# Dynamic Statistical Comparisons

**Gao Wang & Matthew Stephens**

Stephens Lab
University of Chicago

September 28, 2016

THE UNIVERSITY OF
CHICAGO

# DSC aids in reproducible research

Dynamic Statistical Comparisons (DSCs):

- Comparisons typically performed is suboptimal in many ways
- The idea of DSC is an attempt to make statistical comparisons **easily extensible** and **reproducible**

Our desired products:

# DSC aids in reproducible research

Dynamic Statistical Comparisons (DSCs):

- Comparisons typically performed is suboptimal in many ways
- The idea of DSC is an attempt to make statistical comparisons **easily extensible** and **reproducible**

Our desired products:

- A *platform* to make it **simple**, even **fun** to carry out DSC
- A *DSC repository* to facilitate research in Stephens Lab
  - ... and the entire research community

# DSC2: the new DSC platform

## DSCR

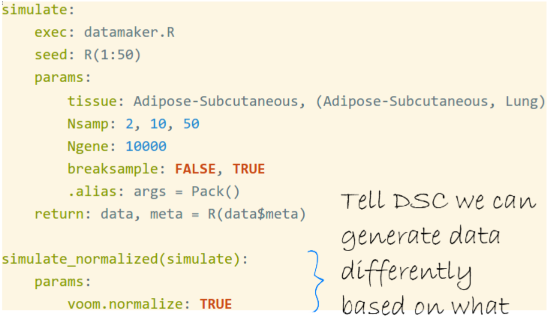
- Our first attempt to a DSC platform in the `R` language
- A successful proof-of-concept implementation to DSC model
- Lacks flexibility and capacity for complex and large scale DSC

## DSC2

- Our recent attempt to a multi-language DSC platform
  - Mix-and-match `R`, `Python` and `Shell` programs
  - Assembly of statistical procedures like LEGO
  - Engineered by modern workflow management system standards

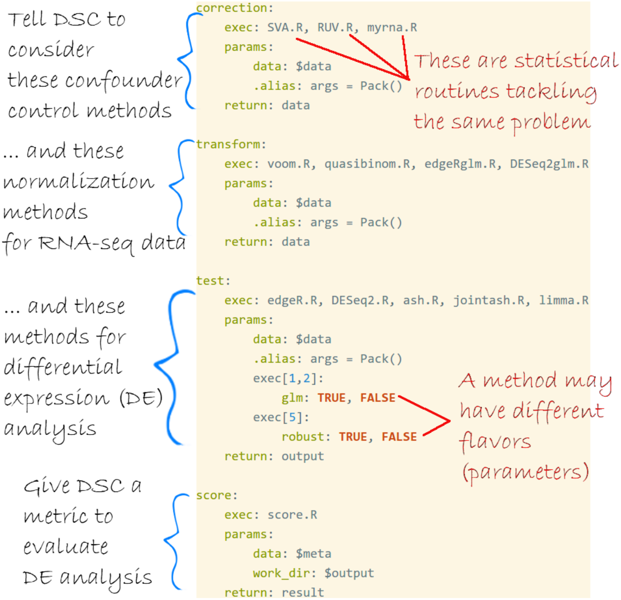
# DSC2: key idea illustrated

Tell DSC how benchmark data is generated



```
simulate:
    exec: datamaker.R
    seed: R(1:50)
    params:
        tissue: Adipose-Subcutaneous, (Adipose-Subcutaneous, Lung)
        Nsamp: 2, 10, 50
        Ngene: 10000
        breaksample: FALSE, TRUE
        .alias: args = Pack()
    return: data, meta = R(data$meta)

simulate_normalized(simulate):
    params:
        voom.normalize: TRUE
```

Tell DSC we can generate data differently based on what we've done



Tell DSC to consider these confounder control methods

... and these normalization methods for RNA-seq data

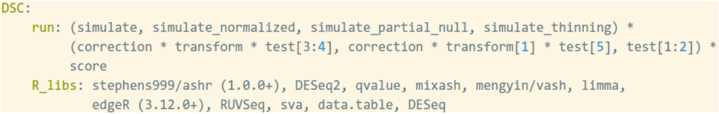
... and these methods for differential expression (DE) analysis

