

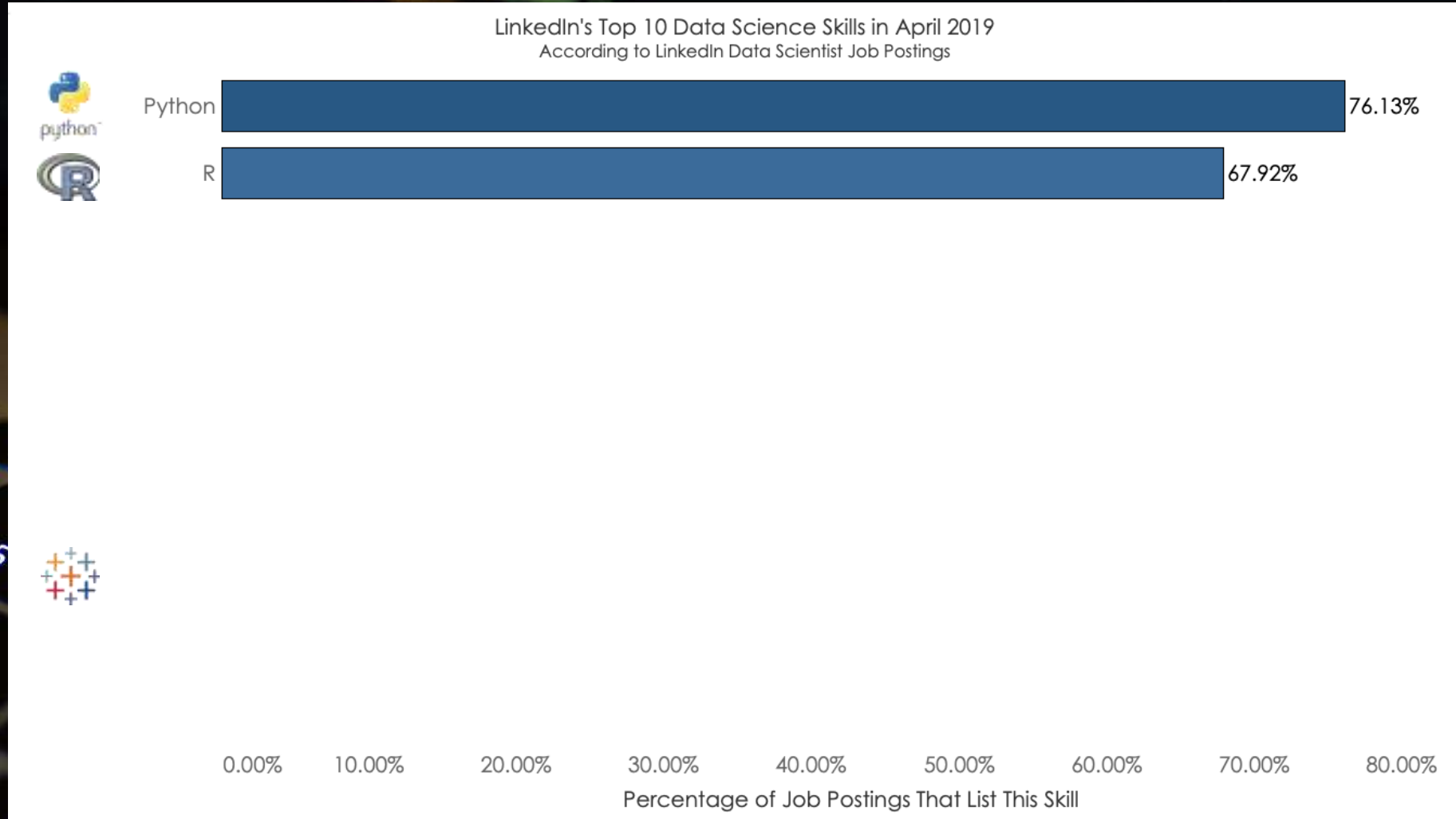
Tableau for Data Scientists

Joel Hutchison
Customer Consultant
Tableau

Understanding the Why

The background is a solid orange gradient. It features several decorative elements: a small circle in the top right, a medium circle in the bottom center, a large circle in the bottom right, and a complex shape on the far right composed of overlapping circles and lines. The text 'Understanding the Why' is centered in the upper half of the image.

Why Python? Why R? Why Tableau?



