Script of Scripts

for the writing and execution of scripts in multiple languages

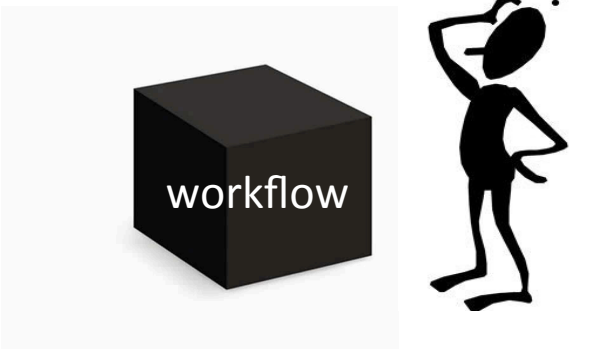
Bo Peng, Ph.D.

Department of Bioinformatics and Computational Biology
The University of Texas, MD Anderson Cancer Center

Jan. 23rd, 2017

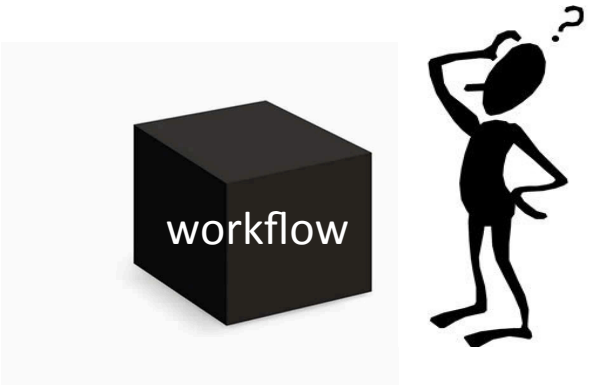
Multi Language Anxiety

Multi Language Anxiety



Others' workflow

Multi Language Anxiety



Others' workflow

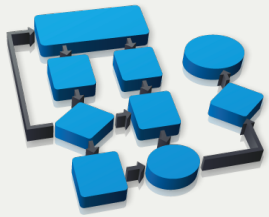
Implementation Hurdles

What is SoS?

Multi-language File Formats

What is SoS?

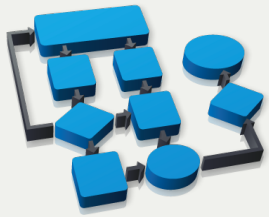
Multi-language File Formats



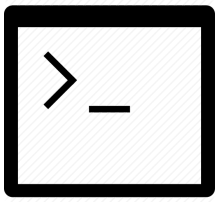
Interpreter and workflow engine

What is SoS?

Multi-language File Formats



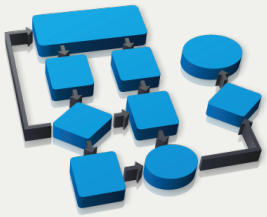
Interpreter and workflow engine



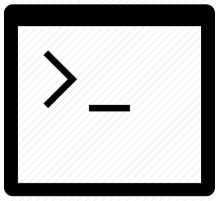
Command line

What is SoS?

Multi-language File Formats



Interpreter and workflow engine

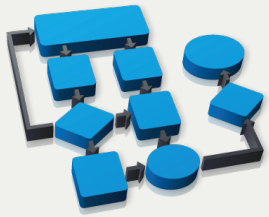


Command line

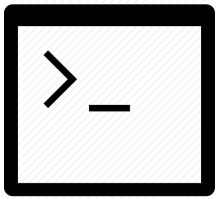
Jupyter notebook

What is SoS?

Multi-language File Formats



Interpreter and workflow engine



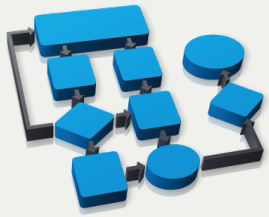
Command line

Jupyter notebook

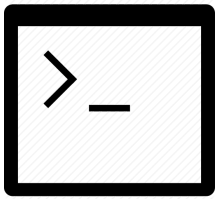
IDE

What is SoS?

Multi-language File Formats



Interpreter and workflow engine



Command line

Jupyter notebook

IDE

Task Queues

A Multi-Language Environment

1.1 Using Jupyter notebook

Let us assume that you are a bioinformaticist needed to compare the expression levels between two samples. You can use a Jupyter notebook with SoS kernel to perform the analysis using different script, the trick here is to select the right kernel for each cell. For example, you can run the following cell in bash if you choose `bash` as the kernel.

In [2]: Bash

```
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
    --genomeDir STAR_index
# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated
```

```
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
```

The first command builds an index of the reference genome in preparation for the latter steps, the second command aligns reads from the first sample to the reference genome, and the third command aligns reads from the second sample to the reference genome. Do not panic if you do not know what these commands are doing, this is just an example.

These commands generate, among other files, two files named `aligned/control.out.tab` and `aligned/mutated.out.tab` with expression counts of all genes. You then wrote a R script to analyze the results, something like

In [3]: R

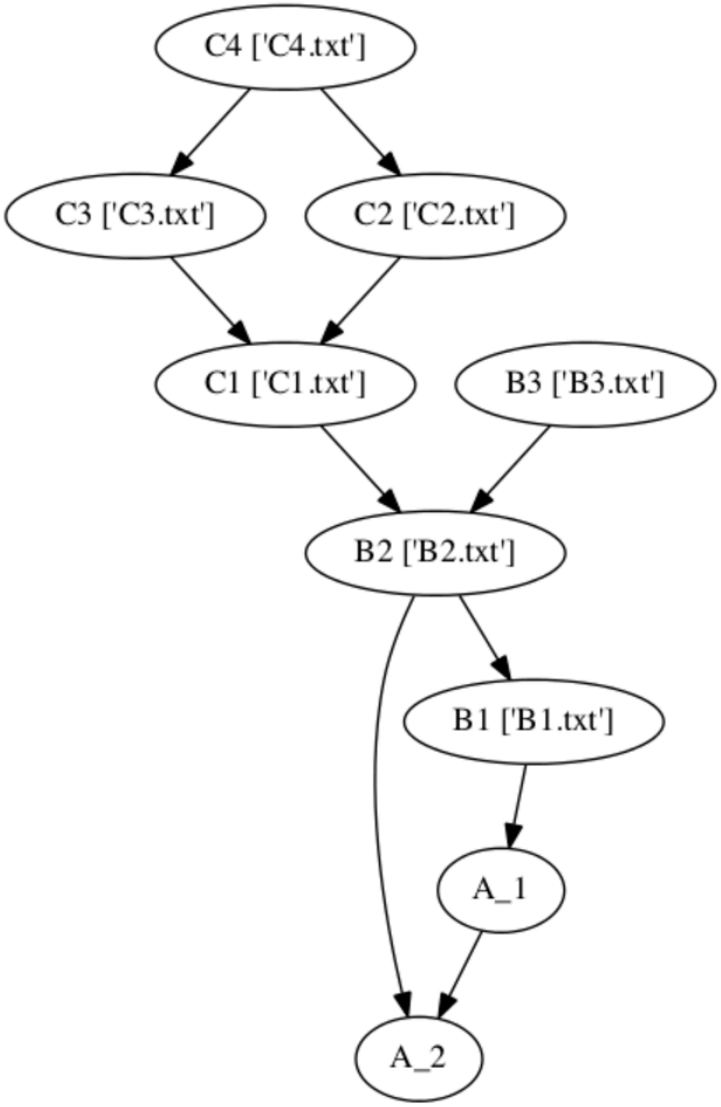
```
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

pdf: 2

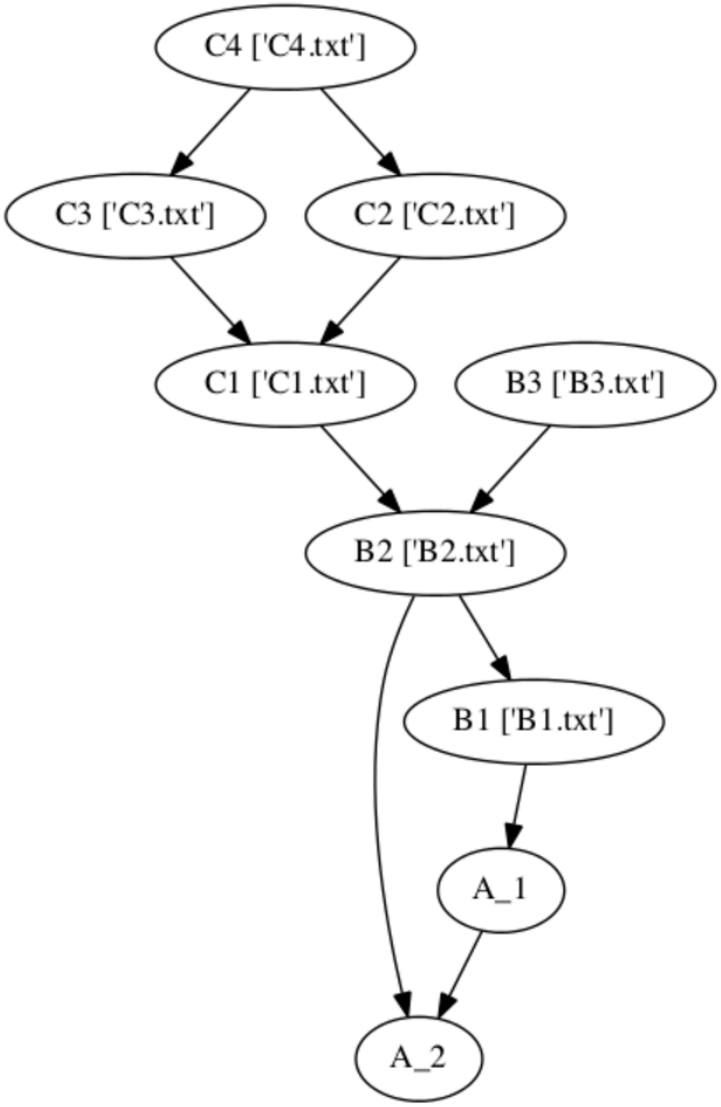
A Readable Workflow Format

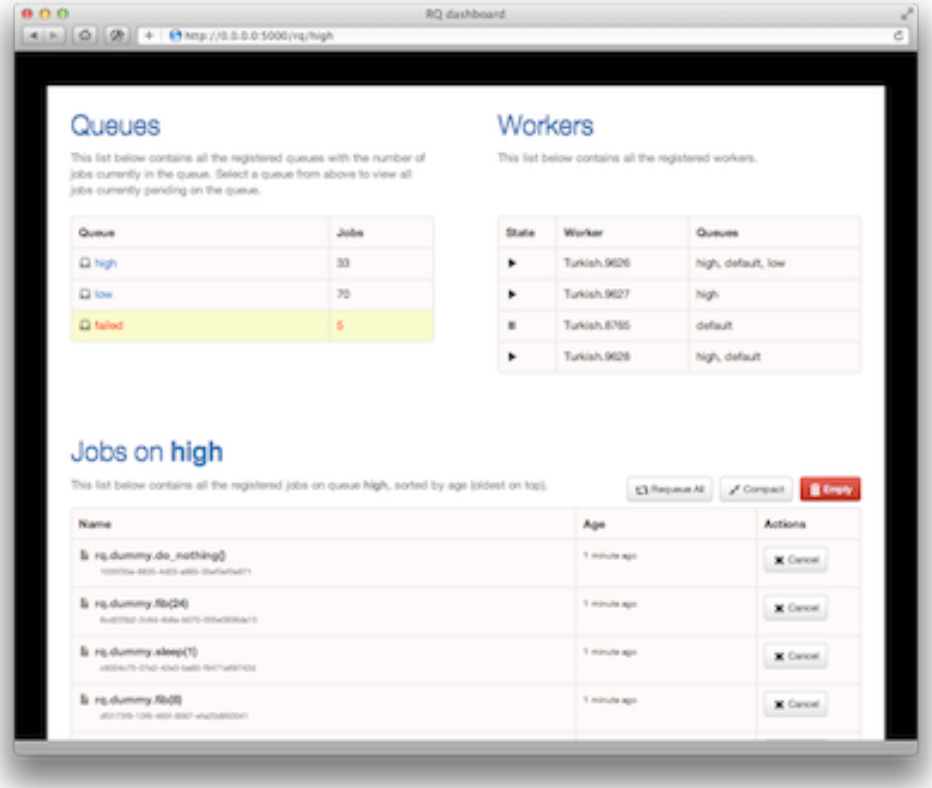
```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

