Performance and Usage Analytics for NCAR's Climate Model

Lolita Mannik

Regis University

National Center for Atmospheric Research (NCAR)

Summer Internships in Parallel Computational Science (SIParCS)

October 24, 2019

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Overview

1. Background on Community Earth System Model (CESM)

2. Model's configuration

3. Data preparation and analysis

4. Key findings

5. Conclusion and future work

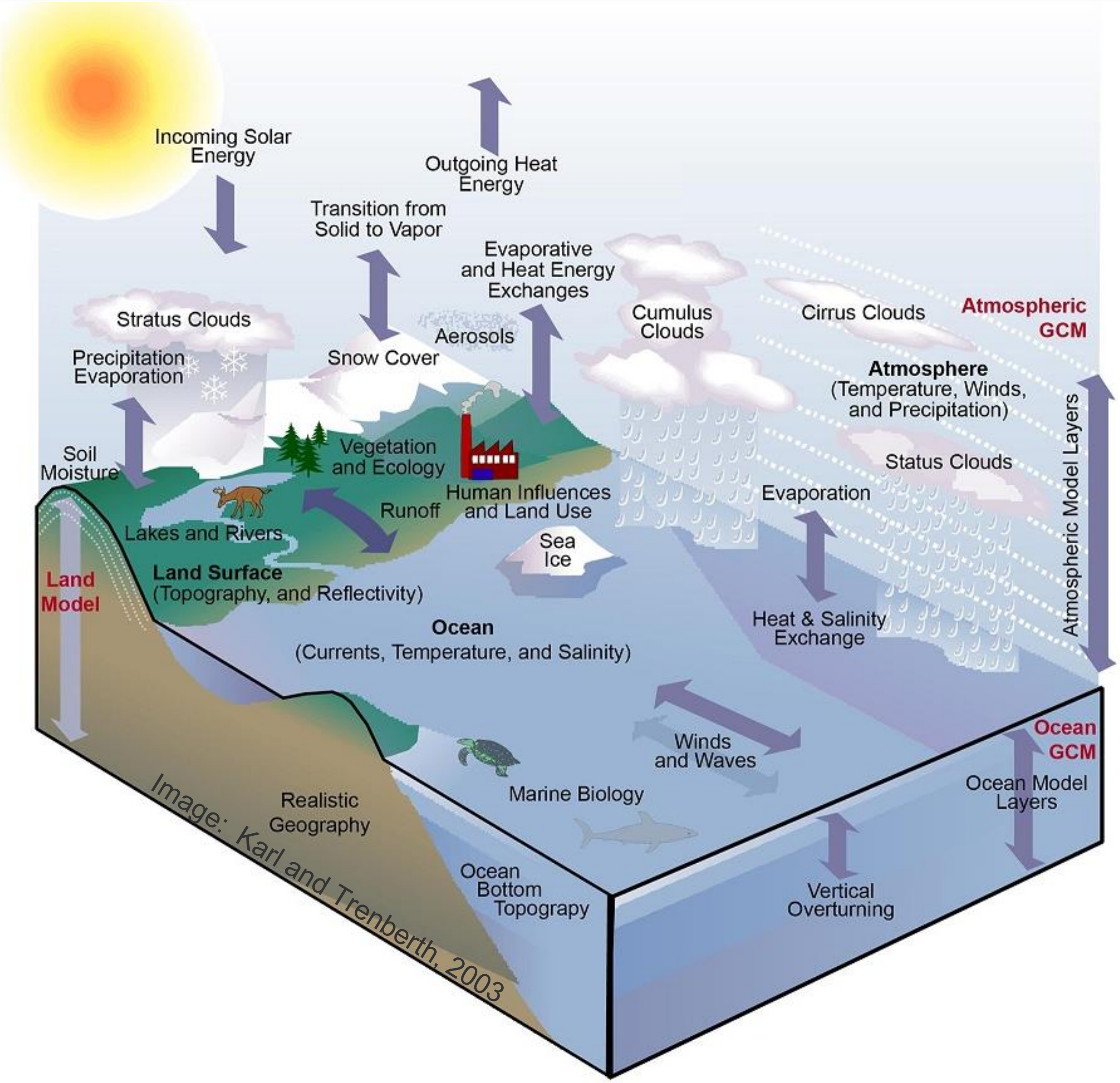
Image: http://www.cesm.ucar.edu/models/

Goal

Analyze CESM performance metadata

- ❑ Establish basic metrics
- ❑ Effect of a system upgrade on performance

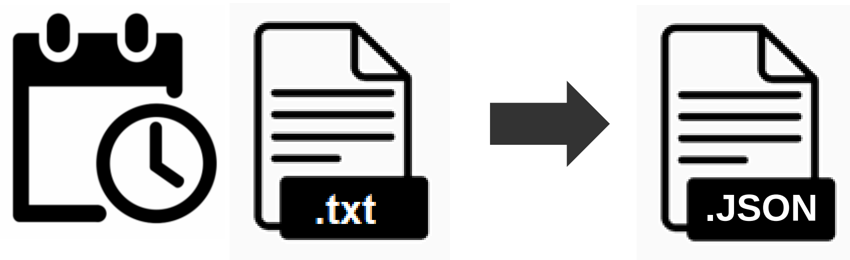
CESM Climate Model



- **Virtual laboratory**
- **Freely available**
- **Components:**
  - Atmosphere
  - Land
  - Ocean
  - River
  - Sea and Land Ice
  - Wave

CESM = Community Earth System Model

Method



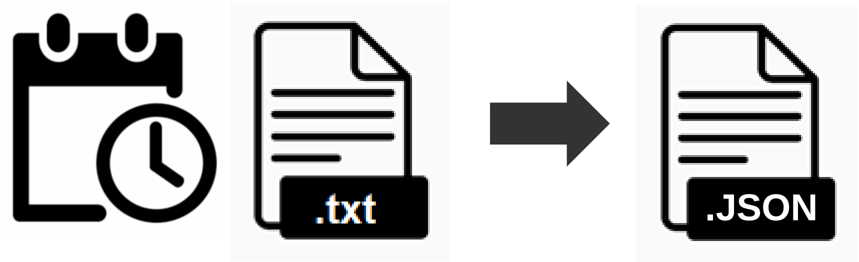
1

Data Engineering

- **Acquiring**
- **Saving**

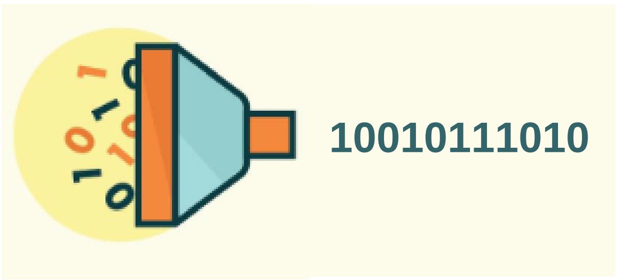
```
|---------------- TIMING PROFILE ---------------------
  Case          : b.e21.BHIST.f09_g17.CMIP6-historical.001
  LID           : 2979765.chadmin1.181015-050236

  User          : cmip6
  Curr Date     : Mon Oct 15 10:01:22 2018
  grid          :
a%0.9x1.25_l%0.9x1.25_oi%gx1v7_r%r05_g%gland4_w%ww3a_m%gx1v7
  compset       : HIST_CAM60_CLM50%BGC-CROP_CICE_POP2%ECO%ABIO-
                  DIC_MOSART_CISM2%NOEVOLVE_WW3_BGC%BDRD
  run_type      : hybrid, continue_run = TRUE (inittype = FALSE)
  stop_option   : nyears, stop_n = 5
  run_length    : 1825 days (1825.0 for ocean)

  Init Time     :       63.817 seconds
  Run Time      :    17837.627 seconds        9.774 seconds/day
  Final Time    :        0.057 seconds
```

Method



1

Data Engineering
- Acquiring
- Saving



2

Data Wrangling
- Reindexing
- More parsing
- Set data types
- Intuitive columns
- Calculations

Data Wrangling

Parse Run_Length

```
3650 days (3650.0 for ocean)
```

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
   if 'length_days' in col:
      df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650 days (3650.0 for ocean)

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
   if 'length_days' in col:
      df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650 days (3650.0 for ocean)

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650 days

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
   if 'length_days' in col:
      df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650 $\boxed{\text{days}}$

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```

Data Wrangling

Parse Run_Length

3650

```
#1. Strip everything after "days" in run_length column
df['run_length_temp'] = df['run_length'].str.split('(').str[0]

#Confirm every run_length contains the same units of days
substr = 'days'
print ("Rows in df:", len(df))
print ("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

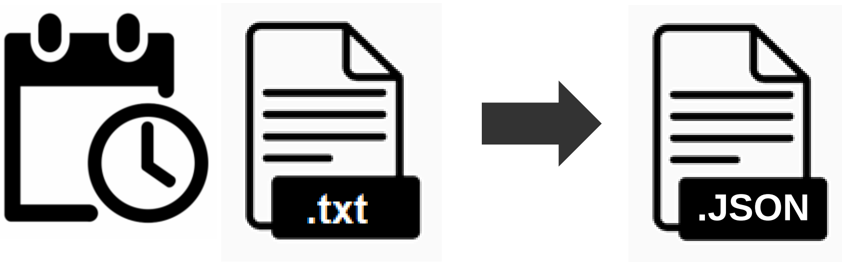
```
Rows in df: 5160
Rows with units of days: 5160
```

```
#2. Strip "days" in run_length column
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