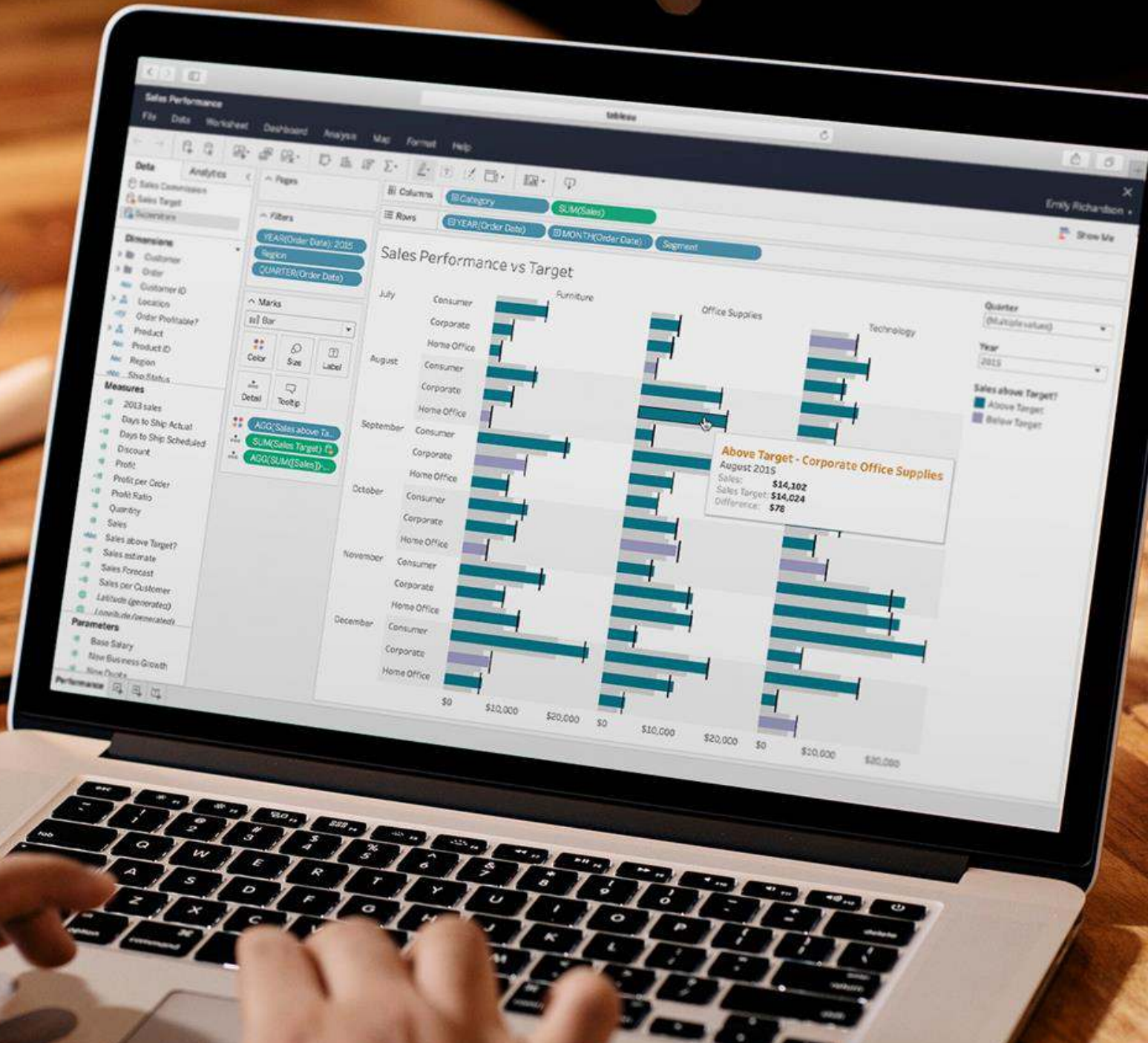
“Visualization of data (static or interactive).

Storytelling with data. This is a critical skill.

In essence, can someone with no background in whatever area your project is in look at your project and gain some new understandings from it?”



We help
people see
and
understand
their data.



Telling your story.



Advanced Analytical Languages

- Peer-reviewed mathematical and statistics packages built by domain experts
- Enrich data with machine learning and natural language processing libraries
- Perform heavy statistical testing
- Create and iterate on regression model



Visual Analytics in Tableau

- Tableau's visual analytics makes it faster and easier to identify patterns, trends and relationships
- Tableau allows users to easily share and communicate insights
- Tableau enables users to ask and answer their own questions

Combined Benefits



- Enable broader audiences to use sophisticated models and statistics in decision-making
- Empower analytical package power-users to uncover more through fluid data exploration
- Enhance the OOTB function-library with available statistical libraries and centralized algorithms
- Easily tell your data story!