Give DSC a metric to evaluate DE analysis

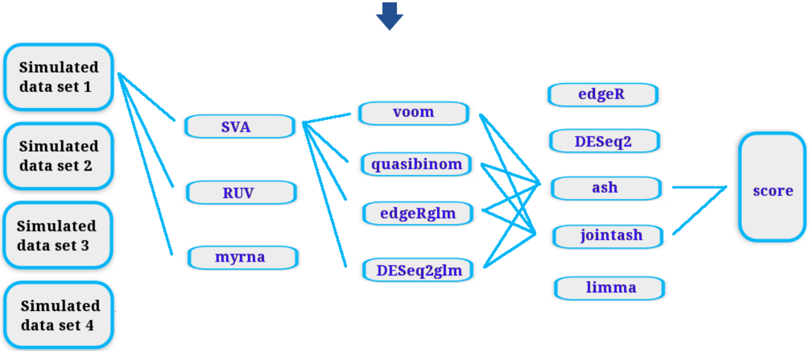
```
correction:
    exec: SVA.R, RUV.R, myrna.R
    params:
        data: $data
        .alias: args = Pack()
    return: data

transform:
    exec: voom.R, quasibinom.R, edgeRglm.R, DESeq2glm.R
    params:
        data: $data
        .alias: args = Pack()
    return: data

test:
    exec: edgeR.R, DESeq2.R, ash.R, jointash.R, limma.R
    params:
        data: $data
        .alias: args = Pack()
        exec[1,2]:
            glm: TRUE, FALSE
        exec[5]:
            robust: TRUE, FALSE
    return: output

score:
    exec: score.R
    params:
        data: $meta
        work_dir: $output
    return: result
```

These are statistical routines tackling the same problem

A method may have different flavors (parameters)



```
DSC:
    run: (simulate, simulate_normalized, simulate_partial_null, simulate_thinning) *
         (correction * transform * test[3:4], correction * transform[1] * test[5], test[1:2]) *
         score
    R_libs: stephens999/ashr (1.0.0+), DESeq2, qvalue, mixash, mengyin/vash, limma,
            edgeR (3.12.0+), RUVSeq, sva, data.table, DESeq
```



Simulated data set 1

Simulated data set 2

Simulated data set 3

Simulated data set 4

SVA

RUV

myrna

voom

quasibinom

edgeRglm

DESeq2glm

edgeR

DESeq2

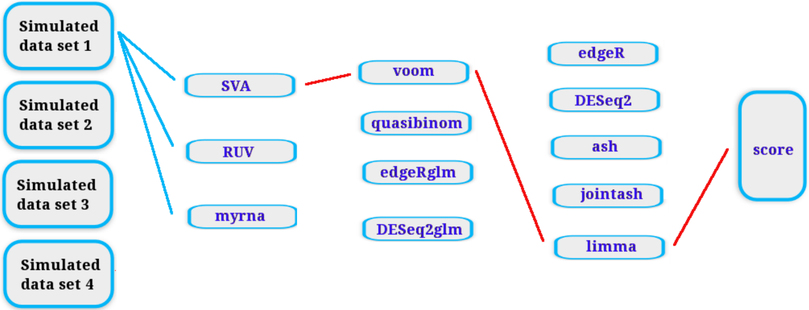
ash

jointash

limma

score

```
DSC:
    run: (simulate, simulate_normalized, simulate_partial_null, simulate_thinning) *
         (correction * transform * test[3:4], correction * transform[1] * test[5], test[1:2]) *
         score
    R_libs: stephens999/ashr (1.0.0+), DESeq2, qvalue, mixash, mengyin/vash, limma,
            edgeR (3.12.0+), RUVSeq, sva, data.table, DESeq
```