A Workflow System with Job Queues

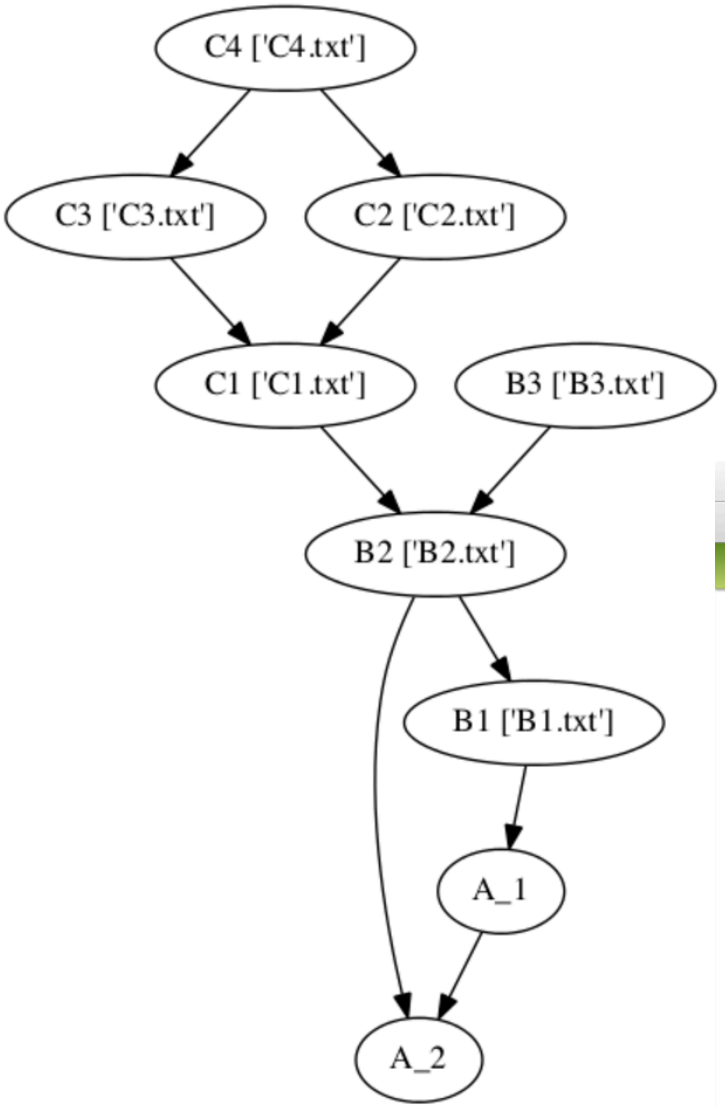


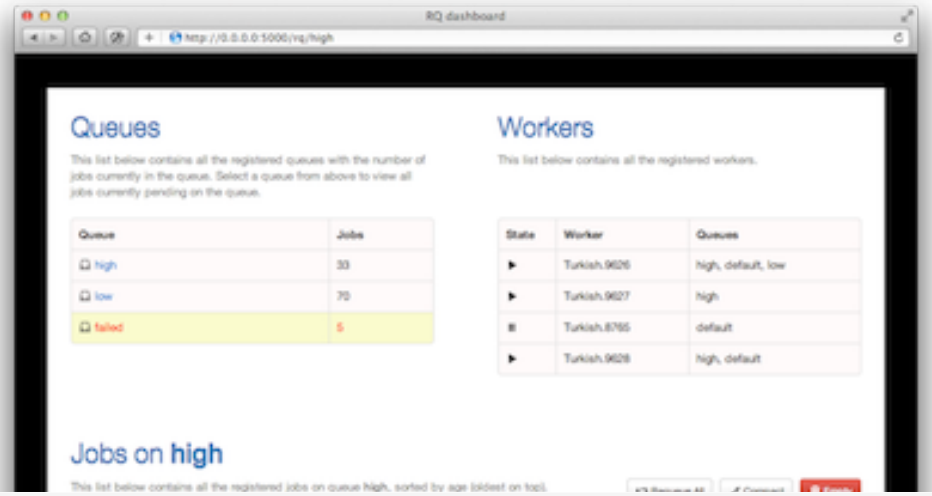
A Workflow System with Job Queues

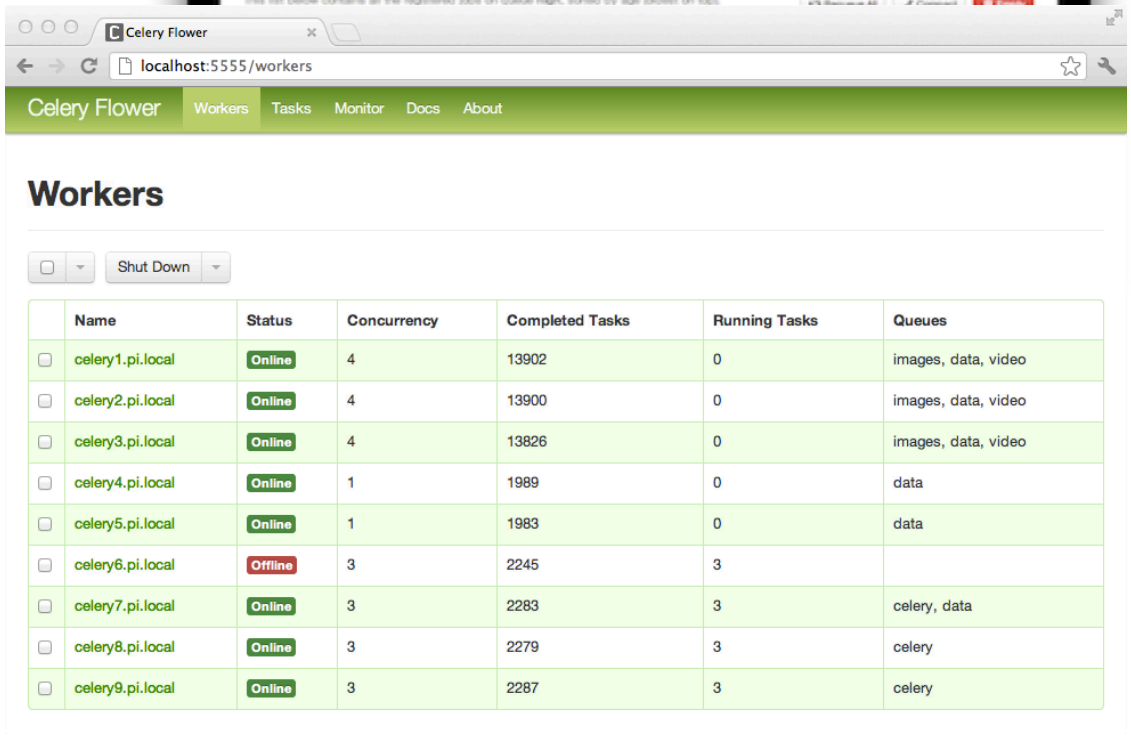




A Workflow System with Job Queues







	Name	Status	Concurrency	Completed Tasks	Running Tasks	Queues
☐	celery1.pi.local	Online	4	13902	0	images, data, video
☐	celery2.pi.local	Online	4	13900	0	images, data, video
☐	celery3.pi.local	Online	4	13826	0	images, data, video
☐	celery4.pi.local	Online	1	1989	0	data
☐	celery5.pi.local	Online	1	1983	0	data
☐	celery6.pi.local	Offline	3	2245	3	
☐	celery7.pi.local	Online	3	2283	3	celery, data
☐	celery8.pi.local	Online	3	2279	3	celery
☐	celery9.pi.local	Online	3	2287	3	celery

Interactive SoS

- Multi-language Notebook
- String interpolation
- SoS Magics
- Explicit Data Exchange
- Automatic Data Exchange
- Variable and File Preview

Multi-language notebook

1.1 Using Jupyter notebook

Let us assume that you are a bioinformaticist needed to compare the expression levels between two samples. You can use a Jupyter notebook with SoS kernel to perform the analysis using different script, the trick here is to select the right kernel for each cell. For example, you can run the following cell in bash if you choose `bash` as the kernel.

In [2]: Bash

```
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
    --genomeDir STAR_index
# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated
```

```
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
```

The first command builds an index of the reference genome in preparation for the latter steps, the second command aligns reads from the first sample to the reference genome, and the third command aligns reads from the second sample to the reference genome. Do not panic if you do not know what these commands are doing, this is just an example.

These commands generate, among other files, two files named `aligned/control.out.tab` and `aligned/mutated.out.tab` with expression counts of all genes. You then wrote a R script to analyze the results, something like

In [3]: R

```
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

pdf: 2

Multi-language notebook

1.1 Using Jupyter notebook

Let us assume that you are a bioinformaticist needed to compare the expression levels between two samples. You can use a Jupyter notebook with SoS kernel to perform the analysis using different script, the trick here is to select the right kernel for each cell. For example, you can run the following cell in bash if you choose `bash` as the kernel.

In [2]: Bash

```
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
    --genomeDir STAR_index
# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated
```

```
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
```

Select to change kernel

The first command builds an index of the reference genome in preparation for the latter steps, the second command aligns reads from the first sample to the reference genome, and the third command aligns reads from the second sample to the reference genome. Do not panic if you do not know what these commands are doing, this is just an example.

These commands generate, among other files, two files named `aligned/control.out.tab` and `aligned/mutated.out.tab` with expression counts of all genes. You then wrote a R script to analyze the results, something like

In [3]: R

```
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

pdf: 2

Multi-language notebook

1.1 Using Jupyter notebook

Let us assume that you are a bioinformaticist needed to compare the expression levels between two samples. You can use a Jupyter notebook with SoS kernel to perform the analysis using different script, the trick here is to select the right kernel for each cell. For example, you can run the following cell in bash if you choose `bash` as the kernel.

In [2]:

```
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
    --genomeDir STAR_index
# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fastq --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated
```

Bash

```
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
```

Select to change kernel

kernel specific prompt color

The first command builds an index of the reference genome in preparation for the latter steps, the second command aligns reads from the first sample to the reference genome, and the third command aligns reads from the second sample to the reference genome. Do not panic if you do not know what these commands are doing, this is just an example.

These commands generate, among other files, two files named `aligned/control.out.tab` and `aligned/mutated.out.tab` with expression counts of all genes. You then wrote a R script to analyze the results, something like