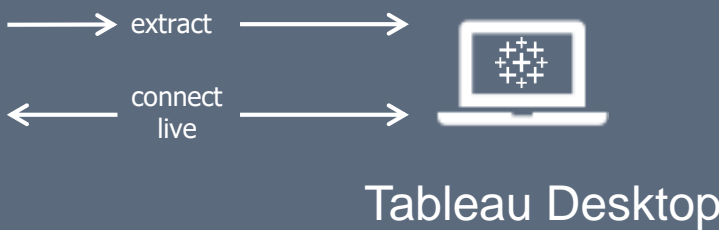
Understanding the How

The background is a solid orange gradient. It features several decorative elements: a small circle in the top right, a medium circle in the bottom center, a large circle in the bottom right, and a complex shape on the far right composed of overlapping circles and lines. The text "Understanding the How" is centered in the upper half of the image.

How does it work?

Data Sources

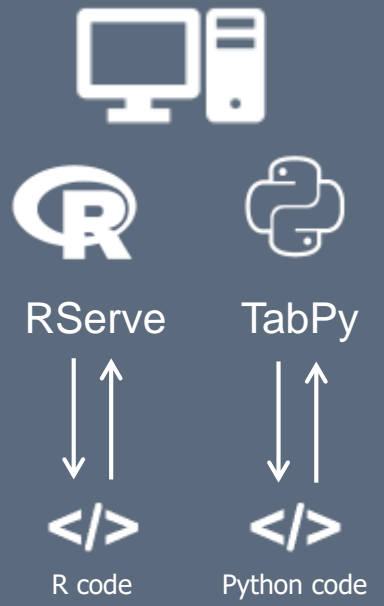
- Files
- Databases
- Big Data
- Cloud
- Apps



workbooks with script_ functions



External Services



Preprocessing the data

Data Sources

- Files
- Databases
- Big Data
- Cloud
- Apps

Preprocess Data



Write to a database or a Tableau Hyper Extract



Tableau Desktop



Tableau Server

External Services



TabPy

The TabPy server **allows for the remote execution of Python code**. It has two components:

- A server process built on Tornado, which allows for the remote execution of Python code through a set of REST APIs.
- A tools library that enables the deployment of such endpoints, based on Python functions

<https://github.com/tableau/TabPy/blob/master/docs/about.md>



Rserve

Rserve is a **TCP/IP server** which allows other programs to use facilities of R from various languages **without the need to initialize R** or link against R library.

- Rserve supports remote connection, authentication and file transfer.

<https://www.rforge.net/Rserve/>

SCRIPT_*() functions in Tableau

```
SCRIPT_BOOL("
library(AnomalyDetection)

timestamp = .arg1
tweets = .arg2

"
MAX([Timestamp]),
SUM([Tweets]))
```

1. Functions telling Tableau to use an external service.

- `SCRIPT_REAL()` returns real or decimal numbers
- `SCRIPT_INT()` returns integers or whole numbers
- `SCRIPT_STR()` returns strings (words and text)
- `SCRIPT_BOOL()` returns Booleans (true/false)

SCRIPT_*() functions in Tableau



2. The actual R / Python code to be executed.

- Tableau treats this as a **string**, sends it to Rserve / TabPy to interpret

SCRIPT_*() functions in Tableau

```
Outlier
Results are computed along Table (across).
SCRIPT_BOOL("
library(AnomalyDetection)

timestamp = .arg1
tweets = .arg2

"
MAX([Timestamp]),
SUM([Tweets]))

```

The screenshot shows the 'Outlier' dialog box in Tableau. The title bar says 'Outlier'. Below the title bar, there is a yellow highlighted area with the text 'Results are computed along Table (across)'. Below this, there is a text area containing a Tableau script. The script starts with 'SCRIPT_BOOL("' and ends with '")'. Inside the quotes, there is a library load 'library(AnomalyDetection)', two variable assignments 'timestamp = .arg1' and 'tweets = .arg2', and a large block of commented-out R code. Below the script, there are two aggregation functions: 'MAX([Timestamp]),' and 'SUM([Tweets]))'. Red circles with numbers 1, 2, 3, and 4 point to specific parts of the script: 1 points to the opening quote, 2 points to the closing quote, 3 points to the aggregation functions, and 4 points to the variable assignments. At the bottom of the dialog, there is a status bar that says 'The calculation is valid.', a dropdown menu for 'Sheets Affected', and two buttons: 'Apply' and 'OK'. The 'Default Table Calculation' text is visible above the 'Apply' and 'OK' buttons.