Simulated data set 1

Simulated data set 2

Simulated data set 3

Simulated data set 4

SVA

RUV

myrna

voom

quasibinom

edgeRglm

DESeq2glm

edgeR

DESeq2

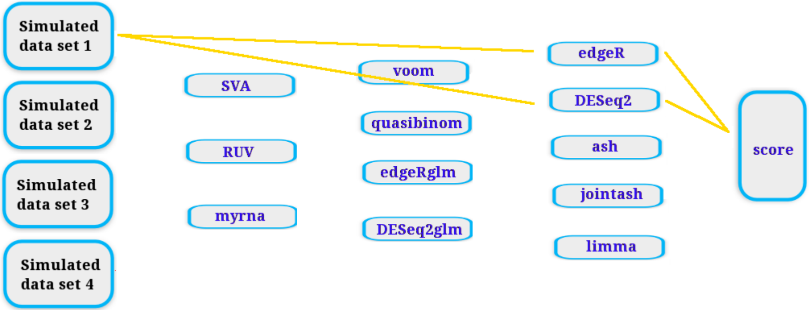
ash

jointash

limma

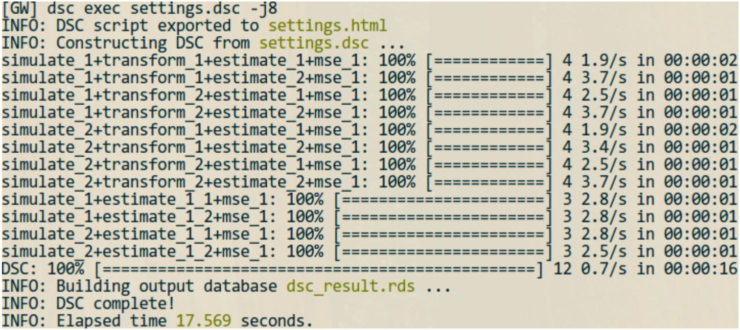
score

```
DSC:
    run: (simulate, simulate_normalized, simulate_partial_null, simulate_thinning) *
         (correction * transform * test[3:4], correction * transform[1] * test[5], test[1:2]) *
         score
    R_libs: stephens999/ashr (1.0.0+), DESeq2, qvalue, mixash, mengyin/vash, limma,
            edgeR (3.12.0+), RUVSeq, sva, data.table, DESeq
```



# User interface & DSC browser

- A command line tool that generates HTML reports



```
[GW] dsc exec settings.dsc -j8
INFO: DSC script exported to settings.html
INFO: Constructing DSC from settings.dsc ...
simulate_1+transform_1+estimate_1+mse_1: 100% [============] 4 1.9/s in 00:00:02
simulate_1+transform_1+estimate_2+mse_1: 100% [============] 4 3.7/s in 00:00:01
simulate_1+transform_2+estimate_1+mse_1: 100% [============] 4 2.5/s in 00:00:01
simulate_1+transform_2+estimate_2+mse_1: 100% [============] 4 3.7/s in 00:00:01
simulate_2+transform_1+estimate_1+mse_1: 100% [============] 4 1.9/s in 00:00:02
simulate_2+transform_1+estimate_2+mse_1: 100% [============] 4 3.4/s in 00:00:01
simulate_2+transform_2+estimate_1+mse_1: 100% [============] 4 2.5/s in 00:00:01
simulate_2+transform_2+estimate_2+mse_1: 100% [============] 4 3.7/s in 00:00:01
simulate_1+estimate_1_1+mse_1: 100% [======================] 3 2.8/s in 00:00:01
simulate_1+estimate_1_2+mse_1: 100% [======================] 3 2.8/s in 00:00:01
simulate_2+estimate_1_1+mse_1: 100% [======================] 3 2.8/s in 00:00:01
simulate_2+estimate_1_2+mse_1: 100% [======================] 3 2.5/s in 00:00:01
DSC: 100% [===============================================] 12 0.7/s in 00:00:16
INFO: Building output database dsc_result.rds ...
INFO: DSC complete!
INFO: Elapsed time 17.569 seconds.
```

- DSC example from a lab project
- ... and the benchmark generated

# Next steps

**Improved DSC report and visualization**

- Make it even more fun to build DSC

**Support for cluster computing**

- Implement Directed Acyclic Graph (DAG) for DSC jobs
- Job management and signature tracking via `redis`

**Application to Stephens Lab projects**

- Adapt existing projects to DSC2
- Advocate collaborations via DSC2 within the lab