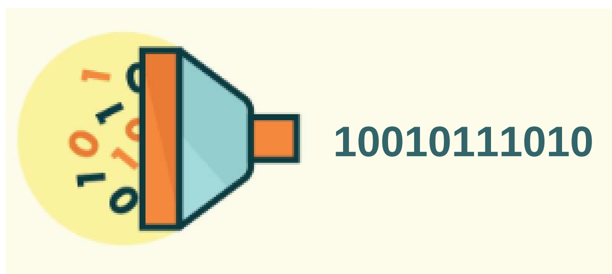
```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
       '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
       '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

```
#Convert necessary columns to numeric format
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```

Method



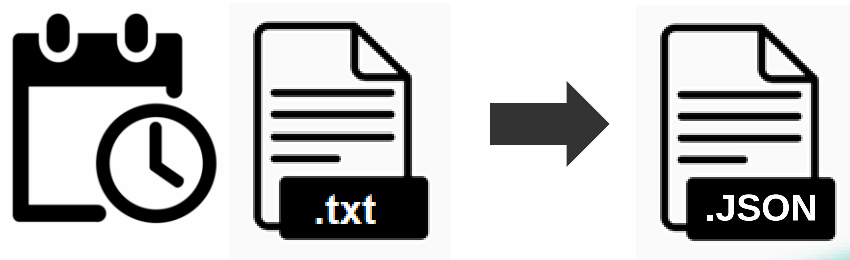
1. **Data Engineering**
   - Acquiring
   - Saving



2. **Data Wrangling**
   - Reindexing
   - More parsing
   - Set data types
   - Intuitive columns
   - Calculations

3. **Data Storage**

Method



1

Data Engineering
- Acquiring

ling

ata types
- Intuitive columns
- Calculations

USABLE DATA!!

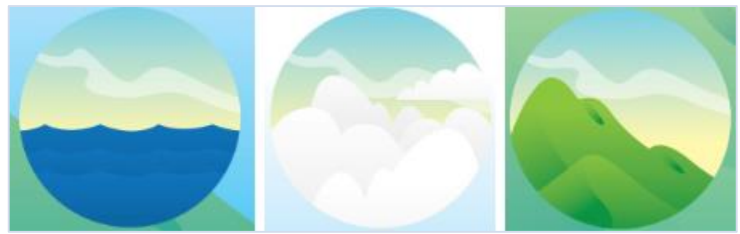
3

Data Storage

Method

4

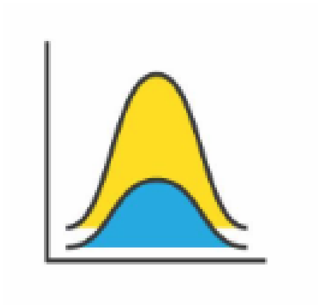
Exploratory Data Analysis



52 unique
component configurations

Method

4 Exploratory Data Analysis

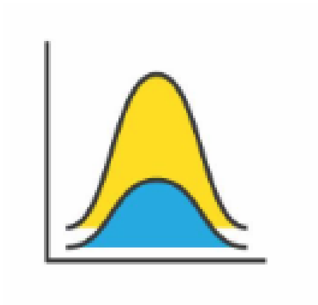


5 Statistical Analysis

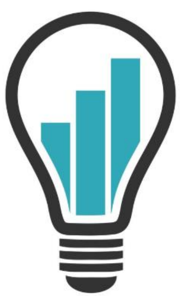
Pandas
Numpy
Scipy Stats
StatsModels

Method

4 Exploratory Data Analysis



5 Statistical Analysis



6 Visualization

MatPlotLib
Seaborn
Plotly

Analysis: Dataset Totals

416 Days

948
Unique Cases

21,785
Simulated Years

137,112,802
CPU Hours

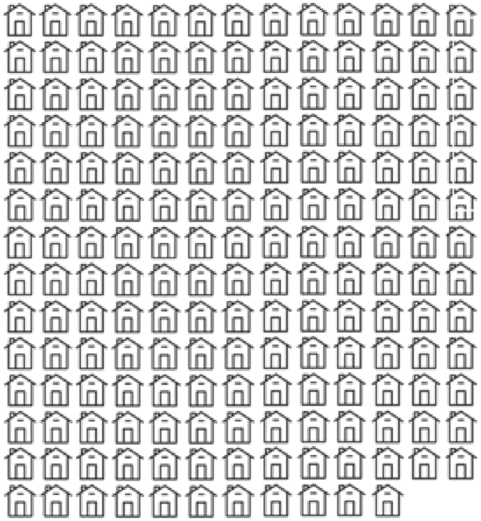
Power Equivalence

137,112,802

CPU Hours

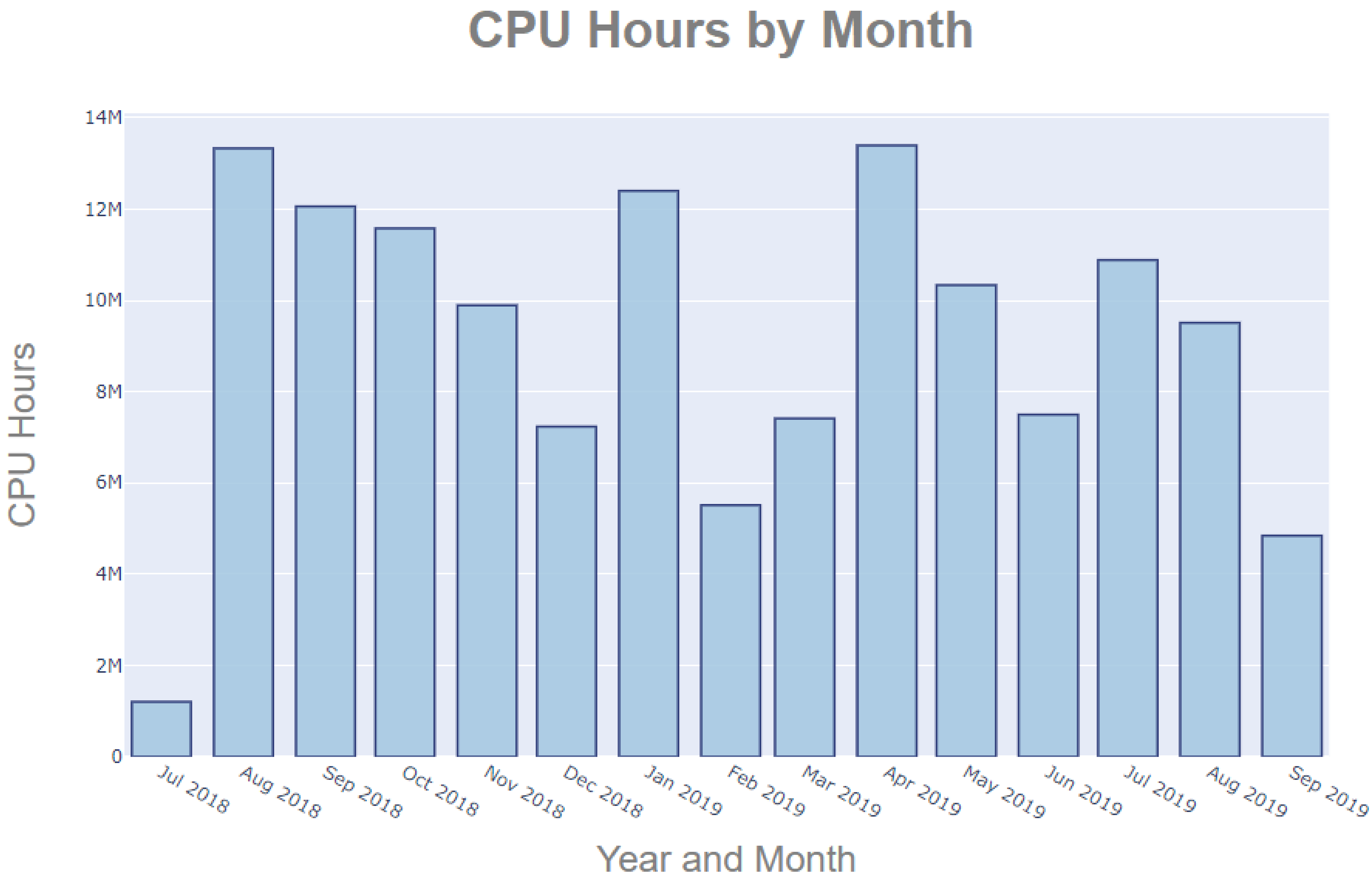
218 trips around the equator in a Nissan Leaf

or



Annual power for 180 Colorado homes

Analysis: Monthly Totals



Analysis: Grouping Atmospheric Configurations

Component string = compset

`'1850_CAM60%1PCT_CLM50%BGC-CROP_CICE%CMIP6_POP2%ECO_MOSART_CISM2%EVOLVE_WW3_BGC%BDRD'`

Init **Atm** **Land** **Sea Ice** **Ocean** **River** **Land Ice** **Wave** **OBGC***