3. The data from Tableau.

- As many arguments as needed
- Can be [fields] or [parameters]
- All fields must be aggregated

MIN(), MAX(), SUM(), etc.



SCRIPT_*() functions in Tableau



```
Outlier
Results are computed along Table (across).
SCRIPT_BOOL("
library(AnomalyDetection)

timestamp = .arg1
tweets = .arg2

# Example: I generated program (timestamp, "2014-01-01T00:00:00", 1000)
# tweets = new [timestamp, tweet]

# Example: I generated program (timestamp, "2014-01-01T00:00:00", 1000)
# tweets = new [timestamp, tweet]

# Example: I generated program (timestamp, "2014-01-01T00:00:00", 1000)
# tweets = new [timestamp, tweet]

",
MAX([Timestamp]),
SUM([Tweets]))
```

The calculation is valid. Sheets Affected Apply OK

4. The data from Tableau is passed in the code as arguments

- arg1, arg2, arg3, etc. indicates where to put the data into the code
- In example on the left
.arg1 = MAX([Timestamp]), .arg2 = SUM([Tweets])
- R: .arg1, .arg2, etc.
- Python: _arg1, _arg2, etc.

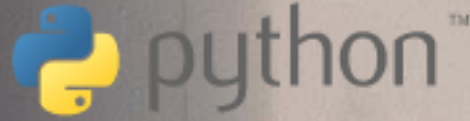


The Nuts and Bolts

The background is a solid orange gradient. It features several decorative elements: a small circle in the upper right, a medium circle in the lower center, a large circle in the lower right, and a complex shape on the far right composed of overlapping circles and lines. A partial circle is visible on the left edge.

