

Package ‘vmr’

February 20, 2021

Type Package

Encoding UTF-8

Title Virtual Machines for R

Version 0.0.2

Date 2021-02-19

Maintainer Jean-François Rey <jf.rey.public@gmail.com>

Description Manage, provision and use Virtual Machines pre-configured for R. Develop, test and build package in a clean environment. 'Vagrant' tool and a provider (such as 'Virtualbox') have to be installed.

URL <https://gitlab.com/rstuff/vmr>, <https://rstuff.gitlab.io/vmr>

BugReports <https://gitlab.com/rstuff/vmr/-/issues>

License GPL (>= 3)

LazyData true

BuildVignettes true

NeedsCompilation no

Biarch true

SystemRequirements Vagrant <<https://www.vagrantup.com>>

Depends utils, R (>= 3.3.0)

Imports jsonlite, curl

Collate 'vmr.R' 'package.R' 'virtualbox.R' 'vagrantcloudAPI.R' 'vagrant.R' 'vmr-methods.R'

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ssh

Config/testthat/edition 3

VignetteBuilder knitr

Author Jean-François Rey [cre, aut]

Repository CRAN

Date/Publication 2021-02-20 01:30:03 UTC

R topics documented:

vmr-package	3
getProviderOptions	4
print.vmr	4
summary.vmr	5
virtualboxGitlabRunner	5
virtualboxOptions	6
vmrBoxDownload	7
vmrConnect	8
vmrCreate	8
vmrDestroy	10
vmrDisconnect	11
vmrExec	11
vmrInfo	12
vmrInitEnv	12
vmrInstallPackages	13
vmrIsRunning	14
vmrList	14
vmrListBox	15
vmrListSnapshot	16
vmrLoad	16
vmrLocalBoxList	17
vmrLocalBoxPrune	17
vmrLocalBoxRemove	18
vmrLocalBoxUpdate	19
vmrMountDir	19
vmrPackageBuild	20
vmrPackageCheck	20
vmrPackageTest	21
vmrProvision	22
vmrRestoreSnapshot	22
vmrResume	23
vmrSend	23
vmrSetVerbose	24
vmrStart	24
vmrStatus	25
vmrStop	25
vmrSuspend	26
vmrTakeSnapshot	26
vmrUpdateEnvVersion	27
vmrUpdatePackages	28

`vmr-package`*Virtual Machines for R*

Description

Manage, provision and use Virtual Machines pre-configured for R. Develop, test and build package in a clean environment. 'Vagrant' tool and a provider (such as 'Virtualbox') have to be installed.

Details

Package: vmr
Type: Package
Version: 0.0.2
Date: 2021-02-19
License: GPL (>=3)

This package is a wrap of the **Vagrant** tool and more. It allows to manage, provision and use Virtual Machines pre-configured for R. It currently only uses 'Virtualbox' (>= 6.1.14) as provider. Vagrant tool have to be installed too. Used VMs come from <https://app.vagrantup.com/VMR> repository and the sources use to generate them can be found at <https://gitlab.com/rstuff/vms>. See vignettes for the documentations `browseVignette("vmr")`.

Author(s)

Jean-François Rey <jf.rey.public@gmail.com>

Maintainer: Jean-François Rey <jf.rey.public@gmail.com>

See Also

Useful links:

- <https://gitlab.com/rstuff/vmr>
- <https://rstuff.gitlab.io/vmr>
- Report bugs at <https://gitlab.com/rstuff/vmr/-/issues>

Examples

```
## Not run:  
library("vmr")  
  
## End(Not run)
```

getProviderOptions *List provider options*

Description

List a provider available options.

Usage

```
getProviderOptions(provider = "virtualbox", details = FALSE)
```

Arguments

provider	a provider name
details	if TRUE print options, otherwise return default options

Details

It return a list of options name and value for a specific provider. To get the help page do ?<provider_name>Options(), for example [[virtualboxOptions\(\)](#)].