14 unique atmospheric components

***OBGC = Ocean Bio-geo-chemsitry**

Analysis: Grouping Atmospheric Configurations

Component string = compset

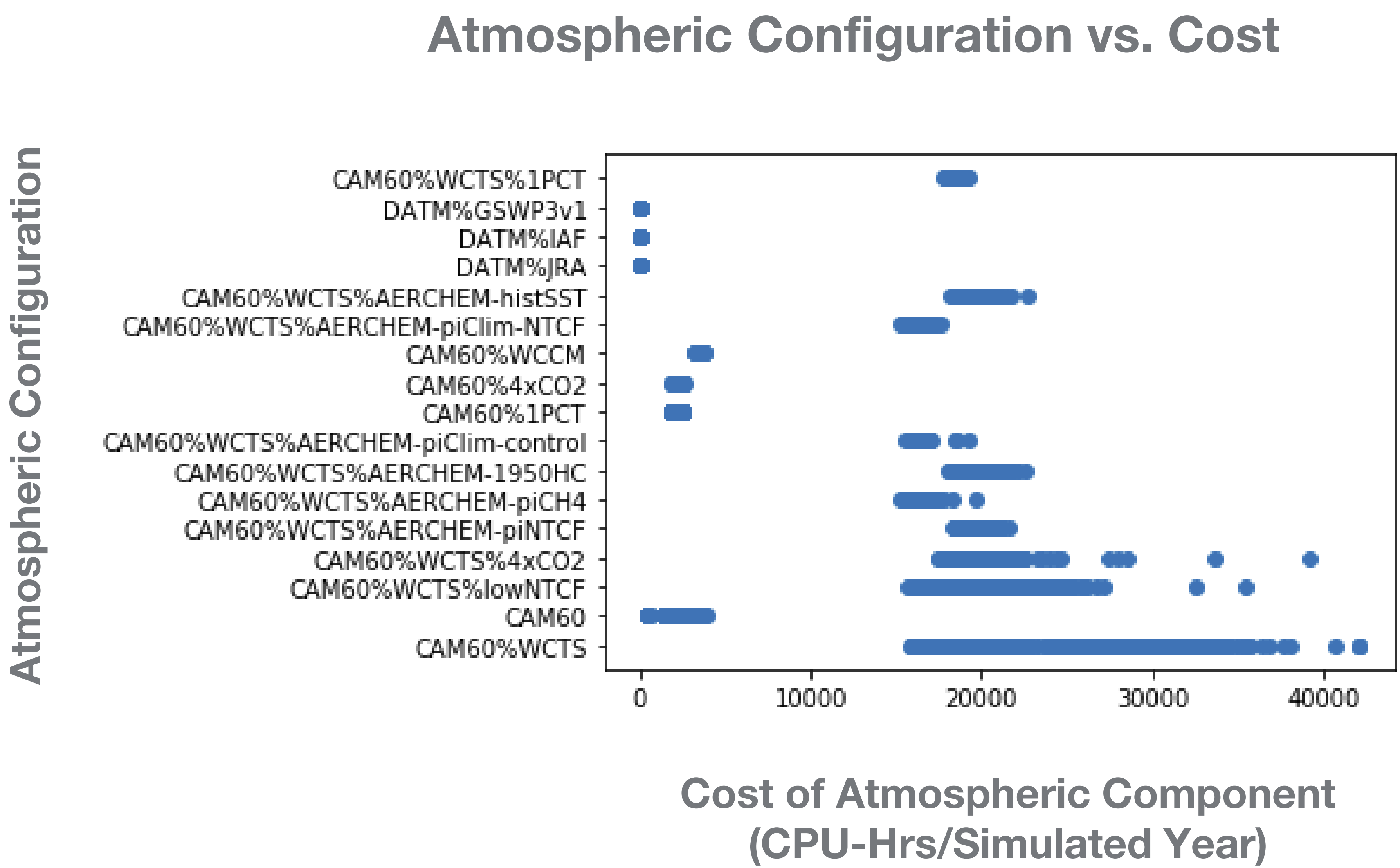
'1850_CAM60%1PCT_CLM50%BGC-CROP_CICE%CMIP6_POP2%ECO_MOSART_CISM2%EVOLVE_WW3_BGC%BDRD'