Installing TabPy

1. Install Python



2. Install TabPy

- `pip install tabpy-server`

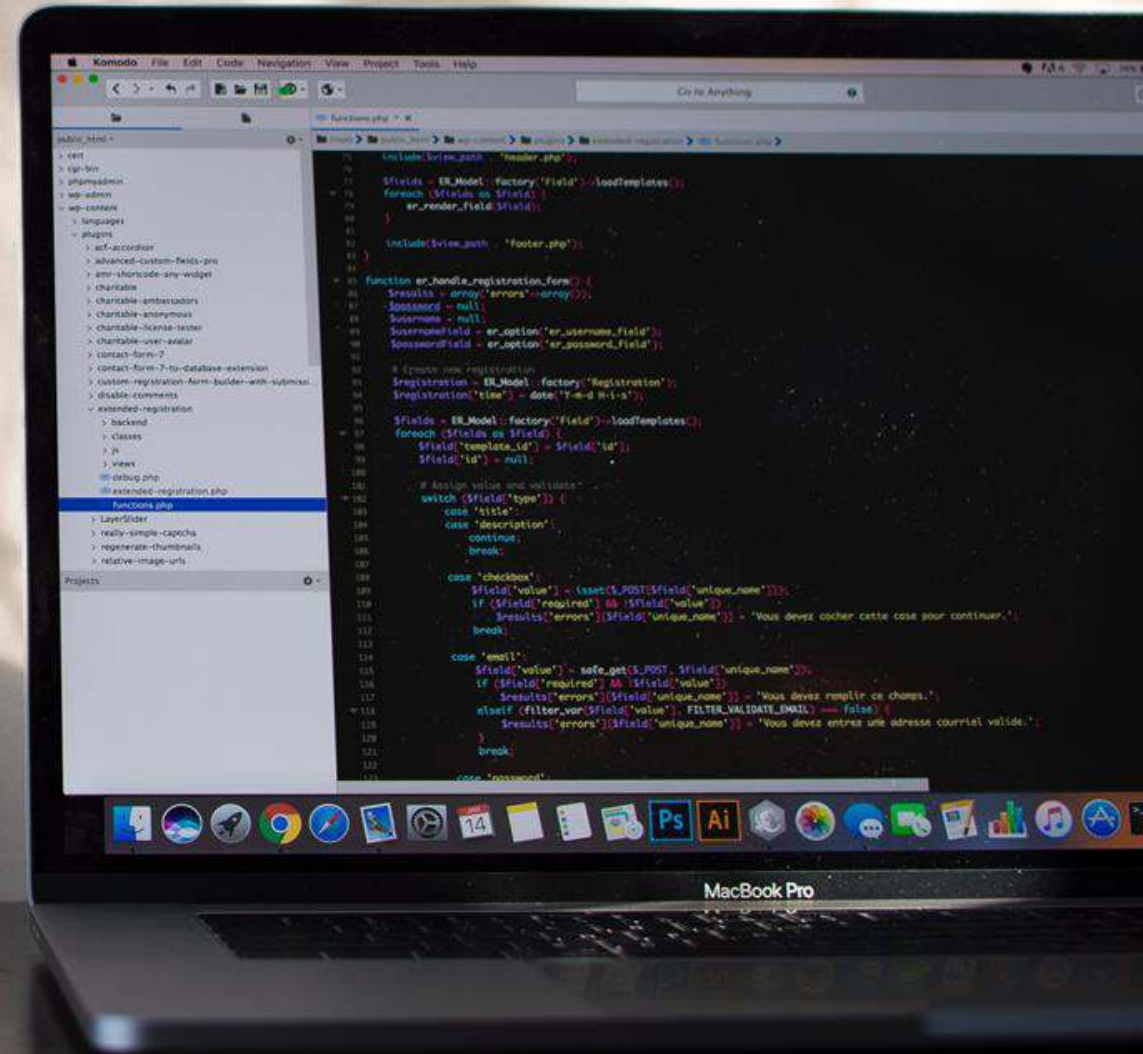
1. Install required python modules

- `python -m pip install numpy scipy pandas statsmodels patsy sklearn nltk`

2. Initialize sentiment lexicon on Python console

- `import nltk`
`nltk.download('vader_lexicon')`

3. Start Tabpy from the command line



More details on the install can be found on [Github](#).

Install RServe

1. Install R

2. Optionally install Rstudio

3. Run R (IDE like RStudio, GUI, CLI)

4. Install required packages

- `install.packages(c("Rserve", "forecast", "dbscan", "dplyr", "tidytext"))`

5. Start Rserve session

- `library(Rserve)`
`run.Rserve()`



Connect Tableau Desktop to Rserve / TabPy

External Service Connection


Select an External Service
Rserve

Specify a server name and a port
Server: localhost Port: 6311

Sign in with a username and password
Username: Password:

Require SSL

Test Connection OK Cancel



Help

- Open Help F1
- Get Support...
- Check for Product Updates...
- Watch Training Videos
- Sample Workbooks
- Sample Gallery
- Choose Language
- Settings and Performance
- Manage Product Keys...
- About Tableau

- Reset Ignored Messages
- Clear Saved Server Sign-ins
- Enable Automatic Product Updates
- Enable Autosave
- Enable Accelerated Graphics
- Manage External Service Connection...
- Set Dashboard Web View Security
- Start Performance Recording

External Service Connection


Select an External Service
TabPy/External API

Specify a server name and a port
Server: localhost Port: 9004

Sign in with a username and password
Username: Password:

Require SSL

Test Connection OK Cancel



Connect Tableau Server to Rserve / TabPy



```
tsm configuration set -k vizqlserver.extsvc.host -v <IP>
```

