# The CloudyR Project: Statistical Cloud Computing in R with Amazon and Google

**Thomas J. Leeper**

London School of Economics and Political Science

Twitter: **@thosjleeper @cloudyrproject**
GitHub: **@leeper @cloudyr**

**thosjleeper@gmail.com**

This talk is about cloud computing.

What is that?

*Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it. . .*

– Dan Ariely, 2013

# Cloud computing

*~~Big data~~ is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it. . .*

– Dan Ariely, 2013


There is no cloud
it's just someone else's computer

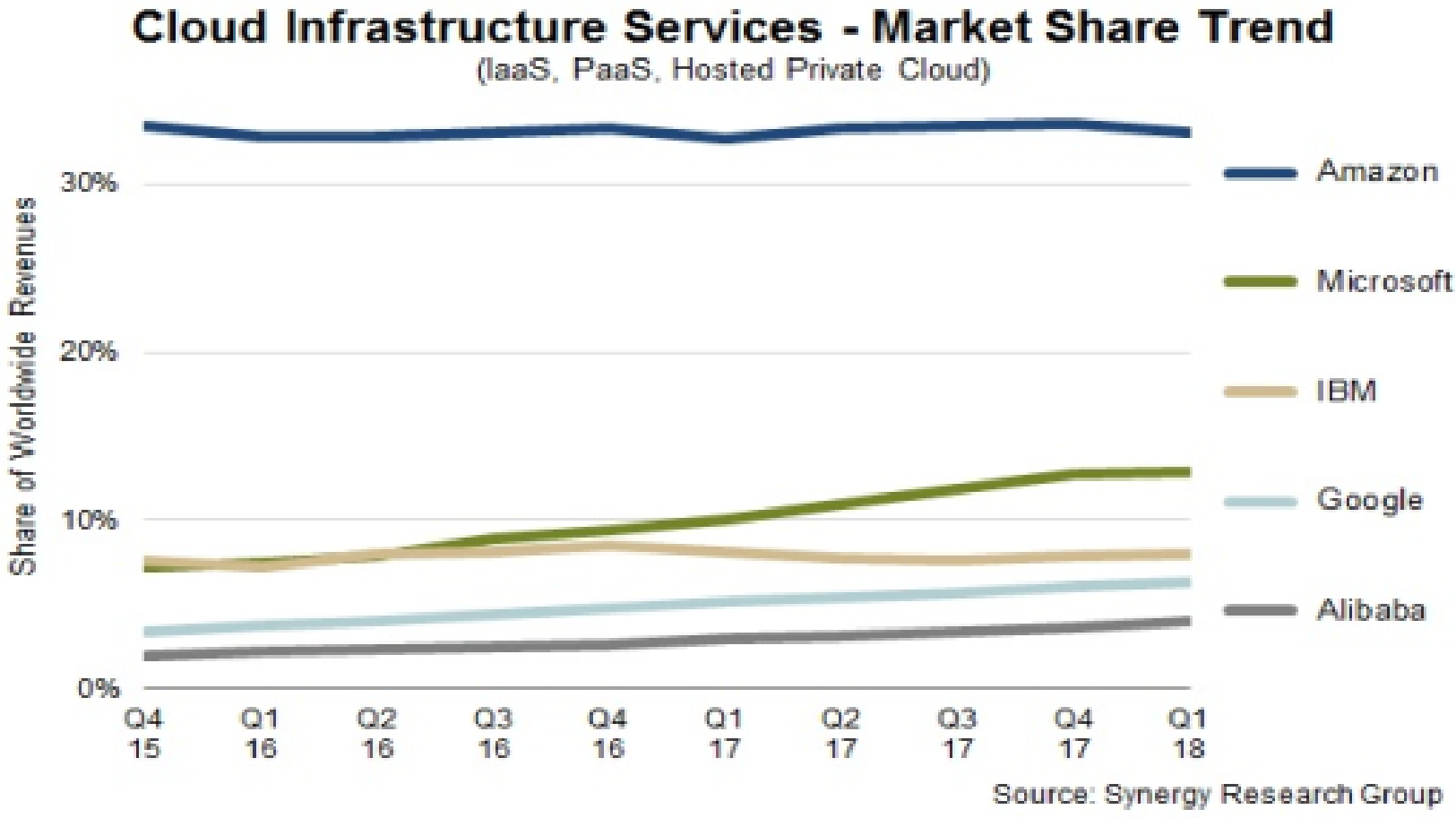
# Cloud Computing 101

Cloud computing refers to a variety of ideas:

- Software-as-a-Service (SaaS)
- Platform-as-a-Service (PaaS)
- Infrastructure-as-a-Service (IaaS)

All of these shift computational tasks from a local machine to a server.

