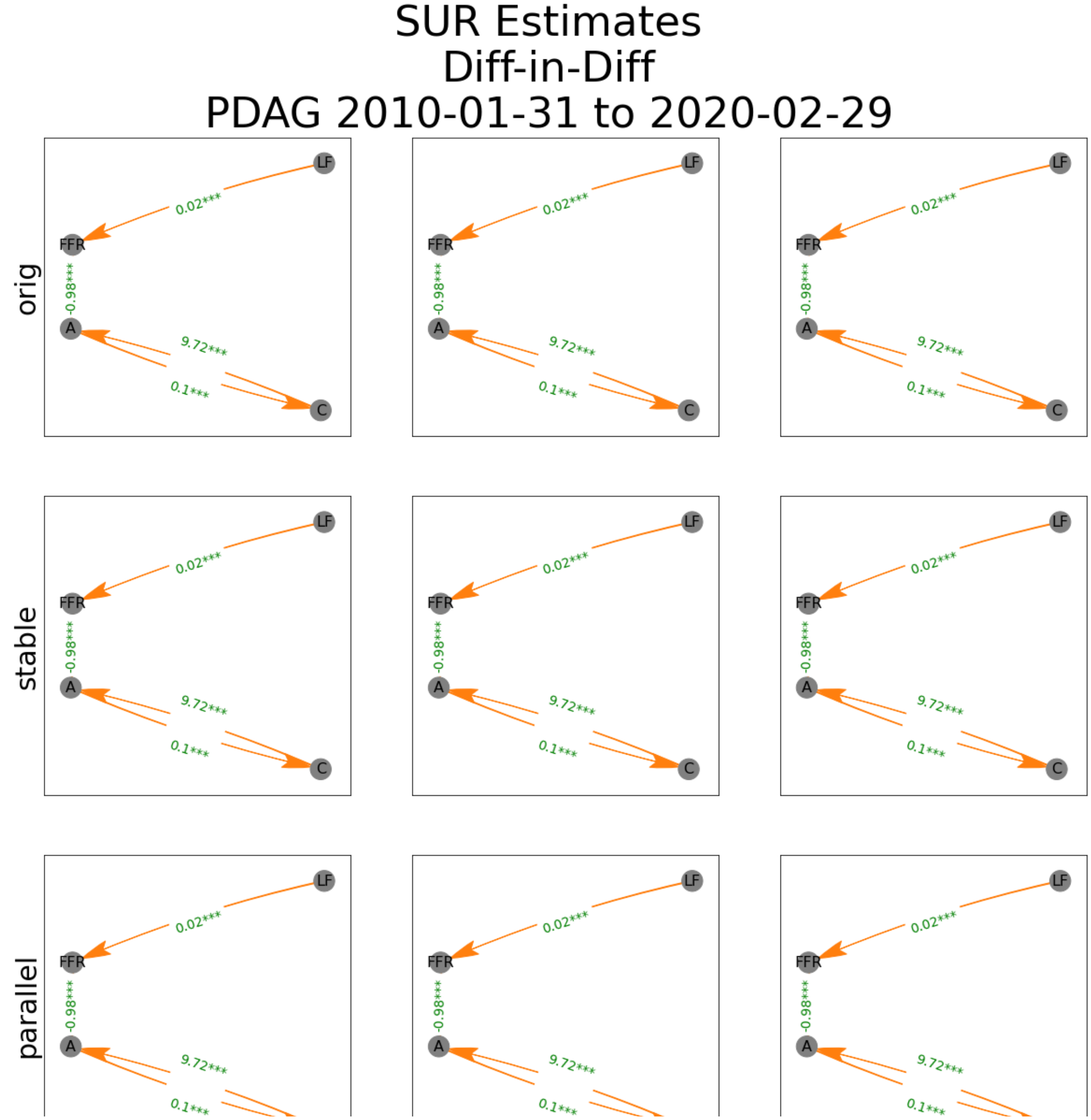
SUR Estimates
Diff-in-Diff
PDAG 2006-12-31 to 2008-09-30