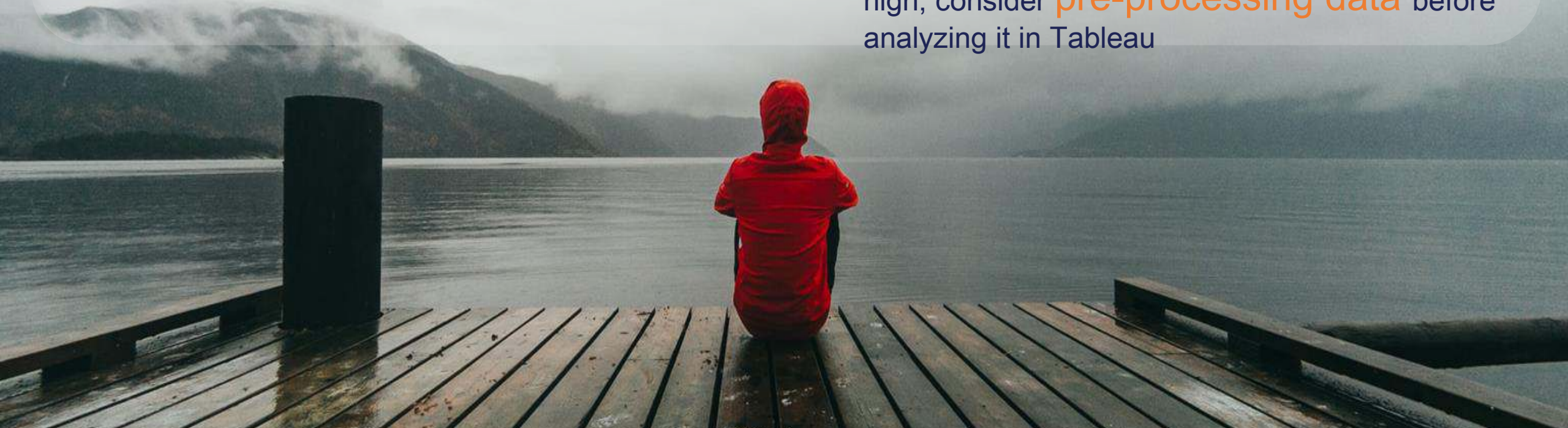
```
tsm configuration set -k vizqlserver.extsvc.port -v <port>
```

Additional Considerations

The background is a solid orange gradient. It features several decorative elements: a small circle in the upper right, a medium circle in the lower center, a large circle in the lower right, and a complex shape on the far right composed of overlapping circles and lines. The text "Additional Considerations" is centered in white.

Additional Considerations

1. Tableau Desktop and Server currently only support **one External Service**
2. No support for External Services with **Tableau Online** and **Tableau Public**
3. Security and best practices require putting External Services on a **separate machine** and limiting access
4. If latency for calculation processing times are high, consider **pre-processing data** before analyzing it in Tableau

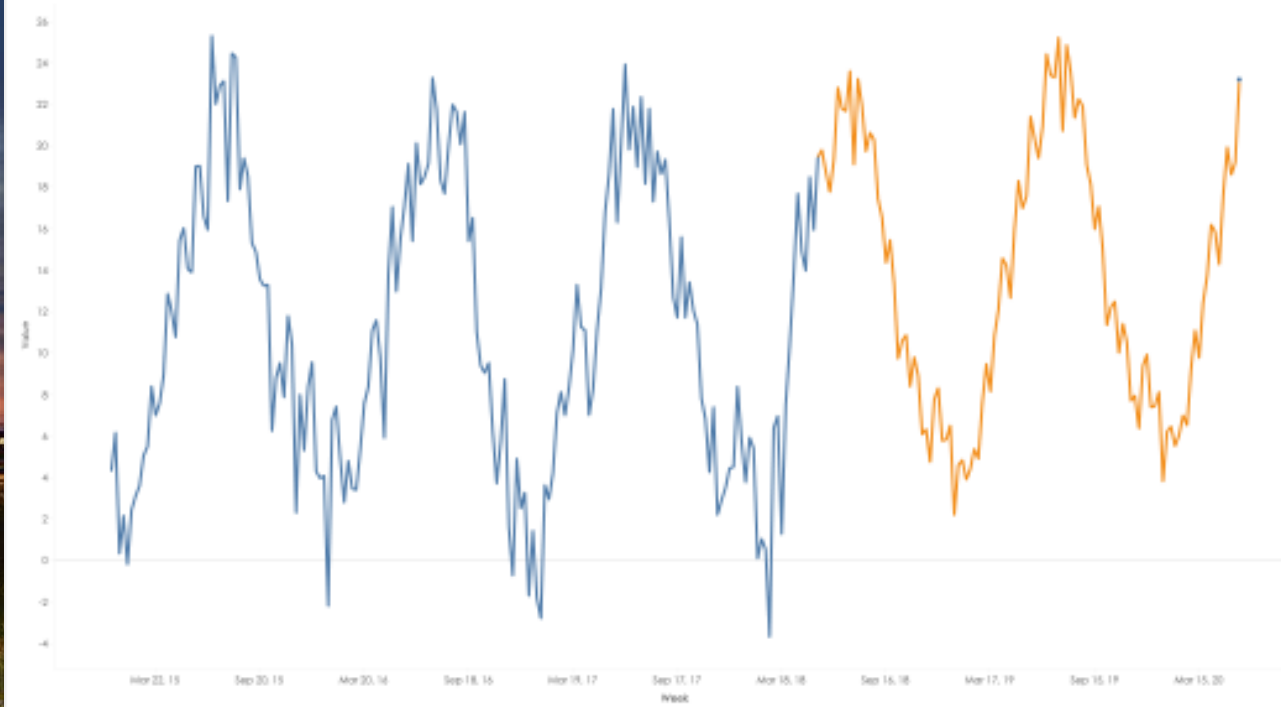


Use Cases

The background is a solid orange color. It features several decorative elements: a small white circle in the upper right, a medium white circle in the lower center, and a large white circle on the right side that overlaps with other lines. There are also several overlapping white lines of varying thicknesses that form abstract shapes and patterns across the lower half of the page.