# Who are the major players?



# Why cloud computing?

# Why cloud computing?

- Storage

# Why cloud computing?

- Storage
- Memory

# Why cloud computing?

- Storage
- Memory
- Explicit parallelism

# Why cloud computing?

- Storage
- Memory
- Explicit parallelism
- Security/Collaboration

# Why cloud computing?

- Storage
- Memory
- Explicit parallelism
- Security/Collaboration
- Reproducibility

# Why cloud computing?

- Storage
- Memory
- Explicit parallelism
- Security/Collaboration
- Reproducibility
- Data pipelines

# Why cloud computing?

- Storage
- Memory
- Explicit parallelism
- Security/Collaboration
- Reproducibility
- Data pipelines
- SaaS

# Why cloud computing?

**This Laptop**

- Intel Core i7 (4 cores)
- 8 GB memory
- 100 GB of usable storage

**What you can get on AWS**

- Equivalent AWS instance costs $0.0928/hour
- 96 cores and 384 GB memory costs $4.608/hour
- In theory unlimited number of instances
- Storage is basically unlimited
  - S3: $0.023/GB-month
  - EBS: $0.10/GB-month

# Simplest Use Case: Execute Code in the Cloud

1. Reserve an "instance" in the cloud
2. Fire up your favorite statistical software
3. Execute code as if you were running locally
4. Retrieve results

# Why aren't researchers using cloud computing resources?

*I started using SPSS in 1979, while studying cognitive psychology at the Leiden University. In these days I had to program SPSS-syntax on punched cards. The worst thing was not this card-interface, but it was the IBM job control language you had to include: total gibberish language that was needed to make your SPSS-job run on a mainframe somewhere in one of the university buildings.*

Source: Gerard van Meurs,

`https://50-years-spss.com/user-stories/`

# Why aren't researchers using cloud computing resources?

## Why aren't researchers using cloud computing resources?

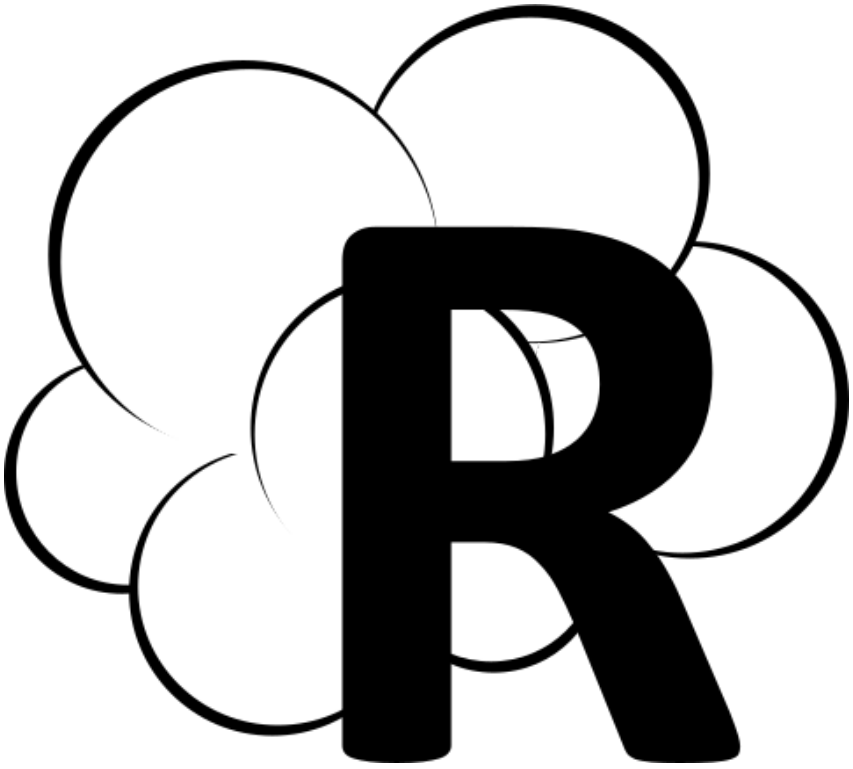
Statisticians and scientists may not know anything about how to set up high-performance computing infrastructure!

**Why aren't researchers using cloud computing resources?**

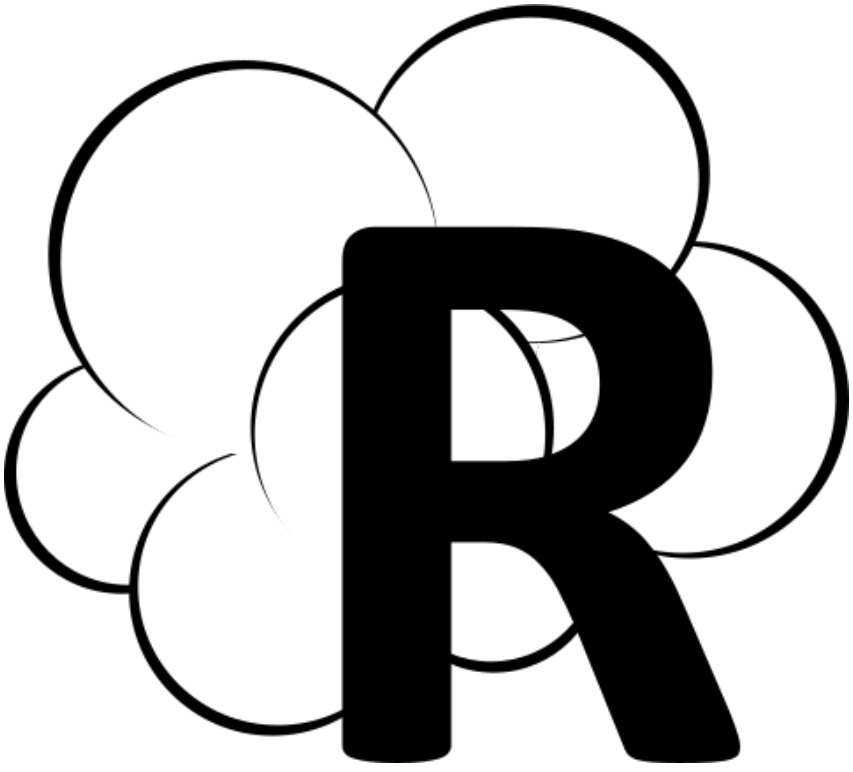
Statisticians and scientists may not know anything about how to set up high-performance computing infrastructure!