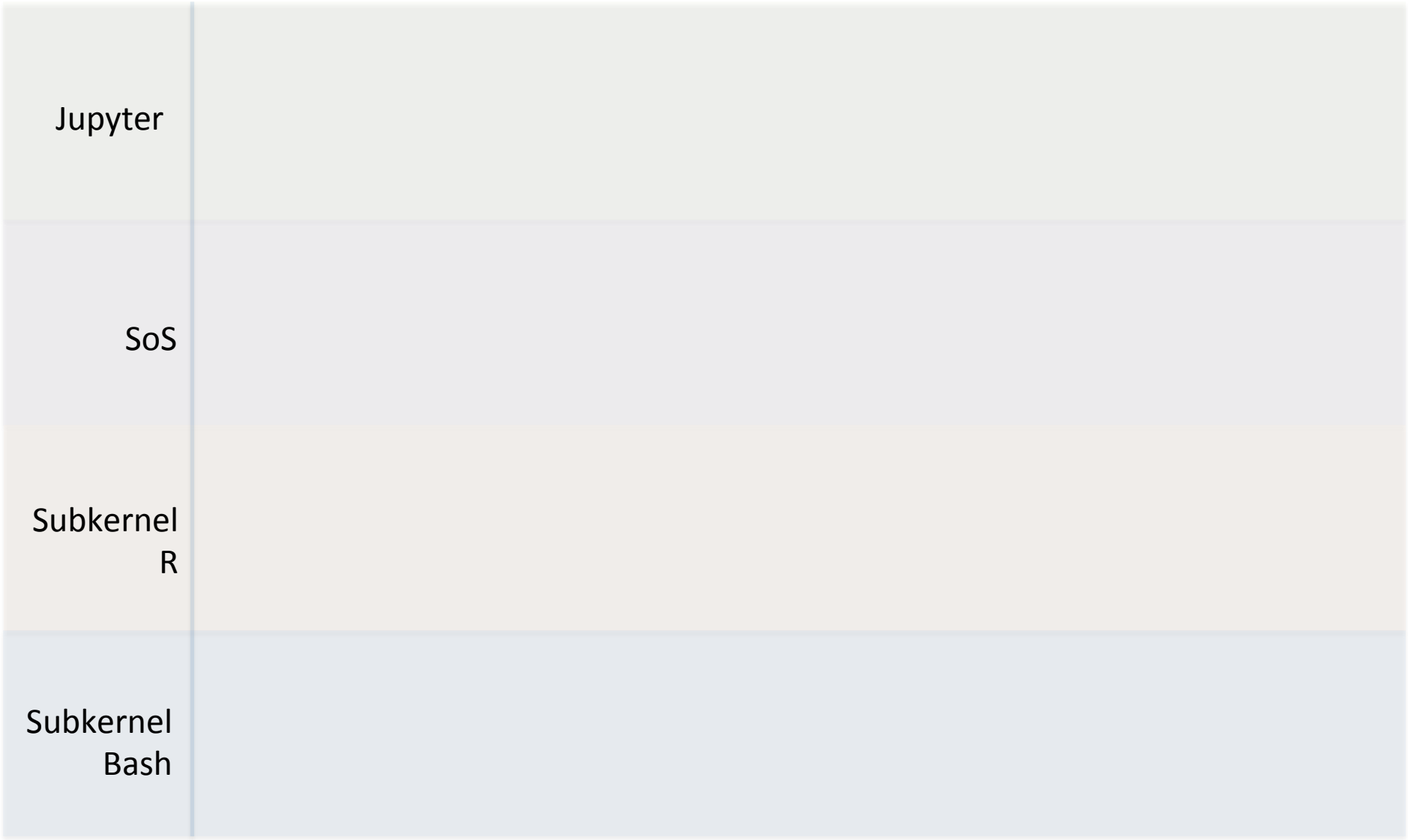
In [3]:

```
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

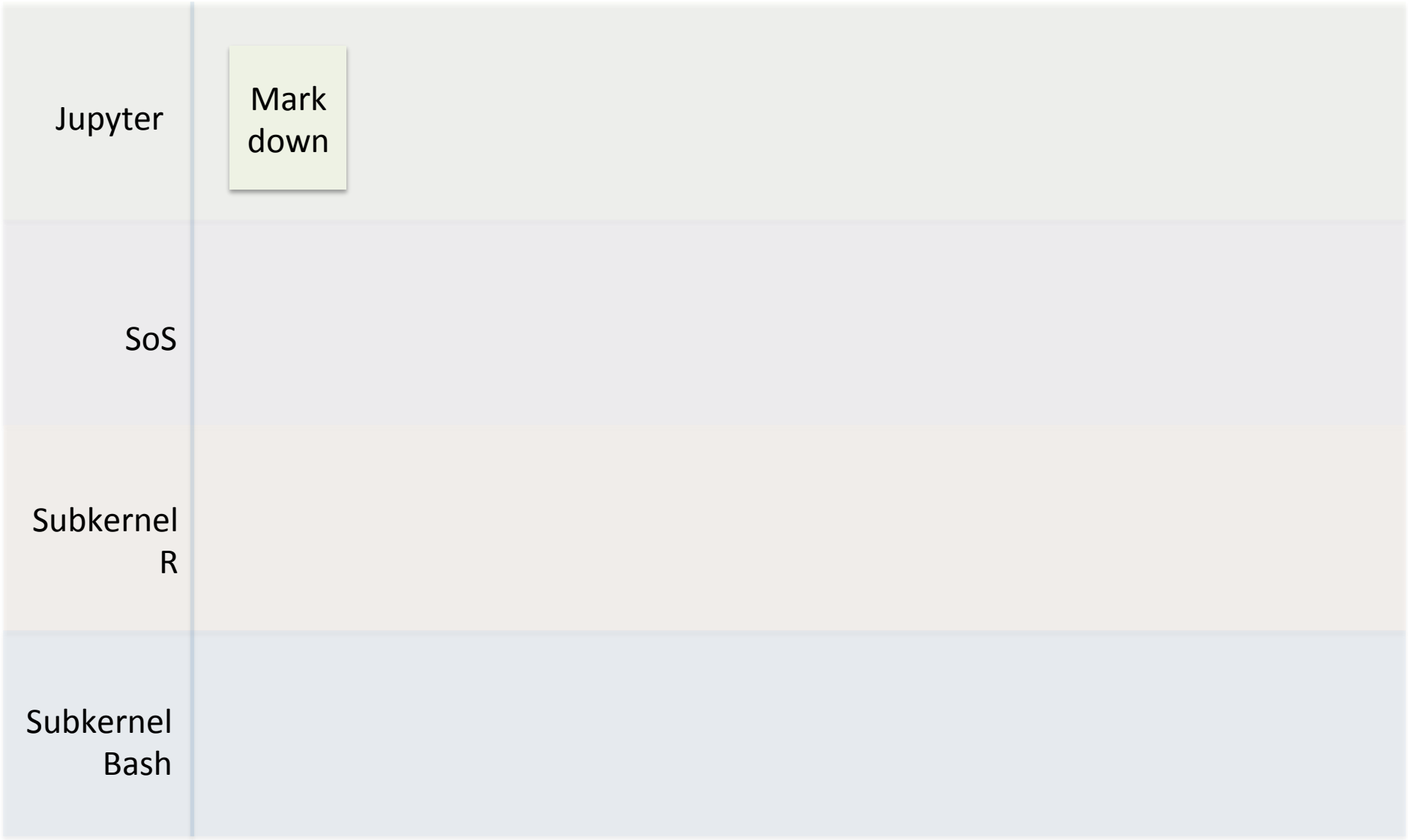
R

pdf: 2

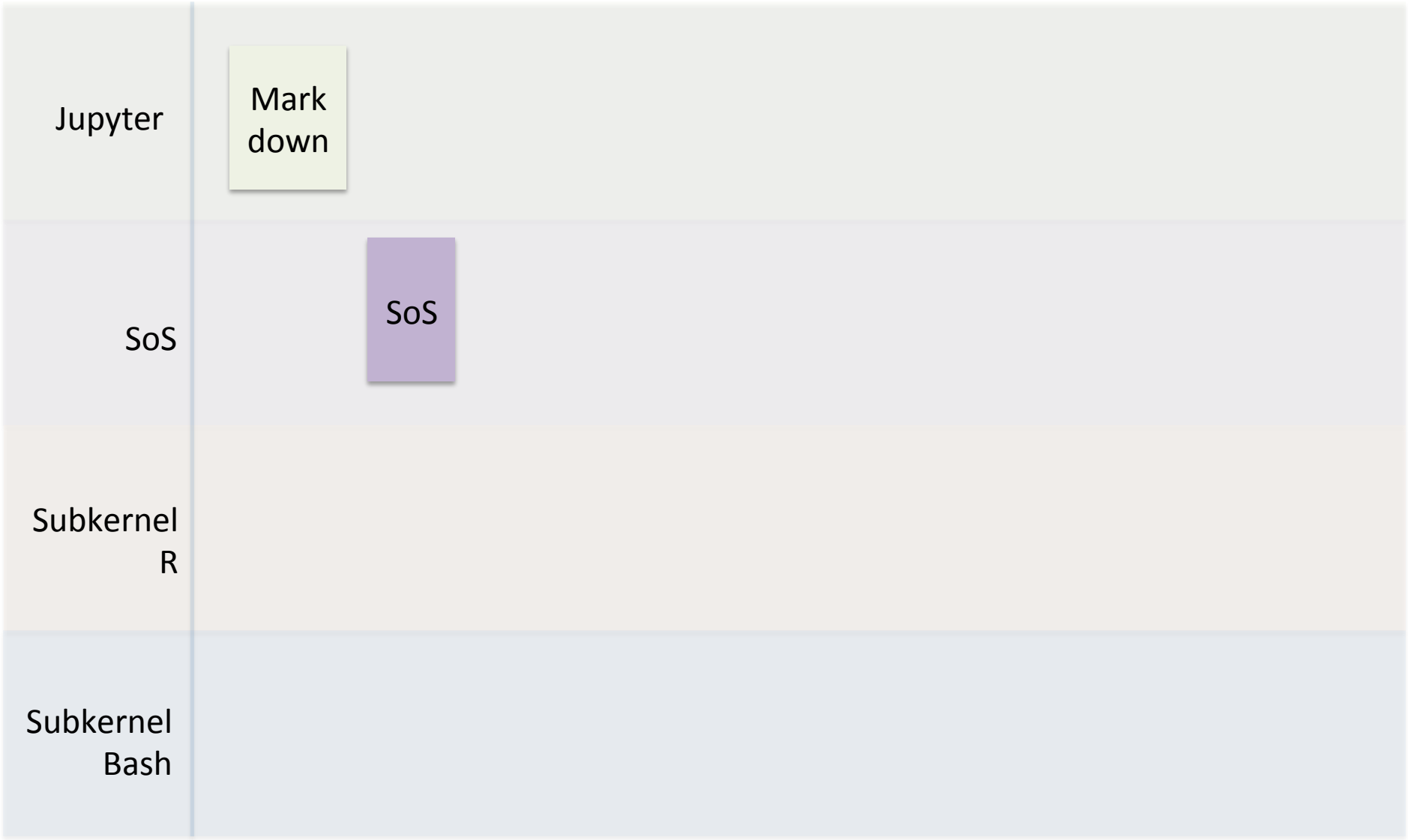
SoS Magics: %use and %with



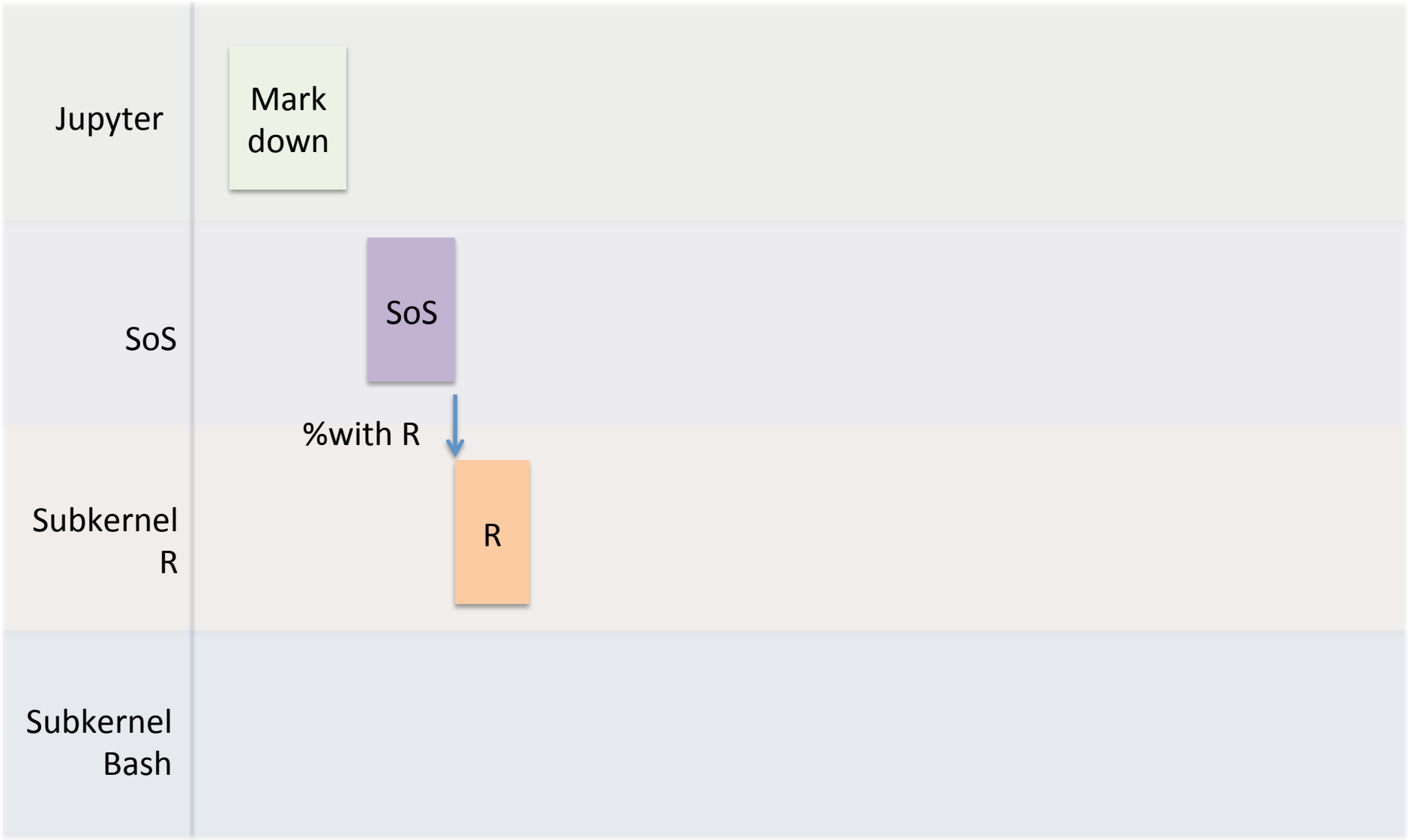
SoS Magics: %use and %with



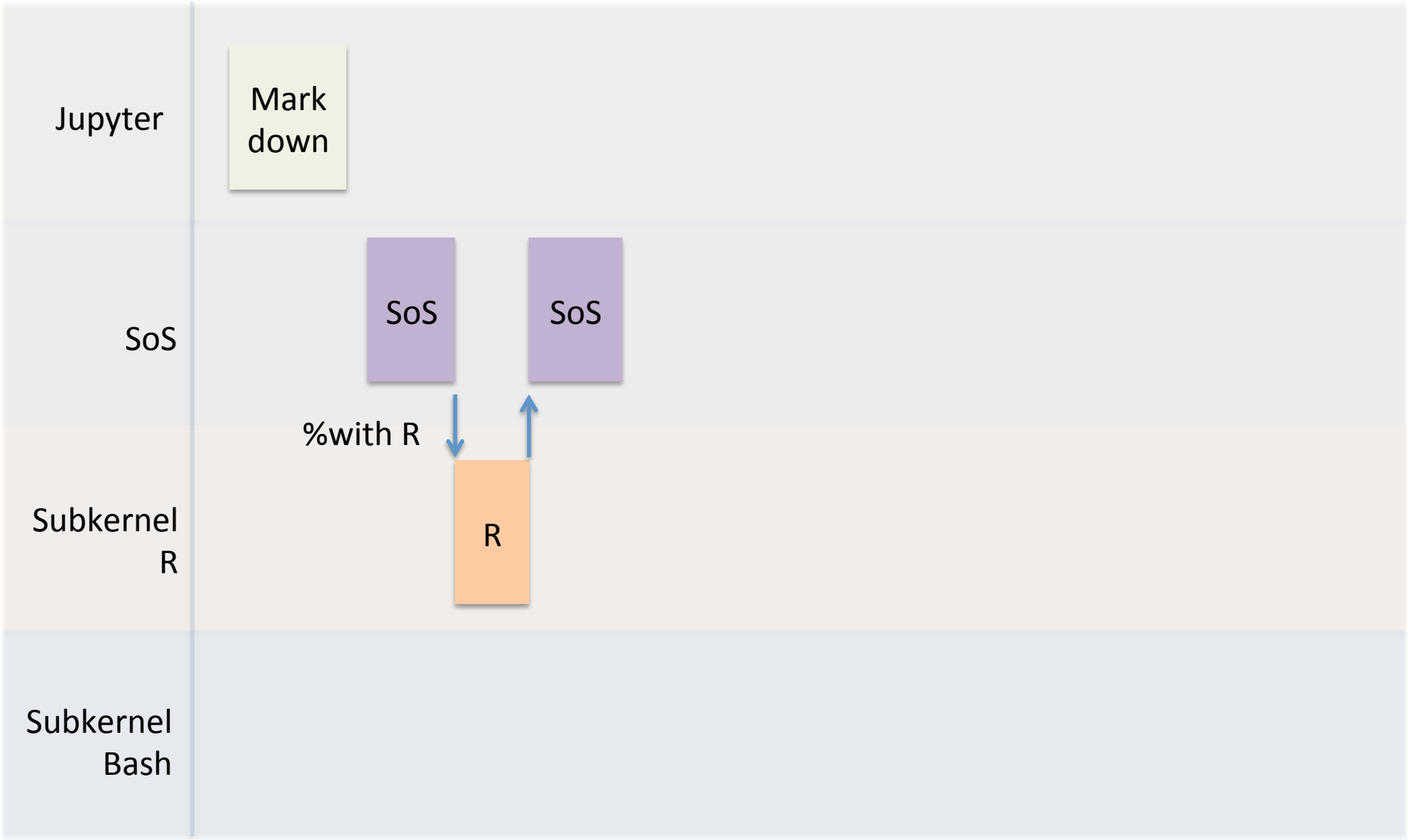
SoS Magics: %use and %with



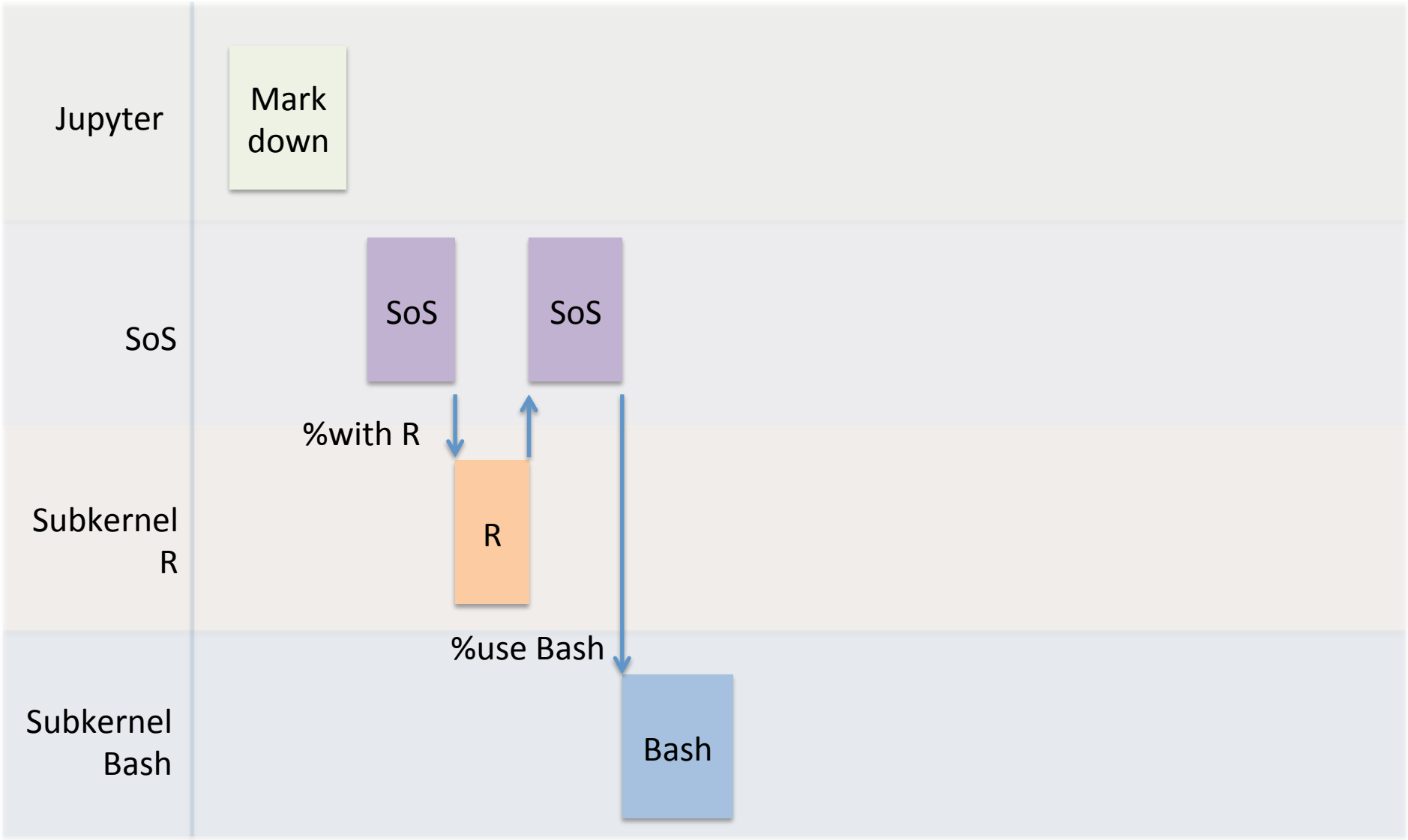
SoS Magics: %use and %with