Value

a list of options

Examples

```
vbOpts <- getProviderOptions(provider = "virtualbox")
print(vbOpts)
```

print.vmr *Print vmr object information*

Description

print information from a **vmr** object

Usage

```
## S3 method for class 'vmr'
print(x, ...)
```

Arguments

x	a vmr object
...	optional print arguments

Value

the **vmr** object (via invisible(x))

summary.vmr

Summary vmr object information

Description

print information from a **vmr** object

Usage

```
## S3 method for class 'vmr'
summary(object, ...)
```

Arguments

object a **vmr** object
 ... optional print arguments

Value

the **vmr** object (via invisible(x))

virtualboxGitlabRunner

Configure the guest VM to be use as a Gitlab-Runner

Description

Configure the guest VM to be use as a GitLab Runner and return the command to run in shell to register it.

Usage

```
virtualboxGitlabRunner(
  vmr,
  gitlab_url,
  gt_token,
  snapshot_name = "",
  vm_name = ""
)
```

Arguments

vmr	a vmr object
gitlab_url	a GitLab URL
gt_token	a GitLab registration token
snapshot_name	name of a snapshot to use if any
vm_name	the 'VirtualBox' VM name if not specified in 'vmr' object provider_options.

Value

Character command to run in shell to register it

Examples

```
## Not run:
cmd <- virtualboxGitLabRunner(vmr, "gitlab.com", "mytoken")
system(cmd)

## End(Not run)
```

virtualboxOptions *List 'VirtualBox' options available*

Description

List available options for 'VirtualBox' provider

Usage

```
virtualboxOptions(details = TRUE)
```

Arguments

details if TRUE print options (default), otherwise only return default options

Details

Get the 'VirtualBox' default options. It return a list as follow:

```
list(
  gui = TRUE,
  name = NULL,
  nic_type = NULL,
  linked_clone = FALSE,
  check_guest_additions = TRUE,
  modifyvm = list(cpus = "2", memory = "4096")
)
```

- **gui**: if TRUE show the GUI, otherwise headless mode is activated
- **name**: the 'VirtualBox' instance name
- **nic_type**: the NIC type for the network interface to use, by default use the default one. see [VirtualBox Networking](#)
- **linked_clone**: if TRUE, linked clones are based on a master VM, which is generated by importing the base box only once the first time it is required. For the linked clones only differencing disk images are created where the parent disk image belongs to the master VM. (Be careful, master VM can't be remove until linked_clone still exists)
- **check_guest_additions**: If TRUE (default) check if guest have guest additions installed.
- **modifyvm**: list of 'VirtualBox' properties for the guest VM (such as number of cpus, memory size,...). see ['VirtualBox' modifyvm](#)

Value

A default list of options

```
list(
  gui = TRUE,
  name = NULL,
  nic_type = NULL,
  linked_clone = FALSE,
  check_guest_additions = TRUE,
  modifyvm = list(cpus = "2", memory = "4096")
)
```

Examples

```
## Not run:
vb.opts <- virtualboxOptions(details = FALSE)
vb.opts$modifyvm$cpus <- "4"
vb.opts$modifyvm$memory <- "8192"
vb.opts

## End(Not run)
```

vnrBoxDownload

Download a Box

Description

Download a box from a **vnr** object.

Usage

```
vnrBoxDownload(vnr)
```

Arguments

vmr a **vmr** object

Value

a **vmr** object

vmrConnect *Open a ssh connection to guest machine*

Description

Open a ssh connection to guest machine

Usage

```
vmrConnect(vmr)
```

Arguments

vmr a **vmr** object

Details

To open a ssh connection 'ssh' package have to be installed.

Value

a **vmr** object

vmrCreate *Create a **vmr** environment class*

Description

Create a **vmr** object.

Usage

```
vmrCreate(  
  name,  
  provider = "virtualbox",  
  version = "latest",  
  provider.options = virtualboxOptions(FALSE)  
)
```


Arguments

name	a box name
provider	the box provider (default: "virtualbox")
version	the box version (default : "latest")
provider.options	provider options (call [getProviderOptions()] to get values)

Details

Create a S3 **vmr** object (a simple list). The object contains all information needed to configure and manage a **vmr** environment (a vagrant environment).