**I am one of those people!**

# The CloudyR Project

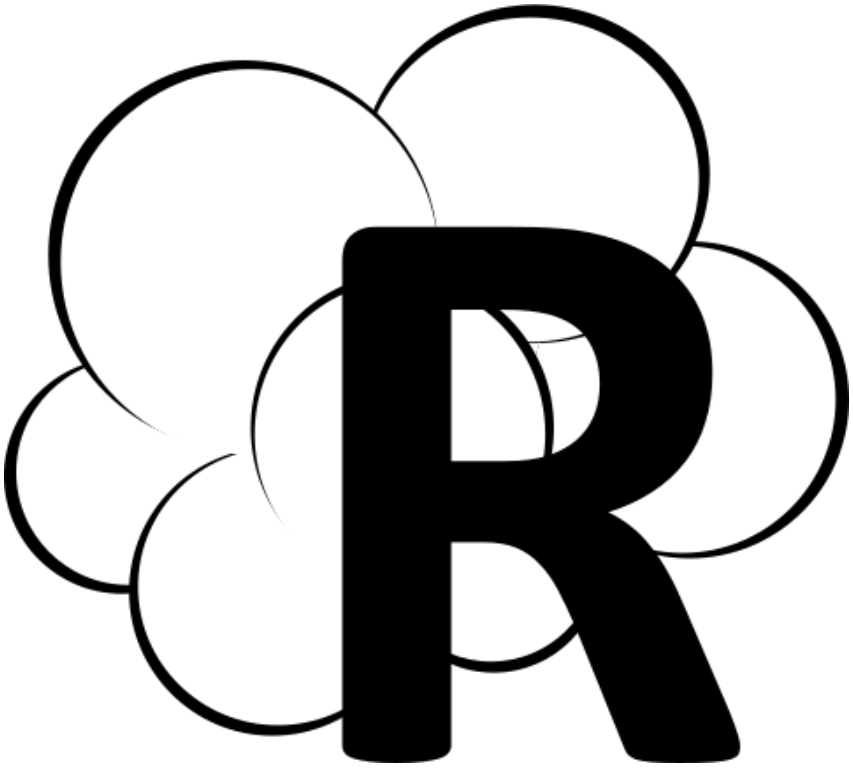


# The CloudyR Project



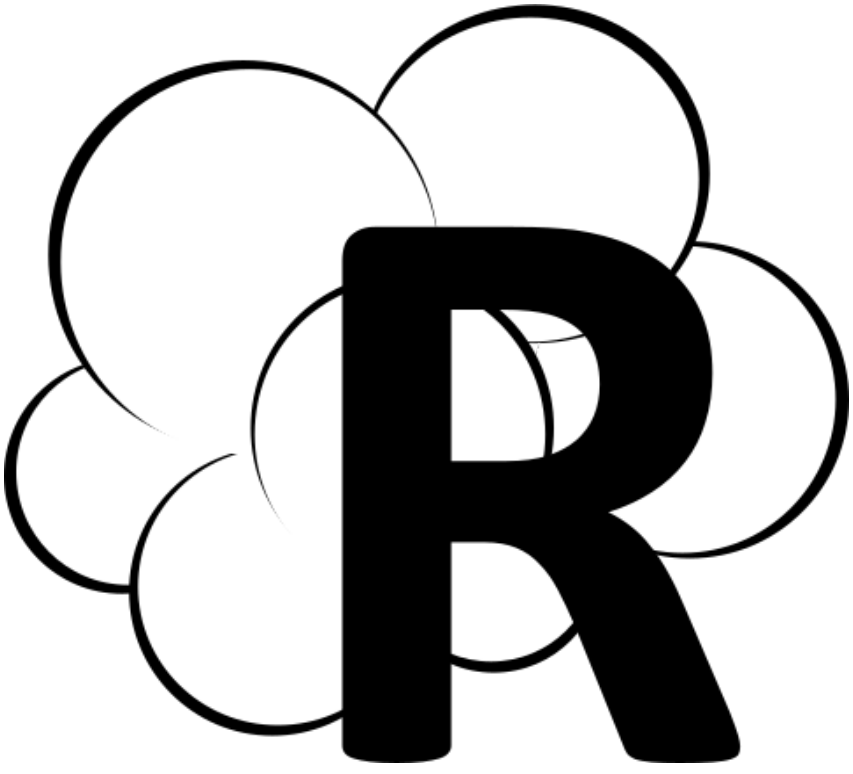

- Make R Cloudier!

