



Rapid Deployment of Automatic Scoring Models to Hadoop Production Systems

System Overview, Case Study, and Recommendations

 @amitaigolub

 amitai.golub@innogames.com

Who am I? Who is InnoGames?

About me:

- Studied Statistics in the Hebrew University in Jerusalem, with a focus on non-numerical data and graphical models.
- Consultant for bio-tech, gambling and IT companies.
- Since 2014 data mining analyst at InnoGames GmbH in Hamburg, Germany

About InnoGames:

- Started as a hobby-project in 2003 by our managing directors; The first game “Tribal Wars” is still growing.
- F2P business model: moved from browser to cross platform developer & publisher.
- +350 employees from 30 nations.
- 135M registered player.

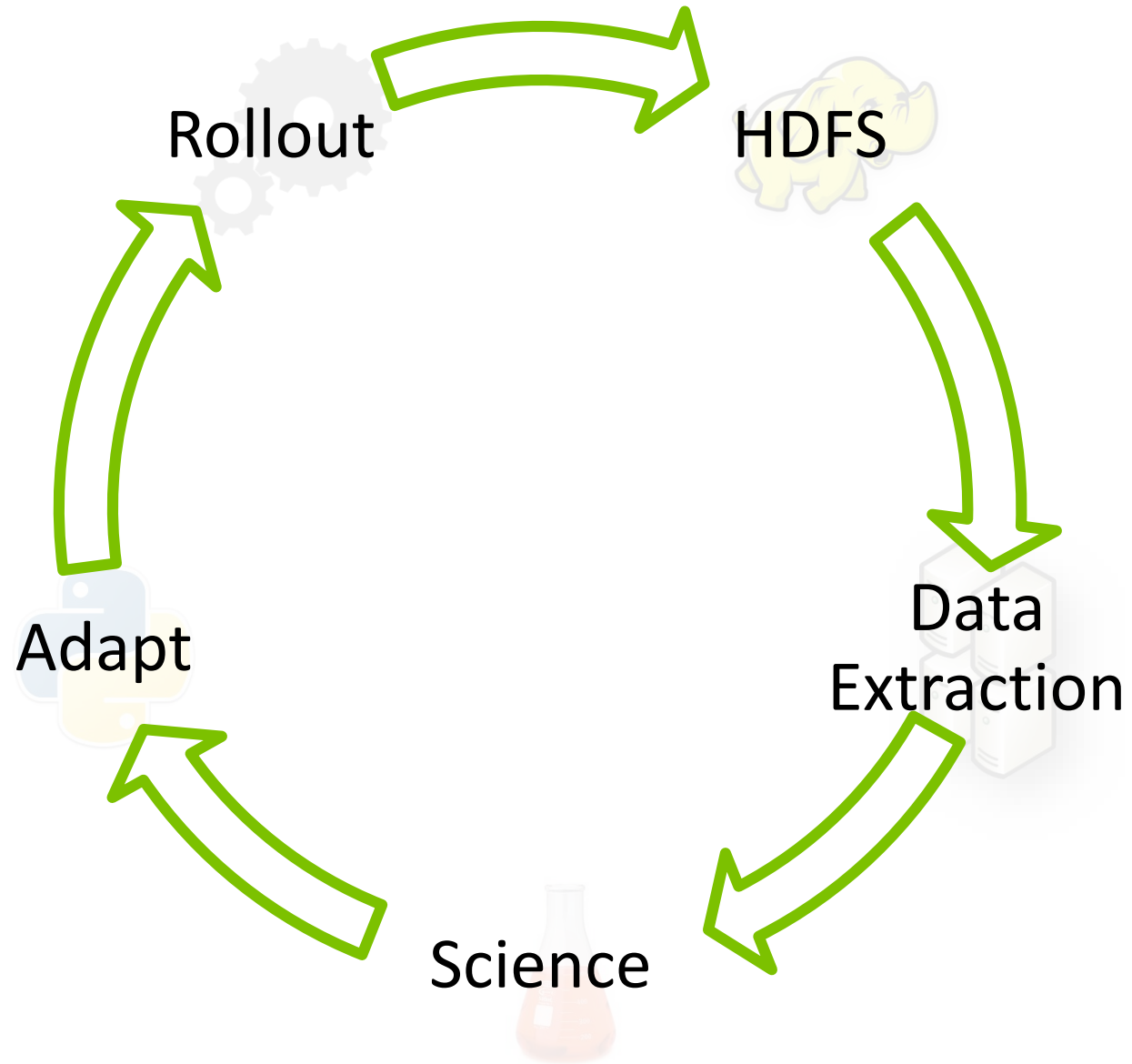




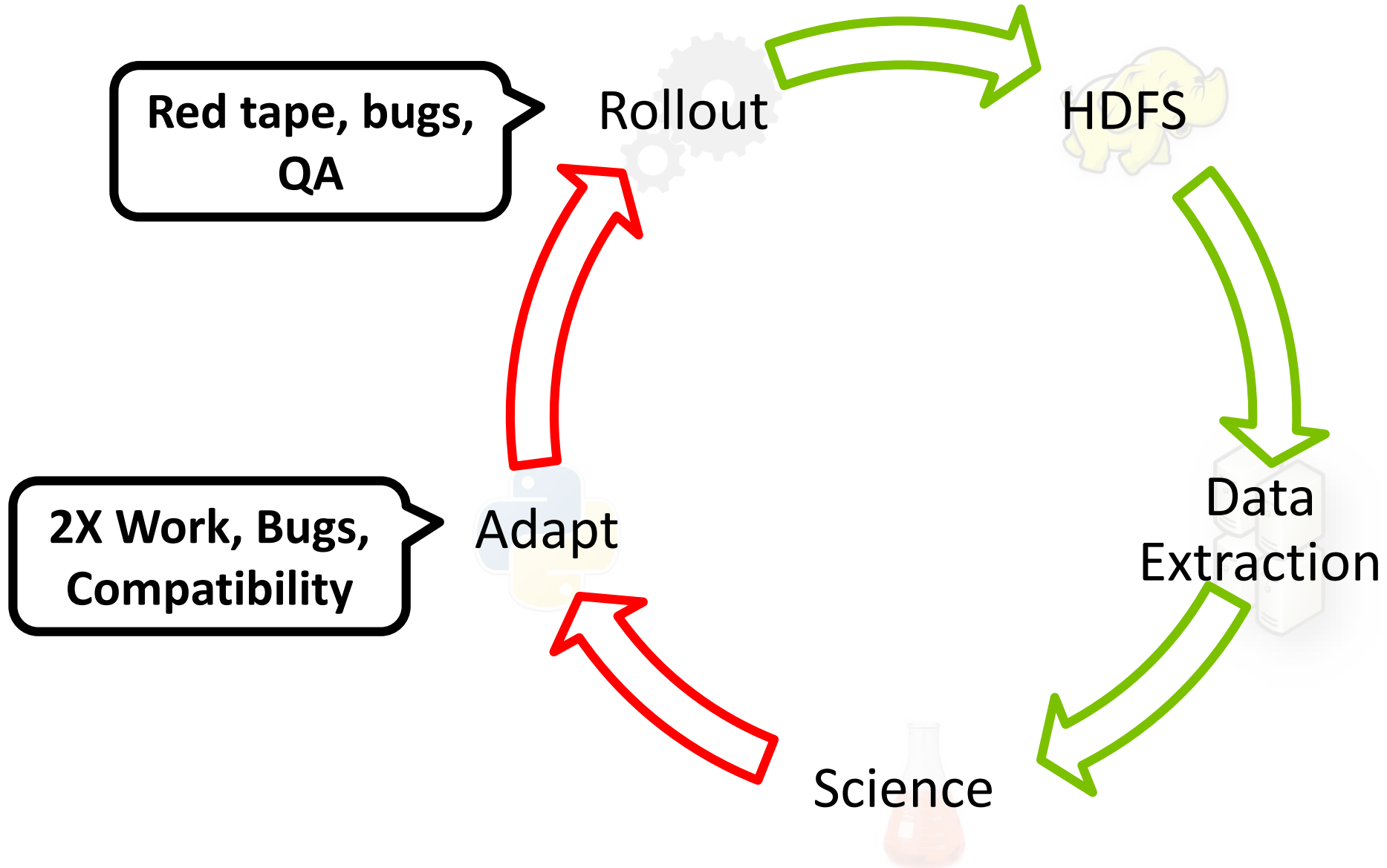
Motivation

Data Scientists are data consumers, but have a hard time becoming data suppliers. I want to help data scientists become efficient suppliers of data.