A **vmr** environment need mostly a box *name* and a *provider*. The environment is attached to the current working directory.

vmr object main attributs:

- **path**: working directory
- **org**: Vagrant cloud user/organization name 'VMR'
- **box**: the box name
- **version**: the box version
- **provider**: the provider
- **provider_options**: the provider options (see [[getProviderOptions\(\)](#)])
- **vagrantName**: Vagrant environment name
- **ID** <- Vagrant environment ID
- **synced_folder**: a list with source and destination
- **ssh_user**: the ssh user
- **ssh_pwd**: the ssh user password

Value

a **vmr** object (see details)

Examples

```
## Not run:
# List boxes available
boxes <- vmrList()
# Create a vmr object
vmr <- vmrCreate(boxes$Name[1])

# to customize the guest machine for virtualbox
virtualboxOpts <- getProviderOptions(provider = "virtualbox")
virtualboxOpts$modifyvm <- list(cpus = 4, memory = 4096)
virtualboxOpts$name <- "My VM Cool Name"
# To specify a provider and version
vmr <- vmrCreate(
```

```
    name = boxes$Name[1],
    provider = "virtualbox",
    version = boxes$Version[1],
    provider.options = virtualboxOpts
  )

  ## End(Not run)
```

vmrDestroy

*Remove all resources created in a **vmr** environment*

Description

Remove all resources created by [vmrStart\(\)](#)

Usage

```
vmrDestroy(id = "", force = FALSE)
```

Arguments

id	a vmr environment id (default : "" id from the current environment)
force	if TRUE force to remove

Details

Will by default remove all resources created from the current **vmr** environment. By specifying the *id* any environment with this *id* will be remove.

Value

the vagrant environment id

Examples

```
## Not run:
vmrStop()
vmrDestroy()

## End(Not run)
```

vnrDisconnect	<i>Disconnect ssh connection to guest machine</i>
---------------	---

Description

Close a ssh connection to the guest machine

Usage

```
vnrDisconnect(vnr)
```

Arguments

vnr a **vnr** object

Details

'ssh' package need to be installed.

Value

a **vnr** object

vnrExec	<i>Execute R methods into guest machine</i>
---------	---

Description

Run R method into guest machine.

Usage

```
vnrExec(cmd = c())
```

Arguments

cmd list of R command

Details

call Rscript -e "cmd" into the guest machine from the current **vnr** environment. Command are independents and do not keep memory of past commands.

Value

NULL

Examples

```
## Not run:
cmd <- c("Sys.info()", 'print("Hello World!")')
vnrExec(cmd)

## End(Not run)
```

vnrInfo *Get guest machine information*

Description

Get guest machine information. Print OS, R, R-devel and R packages information. Still in development.

Usage

```
vnrInfo()
```

Value

NULL

Examples

```
## Not run:
boxes <- vnrList()
vnr <- vnrCreate(boxes$Name[1])
vnr <- vnrInitEnv(vnr)
vnrStart()
vnrInfo()

## End(Not run)
```

vnrInitEnv *Initialize the **vnr** environment*

Description

Create **vnr** environment in the current directory. Set configuration into a template file name "Vagrantfile" and download the box if needed.

Usage

```
vnrInitEnv(vnr, force.vagrantfile = FALSE, force.download = FALSE)
```

Arguments

vmr a **vmr** object
 force.vagrantfile if TRUE force to overwrite environment configuration (default FALSE)
 force.download if TRUE force to download the box, otherwise do not (default FALSE).

Details

The **vmr** environment consist of a directory (the working directory) and a template file name *Vagrantfile*. If the box is not present in localhost it will be download.

Value

the **vmr** object

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1])
vmr <- vmrInitEnv(vmr)

## End(Not run)
```

vmrInstallPackages *Install R packages into guest machine*

Description

Install a list of R packages into the guest machine of the current **vmr** environment.

Usage

```
vmrInstallPackages(pkgs = c())
```

Arguments

pkgs list of R packages

Value

installed packages vector

Examples

```
## Not run:
vmrInstallPackages(c("vmr"))

## End(Not run)
```

vmrIsRunning	<i>Is vmr environment running</i>
--------------	--

Description

Check if a guest machine in a **vmr** environment is running

Usage

```
vmrIsRunning()
```

Value

TRUE if running, otherwise FALSE

Examples

```
## Not run:
lboxes <- vmrList()
vmr <- vmrCreate(lboxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrIsRunning()
vmrStop()
vmrIsRunning()