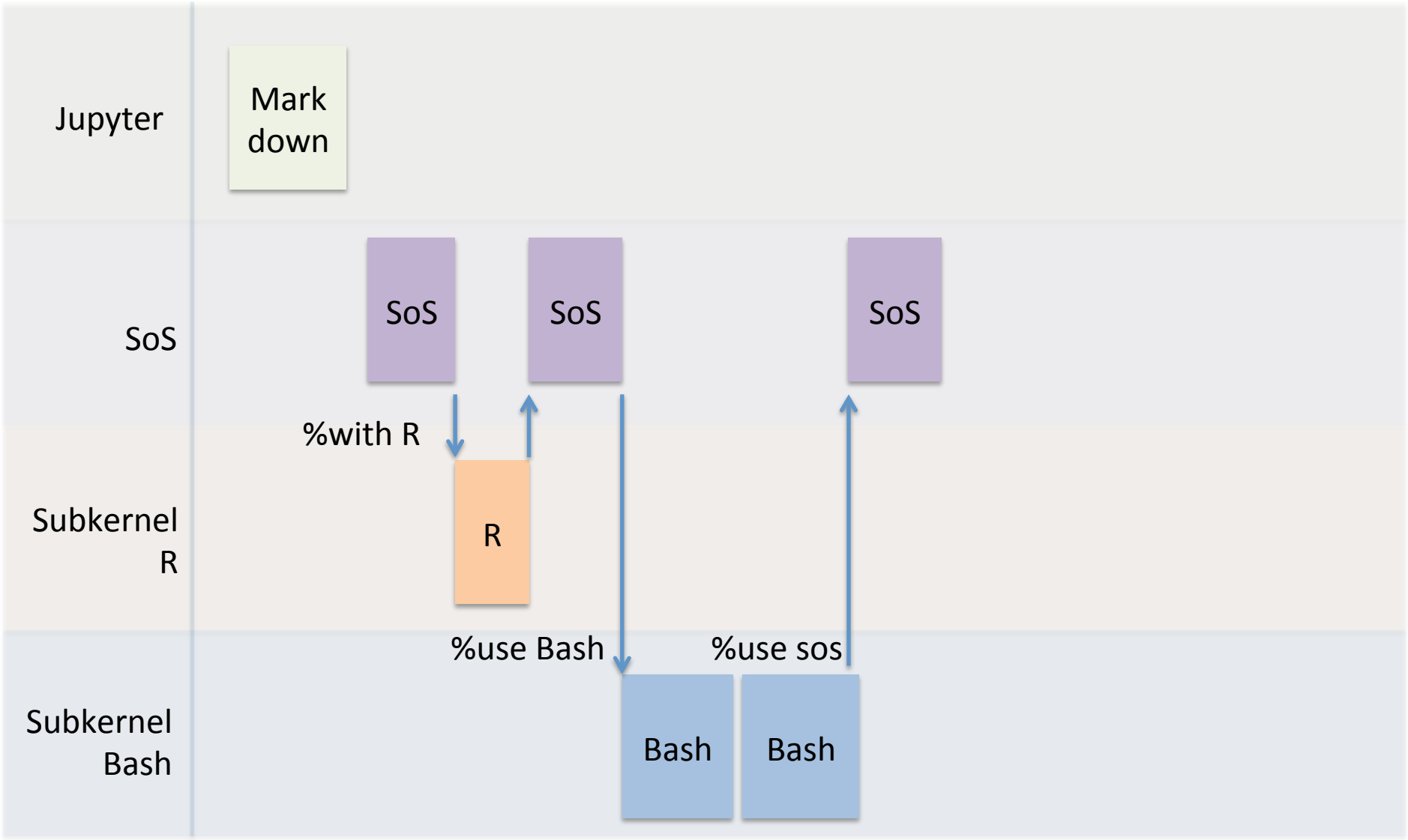
SoS Magics: %use and %with



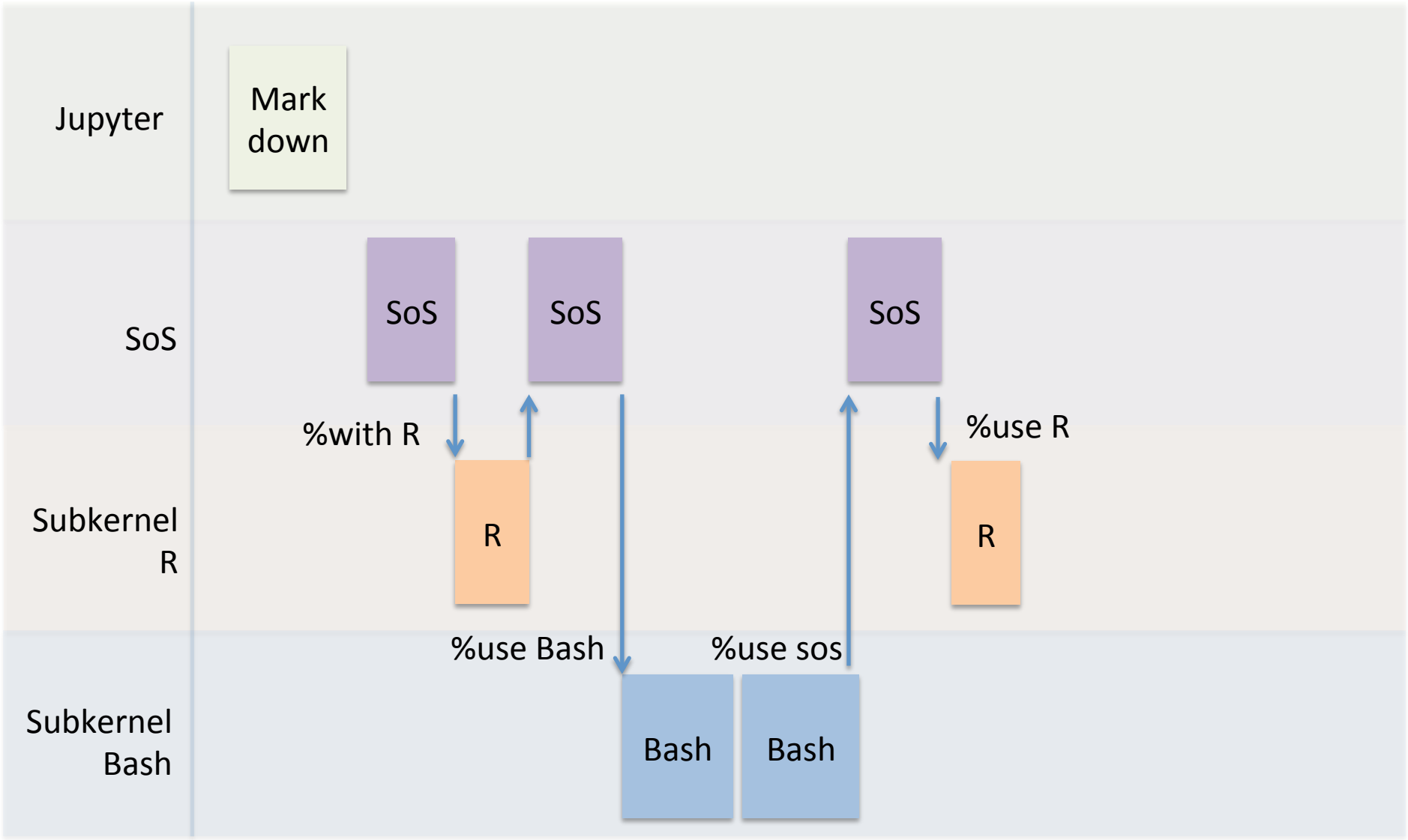
SoS Magics: %use and %with



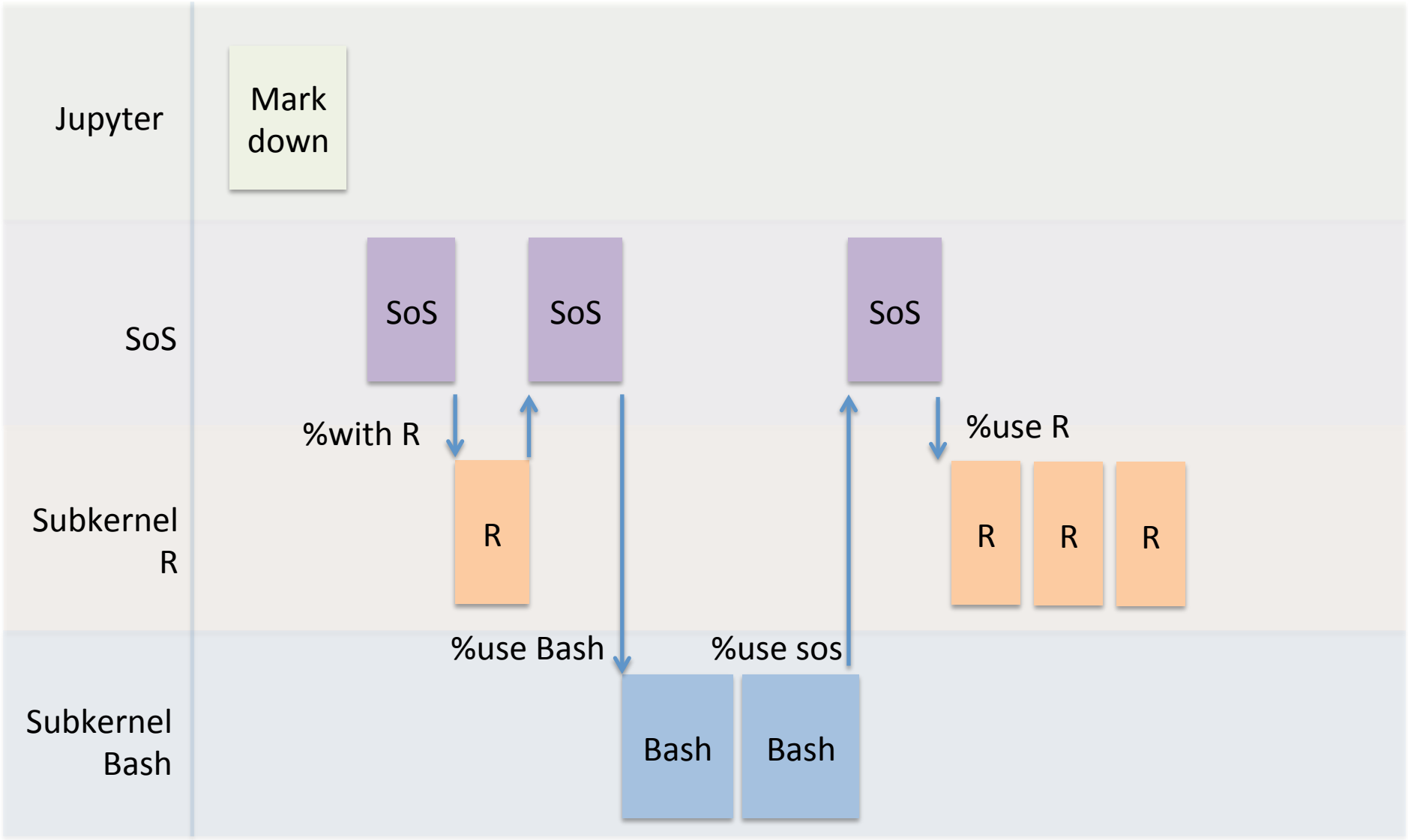
SoS Magics: %use and %with



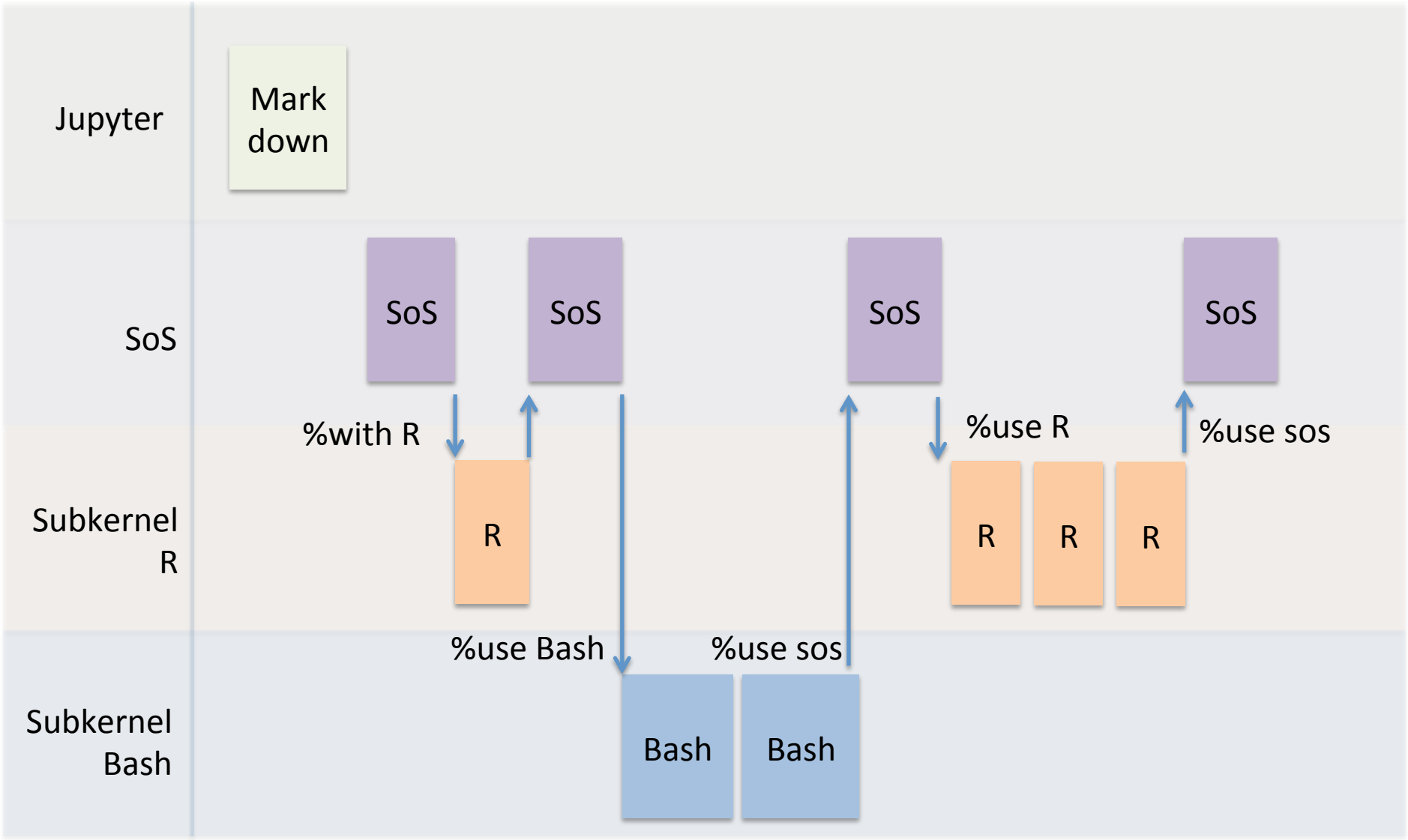
SoS Magics: %use and %with



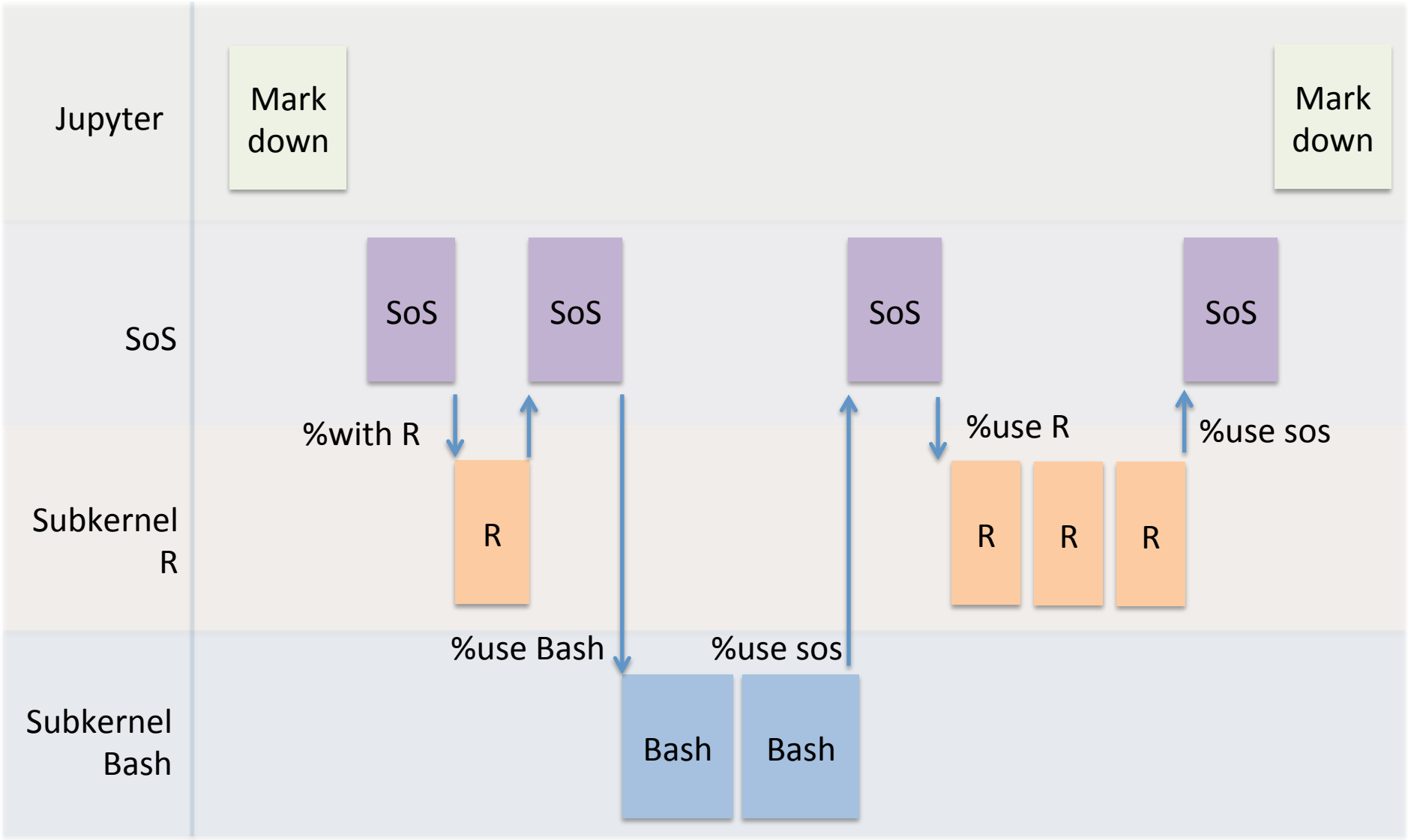
SoS Magics: %use and %with



SoS Magics: %use and %with



SoS Magics: %use and %with



String Interpolation

In [3]:

```
%preview ref_genome title all_names single_quoted formatted

resource_path = '~/.sos/resources'
ref_genome    = "${resource_path}/hg19/refGenome.fasta"

sample_names  = ['A', 'B', 'C']
title         = "Sample ${sample_names[0]} results"
all_names     = "Samples ${sample_names}"

single_quoted = '${sample_names} is not interpolated'

formatted     = "${len(sample_names)/.11:.1f}% samples are processed"
```

SoS

```
## %preview ref_genome title all_names single_quoted formatted

> ref_genome:

'~/.sos/resources/hg19/refGenome.fasta'

> title:

'Sample A results'

> all_names:

'Samples A B C'

> single_quoted:

'${sample_names} is not interpolated'

> formatted:

'27.3% samples are processed'
```

- Double or triple double quoted string
- Default sigil ${ }
- Arbitrary Python expression allowed
- Python format specification allowed

String Conversion

```
In [22]: %preview name filename basefilename expanded parparname
         file = '~/sos/examples/update_toc.sos'
         name = "${file!n}"
         filename = "${file!b}"
         basefilename = "${file!bn}"
         expanded = "${file!e}"
         parparname = "${file!ddb}"

         ## %preview name filename basefilename expanded parparname

         > name:

         '~/sos/examples/update_toc'

         > filename:

         'update_toc.sos'

         > basefilename:

         'update_toc'

         > expanded:

         '/Users/bpeng1/sos/examples/update_toc.sos'

         > parparname:

         'sos'
```

convertor	effect
s	str()
r	repr()
q	quoted()
e	expanduser()
a	abspath(expanduser())
b	basename())
d	dirname())
n	splitext()[0]
,	', '.join()

String Conversion

```
In [22]: %preview name filename basefilename expanded parparname
         file = '~/sos/examples/update_toc.sos'
         name = "${file!n}"
         filename = "${file!b}"
         basefilename = "${file!bn}"
         expanded = "${file!e}"
```

```
In [23]: salary = {'James': 20, 'Bob': 25, 'Kathy': 18}
         R:
             employee = c(${salary!,r})
             print(employee)

         [1] "James" "Bob"   "Kathy"
```

```
> filename:

'update_toc.sos'

> basefilename:

'update_toc'

> expanded:

'/Users/bpeng1/sos/examples/update_toc.sos'

> parparname:

'sos'
```

r	repr()
q	quoted()
e	expanduser()
a	abspath(expanduser())
b	basename())
d	dirname())
n	splitext()[0]
,	', '.join()

Explicit Exchange of Variables

From subkernel

- `%get var1 var2 …`
- `%put var1 var2 …`

Explicit Exchange of Variables

From subkernel

- `%get var1 var2 …`
- `%put var1 var2 …`

From switching kernel

- `%with kernel --in var1 --out var2`
- `%use kernel --in var1 --out var2`

Explicit Exchange of Variables

From subkernel

- `%get var1 var2 …`
- `%put var1 var2 …`

From switching kernel

- `%with kernel --in var1 --out var2`
- `%use kernel --in var1 --out var2`

```
In [1]: n = 5

In [2]: %with R -i n -o rn
        rn <- rnorm(n)

In [3]: rn

Out[3]: [-0.751548450771555,
         -0.286746622784046,
         -0.142180678559069,
         1.40603706962753,
         0.0322491935376987]
```

Automatic Data Exchange

Variables with names starting with sos will be automatically exchanged

Automatic Data Exchange

Variables with names starting with sos will be automatically exchanged

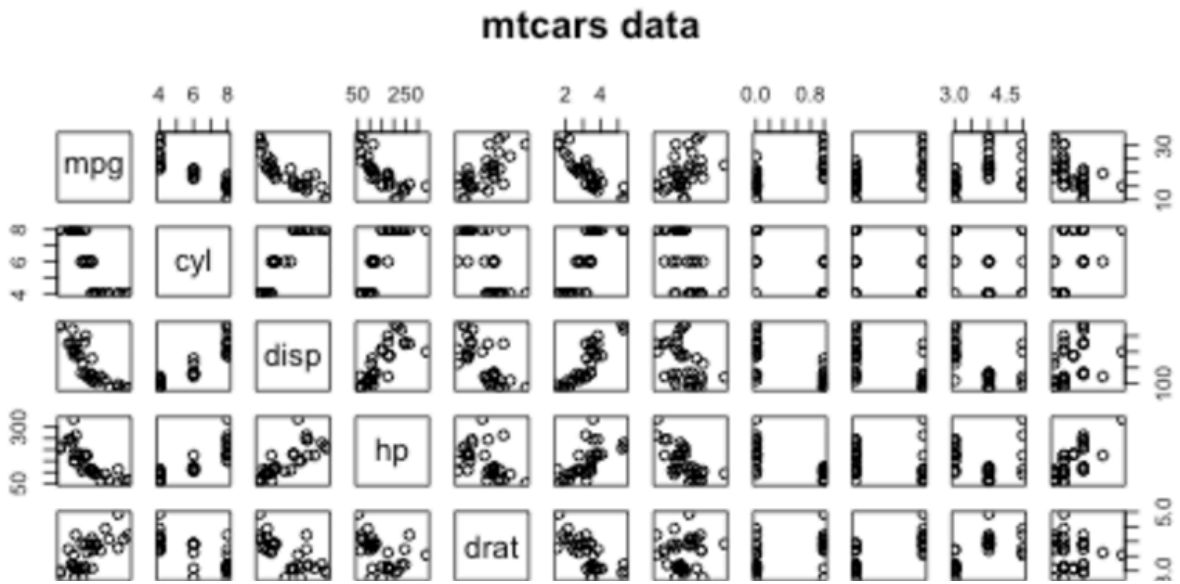
```
In [4]: sos_n = 5

In [5]: sos_rn <- rnorm(sos_n)

In [6]: sos_rn

Out[6]: [-0.15450269851469,
         -0.883153570911902,
         1.52965785081021,
         -1.82410951833293,
         0.558328894828463]
```

Explicit File and Expression Preview

In [4]:
```
%preview cars.png
%with R
require(graphics)
png('cars.png')
pairs(mtcars, main = "mtcars data")
dev.off()
```

pdf: 2

```
## %preview cars.png

> cars.png (83.0 KiB):
```



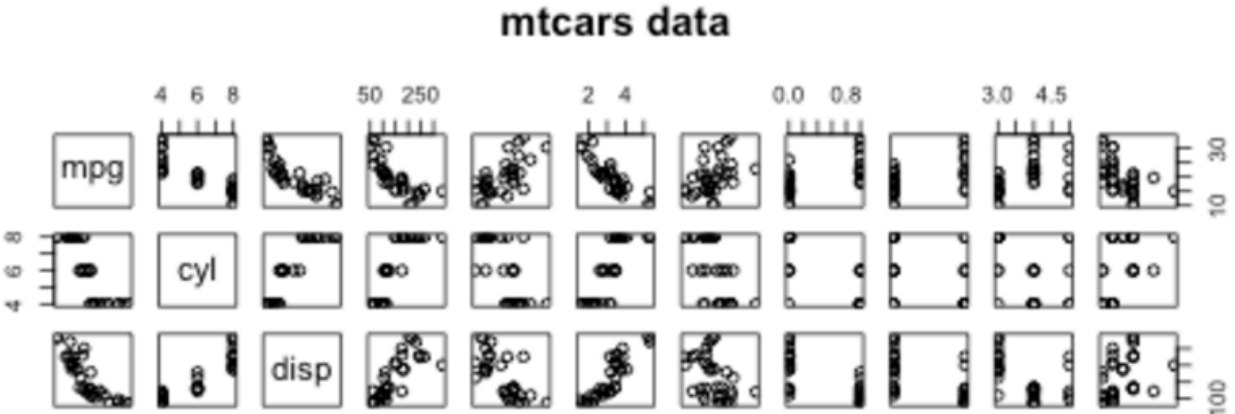
%preview magic previews files and (SoS) variables and expressions after the execution of the cell.

Automatic File Preview

In [6]:
```
output: 'cars.png'
R:
    require(graphics)
    png("${output}")
    pairs(mtcars, main = "mtcars data")
    dev.off()
```

```
null device
          1

## -- Preview output --

 input:
output: cars.png

> cars.png (83.0 KiB):
```

Output files of workflows will be automatically previewed.



Demonstration of Jupyter Notebook with SoS Kernel

SoS in batch mode

- SoS scripts
- Command line argument
- Input and output files
- Packing and unpacking
- Parallelization

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0
```

Header (optional)

```
# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

Description (optional)

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

Shell script

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

R script

The First SoS Script

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
# index reference genome
STAR --runMode genomeGenerate --genomeFastaFile human38.fasta \
    --genomeDir STAR_index

# align reads to the reference genome
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn control.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/control
STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
    --readFilesIn mutated.fasta --quantMode GeneCounts \
    --outFileNamePrefix aligned/mutated

R:
control.count <- read.table('aligned/control.out.tab')
mutated.count <- read.table('aligned/mutated.out.tab')
# normalize, compare, output etc, ignored.
pdf('myfigure.pdf')
# plot results
dev.off()
```

Indented Scripts

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
    # index reference genome
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq\
        --genomeDir STAR_index

    # align reads to the reference genome
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Shell script

Indented Scripts

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
    # index reference genome
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq\
        --genomeDir STAR_index

    # align reads to the reference genome
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

R script

Indented Scripts

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

run:
    # index reference genome
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq\
        --genomeDir STAR_index

    # align reads to the reference genome
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Execute the script

```
$ sos run myscript_1
INFO: Executing default_0:
INFO: default_0 input:    []
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
Generating myfigure.pdf
INFO: Workflow default (ID=faf24decb10430c2) is executed successfully.
```

Execute the script

```
$ sos run myscript_1
INFO: Executing default_0:
INFO: default_0 input:    []
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
Generating myfigure.pdf
INFO: Workflow default (ID=faf24decb10430c2) is executed successfully.
```

```
$ chmod +x myscript_1.sos
$ ./myscript_1.sos
INFO: Executing default_0:
INFO: default_0 input:    []
Generating STAR_index/chrName.txt
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
Generating myfigure.pdf
INFO: Workflow default (ID=faf24decb10430c2) is executed successfully.
```

Separate Scripts into Steps

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Separate Scripts into Steps

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Step 1

Separate Scripts into Steps

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Step 2

Separate Scripts into Steps

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Step 3

Separate Scripts into Steps

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn control.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn mutated.fastq --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Step descriptions

Separate Scripts into Steps

```
$ sos run myscript_3
INFO: Executing default_1: index reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: Executing default_2: align reads to the reference genome
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
INFO: Executing default_3: compare expression values
Generating myfigure.pdf
INFO: Workflow default (ID=5120e227d6e4adb3) is executed successfully.
```

Command Line Arguments

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Command Line Arguments

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Parameter

Command Line Arguments

```
#!/usr/bin/env sos-runner
#fileformat=SOS1.0

# This script aligns raw reads of a control and a mutated sample
# to the reference genome and compare the expression values
# of the samples at genes A, B and C.

# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# index reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align reads to the reference genome
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${fastq_files[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
R:
    control.count <- read.table('aligned/control.out.tab')
    mutated.count <- read.table('aligned/mutated.out.tab')
    # normalize, compare, output etc, ignored.
    pdf('myfigure.pdf')
    # plot results
    dev.off()
```

Use of variable
(string interpolation)

Command Line Arguments

```
$ sos run myscript_4
INFO: Executing default_1: index reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: Executing default_2: align reads to the reference genome
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
INFO: Executing default_3: compare expression values
Generating myfigure.pdf
INFO: Workflow default (ID=60e7bac0e231b6fd) is executed successfully.
```

Command Line Arguments

```
$ sos run myscript_4
INFO: Executing default_1: index reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: Executing default_2: align reads to the reference genome
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
INFO: Executing default_3: compare expression values
Generating myfigure.pdf
INFO: Workflow default (ID=60e7bac0e231b6fd) is executed successfully.
```

```
$ sos run myscript_4 --fastq-files ctrl.fastq case.fastq
INFO: Executing default_1: index reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: Executing default_2: align reads to the reference genome
Generating aligned/control.out.tab from ctrl.fastq
Generating aligned/mutated.out.tab from case.fastq
INFO: Executing default_3: compare expression values
Generating myfigure.pdf
INFO: Workflow default (ID=0757f512161e77d6) is executed successfully.
```

Specify input and output files

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# create a index for reference genome
output: 'STAR_index/chrName.txt'
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Specify input and output files

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# create a index for reference genome
output: 'STAR_index/chrName.txt'
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Output of step 1
(No input)

Specify input and output files

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# create a index for reference genome
output: 'STAR_index/chrName.txt'
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Input, output and dependent files of step 2

Specify input and output files

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# create a index for reference genome
output: 'STAR_index/chrName.txt'
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Output of step 3
(Input is the output of step 2)

Specify input and output files

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[1]
# create a index for reference genome
output: 'STAR_index/chrName.txt'
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[2]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[3]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Use of SoS variables `input` and `output`

Ignore executed steps

```
$ sos run myscript_5
INFO: Executing default_1: create a index for reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: default_1 output:   ['STAR_index/chrName.txt']
INFO: Executing default_2: align the reads to the reference genome
INFO: default_2 input:    ['control.fastq', 'mutated.fastq']
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
INFO: default_2 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_3: compare expression values
INFO: default_3 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
Generating myfigure.pdf
INFO: default_3 output:   ['myfigure.pdf']
INFO: Workflow default (ID=f0a8e4ef98542ae9) is executed successfully.
```

Ignore executed steps

```
$ sos run myscript_5
INFO: Executing default_1: create a index for reference genome
INFO: default_1 input:    []
Generating STAR_index/chrName.txt
INFO: default_1 output:   ['STAR_index/chrName.txt']
INFO: Executing default_2: align the reads to the reference genome
INFO: default_2 input:    ['control.fastq', 'mutated.fastq']
Generating aligned/control.out.tab from control.fastq
Generating aligned/mutated.out.tab from mutated.fastq
INFO: default_2 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_3: compare expression values
INFO: default_3 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
Generating myfigure.pdf
INFO: default_3 output:   ['myfigure.pdf']
INFO: Workflow default (ID=f0a8e4ef98542ae9) is executed successfully.
```

```
$ sos run myscript_5
INFO: Executing default_1: create a index for reference genome
INFO: default_1 input:    []
INFO: Step default_1 (index=0) is ignored due to saved signature
INFO: default_1 output:   ['STAR_index/chrName.txt']
INFO: Executing default_2: align the reads to the reference genome
INFO: default_2 input:    ['control.fastq', 'mutated.fastq']
INFO: Step default_2 (index=0) is ignored due to saved signature
INFO: default_2 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_3: compare expression values
INFO: default_3 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Step default_3 (index=0) is ignored due to saved signature
INFO: default_3 output:   ['myfigure.pdf']
INFO: Workflow default (ID=f0a8e4ef98542ae9) is executed successfully.
```

Packing and Unpacking an Analysis

```
$ sos pack f0a8 -o my_analysis.sar -y
INFO: Checking mutated.fastq
INFO: Checking aligned/mutated.out.tab
INFO: Checking aligned/control.out.tab
INFO: Checking myfigure.pdf
INFO: Checking control.fastq
INFO: Checking STAR_index/chrName.txt
INFO: Archiving 6 files (110 B)...
INFO: Adding myscript_5.sos
INFO: Adding mutated.fastq
INFO: Adding aligned/mutated.out.tab
INFO: Adding aligned/control.out.tab
INFO: Adding myfigure.pdf
INFO: Adding control.fastq
INFO: Adding STAR_index/chrName.txt
INFO: Adding runtime files
```

Command pack archives scripts, tracked (input, output, depends files) and optional untracked files, runtime signatures to a single compressed archive.