# The CloudyR Project




- Make R Cloudier!
- Build easy-to-use, dependency-free software tools for working with any cloud service from R

# The CloudyR Project




- Make R Cloudier!
- Build easy-to-use, dependency-free software tools for working with any cloud service from R

Eventual goal: `eval_cloud("script.R")`

# The CloudyR Project

- 100% volunteer effort
- We receive no funding from any cloud service
- We build free and open source tools
- Many contributors!
  - Main AWS developer: Thomas Leeper
  - Main GCS developer: Mark Edmondson
  - Lots of PRs, bug reports, and documentation fixes from many, many people

# Why bother?

Cloud providers have broad language support:

- AWS SDKs: Java .Net Node.js PHP Python Ruby Go (C++)
- GCS SDKs: Java .Net Node.js PHP Python Ruby Go (C++)

# Why bother?

Cloud providers have broad language support:

- AWS SDKs: Java .Net Node.js PHP Python Ruby Go (C++)
- GCS SDKs: Java .Net Node.js PHP Python Ruby Go (C++)

**But where's R?**

# R is a first-class statistics and data science language!

| Jan 2018 | Jan 2017 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 14.215% | -3.06% |
| 2 | 2 | | C | 11.037% | +1.69% |
| 3 | 3 | | C++ | 5.603% | -0.70% |
| 4 | 5 | ^ | Python | 4.678% | +1.21% |
| 5 | 4 | v | C# | 3.754% | -0.29% |
| 6 | 7 | ^ | JavaScript | 3.465% | +0.62% |
| 7 | 6 | v | Visual Basic .NET | 3.261% | +0.30% |
| 8 | 16 | ^^ | R | 2.549% | +0.76% |
| 9 | 10 | ^ | PHP | 2.532% | -0.03% |
| 10 | 8 | v | Perl | 2.419% | -0.33% |

# Building R packages for cloud computing is difficult

# Building R packages for cloud computing is difficult

- Wrap an existing SDK
  - `https://github.com/hrbrmstr/roto.s3` (Requires Python 😐)
  - `https://cran.r-project.org/package=AWR` (Requires Java 🤢)

# Building R packages for cloud computing is difficult

- Wrap an existing SDK
  - `https://github.com/hrbrmstr/roto.s3` (Requires Python 😐)
  - `https://cran.r-project.org/package=AWR` (Requires Java 🤢)
- Wrap the AWS Command Line Tools
  - AWS.tools, awsConnect
  - Requires a system dependency ☹️
  - Very difficult to maintain ☹️

# Building R packages for cloud computing is difficult

- Wrap an existing SDK
  - `https://github.com/hrbrmstr/roto.s3` (Requires Python 😐)
  - `https://cran.r-project.org/package=AWR` (Requires Java 🤢)
- Wrap the AWS Command Line Tools
  - AWS.tools, awsConnect
  - Requires a system dependency ☹️
  - Very difficult to maintain ☹️
- Build native R packages using web APIs 😀

## Guides and API References

**Compute**
- AWS Batch
- Amazon EC2
- Amazon ECR
- Amazon ECS
- Amazon EKS
- AWS Elastic Beanstalk
- AWS Lambda
- Amazon Lightsail
- AWS Serverless Application Repository
- Amazon VPC

**Storage**
- Amazon EBS
- Amazon EFS
- Amazon Glacier
- Amazon S3
- AWS Snowball
- AWS Storage Gateway

**Database**
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon Neptune
- Amazon RDS
- Amazon Redshift

**Migration**
- AWS Application Discovery Service
- AWS Database Migration Service
- AWS Migration Hub
- AWS Schema Conversion Tool
- AWS Server Migration Service
- AWS Snowball

**Networking & Content Delivery**
- Amazon API Gateway
- Amazon CloudFront
- AWS Direct Connect
- Elastic Load Balancing
- Amazon Route 53
- Amazon VPC

**Application Integration**
- Amazon MQ
- Amazon SNS
- Amazon SQS
- AWS Step Functions
- Amazon SWF

**Developer Tools**
- AWS Cloud9
- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- AWS CodeStar
- AWS Tools & SDKs
- AWS X-Ray

**Management Tools**
- AWS Auto Scaling
- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch
- AWS Command Line Interface
- AWS Config
- Amazon Data Lifecyle Manager
- AWS Health
- AWS Management Console
- AWS OpsWorks
- AWS Service Catalog
- AWS Systems Manager
- AWS Tools for Windows Powershell
- Trusted Advisor

**Media Services**
- Amazon Elastic Transcoder
- AWS Elemental MediaConvert
- AWS Elemental MediaLive
- AWS Elemental MediaPackage
- AWS Elemental MediaStore
- AWS Elemental MediaTailor

**Machine Learning**
- Apache MXNet on AWS
- Amazon Comprehend
- AWS Deep Learning AMIs
- AWS DeepLens
- Amazon Lex
- Amazon Machine Learning
- Amazon Polly
- Amazon Rekognition
- Amazon SageMaker
- Amazon Transcribe
- Amazon Translate

**Internet of Things**
- Amazon FreeRTOS
- AWS Greengrass
- AWS IoT 1-Click
- AWS IoT Analytics
- AWS IoT Core
- AWS IoT Device Management

**Analytics**
- Amazon Athena
- Amazon CloudSearch
- AWS Data Pipeline
- Amazon Elasticsearch Service
- Amazon EMR
- AWS Glue
- Amazon Kinesis
- Amazon QuickSight
- Amazon Redshift