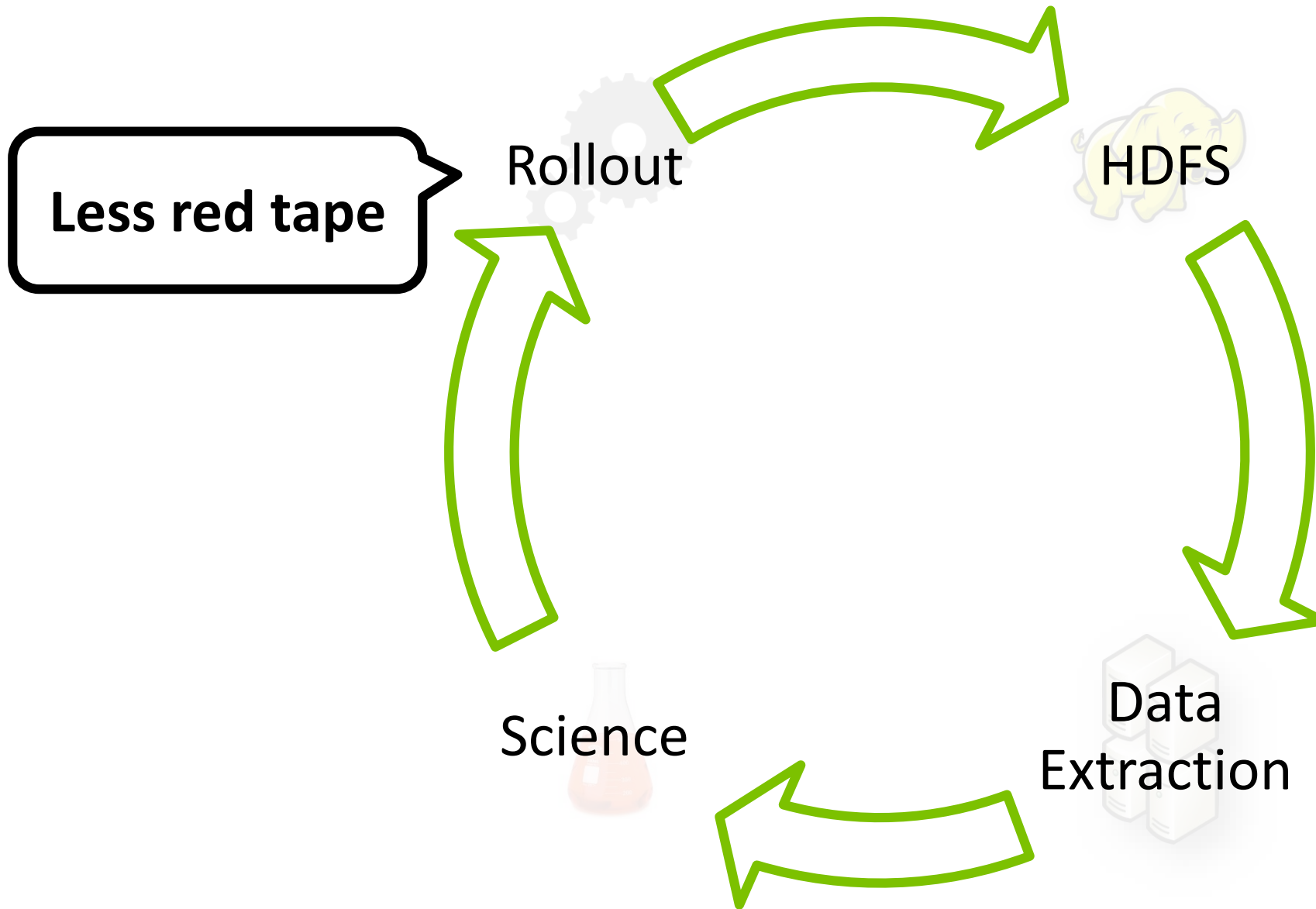
Production Cycle



Production Cycle



Production Cycle



The New Packages You Need + Some Setup



rmr2

- The basic building blocks of RHadoop.
- Allows building MapReduce functions in R

Rhdfs

- HDFS file management from within R (hdfs terminal)
- Read, write, copy, rename, etc., etc.