## End(Not run)
```

vmrList	<i>List available boxes from VagrantCloud</i>
---------	---

Description

List of available boxes from a VagrantCloud organization account.

Usage

```
vmrList(org = .VagrantCloudOrganization)
```

Arguments

org Vagrant Cloud organization name (default : 'VMR')

Details

Default usage lists boxes preconfigured with R from **VMR organization account**.

Value

a data.frame with Name, Provider, Version and Description of available boxes

vnrListBox	<i>List all available version of a box</i>
------------	--

Description

List all versions and providers available of a box.

Usage

```
vnrListBox(box_name, org = .VagrantCloudOrganization)
```

Arguments

box_name	the box name
org	Vagrant Cloud organization name (default : 'VMR')

Details

List information of a box from VagrantCloud. Default usage list information of a box preconfigured with R from [VMR organization account](#).

Value

a data.frame with "Name", "Version", "Description", "Provider" and "Date" of the box

Examples

```
## Not run:  
# List Boxes  
boxes <- vnrList()  
# Box informaion  
box_info <- vnrListBox(boxes$Name[1])  
box_info  
  
## End(Not run)
```

vmrListSnapshot	<i>List snapshot of the guest machine</i>
-----------------	---

Description

Print all snapshot name of the guest machine

Usage

```
vmrListSnapshot()
```

Value

NULL

vmrLoad	<i>Load a vmr environment containing a Vagrant file</i>
---------	--

Description

Load a **vmr** environment containing a VagrantFile and create a **vmr** object (see [[vmrCreate\(\)](#)] for object details).

Usage

```
vmrLoad(dir = "./", vagrantfileName = "Vagrantfile")
```

Arguments

dir	the vmr environment directory (default: ".")
vagrantfileName	a Vagrantfile name (default: "Vagrantfile")

Details

It read a Vagrant file template with **vmr** compatible parameters. It's an experimental Vagrant file reading, some parameters may not be loaded.

Value

a **vmr** object

Examples

```
## Not run:  
# load the Vagrantfile in the current directory  
vnr <- vnrLoad(getwd())  
  
## End(Not run)
```

vnrLocalBoxList	<i>List downloaded boxes</i>
-----------------	------------------------------

Description

List all boxes downloaded in localhost

Usage

```
vnrLocalBoxList()
```

Value

a data.frame with boxes Name, Providers and Version

Examples

```
## Not run:  
localBoxes <- vnrLocalBoxList()  
print(localBoxes)  
  
## End(Not run)
```

vnrLocalBoxPrune	<i>Remove old installed boxes</i>
------------------	-----------------------------------

Description

Removes old versions of installed boxes.

Usage

```
vnrLocalBoxPrune()
```

Value

a data.frame of still installed boxes (Name, Providers and Version)

Examples

```
## Not run:  
vnrLocalBoxPrune()  
  
## End(Not run)
```

vnrLocalBoxRemove	<i>Remove a box from localhost</i>
-------------------	------------------------------------

Description

Remove a specific box from localhost.

Usage

```
vnrLocalBoxRemove(name, provider = "", version = "", force = FALSE)
```

Arguments

name	the box name
provider	the box provider (default: first provider found)
version	the box version (default: version available)
force	if TRUE force to remove

Value

execution code or message

Examples

```
## Not run:  
lboxes <- vnrLocalBoxList()  
vnrLocalBoxRemove(lboxes$Name[[1]])  
# if multiple providers and versions  
vnrLocalBoxRemove(lboxes$Name[[1]], lboxes$Provider[[1]], lboxes$Version[[1]])  
  
## End(Not run)
```

vnrLocalBoxUpdate	<i>Update local box version</i>
-------------------	---------------------------------

Description

Download the latest version of the box use in the current **vnr** environment.

Usage

```
vnrLocalBoxUpdate()
```

Value

execution code or message

vnrMountDir	<i>Mount a host directory to guest</i>
-------------	--

Description

Mount a host directory to the guest machine.

Usage

```
vnrMountDir(vnr, src = "", dest = "")
```

Arguments

vnr	a vnr object
src	a host directory
dest	a destination guest directory

Details

If the option of mounting a directory is available in the guest provider, it will mount *src* to *destination* directory. Calling with no arguments will disable this option.

Value

a **vnr** object

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1])
vmr <- vmrMountDir(vmr, src = getwd(), dest = "/vmr")
vmr <- vmrInitEnv(vmr)
vmrStart()

## End(Not run)
```

vmrPackageBuild	<i>Build a package in the guest machine</i>
-----------------	---

Description

Build a package bundle or binary into the guest machine.