Packing and Unpacking an Analysis

```
$ sos pack f0a8 -o my_analysis.sar -y
INFO: Checking mutated.fastq
INFO: Checking aligned/mutated.out.tab
INFO: Checking aligned/control.out.tab
INFO: Checking myfigure.pdf
INFO: Checking control.fastq
INFO: Checking STAR_index/chrName.txt
INFO: Archiving 6 files (110 B)...
INFO: Adding myscript_5.sos
INFO: Adding mutated.fastq
INFO: Adding aligned/mutated.out.tab
INFO: Adding ali
INFO: Adding myfi
INFO: Adding cont
INFO: Adding STAR
INFO: Adding runt
```

Command pack archives scripts, tracked (input, output, depends files) and optional untracked files, runtime signatures to a single compressed archive.

```
$ sos unpack -l my_analysis.sar
    Length  Date      Time  Type     Name
    ------  ----      ----  ----     ----
   1.3 KiB  01-23-17 00:44 SCRIPTS  myscript_5.sos
       0 B  01-23-17 05:54 TRACKED  mutated.fastq
      43 B  01-23-17 17:25 TRACKED  aligned/mutated.out.tab
      43 B  01-23-17 17:25 TRACKED  aligned/control.out.tab
       8 B  01-23-17 17:25 TRACKED  myfigure.pdf
       0 B  01-23-17 05:54 TRACKED  control.fastq
      16 B  01-23-17 17:25 TRACKED  STAR_index/chrName.txt
    ------                          ----
   1.4 KiB                          7 files
```

Makefile-style Rules

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Makefile-style Rules

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Auxiliary step

Makefile-style Rules

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Triggered by input or depends rules

Makefile-style Rules

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
input:    fastq_files
depends:  'STAR_index/chrName.txt'
output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[0]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/control
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${input[1]} --quantMode GeneCounts \
        --outFileNamePrefix aligned/mutated

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Makefile-style Rules

```
$ sos run myscript_6
INFO: Adding step build_index with output STAR_index/chrName.txt
INFO: Executing build_index: create a index for reference genome
INFO: Step build_index (index=0) is ignored due to saved signature
INFO: build_index output:   ['STAR_index/chrName.txt']
INFO: Executing default_1: align the reads to the reference genome
INFO: default_1 input:    ['control.fastq', 'mutated.fastq']
INFO: Step default_1 (index=0) is ignored due to saved signature
INFO: default_1 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_2: compare expression values
INFO: default_2 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Step default_2 (index=0) is ignored due to saved signature
INFO: default_2 output:   ['myfigure.pdf']
INFO: Workflow default (ID=823c0fe0cd56387b) is executed successfully.
```

Makefile-style Rules

```
$ sos run myscript_6
INFO: Adding step build_index with output STAR_index/chrName.txt
INFO: Executing build_index: create a index for reference genome
INFO: Step build_index (index=0) is ignored due to saved signature
INFO: build_index output:   ['STAR_index/chrName.txt']
INFO: Executing default_1: align the reads to the reference genome
INFO: default_1 input:    ['control.fastq', 'mutated.fastq']
INFO: Step default_1 (index=0) is ignored due to saved signature
INFO: default_1 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_2: compare expression values
INFO: default_2 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Step default_2 (index=0) is ignored due to saved signature
INFO: default_2 output:   ['myfigure.pdf']
INFO: Workflow default (ID=823c0fe0cd56387b) is executed successfully.
```

Makefile-style Rules

```
$ sos run myscript_6
INFO: Adding step build_index with output STAR_index/chrName.txt
INFO: Executing build_index: create a index for reference genome
INFO: Step build_index (index=0) is ignored due to saved signature
INFO: build_index output:   ['STAR_index/chrName.txt']
INFO: Executing default_1: align the reads to the reference genome
INFO: default_1 input:    ['control.fastq', 'mutated.fastq']
INFO: Step default_1 (index=0) is ignored due to saved signature
INFO: default_1 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_2: compare expression values
INFO: default_2 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Step default_2 (index=0) is ignored due to saved signature
INFO: default_2 output:   ['myfigure.pdf']
INFO: Workflow default (ID=823c0fe0cd56387b) is executed successfully.
```

Makefile-style Rules

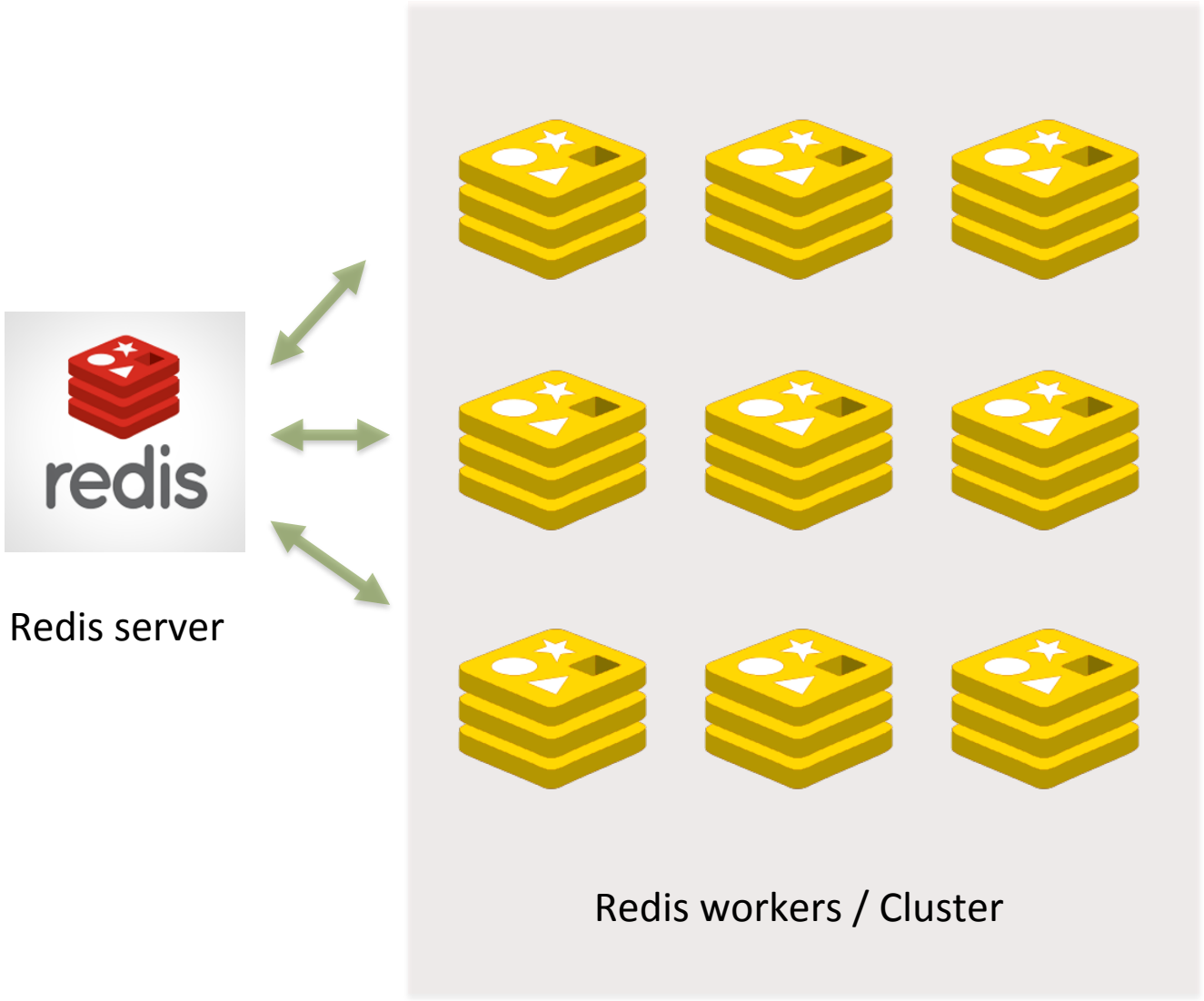
```
$ sos run myscript_6
INFO: Adding step build_index with output STAR_index/chrName.txt
INFO: Executing build_index: create a index for reference genome
INFO: Step build_index (index=0) is ignored due to saved signature
INFO: build_index output:   ['STAR_index/chrName.txt']
INFO: Executing default_1: align the reads to the reference genome
INFO: default_1 input:    ['control.fastq', 'mutated.fastq']
INFO: Step default_1 (index=0) is ignored due to saved signature
INFO: default_1 output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_2: compare expression values
INFO: default_2 input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Step default_2 (index=0) is ignored due to saved signature
INFO: default_2 output:   ['myfigure.pdf']
INFO: Workflow default (ID=823c0fe0cd56387b) is executed successfully.
```

Execution unit is independent of scripts!