**Security, Identity, & Compliance**
- AWS Artifact
- AWS Certificate Manager
- AWS CloudHSM
- Amazon Cognito
- AWS Crypto Tools
- AWS Directory Service
- AWS Firewall Manager
- Amazon Cloud Directory
- Amazon GuardDuty
- Identity & Access Management
- Amazon Inspector
- AWS Key Management Service
- Amazon Macie
- AWS Organizations
- AWS Secrets Manager
- AWS Shield
- AWS Single Sign-On
- AWS WAF

**Mobile Services**
- AWS AppSync
- AWS Device Farm
- Amazon Mobile Analytics
- AWS Mobile Hub
- AWS Mobile SDK for Android
- AWS Mobile SDK for iOS
- AWS Mobile SDK for Unity
- AWS Mobile SDK for Xamarin
- Amazon Pinpoint
- Amazon SNS

**Desktop & App Streaming**
- Amazon AppStream 2.0
- Amazon WAM
- Amazon WorkSpaces
- NICE Desktop Cloud Visualization

**Business Productivity**
- Alexa for Business
- Amazon Chime
- Amazon WorkDocs
- Amazon WorkMail

**AR & VR**
- Amazon Sumerian

**Customer Engagement**
- Amazon Connect
- Amazon Pinpoint
- Amazon Simple Email Service (SES)

**Game Development**
- Amazon GameLift
- Amazon Lumberyard (Beta)

**SDKs & Toolkits**
- AWS Crypto Tools
- AWS Guide for .NET Developers
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto 3)
- AWS SDK for Ruby
- AWS Toolkit for Eclipse
- AWS Toolkit for Visual Studio
- AWS Tools for Visual Studio Team Services

**General Reference**
- ARNs & Service Namespaces
- AWS Glossary
- Regions and Endpoints
- Security Credentials
- Service Limits

**Additional References**
- Alexa Top Sites
- Alexa Web Information Service
- AWS Billing and Cost Management
- AWS Blockchain Templates
- AWS General Reference
- AWS GovCloud (US)
- AWS Marketplace
- AWS Quick Starts
- Amazon Silk

**AWS Management Console**
- Resource Groups
- Resource Groups Tagging API
- Tag Editor

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3)

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM)

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓
- Cloud computing tools (EC2)

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓
- Cloud computing tools (EC2) ✓

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓
- Cloud computing tools (EC2) ✓
- Secure shell connections[1]

[1] `https://github.com/ropensci/ssh`

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓
- Cloud computing tools (EC2) ✓
- Secure shell connections[1] ✓

[1] `https://github.com/ropensci/ssh`

# Simplest Use Case

End goal: `eval_cloud("script.R")`

What do we need in order to make that happen?

- Low-level web API (HTTP) handling ✓
- Cloud storage infrastructure (S3) ✓
- User account management (IAM) ✓
- Cloud computing tools (EC2) ✓
- Secure shell connections[1] ✓
- High-level abstractions over the above

[1] `https://github.com/ropensci/ssh`

```
# 1. create an AWS account

# 2. load credentials into R
Sys.setenv("AWS_ACCESS_KEY_ID" = "my_key")
Sys.setenv("AWS_SECRET_ACCESS_KEY" = "my_secret")
Sys.setenv("AWS_DEFAULT_REGION" = "us-east-1")
```

# Storage

```
# cloud storage
library("aws.s3")

# put an R object into the cloud
s3saveRDS(mtcars, "s3://bucket/mtcars.rds")

# get an R object from the cloud
s3readRDS("s3://bucket/mtcars.rds")
```

```
# manipulate buckets
put_bucket()
get_bucket()
delete_bucket()

# manipulate objects
put_object()
get_object()
delete_object()
```

```
# higher-level functions
s3source()

s3save()
s3load()

s3read_using()
s3write_using()

# streaming R connection (rb)
s3connection()
```

# Notifications

```
# notifications
library("aws.sns")

# create a "topic"
topic <- create_topic(name = "jsm-example")

# subscribe to it
subscribe(topic, "me@example.com", "email")
subscribe(topic, "1-111-555-1234", "sms")
```

```
# R script
done <- FALSE
while (!done) {
  # long-running thing
  done <- TRUE
}

# send notification
publish(
  topic = topic,
  message = "Your script is done. -R",
  subject = "Done!"
)
```

# Computing

```
library("aws.ec2") # cloudyr/aws.ec2

# RStudio-configured EC2 image
# http://www.louisaslett.com/RStudio_AMI/
image <- "ami-fd2ffe87"

# create keypair
my_keypair <- create_keypair("jsm-keys")
cat(my_keypair$keyMaterial, file = "my.pem")

my_sg <- create_sgroup(
  "jsm-sg",
  "Allow my IP",
  vpc = describe_vpcs()[[1]]
)
authorize_ingress(my_sg)
```

```
# fire up instance
i <- run_instances(
  image = image, type = "t2.micro",
  sgroup = my_sg,
  subnet = "subnet-b815a6e0",
  keypair = my_keypair
)

ip <- allocate_ip("vpc")
associate_ip(i, ip)

browseURL(paste0("http://", ip$publicIp))
```

```
# log in to instance
library("ssh")
session <- ssh::ssh_connect(
  paste0("ubuntu@", ip$publicIp),
  keyfile = "my.pem",
  passwd = "rstudio"
)

# hello world!
cat("'hello world!'\n", file = "helloworld.R")
# upload it to instance
ssh::scp_upload(session, "helloworld.R")

# execute script on instance
ssh::ssh_exec_wait(session, "Rscript helloworld.R")

# disconnect from instance
ssh_disconnect(session)
```

```
# cleanup
stop_instances(i[[1]])
terminate_instances(i[[1]])

release_ip(ip)

revoke_ingress(my_sg)

delete_sgroup(sgroup = my_sg)
delete_keypair(my_keypair)
```