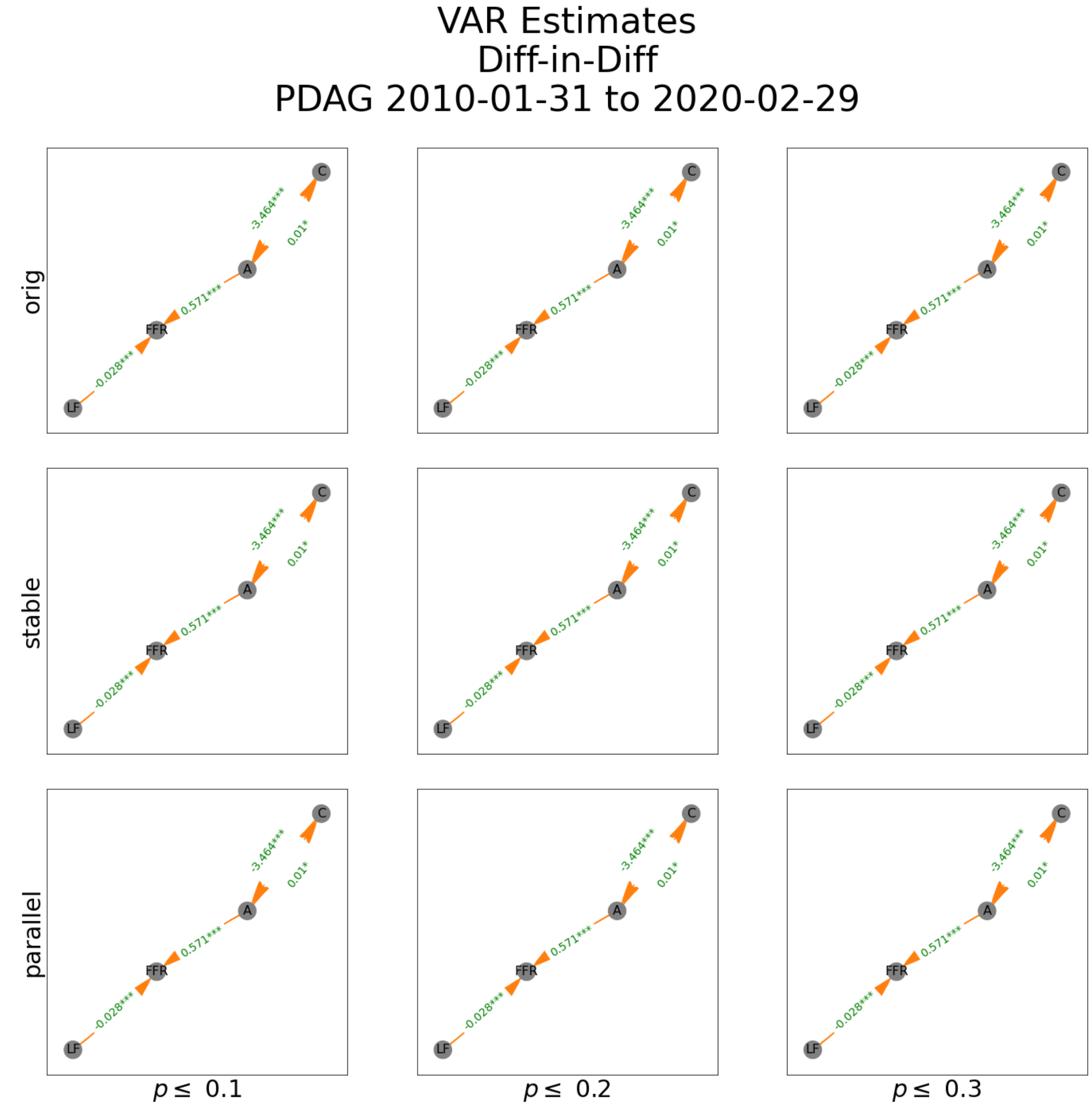
VAR Estimates
Diff-in-Diff
PDAG 2006-12-31 to 2008-09-30





DAG Estimates
Diff-in-Diff
PDAG 2010-01-31 to 2020-02-29


SUR Estimates
Diff-in-Diff
PDAG 2010-01-31 to 2020-02-29
orig
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
stable
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
C
parallel
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***
LF
0.02***
FFR
-0.98***
A
9.72***
0.1***


VAR Estimates
Diff-in-Diff
PDAG 2010-01-31 to 2020-02-29
orig
stable
parallel
C
A
FFR
LF
-3.464***
0.01*
0.571***
-0.028***
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3