Init | Atm | Land | Sea Ice | Ocean | River | Land Ice | Wave | OBGC*

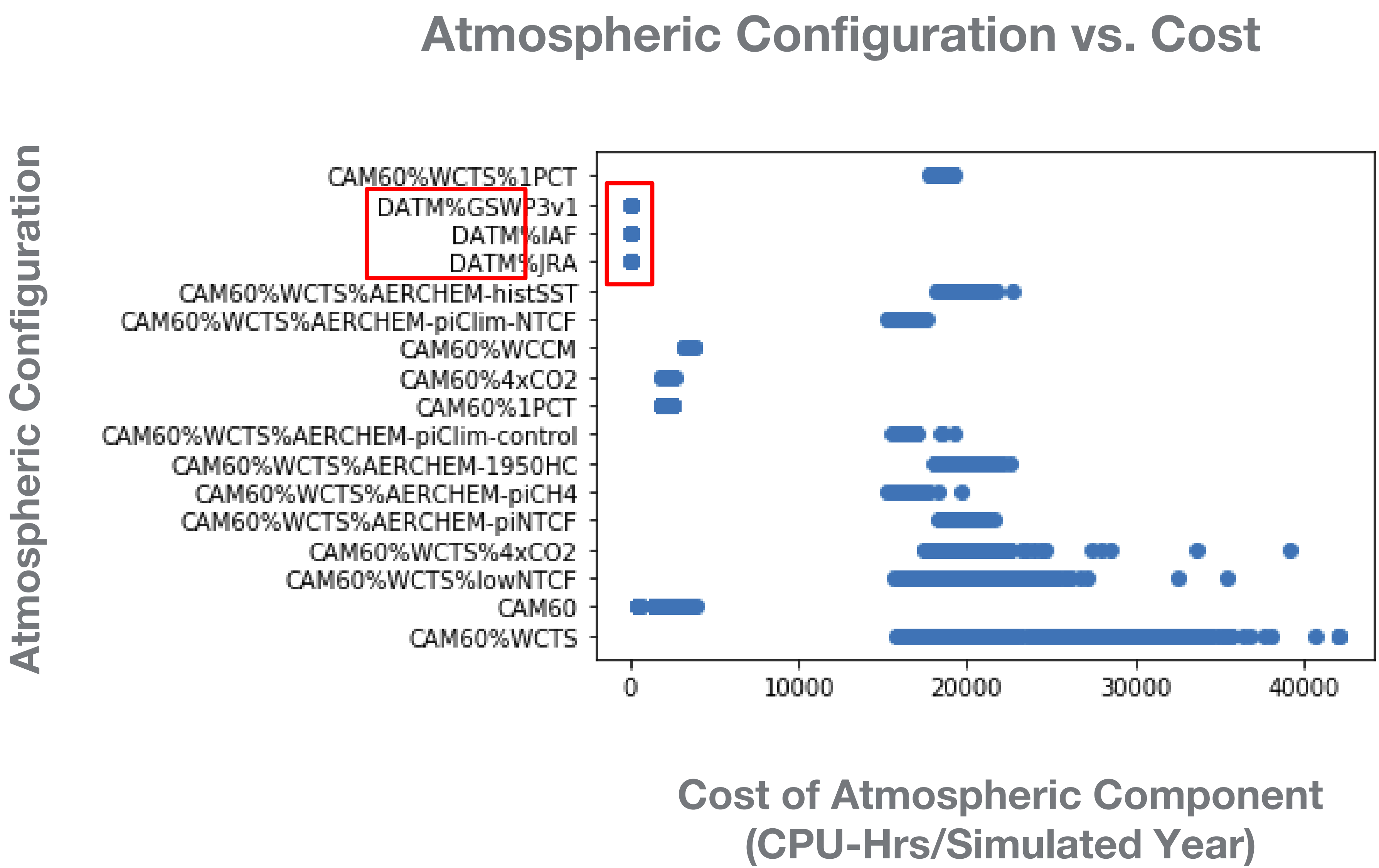
14 unique atmospheric components

*OBGC = Ocean Bio-geo-chemsitry

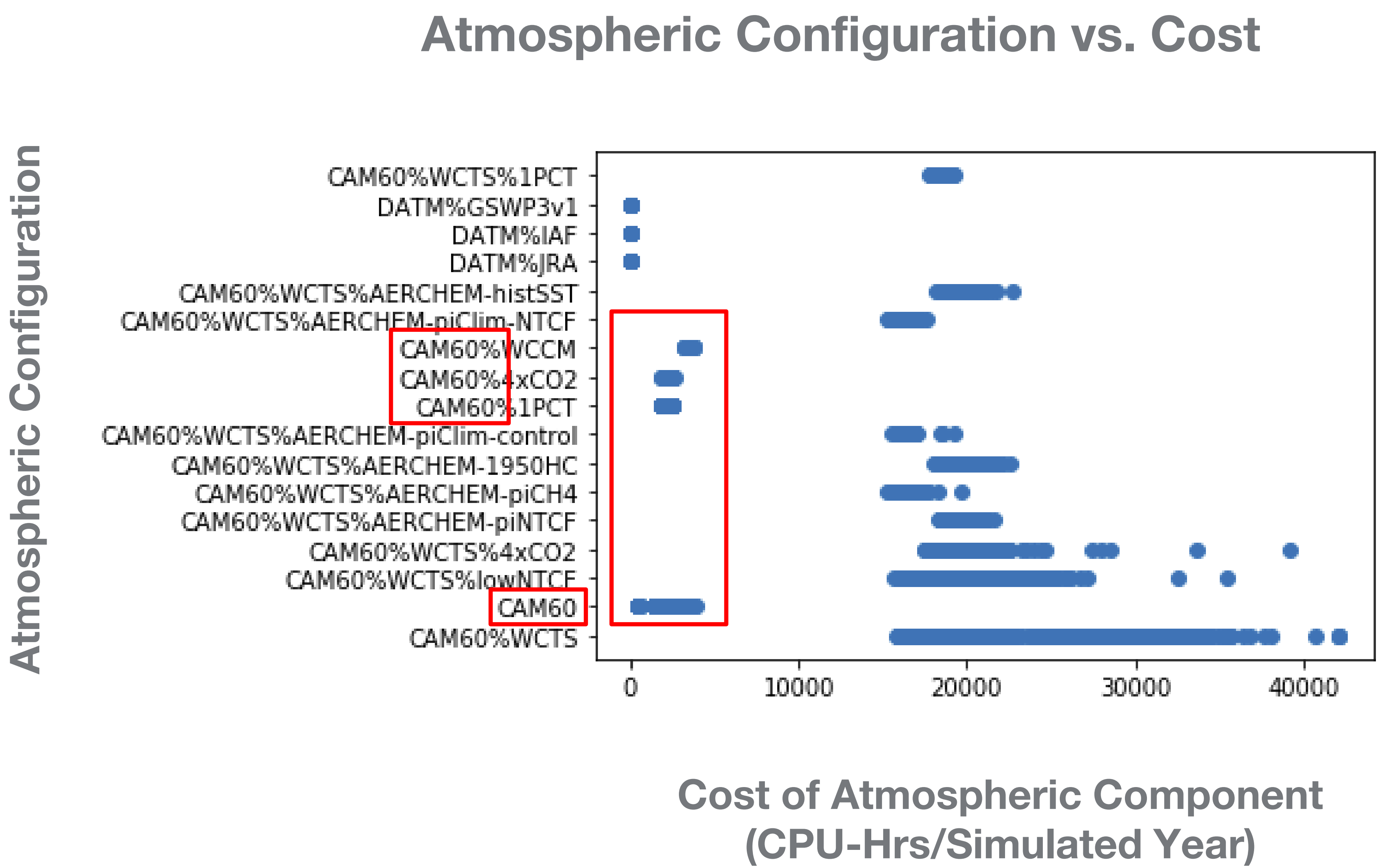
Analysis: Grouping Atmospheric Configurations



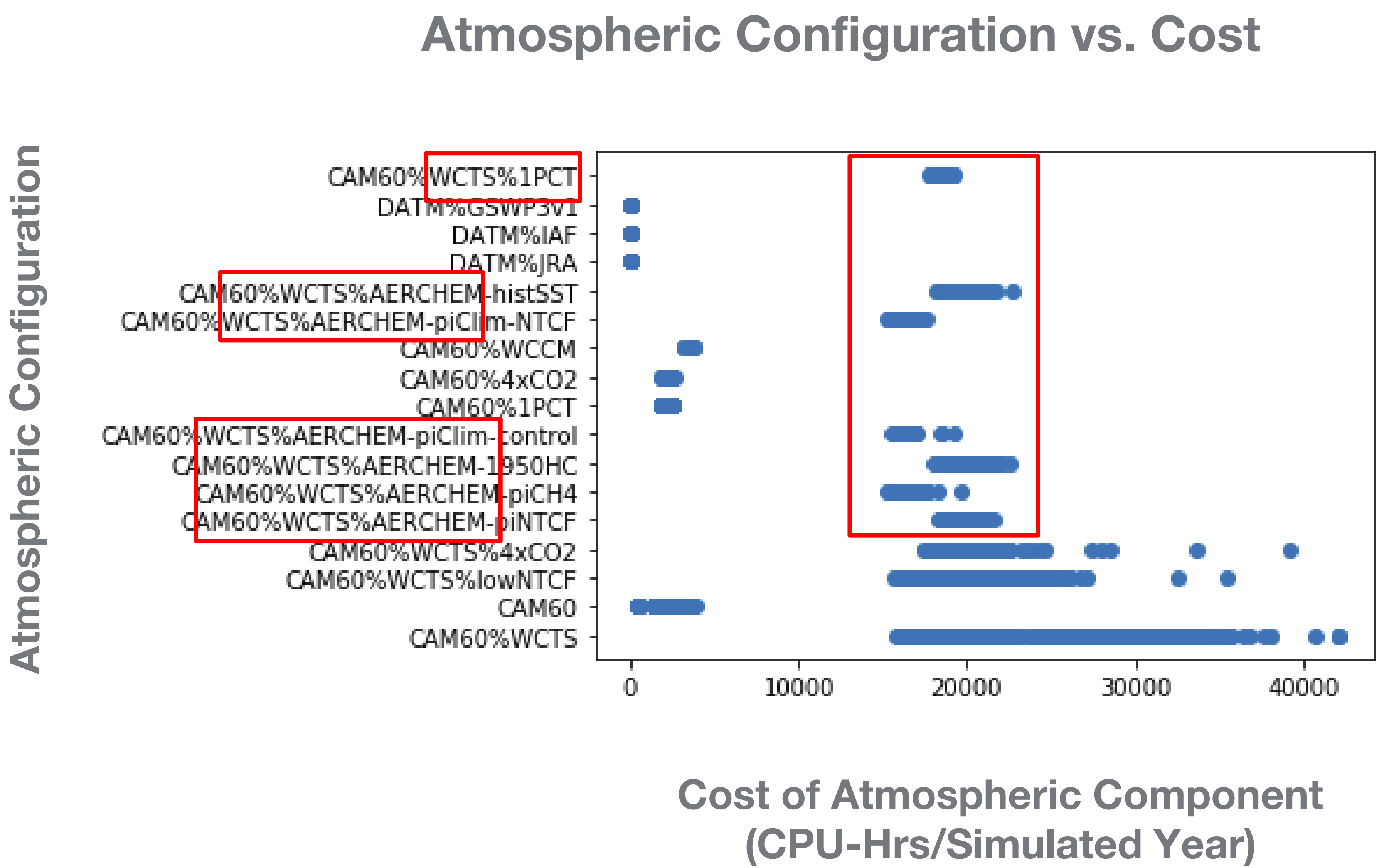
Analysis: Grouping Atmospheric Configurations



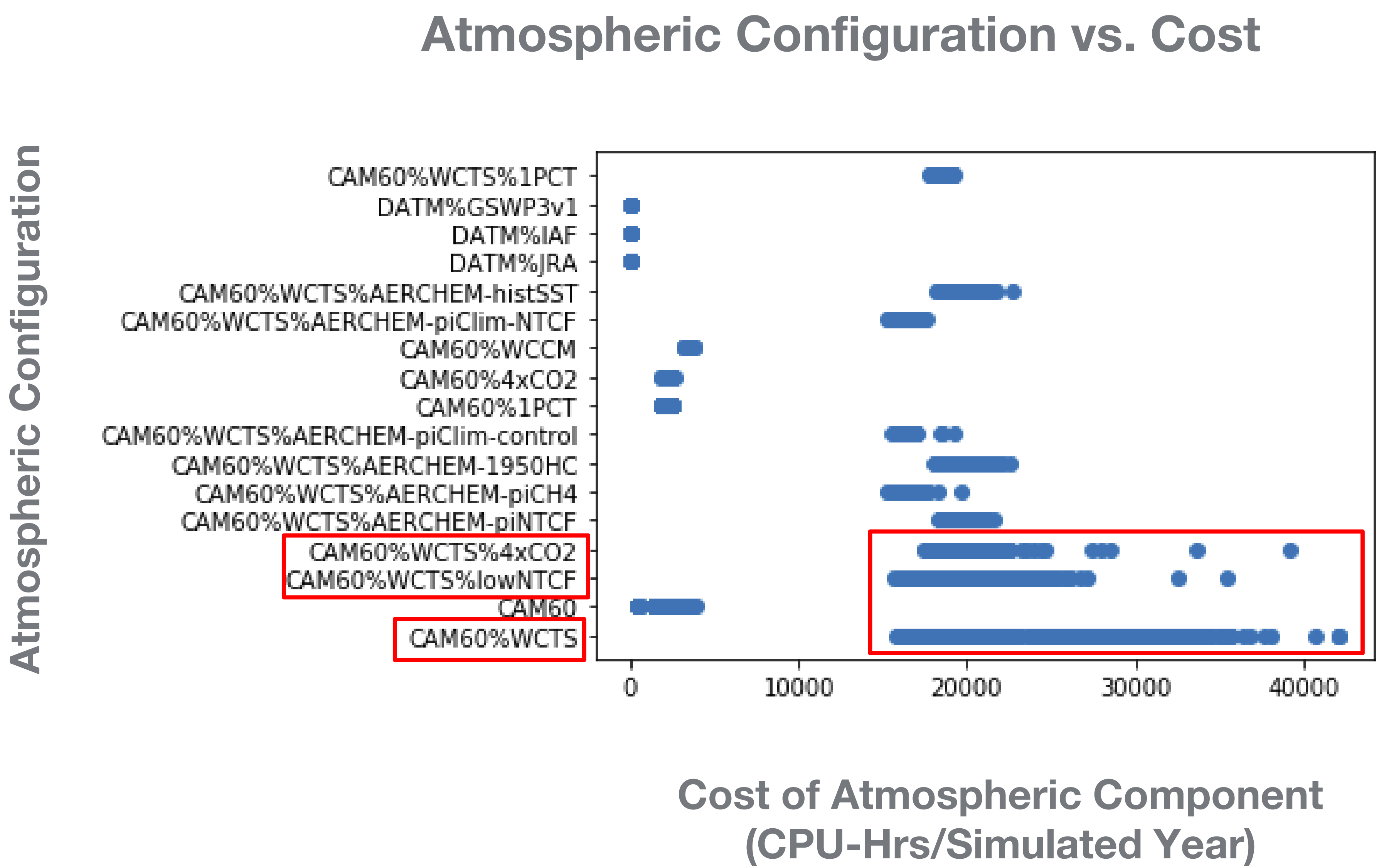
Analysis: Grouping Atmospheric Configurations