Usage

```
vmrPackageBuild(pkg = "./", binary = FALSE)
```

Arguments

pkg	a package directory or a tar.gz file
binary	if TRUE build binary package otherwise FALSE

Details

upload the package and run `devtools::build()` (build available in `$HOME/vmr/package/pkg`) in the current **vmr** environment.

Value

NULL

vmrPackageCheck	<i>Perform a package check on guest</i>
-----------------	---

Description

Perform a package check into the guest

Usage

```
vmrPackageCheck(pkg = "./")
```

Arguments

pkg a package directory or a tar.gz file

Details

upload the package and run `devtools::check()` into the guest machine. (check available in `$HOME/vnr/package/pkg`)

Value

NULL

Examples

```
## Not run:  
vnrPackageCheck("vnr_package.tar.gz")  
  
## End(Not run)
```

vnrPackageTest	<i>Test a package into a guest machine</i>
----------------	--

Description

Test a package into a guest machine

Usage

```
vnrPackageTest(pkg = ".")
```

Arguments

pkg a package directory or tar.gz

Details

Perform a package check into the guest machine of the current **vnr** environment using `devtools::test()`. (tests are available in `$HOME/vnr/package/pkg`)

Value

NULL

vmrProvision *Provision a **vmr** environment*

Description

Provision a **vmr** environment.

Usage

```
vmrProvision(cmd = c(), elts = c(), dest = "")
```

Arguments

cmd	list of shell commands
elts	list of files and/or directories
dest	destination of elts (default HOME/vmr)

Details

Upload 'elts' files and/or directories to the guest machine 'dest' from the current **vmr** environment. And finally run shell commands 'cmd' in the guest machine.

Value

NULL

vmrRestoreSnapshot *Restore a snapshot of the guest machine*

Description

Restore a snapshot of the guest machine.

Usage

```
vmrRestoreSnapshot(snap_name)
```

Arguments

snap_name	the snapshot name
-----------	-------------------

Value

the snapshot name

Examples

```
## Not run:
vnrRestoreSnapshot("my snapshot")

## End(Not run)
```

vnrResume	<i>Resume a stopped guest machine</i>
-----------	---------------------------------------

Description

Resume a stopped guest machine.

Usage

```
vnrResume()
```

Details

In the current **vnr** environment, start a stopped (`[vnrSuspend()]`) guest machine.

Value

NULL

vnrSend	<i>Send files and/or directories to guest machine</i>
---------	---

Description

Send files and/or directories to the guest machine in the current **vnr** environment. They are upload into `~/vnr/` directory.

Usage

```
vnrSend(elt = c())
```

Arguments

`elt` list of files and directories

Value

0 if OK, message otherwise

Examples

```
## Not run:
vmrSend(c("myfile"))

## End(Not run)
```

vmrSetVerbose	<i>Set verbose level</i>
---------------	--------------------------

Description

Set verbose level for vmr package functions

Usage

```
vmrSetVerbose(verbose_mode = "Normal")
```

Arguments

verbose_mode "None", "Normal" or "Full"

Details

Three verboses mode is available:

- "None" : print nothings
- "Normal" : print essential
- "Full" : print all

Value

invisible verbose value

vmrStart	<i>Start a vmr environment</i>
----------	---------------------------------------

Description

Start a guest virtual machine using the current **vmr** environment (directory and Vagrantfile template)

Usage

```
vmrStart()
```

Value

the vmr environment unique id

Examples

```
## Not run:
lboxes <- vnrList()
vnr <- vnrCreate(lboxes$Name[1])
vnr <- vnrInitEnv(vnr)
vnrStart()
vnrStop()

## End(Not run)
```

vnrStatus	<i>Get the state of the guest machine</i>
-----------	---

Description

Print guest machine state in the current **vnr** environment.

Usage

```
vnrStatus()
```

Value

a data.frame with Name, Provider and state

vnrStop	<i>Stop a vnr environment</i>
---------	--------------------------------------

Description

Stop a guest virtual machine in the current **vnr** environment.