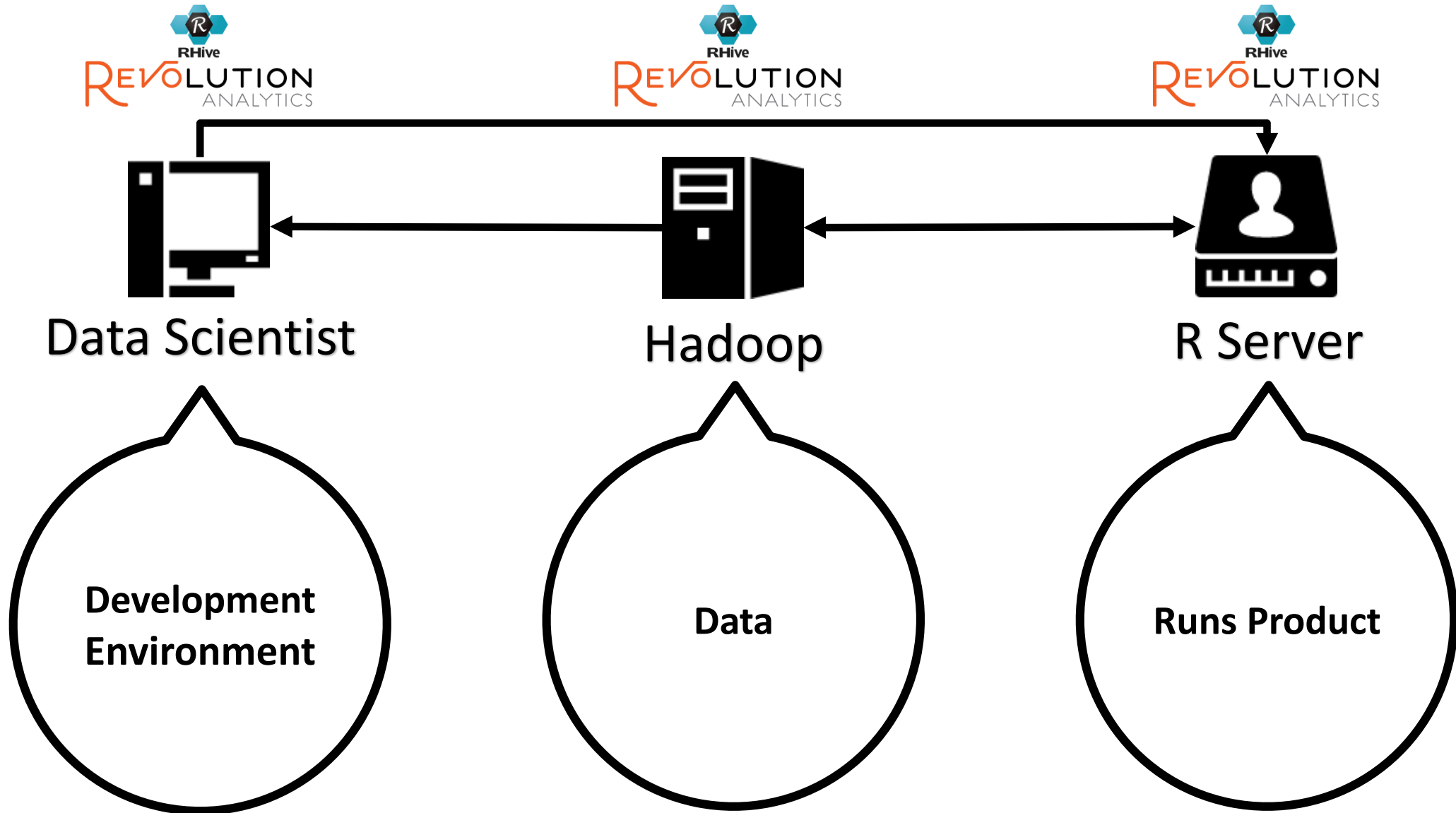
plyrmr

- A higher level function toolkit.
- This is the package the Data Scientist will most likely want to use when analyzing data en masse.

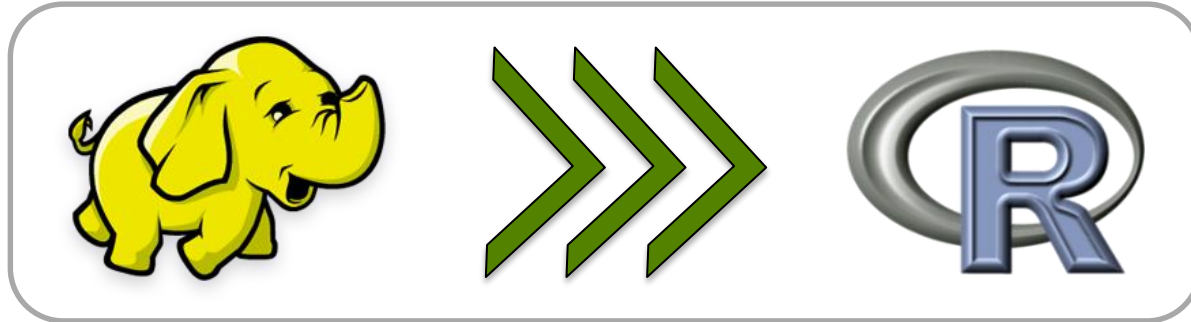
Rhive

- Hive functionality in R.
- Great replacement for ODBC driver issues
- Also contains HDFS functions
- ***Doesn't work on Windows***

Setup




```
get_score(get_data, model_data, make_output, ...)
```



get_score(get_data, **model_data**, make_output, ...)



`get_score(get_data, model_data, make_output, ...)`



Initialization

```
> require('RHive')
> require('mixtools')
>
> Sys.setenv("HADOOP_CMD" = '/usr/bin/hadoop')
> Sys.setenv("HADOOP_STREAMING" = '/usr/lib/hadoop-mapreduce/hadoop-streaming.jar')
>
> require('plyrmr')
>
> rhive.init(hiveHome    = '/usr/lib/hive',
+           hadoopHome  = '/usr/lib/hadoop',
+           hadoopConf  = '/etc/hadoop/conf/')
>
> rhive.connect(host      = '<ip_address>',
+               port      = 10000,
+               hiveServer2 = T,
+               defaultFS  = 'hdfs_namenode_url',
+               user       = 'hdfs',
+               password   = '')
```

Process

```
> get_data_mvnormal <- function(table.name){  
+   query <- paste('select * from', table.name)  
+   return(rhive.query(query = query))  
+ }
```

```
> model_data_mvtnorm_em <- function(data){  
+   res <- mvnormalmixEM(data)  
+   score <- apply(res$posterior, 1, which.max)  
+   return(cbind(data, score))  
+ }
```

```
> make_output_matrix_to_hive <- function(score.df){  
+   rhive.write.table(score.df, 'results', rowName = F)  
+ }
```

```
> get_score <- function(get_data, model_data, make_output, ...){  
  extra_input <- list(...)  
  
  Get_Data      <- match.fun(get_data)  
  Model_Data    <- match.fun(model_data)  
  Make_Output   <- match.fun(make_output)  
  
  Get_Data(extra_input$table_name) %>%  
    Model_Data  
  
  Make_Output(results)  
  
  return(0)  
}
```