Forecasting Time Series Data

Python - Forecasting - Frankfurt Temperatures



R - Forecasting - Frankfurt Temperatures



Forecasting Time Series Data

```
SCRIPT_REAL("
library(forecast)

inputData = na.omit(.arg1)
startDate = as.Date(min(na.omit(.arg2)))

timeSeries = ts(inputData,
                start = startDate,
                deltat = 1/52)

timeSeriesForecast = forecast(timeSeries,
                              h = length(.arg1) -
                                length(inputData),
                              level = 95)

append(inputData,
        timeSeriesForecast$mean)
",
AVG([Temperature]),
MAX([forecastWeek]))
```



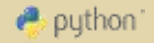
```
SCRIPT_REAL("
import numpy as np
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing

series = pd.DataFrame.from_items([('ts', _arg1), ('y',
_arg2)])
last_week = np.where(pd.isnull(series))[0][0]
weeks_to_forecast = len(series) - last_week

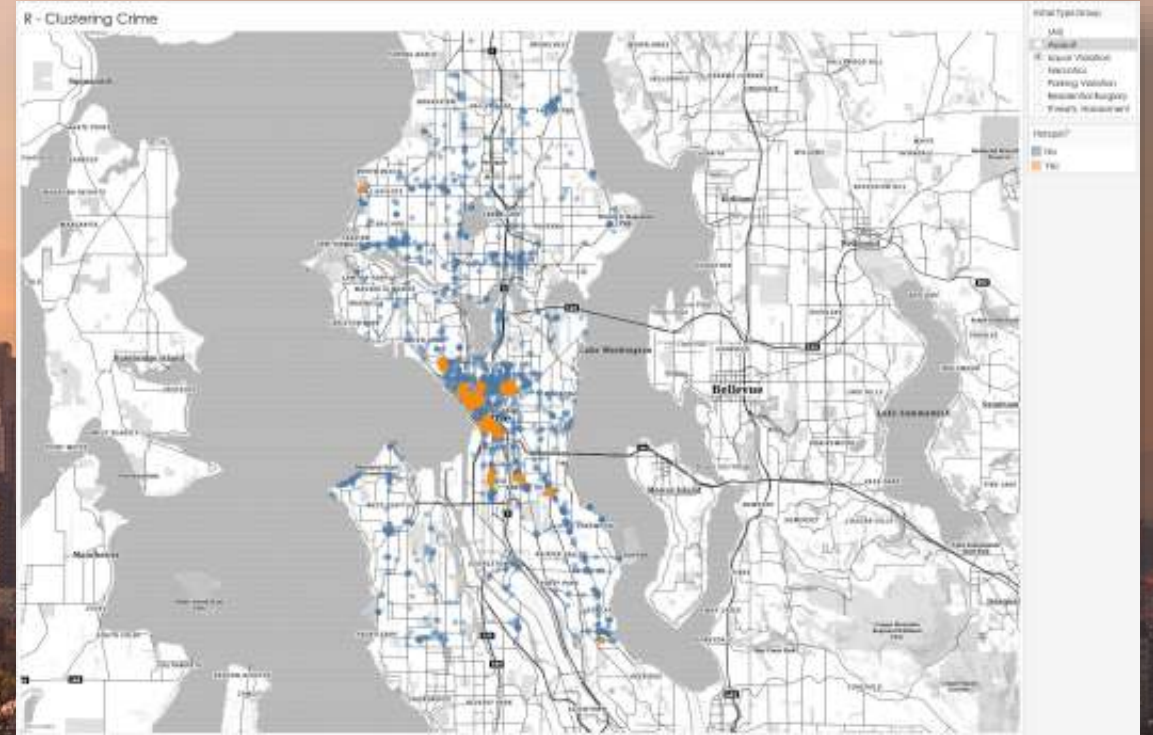
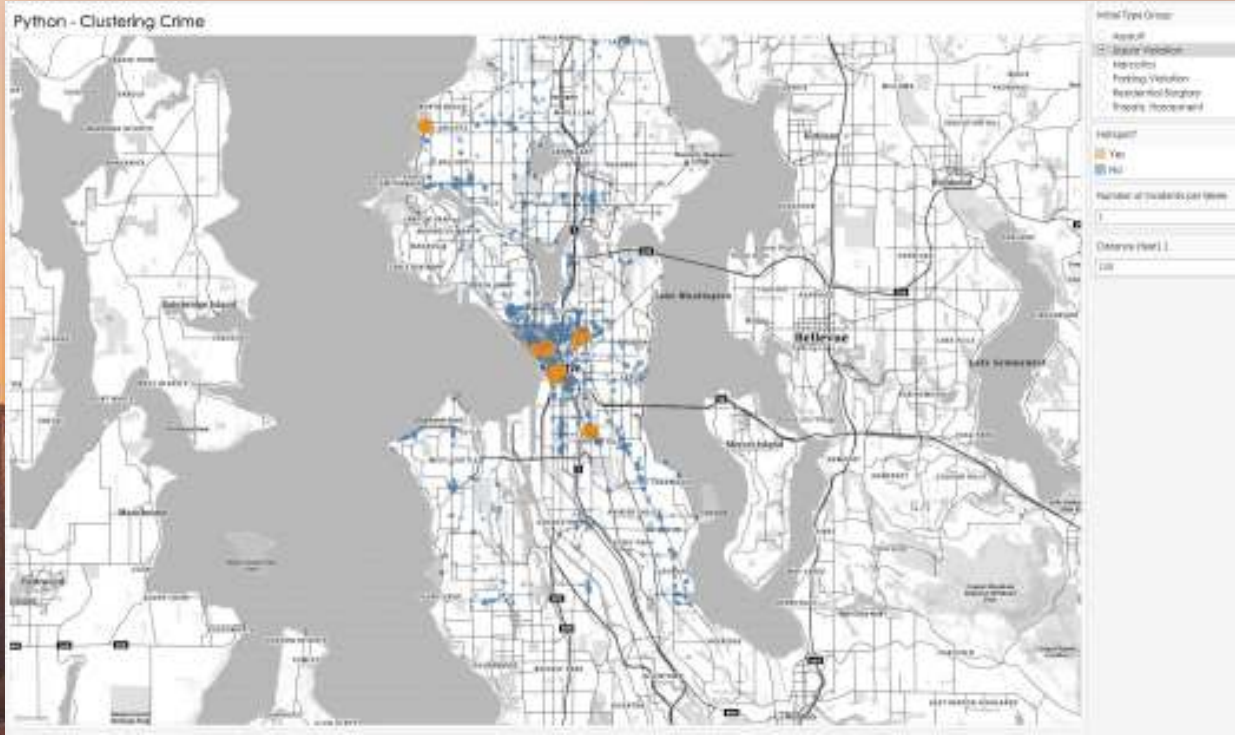