Usage

```
vnrStop(force = FALSE)
```

Arguments

force if TRUE force to stop (powerOff), otherwise FALSE clean shutdown

Value

NULL

Examples

```
## Not run:
lboxes <- vnrList()
vnr <- vnrCreate(lboxes$Name[1])
vnr <- vnrInitEnv(vnr)
vnrStart()
vnrStop()

## End(Not run)
```

vnrSuspend	<i>Save state and stop guest machine</i>
------------	--

Description

Save the guest machine and stop it.

Usage

```
vnrSuspend()
```

Details

In the current **vnr** environment, save the state of the guest machine and stop it.

Value

NULL

vnrTakeSnapshot	<i>Take a snapshot of the guest machine</i>
-----------------	---

Description

Take a snapshot of the guest machine.

Usage

```
vnrTakeSnapshot(snap_name)
```

Arguments

snap_name the name given to the snapshot

Value

the snapshot name (invisible)

Examples

```
## Not run:  
vnrTakeSnapshot("my snapshot")  
  
## End(Not run)
```

vnrUpdateEnvVersion *Update a **vnr** environment.*

Description

Force to use the latest box version of the current **vnr** environment.

Usage

```
vnrUpdateEnvVersion(vnr)
```

Arguments

vnr a **vnr** object

Details

Put **vnr** object version to latest and update the Vagrant File template. Download the new box version if needed.

Value

a **vnr** object

Examples

```
## Not run:  
boxes <- vnrList()  
vnr <- vnrCreate(boxes$Name[1], version = "oldone")  
vnr <- vnrInitEnv(vnr)  
  
# update to latest  
vnr <- vnrUpdateEnvVersion(vnr)  
vnrStart()  
  
## End(Not run)
```

vnrUpdatePackages *Update R packages installed*

Description

Updates R packages installed in the guest machine.

Usage

```
vnrUpdatePackages()
```

Details

Will perform a `update.packages()` in the guest machine of the current **vnr** environment.

Examples

```
## Not run:  
lboxes <- vnrList()  
vnr <- vnrCreate(lboxes$Name[1])  
vnr <- vnrInitEnv(vnr)  
vnrStart()  
vnrUpdatePackages()  
  
## End(Not run)
```

Index

- * **machine**
 - vmr-package, 3
- * **provider**
 - vmr-package, 3
- * **provision**
 - vmr-package, 3
- * **vagrant**
 - vmr-package, 3
- * **virtualbox**
 - vmr-package, 3
- * **virtual**
 - vmr-package, 3
- _PACKAGE (vmr-package), 3
- devtools::build(), 20
- devtools::check(), 21
- devtools::test(), 21
- getProviderOptions, 4
- getProviderOptions(), 9
- print.vmr, 4
- summary.vmr, 5
- update.packages(), 28
- virtualboxGitlabRunner, 5
- virtualboxOptions, 6
- virtualboxOptions(), 4
- vmr (vmr-package), 3
- vmr-package, 3
- vmrBoxDownload, 7
- vmrConnect, 8
- vmrCreate, 8
- vmrCreate(), 16
- vmrDestroy, 10
- vmrDisconnect, 11
- vmrExec, 11
- vmrInfo, 12
- vmrInitEnv, 12
- vmrInstallPackages, 13
- vmrIsRunning, 14
- vmrList, 14
- vmrListBox, 15
- vmrListSnapshot, 16
- vmrLoad, 16
- vmrLocalBoxList, 17
- vmrLocalBoxPrune, 17
- vmrLocalBoxRemove, 18
- vmrLocalBoxUpdate, 19
- vmrMountDir, 19
- vmrPackageBuild, 20
- vmrPackageCheck, 20
- vmrPackageTest, 21
- vmrProvision, 22
- vmrRestoreSnapshot, 22
- vmrResume, 23
- vmrSend, 23
- vmrSetVerbose, 24
- vmrStart, 24
- vmrStart(), 10
- vmrStatus, 25
- vmrStop, 25
- vmrSuspend, 26
- vmrSuspend(), 23
- vmrTakeSnapshot, 26
- vmrUpdateEnvVersion, 27
- vmrUpdatePackages, 28