Process

```
> get_data_mvnormal <- function(table.name){  
+   query <- paste('select * from', table.name)  
+   return(rhive.query(query = query))  
+ }
```

```
> model_data_mvtnorm_em <- function(data){  
+   res <- mvnormalmixEM(data)  
+   score <- apply(res$posterior, 1, which.max)  
+   return(cbind(data, score))  
+ }
```

```
> make_output_matrix_to_hive <- function(score.df){  
+   rhive.write.table(score.df, 'results', rowName = F)  
+ }
```

Process

```
> get_data_mvnormal <- function(tbl_name_1, tbl_name_2){  
+   query_1 <- paste('select * from', tbl_name_1)  
+   query_2 <- paste('select * from', tbl_name_1)  
+   res_1 <- rhive.query(query = query_1)  
+   res_2 <- rhive.query(query = query_2)  
+  
+   p_res <- combine_and_process_data(res_1, res_2)  
+   return(p_res)  
+ }
```

```
> model_data_mvtnorm_em <- function(data){  
+   res <- mvnormalmixEM(data)  
+   score <- apply(res$posterior, 1, which.max)  
+   return(cbind(data, score))  
+ }
```

```
> make_output_matrix_to_hive <- function(score.df){  
+   rhive.write.table(score.df, 'results', rowName = F)  
+ }
```


(Greatly) Increasing Complexity

main.R

```
## main.R
##
## @title Main script
## @author Amitai Golub
## @description
## This script runs the game simulation and outputs the results.
##
## Usage
##   main.R
##
## Parameters
##   --seed: Seed for the random number generator.
##   --n: Number of iterations.
##   --n_players: Number of players.
##   --n_games: Number of games.
##   --n_rounds: Number of rounds per game.
##   --n_steps: Number of steps per round.
##   --n_actions: Number of actions per step.
##   --n_states: Number of states per step.
##   --n_rewards: Number of rewards per step.
##   --n_costs: Number of costs per step.
##   --n_actions_per_step: Number of actions per step.
##   --n_states_per_step: Number of states per step.
##   --n_rewards_per_step: Number of rewards per step.
##   --n_costs_per_step: Number of costs per step.
##   --n_actions_per_round: Number of actions per round.
##   --n_states_per_round: Number of states per round.
##   --n_rewards_per_round: Number of rewards per round.
##   --n_costs_per_round: Number of costs per round.
##   --n_actions_per_game: Number of actions per game.
##   --n_states_per_game: Number of states per game.
##   --n_rewards_per_game: Number of rewards per game.
##   --n_costs_per_game: Number of costs per game.
##   --n_actions_per_simulation: Number of actions per simulation.
##   --n_states_per_simulation: Number of states per simulation.
##   --n_rewards_per_simulation: Number of rewards per simulation.
##   --n_costs_per_simulation: Number of costs per simulation.
##
## Output
##   Results of the simulation.
##
## Notes
##   This script is a simplified version of the game simulation.
##   It does not include the full game logic or the full state space.
##   It is intended for testing and debugging purposes only.
```