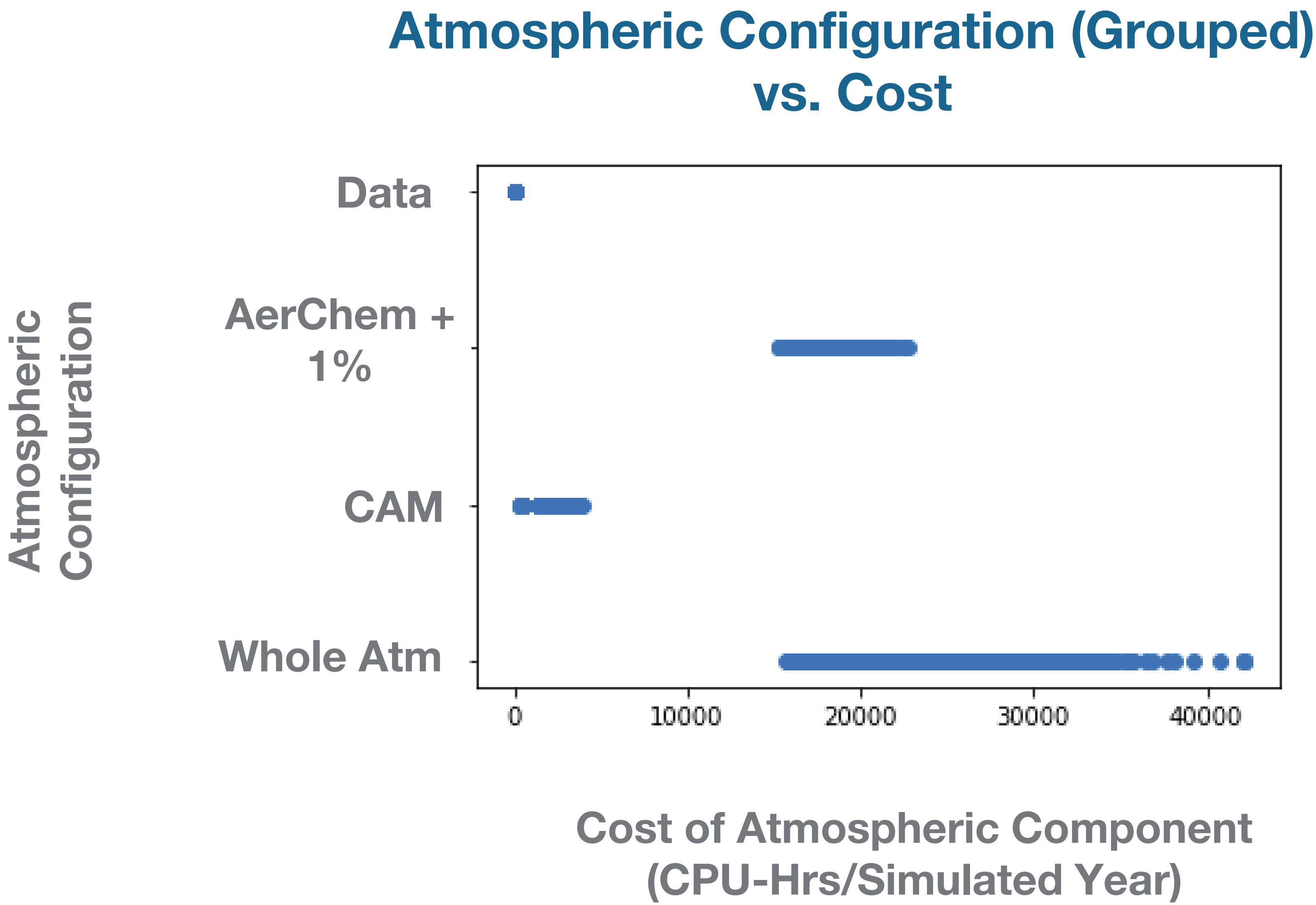
Analysis: Grouping Atmospheric Configurations



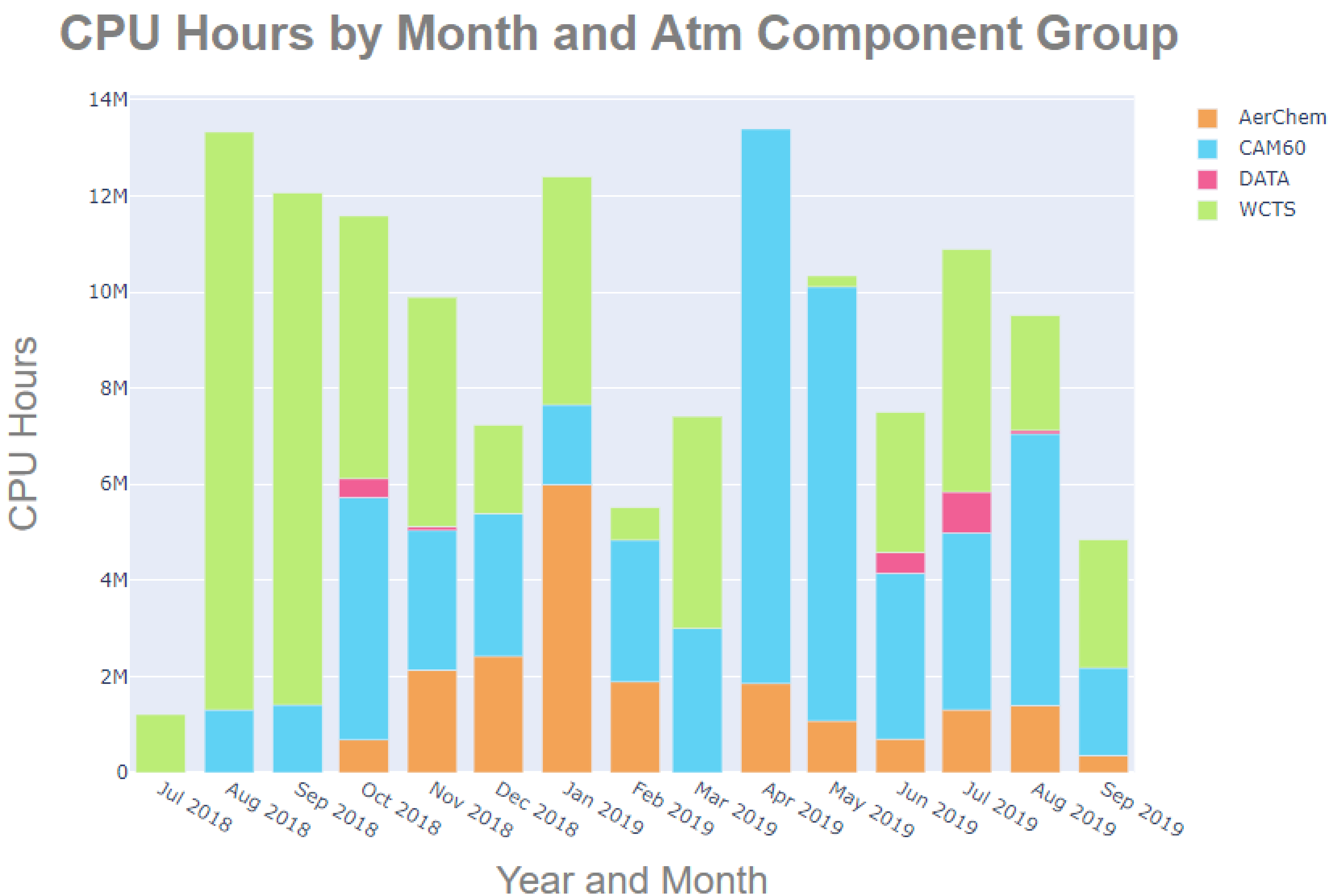
Analysis: Grouping Atmospheric Configurations



Analysis: Grouping Atmospheric Configurations



Analysis: Atmospheric Components



Analysis: System Upgrade

- Cheyenne Supercomputer: 145,152 processors
- Upgrade: June 25-July 5, 2019
- Install SUSE Linux Enterprise Server Service Pack 4 to update security and support

Spoiler Alert!
Name of new supercomputer!

Analysis: System Upgrade

More maniacal subsetting . . .

By Case

```
b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001
```

Analysis: System Upgrade

More maniacal subsetting . . .

By Case

```
b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001
```

By Base

```
b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001
```

Analysis: System Upgrade

More maniacal subsetting . . .

By Case

```
b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001
```

By Base

```
b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001
```

Analysis: System Upgrade

More maniacal subsetting . . .

By Case

`b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001`

By Base

`b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism~~.001~~`

By atmospheric component + atmospheric processors + oceanic processors

compset_atm	comp_pes_atm	comp_pes_ocn	total_submits
CAM60	576	144	10
CAM60	900	751	66
CAM60	1080	1080	912
CAM60%1PCT	1800	432	5
CAM60%1PCT	3456	1536	49
CAM60%WCCM	3456	432	98
CAM60%WCTS	1728	1728	41
CAM60%WCTS%4xCO2	3456	72	64
CAM60%WCTS%4xCO2	3456	108	168

Analysis: System Upgrade

More maniacal subsetting . . .