model_fit = ExponentialSmoothing(series.iloc[:last_week, 1],
seasonal_periods=52, trend='add', seasonal='add').fit()

yhat = model_fit.forecast(weeks_to_forecast)

return np.concatenate([series.iloc[:last_week, 1],
yhat]).tolist()
",
AVG([Temperature]),
MAX([forecastWeek]))
```



Clustering Crime



Clustering Crime

```
SCRIPT_STR("
library(dbscan)

data <- cbind((.arg1 * pi) / 180, (.arg2 * pi) / 180)

db <- dbscan(data,
             eps = 1/39590,
             minPts = .arg3[1])$cluster

db[db > 0] <- 'Yes'
db[db == 0] <- 'No'

db
",
AVG([Latitude]),
AVG([Longitude]),
AVG([Incident Count]))
```

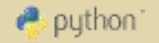


```
SCRIPT_STR("
import numpy as np
from sklearn.cluster import DBSCAN

X = np.column_stack([np.radians(_arg1), np.radians(_arg2)])

db = DBSCAN(eps=_arg3[1], min_samples=_arg4[1],
            metric='haversine').fit(X)

return np.where(db.labels_ == np.array(-1), \
               'No', 'Yes').tolist()
",
AVG([Latitude]),
AVG([Longitude]),
[Distance between incidents]
AVG([Incident Count]))
```



DEMO



Thank You