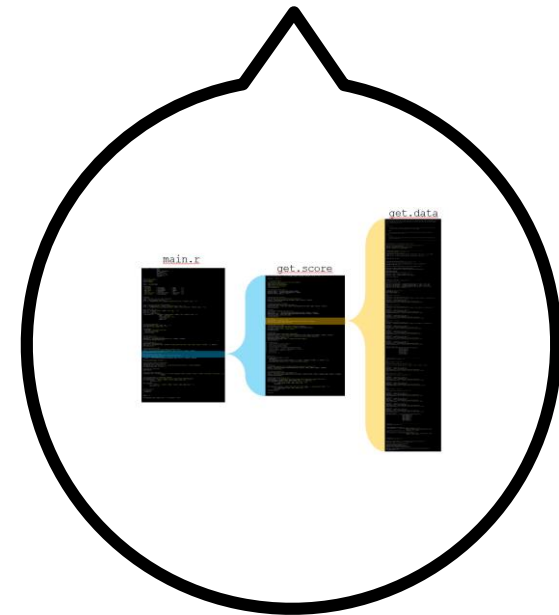
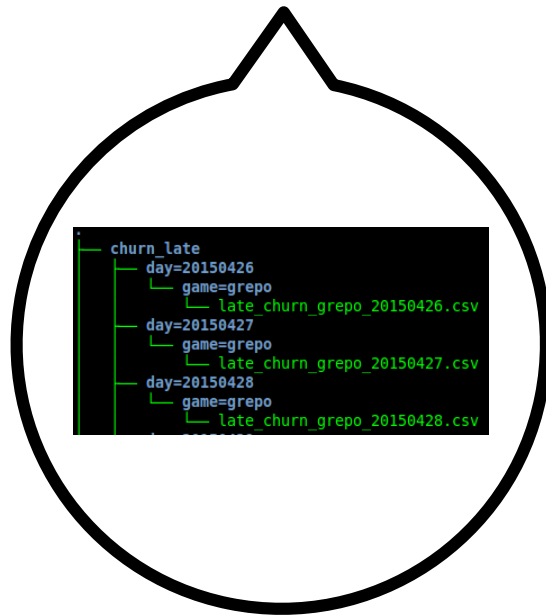
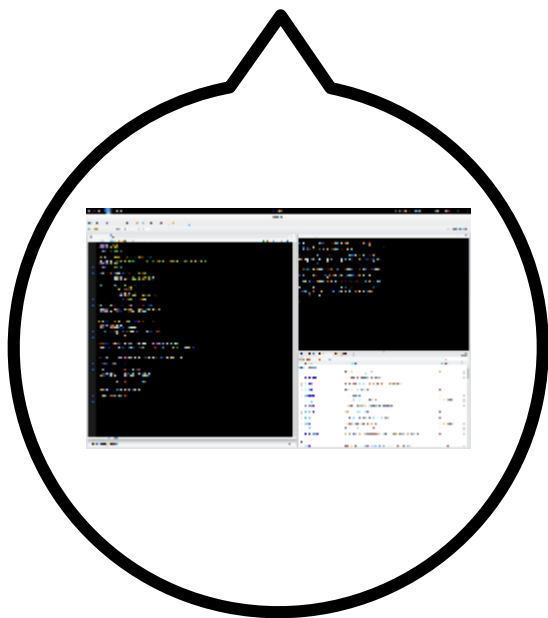
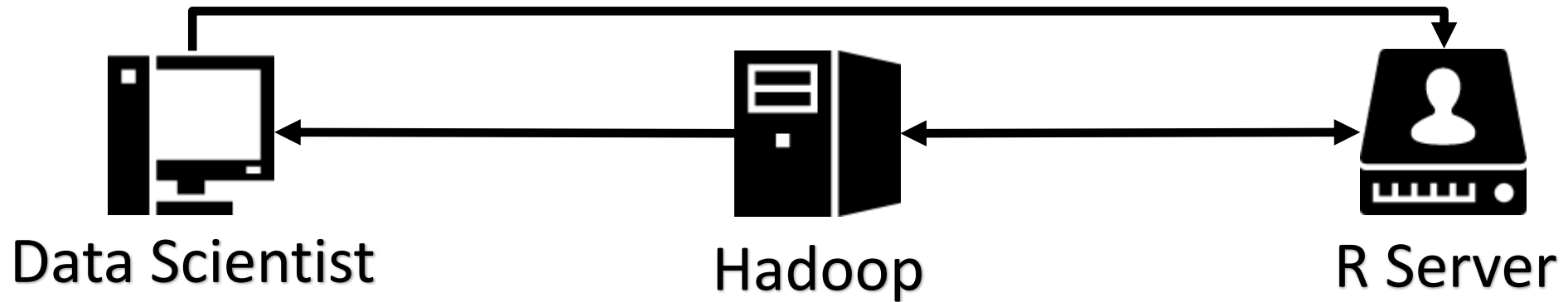
Get_score

```
## Get_score
##
## @title Get score function
## @author Amitai Golub
## @description
## This function calculates the score for a given state.
##
## Usage
##   Get_score(state)
##
## Parameters
##   state: A vector representing the state of the game.
##
## Output
##   A scalar representing the score for the given state.
```