# A couple useful packages

https://cran.r-project.org/package=ssh

https://github.com/cloudyr/rmote

https:
//cran.r-project.org/package=remoter

# SaaS

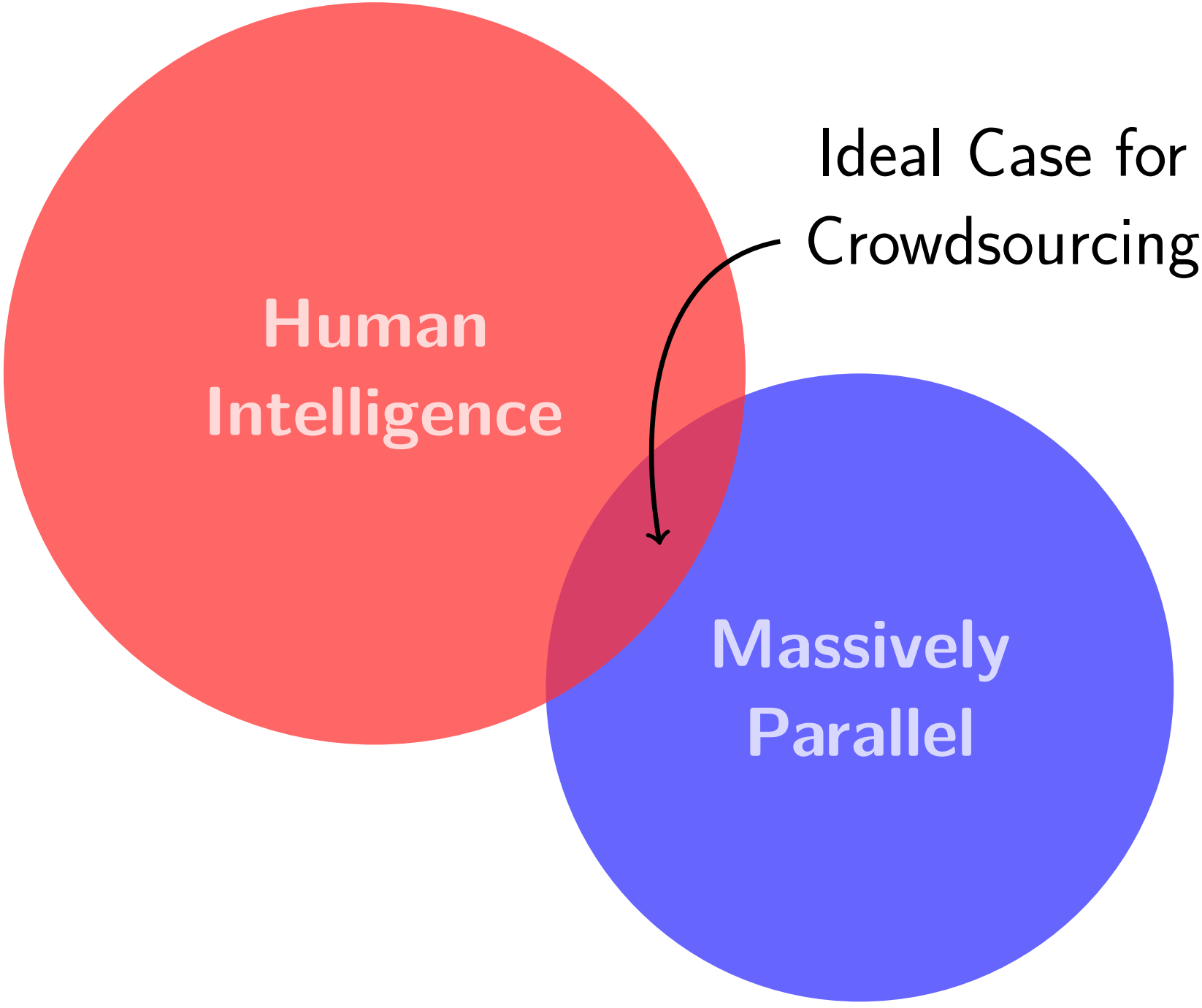
```
library("aws.polly")

msg_en <- "Thanks for attending the Cloud and Distribut

vec_en <- synthesize(msg_en, voice = "Joanna")

tuneR::play(vec_en)
```