By Case (811 data points, 3445 sim years, 14 cases)

`b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism.001`

By Base (1206 data points, 4271 sim years, 14 bases)

b.e21.B1850G.f09_g17_gl4.CMIP6-1pctCO2to4x-withism~~.001~~

By atmospheric component
+ atmospheric processors
+ oceanic processors
(3113 data points,
12,493 sim years,
9 groups)

compset_atm	comp_pes_atm	comp_pes_ocn	total_submits
CAM60	576	144	10
CAM60	900	751	66
CAM60	1080	1080	912
CAM60%1PCT	1800	432	5
CAM60%1PCT	3456	1536	49
CAM60%WCCM	3456	432	98
CAM60%WCTS	1728	1728	41
CAM60%WCTS%4xCO2	3456	72	64
CAM60%WCTS%4xCO2	3456	108	168

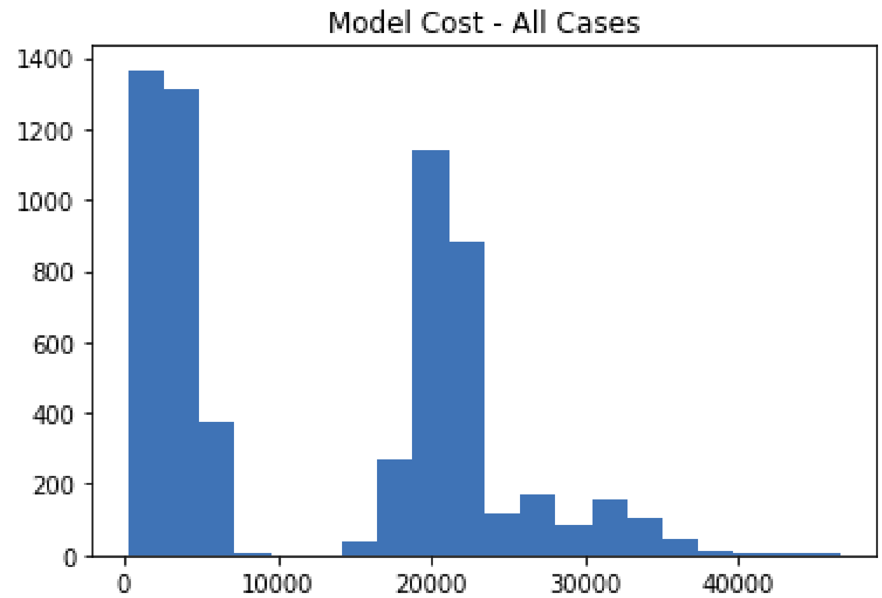
Analysis: System Upgrade

Tests for Normality

All
All - normalized
All - before/after

By case

By base



By group:
(atm compset/atm + ocn processors)

Analysis: System Upgrade

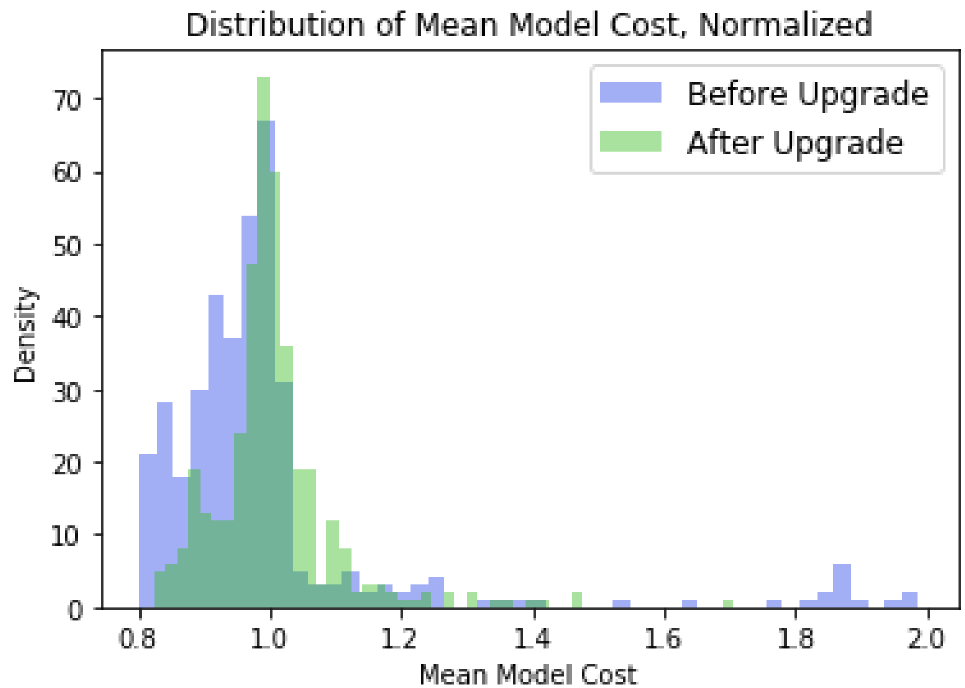
Tests for Normality

All
All - normalized
All - before/after

By case

By base

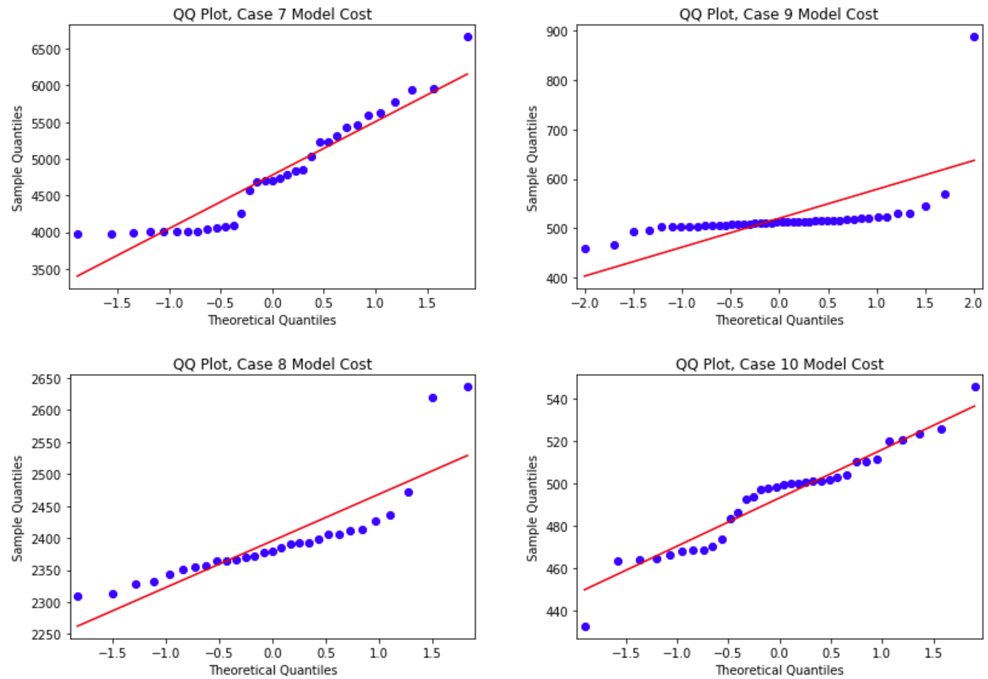
By group:
(atm compset/atm + ocn processors)



Analysis: System Upgrade

Tests for Normality

All
All - normalized
All - before/after



By case

By base

By group:
(atm compset/atm + ocn processors)

Statistical Tests for Normality:

- Kolmogorov-Smirnoff
- Shapiro-Wilk

Analysis: System Upgrade

Mean Model Cost Before Upgrade

vs.

Mean Model Cost After Upgrade

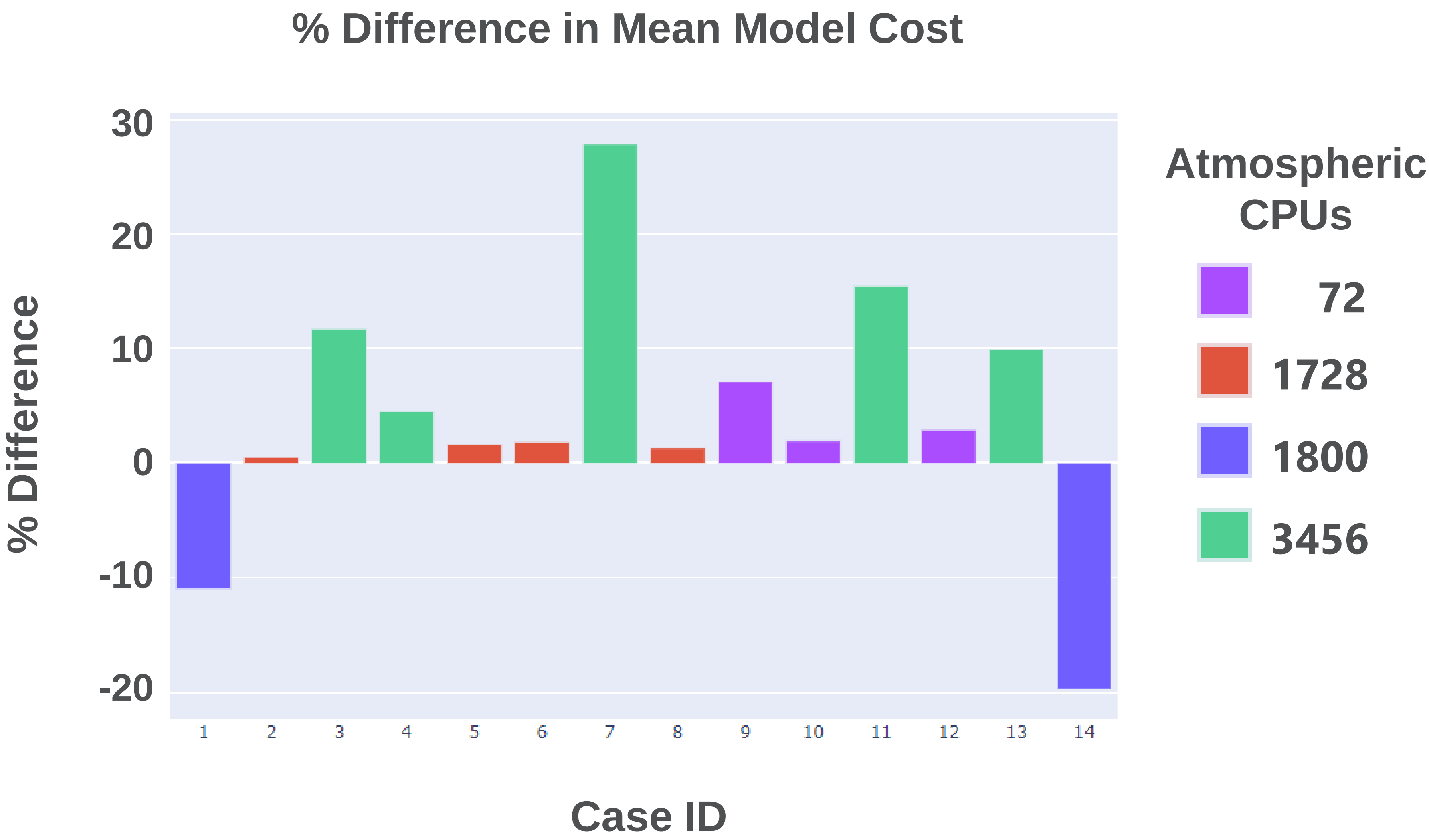
Calculated percent difference (% change) in means before and after the upgrade:

- **By case**
- **By base**
- **By group**

Determined whether there was statistical significance in the means using Kruskal Wallis test

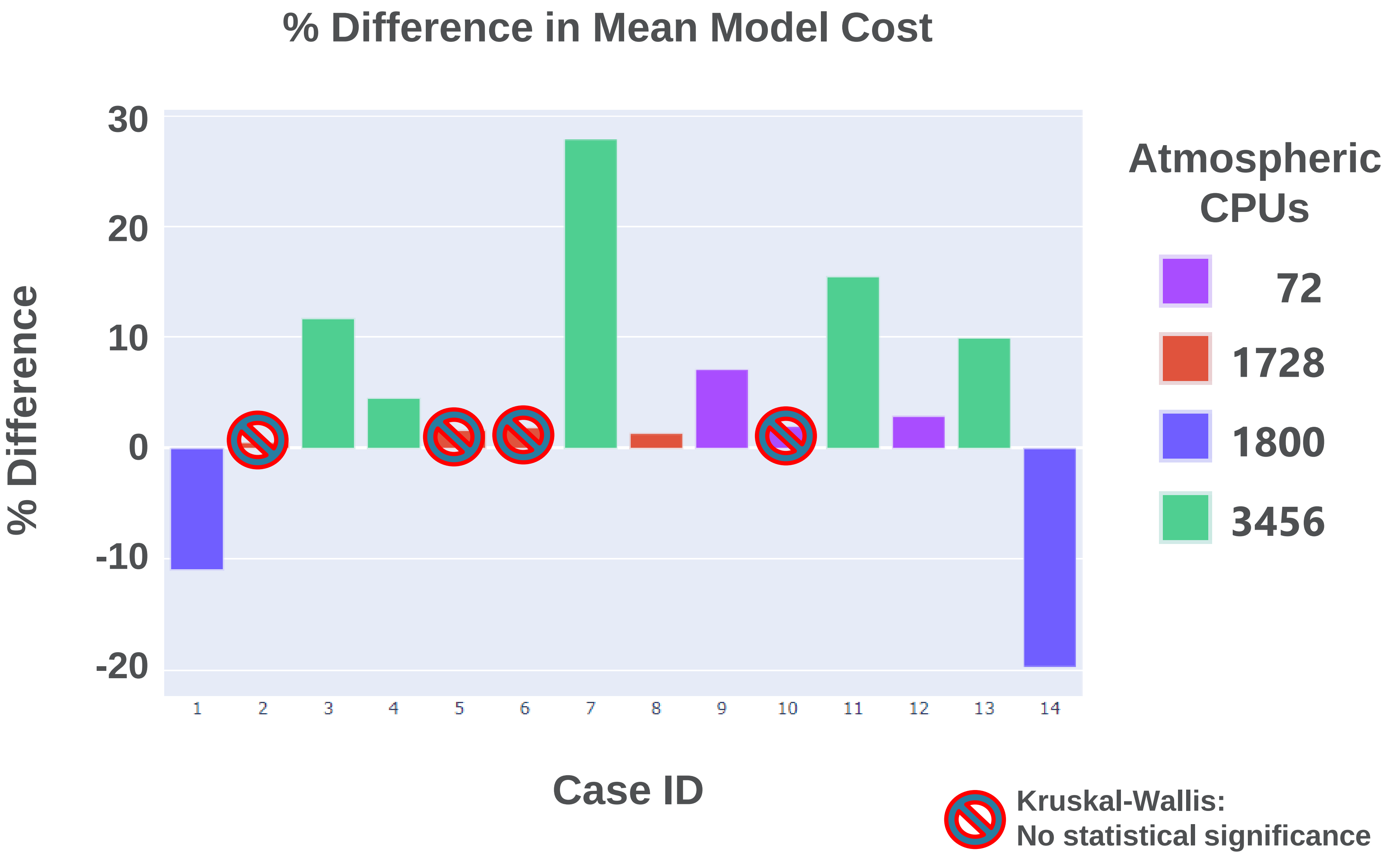
Analysis: System Upgrade

Cases that span the upgrade



Analysis: System Upgrade

Cases that span the upgrade

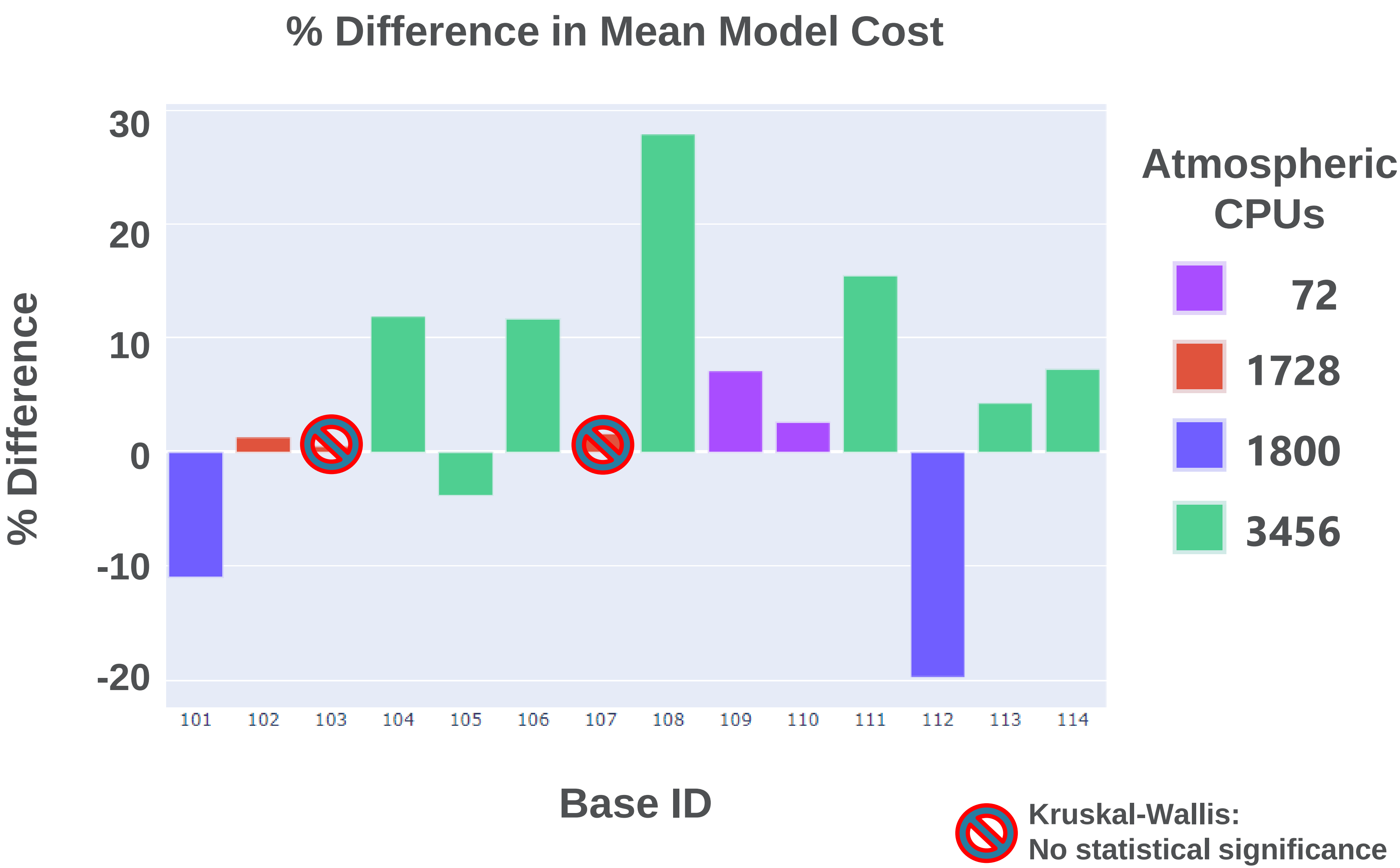


Analysis: System Upgrade

Bases that span the upgrade

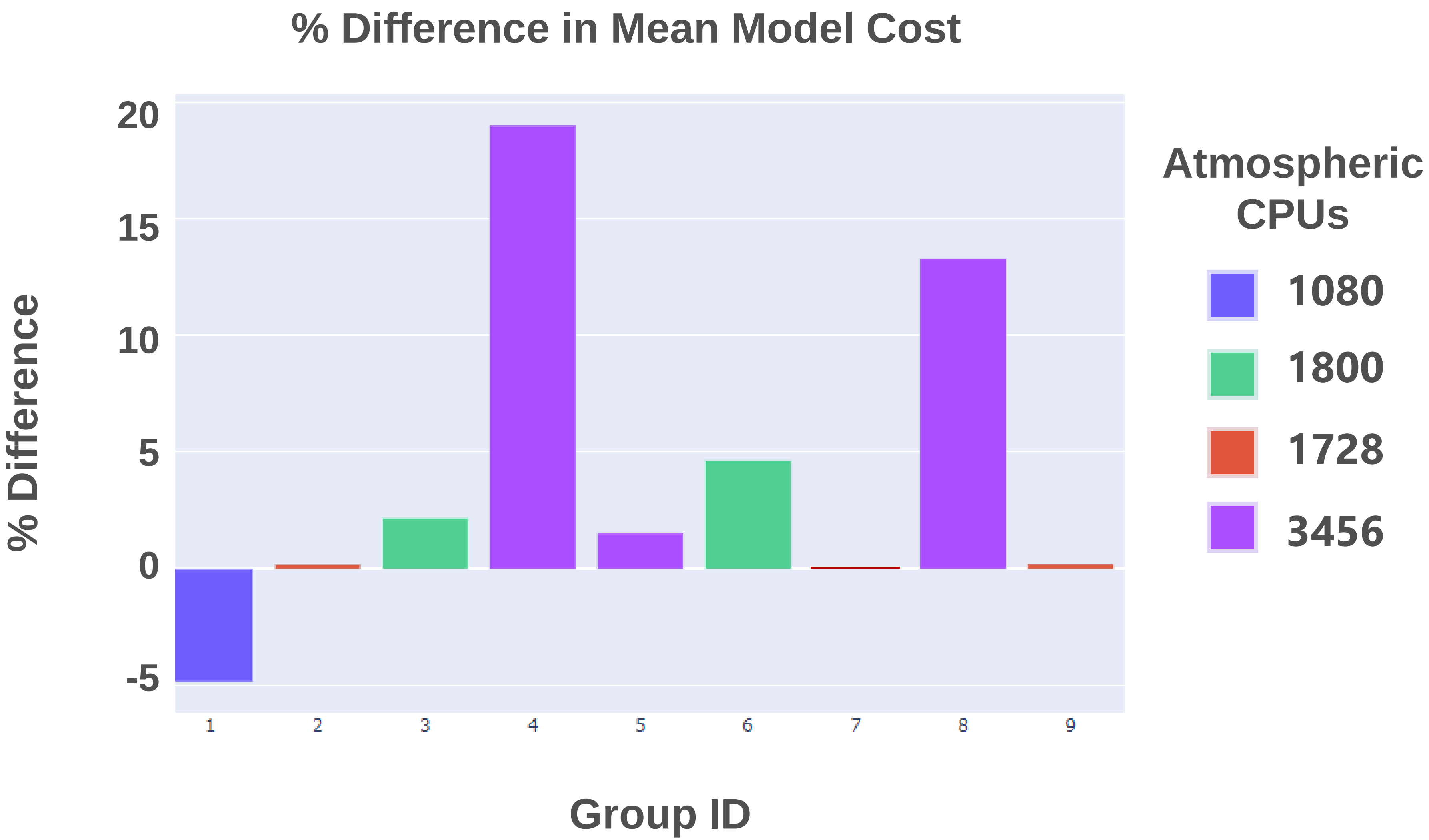
Analysis: System Upgrade

<u>Bases</u> that span the upgrade



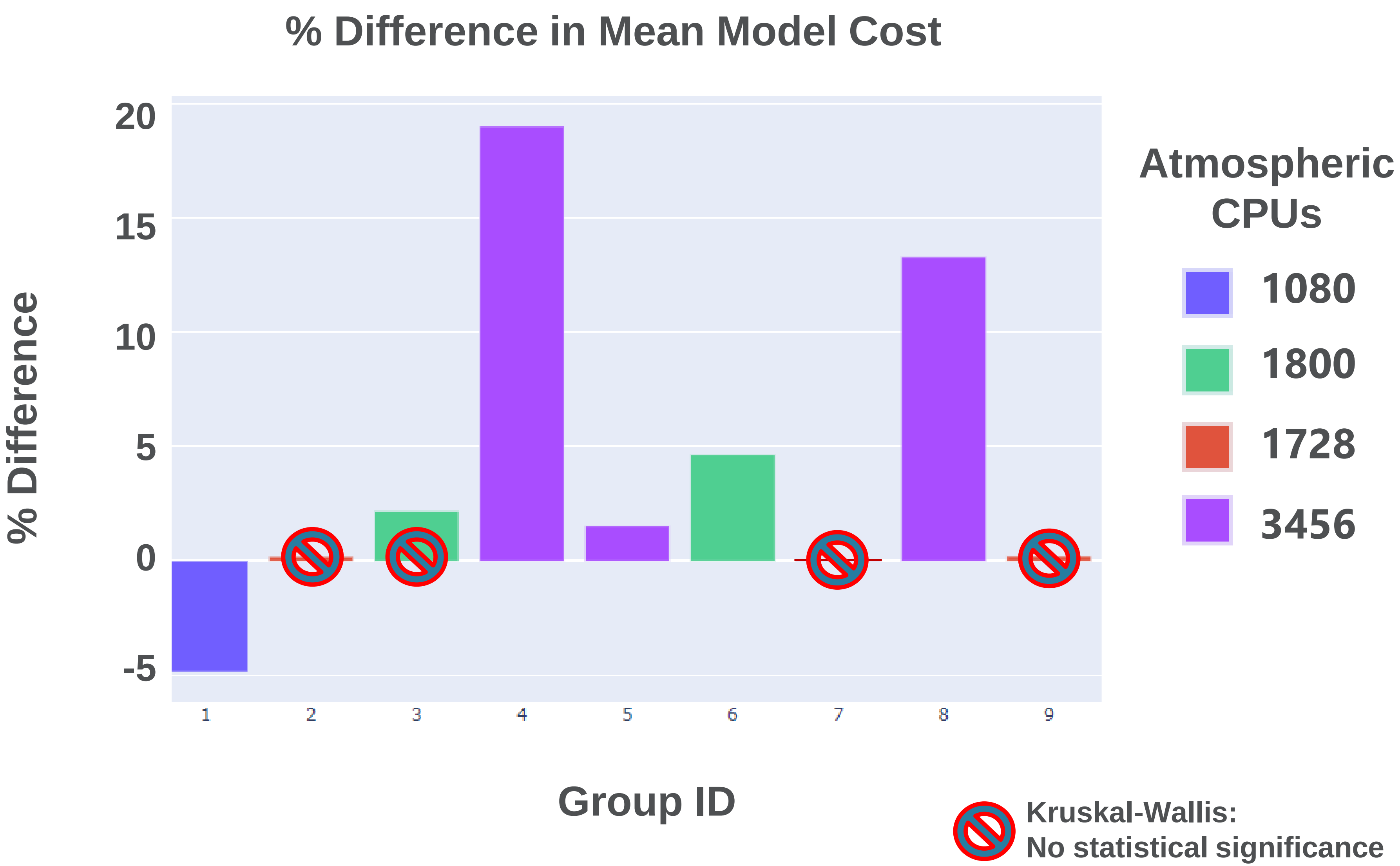
Analysis: System Upgrade

Groups that span the upgrade



Analysis: System Upgrade

<u>Groups</u> that span the upgrade

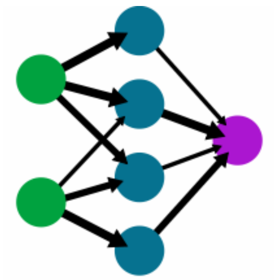


Conclusion

- More analysis in my Jupyter Notebook that points to performance degradation after the upgrade
- Enormous amounts of information to be sliced and diced by each component, physics module, number of processors, and other settings specific to parallel computing
- Answers lead to more questions

Future Work

Machine Learning



- Correlation Plots
- Feature Engineering
- Supervised Learning to Predict Performance (Model Cost)
- Unsupervised Learning (K-Means) to reveal patterns

Acknowledgements

John Dennis
Brian Dobbins
NCAR mentors

AJ Lauer
Virginia Do
NCAR intern managers

Alice Bertini
NCAR SQL Training

Michael Busch
Nate George
Professors at Regis University

References

Balaji, et. al. CPMIP: Measurements of Real Computational Performance of Earth System Models in CMIP6. Geoscience Model Development Issue 10. January 02, 2017. https://www.geosci-model-dev.net/10/19/2017/

Images

Unless otherwise noted, graphics are from www.vecteezy.com

Questions?

Lolita Mannik
lolita@FourWindsPublications.com

This presentation is posted at:
www.FourWindsPublications.com