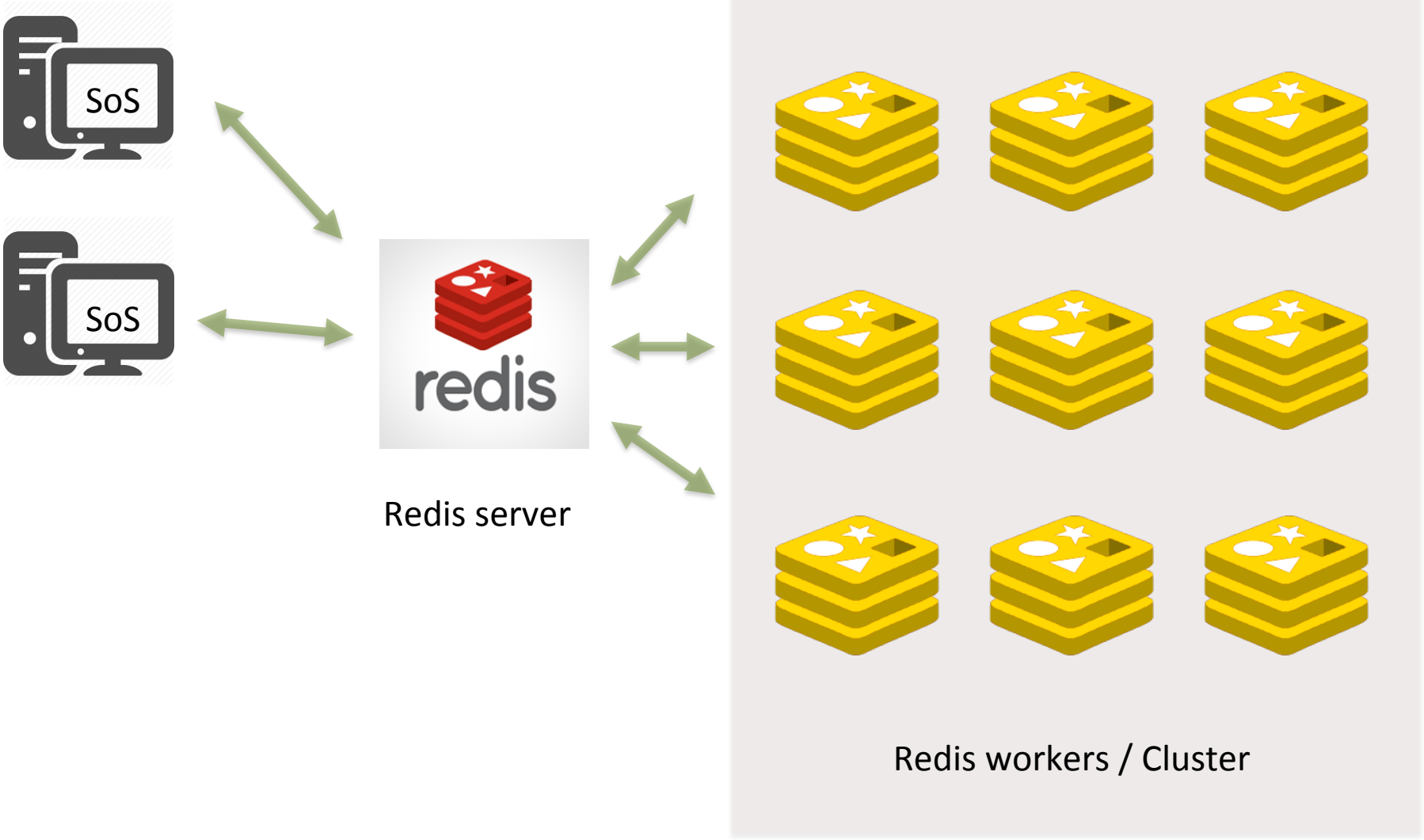
External Execution of Tasks

Redis server

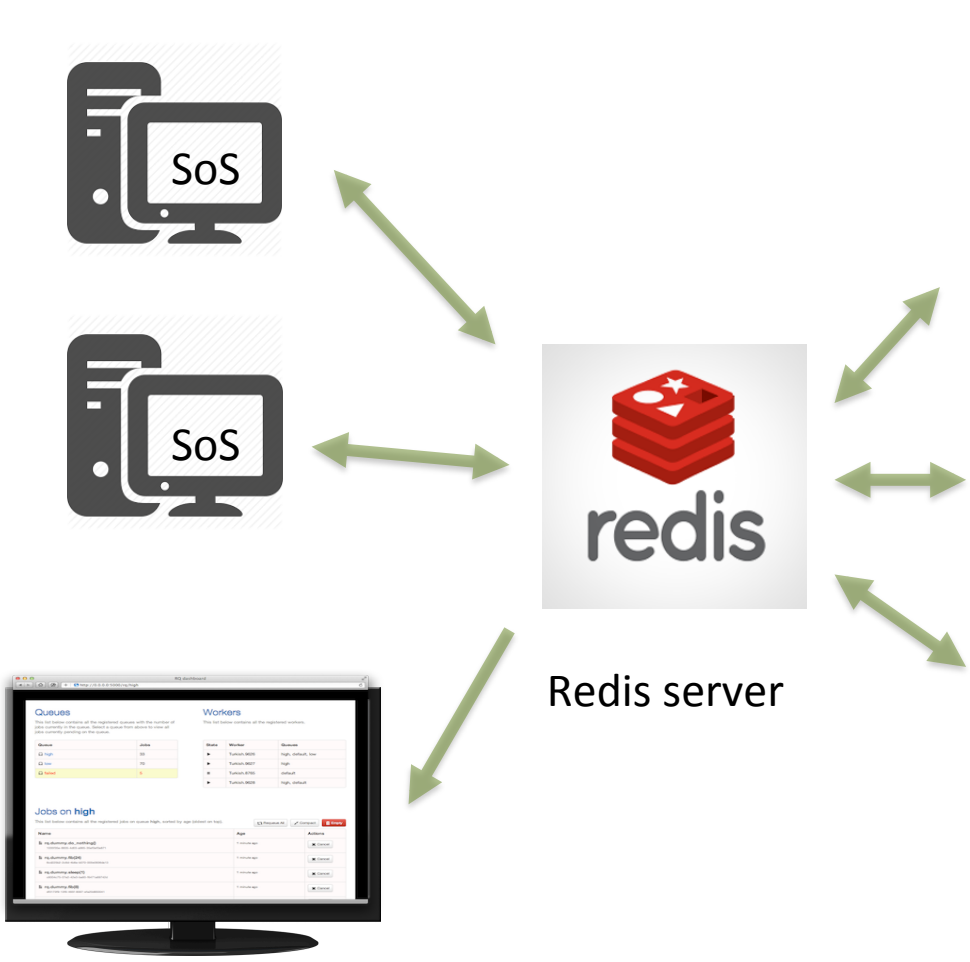
External Execution of Tasks



External Execution of Tasks



External Execution of Tasks



Redis server

Dashboard

Redis workers / Cluster

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

task:     concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

task:     concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

`input` files are sent one by one to `_input`

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:     fastq_files, group_by='single', paired_with='sample_type'
depends:   'STAR_index/chrName.txt'
output:    "aligned/${_sample_type}.out.tab"

task:      concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

`input` files are paired with `sample_type`

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

task:     concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

Each `_input` has a corresponding `_sample_type`

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

task:     concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

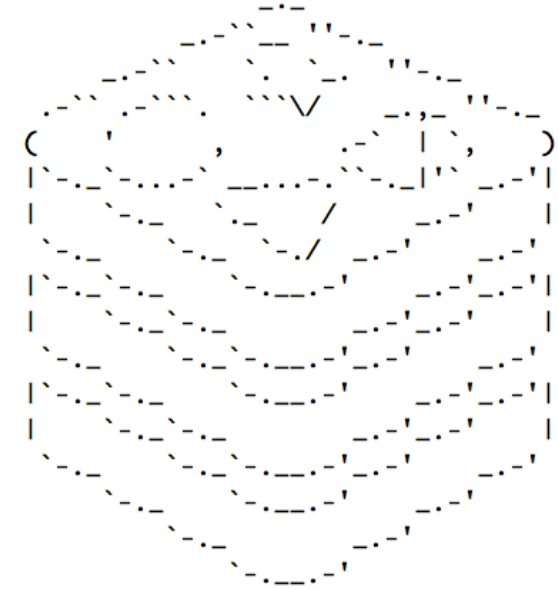
Execute in parallel or in external task queues.

External Tasks

```
# Two input files in .fastq formats. The first one for control sample
# and the second one for mutated sample.
parameter: fastq_files=['control.fastq', 'mutated.fastq']

[build_index: provides='STAR_index/chrName.txt']
# create a index for reference genome
run:
    STAR --runMode genomeGenerate --genomeFastaFile human38.fastq \
        --genomeDir STAR_index

[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

task:     concurrent=True
run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()
```

External Tasks – redis server


```
$ redis-server
74542:C 23 Jan 12:36:23.440 # Warning: no config file specified, using the default config. In order to specify
 a config file use redis-server /path/to/redis.conf
74542:M 23 Jan 12:36:23.441 * Increased maximum number of open files to 10032 (it was originally set to 7168).
```



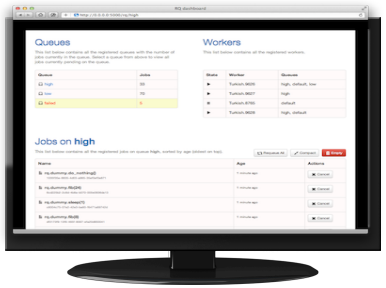
```
Redis 3.2.0 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 74542


        http://redis.io
```

```
74542:M 23 Jan 12:36:23.448 # Server started, Redis version 3.2.0
74542:M 23 Jan 12:36:23.449 * The server is now ready to accept connections on port 6379
```

External Task – redis worker

```
$ rq worker
12:57:43 RQ worker 'rq:worker:bcbm-bpeng.78463' started, version 0.6.0
12:57:43 Cleaning registries for queue: default
12:57:43
12:57:43 *** Listening on default...
```

External Task -- dashboard



```
$ rq-dashboard
RQ Dashboard version 0.3.7
 * Running on http://0.0.0.0:9181/ (Press CTRL+C to quit)
```

Queues

This list below contains all the registered queues with the number of jobs currently in the queue. Select a queue from above to view all jobs currently pending on the queue.

Queue	Jobs
No queues.	

Workers

This list below contains all the registered workers.

State	Worker	Queues
⏸	bcbm-bpeng.78463	default

Jobs on **default**

This list below contains all the registered jobs on queue **default**, sorted by age (oldest on top).

Requeue All | Compact | Empty

Name	Age	Actions
No jobs.		

« »

← Home

Execute Workflow

```
$ sos remove --signature
INFO: 11 runtime signatures removed
$ sos run myscript_8 -q rq
INFO: Adding step build_index with output STAR_index/chrName.txt
INFO: Executing build_index: create a index for reference genome
Generating STAR_index/chrName.txt
INFO: output:   ['STAR_index/chrName.txt']
INFO: Executing default_1: align the reads to the reference genome
INFO: input:    ['control.fastq', 'mutated.fastq']
INFO: output:   ['aligned/control.out.tab', 'aligned/mutated.out.tab']
INFO: Executing default_2: compare expression values
INFO: input:    ['aligned/control.out.tab', 'aligned/mutated.out.tab']
Generating myfigure.pdf
INFO: output:   ['myfigure.pdf']
INFO: Workflow default (ID=b1501d3457e30bc6) is executed successfully.
```

Execute Workflow

```
$ sos remove
INFO: 11 runt
$ sos run mys
INFO: Adding
INFO: Executi
Generating ST
INFO: output:
INFO: Executi
INFO: input:
INFO: output:
INFO: Executi
INFO: input:
Generating my
INFO: output:
INFO: Workflo
```

Queues

This list below contains all the registered queues with the number of jobs currently in the queue. Select a queue from above to view all jobs currently pending on the queue.

Queue	Jobs
default	1

Workers

This list below contains all the registered workers.

State	Worker	Queues
▶	bcbm-bpeng.78463	default

Jobs on **default**

This list below contains all the registered jobs on queue **default**, sorted by age (oldest on top).

Requeue All | Compact | Empty

Name	Age	Actions
sos.sos_step.execute_task(default_1 (index=1)) from **default** 5e0e2d8f-2824-41c3-b7ac-fe374cb0384d Failed 6 hours ago None	4 seconds ago	Requeue Cancel

« 1 »

Home

Generating a Report

```
[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

report:   output='align.md', active=-1
    ## Alignment of reads
    * input file: ${input}
    * output file: ${output}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()

report:   output='analysis.md', active=-1
    ## Statistical analysis
    * input file: ${input}
    * Figure: ${input}

[3]
pandoc: input=['align.md', 'analysis.md'], output='report.html'
    # An analysis of two RNA Seq samples
```

Generating a Report

```
[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

report:   output='align.md', active=-1
    ## Alignment of reads
    * input file: ${input}
    * output file: ${output}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()

report:   output='analysis.md', active=-1
    ## Statistical analysis
    * input file: ${input}
    * Figure: ${input}

[3]
pandoc: input=['align.md', 'analysis.md'], output='report.html'
    # An analysis of two RNA Seq samples
```

Report for step 1

Generating a Report

```
[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

report:   output='align.md', active=-1
    ## Alignment of reads
    * input file: ${input}
    * output file: ${output}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()

report:   output='analysis.md', active=-1
    ## Statistical analysis
    * input file: ${input}
    * Figure: ${input}

[3]
pandoc: input=['align.md', 'analysis.md'], output='report.html'
    # An analysis of two RNA Seq samples
```

Report for step 2

Generating a Report

```
[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

run:
    STAR --genomeDir STAR_index --outSAMtype BAM SortedByCoordinate \
        --readFilesIn ${_input!q} --quantMode GeneCounts \
        --outFileNamePrefix aligned/${_sample_type}

report:   output='align.md', active=-1
    ## Alignment of reads
    * input file: ${input}
    * output file: ${output}

[2]
# compare expression values
output: 'myfigure.pdf'
R:
    control.count <- read.table('${input[0]}')
    mutated.count <- read.table('${input[1]}')
    # normalize, compare, output etc, ignored.
    pdf('${output}')
    # plot results
    dev.off()

report:   output='analysis.md', active=-1
    ## Statistical analysis
    * input file: ${input}
    * Figure: ${input}

[3]
pandoc: input=['align.md', 'analysis.md'], output='report.html'
    # An analysis of two RNA Seq samples
```

Summary report

Generating a Report

```
[1]
# align the reads to the reference genome
sample_type = ['control', 'mutated']
input:    fastq_files, group_by='single', paired_with='sample_type'
depends:  'STAR_index/chrName.txt'
output:   "aligned/${_sample_type}.out.tab"

run:
    STAR --genomeDir S
        --readFilesIn
        --outFileNameP

report:   output='alig
    ## Alignment of re
    * input file: ${in
    * output file: ${o

[2]
# compare expression v
output: 'myfigure.pdf'
R:
    control.count <- r
    mutated.count <- r
    # normalize, compa
    pdf('${output}')
    # plot results
    dev.off()

report:   output='anal
    ## Statistical ana
    * input file: ${input}
    * Figure: ${input}

[3]
pandoc: input=['align.md', 'analysis.md'], output='report.html'
    # An analysis of two RNA Seq samples
```

An analysis of two RNA Seq samples

Alignment of reads

- input file: control.fastq mutated.fastq
- output file: aligned/control.out.tab aligned/mutated.out.tab

Statistical analysis

- input file: aligned/control.out.tab aligned/mutated.out.tab
- Figure: aligned/control.out.tab aligned/mutated.out.tab

More features

- A live Jupyter server with SoS kernel
- Support for docker
- Using Spyder as an IDE
- Configuration files
- Group input files
- Other target types (e.g. executable)
- Sub, combined, and nested workflows
- Dry-run mode
- Advanced signature modes (force, build etc)
- Include steps from other sos scripts
- Celery task queues
- …

Status

More than one year of development (3 years if counting its predecessor Variant Pipeline Tools).

Version 0.9.1 as of today.

http://vatlab.github.io/SOS/

https://github.com/vatlab/SOS

docker

amazon web services


TEST ME!

Acknowledgements

- Dr. Gao Wang
- Jun Ma
- Chris Wakefield



- Dr. John Weinstein
- Dr. Suzanne Leal (BCM)


- Grant R01HG008972
- Grant 1R01HG005859 (Dr. Paul Scheet)
- The Michael and Susan Dell Foundation
- The Chapman Foundation