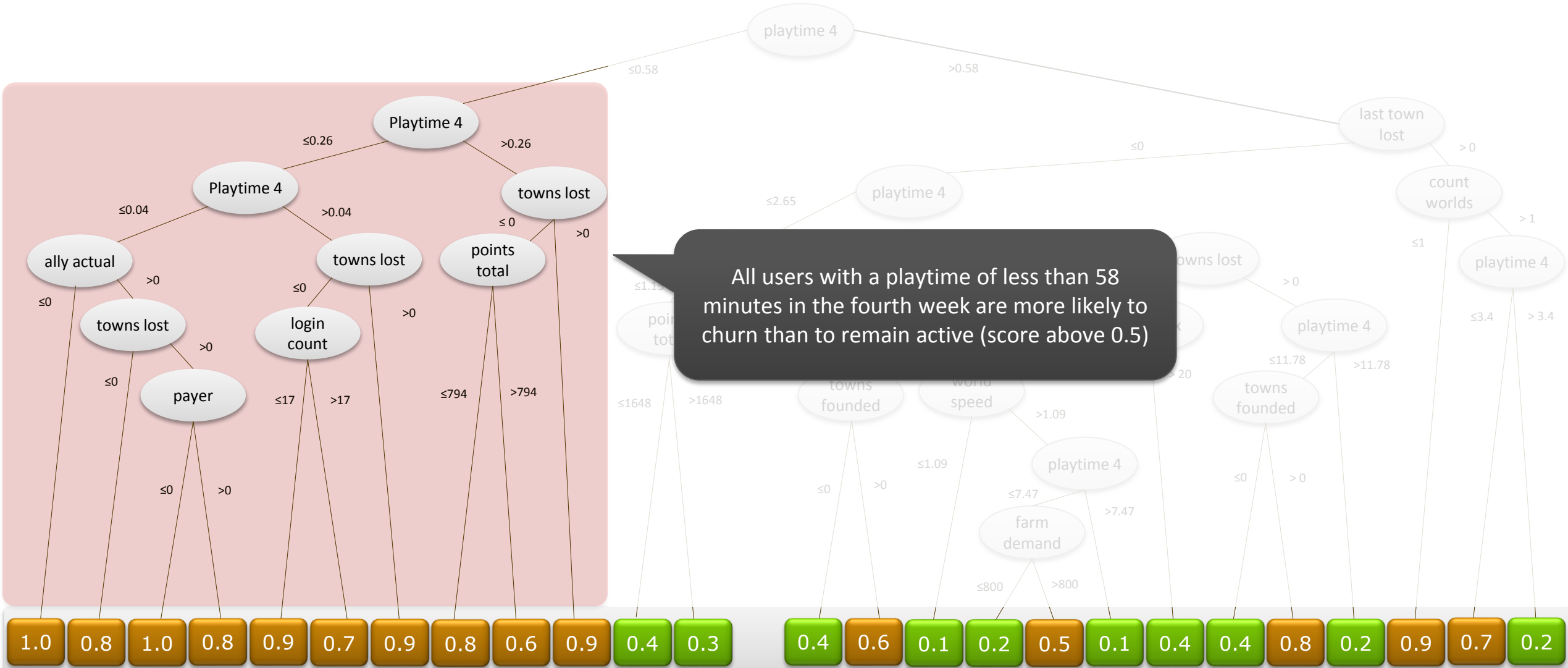
get data

```
## get data
##
## @title Get data function
## @author Amitai Golub
## @description
## This function retrieves the data for a given state.
##
## Usage
##   get_data(state)
##
## Parameters
##   state: A vector representing the state of the game.
##
## Output
##   A list of data for the given state.
```

Result



Churn Models



Social Network Analysis