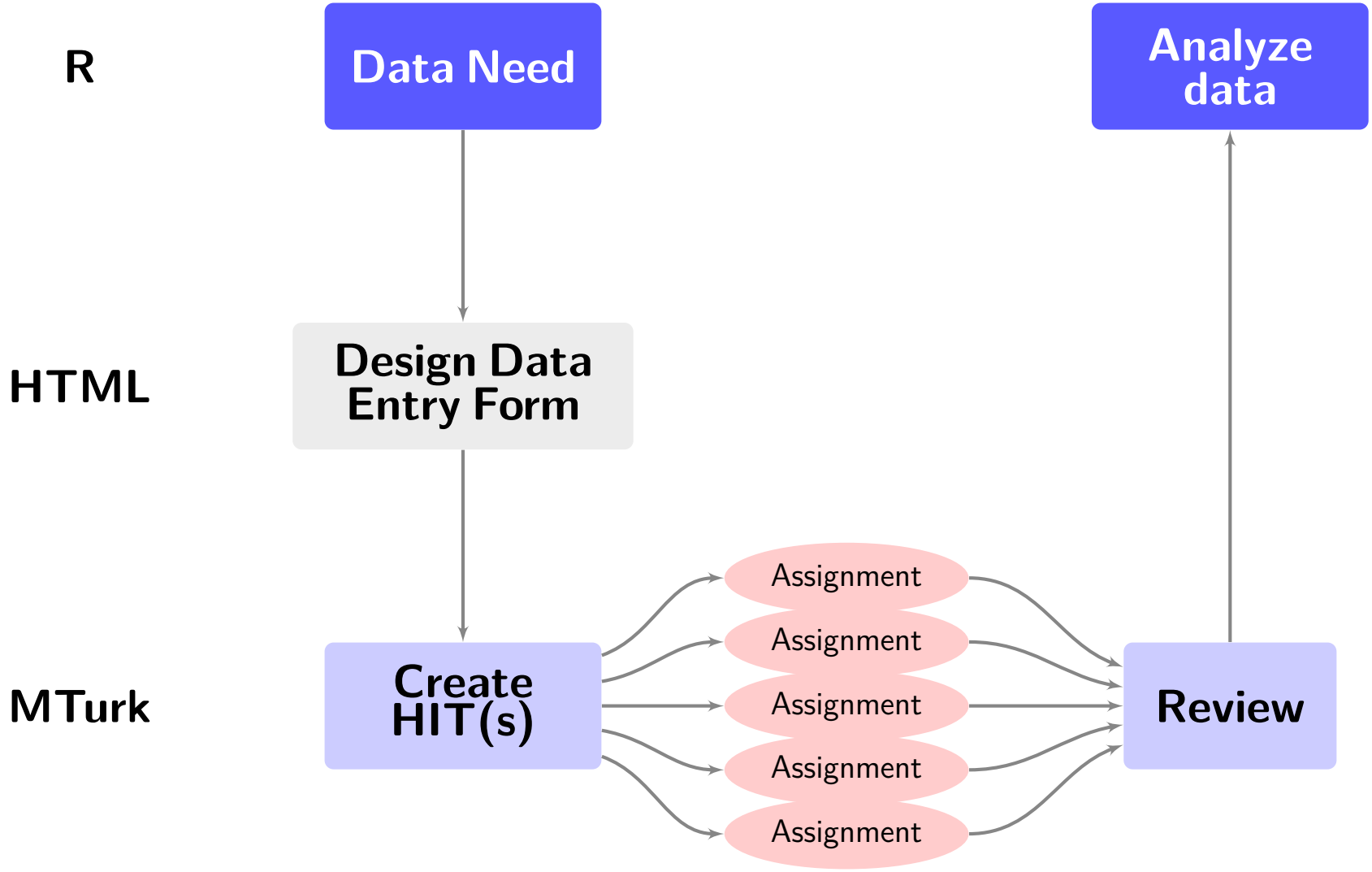
```
library("aws.translate")

msg_es <- translate(msg_en, from = "en", to = "es")
vec_es <- synthesize(msg_es, voice = "Penelope")
tuneR::play(vec_es)

msg_ru <- translate(msg_en, from = "en", to = "ru")
vec_ru <- synthesize(msg_ru, voice = "Maxim")
tuneR::play(vec_ru)
```

```
library("aws.comprehend")

detect_language(msg_en)
detect_language(msg_es)
detect_language(msg_ru)
```

```
library("aws.transcribe")

tuneR::writeWave(vec_en, "english.wav")
aws.s3::put_object(
 "english.wav",
 "s3://jsm2018cloudyrexample/english.wav",
 acl = "public-read"
)

start_transcription(
 "jsm2018-example",
 paste0("https://s3.amazonaws.com/",
        "jsm2018cloudyrexample/",
        "english.wav")
)

tr <- get_transcription("jsm2018")$Transcriptions
cat(strwrap(tr, 60), sep = "\n")
```

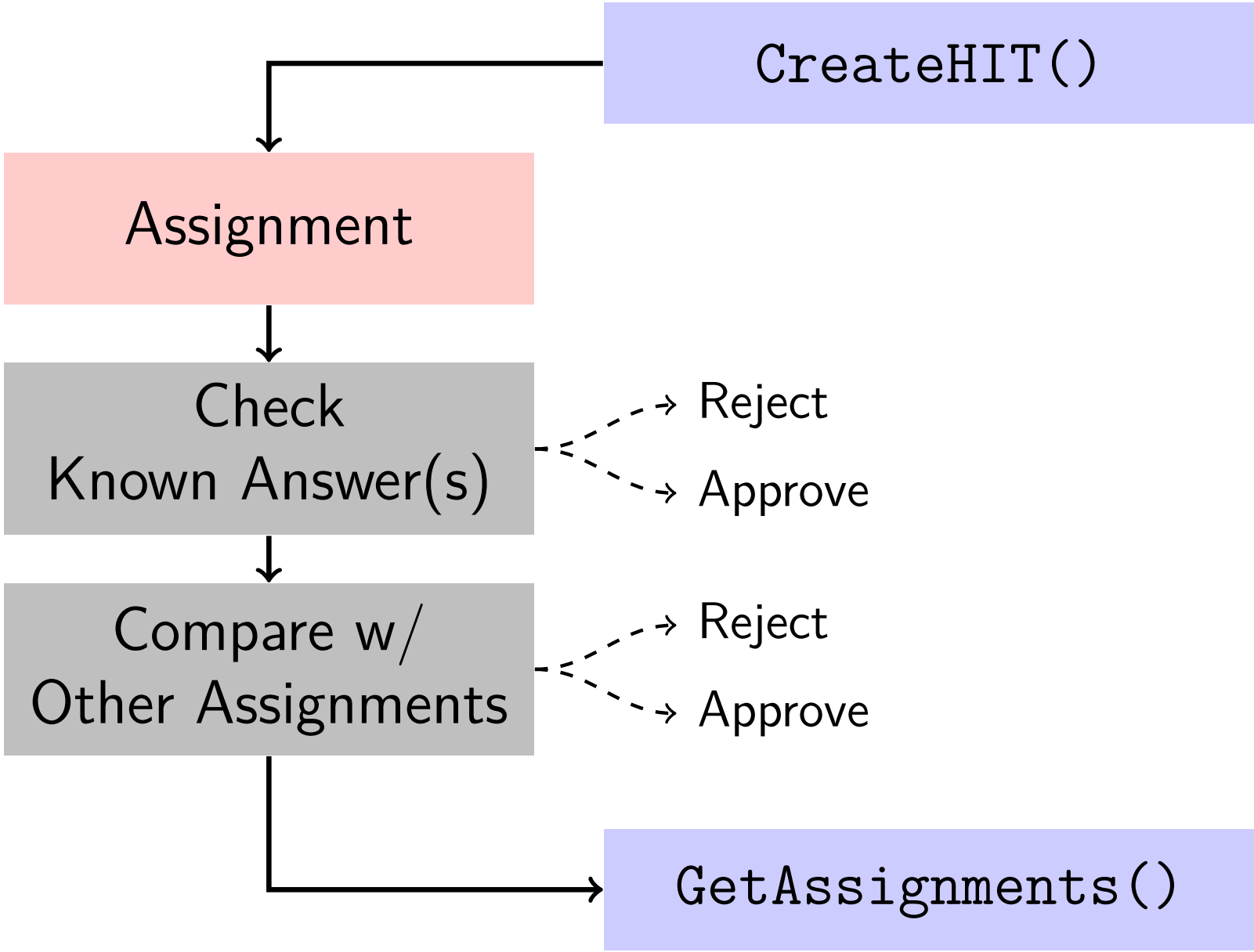
# Crowdsourcing


Human
Intelligence
Ideal Case for
Crowdsourcing
Massively
Parallel


R
HTML
MTurk
Data Need
Design Data Entry Form
Create HIT(s)
Assignment
Assignment
Assignment
Assignment
Assignment
Review
Analyze data

```
a = GenerateHTMLQuestion(file = "hit.html")

hit = CreateHIT(
  title = "Short Survey",
  description = "5 question survey",
  keywords = "survey, questionnaire",
  duration = seconds(hours = 1)
  reward = .10,

  assignments = 5000,
  expiration = seconds(days = 4),
  question = a$string,
)
```

# Anatomy of an MTurkR App



```
BulkCreateFromURLs(
  url = paste0("https://example.com/",1:10,".html"),

  title = "Image Categorization",
  description = "Describe contents of an image",
  keywords = "categorization, image",
  reward = .01,
  duration = seconds(minutes = 5),

  annotation = "My Project",
  expiration = seconds(days = 4),
  auto.approval.delay = seconds(days = 1)
)
```

**Get back a data.frame:**

```
GetAssignments(annotation = "My Project")
```

**Example:**

An image coding task with **27,500 images**

took **225 workers**

about **75 minutes**

and cost **$412.50**

**Pay workers with:**

```
ApproveAssignments(annotation = "My Project")
```

# CloudyR isn't just AWS

- GCS APIs are much cleaner
- Storage: googleCloudStorageR
- Compute: googleComputeEngineR
- Others: gcloudR (client for *any* GCS API)

# CloudyR isn't just AWS

- GCS APIs are much cleaner
- Storage: googleCloudStorageR
- Compute: googleComputeEngineR
- Others: gcloudR (client for *any* GCS API)

- In the pipeline:
  Meta packages to abstract across cloud services

# What's next for CloudyR?

- Databases
  (DynamoDB, Redshift, RDS)
- Machine Learning as a Service
  (AWS Glue, ML, SageMaker)
- Everything!?

# We can always use volunteers!

**Experienced Developers**

- Build packages for new cloud services
- Expand our scope beyond AWS and GCS
- Contribute PRs

**Beginner Developers**

- Feature requests
- Improve our documentation and examples
- Improve our tests
- Use packages and find bugs

```
# Start Cloud Computing


install_github("cloudyr/awspack")
install_github("cloudyr/gcloudR")


# Questions?
# Twitter @thosjleeper @cloudyrproject
# https://github.com/cloudyr
# http://cloudyr.github.io
# thosjleeper@gmail.com
```