|  |  |  |  |
| --- | --- | --- | --- |
| Packages | Install | install.packages(‘package’) | pip install package |
|  | Load | library(package) | import package |
| Working directory | Current directory | wd = getwd() | import os  wd = os.getcwd() |
|  | Change directory | setwd(‘C:/R’) | os.chdir(‘C:\python’) |
| Time code |  | library(tictoc)  tic()  toc() |  |
| Data types | Print | class(x) | type(x) |
|  | Check | is.integer(x)  is.numeric(x)  is.character(x) | isinstance(x,int)  isinstance(x,float)  isinstance(x,str) |
|  | Convert to integer | as.integer(x)  as.numeric(x)  as.character(x) | int(x)  float(x)  str(x) |
| Vectors/Lists | Create | x = c(1,2,3) | x = [1, 2, 3] |
|  | Slice | x[1] | x[0] |
|  | Length | length(x) | len(x) |
|  | Sort | sort(x) | x.sort() |
|  | Reverse | rev(x) | x[::-1] |
|  | Append | x=c(x,4) | x.append(4) |
|  | Concatenate | c(x,y) | x+y |
| Manipulations | Check membership | 4 %in% x | 4 in x |
|  | Index of element | match(x,4) | x.index(4) |
|  | Index of condition | which(x>0) | [i for i,v in enumerate(x) if v>0] |
|  | Filtered by condition | x[x>0] | [n for n in x if n>0] |
| Booleans | Values | TRUE, FALSE | True, False |
|  | And | x==3 & y>2 | x==3 and y>2 |
|  | Or | x==3 | y>2 | x==3 or y>3 |
|  | Not equals | 3 != 2 | 3 != 2 |
|  | Not | !x>2 | not x>2 |
|  | Any | any(TRUE, FALSE) | any([True, False]) |
|  | All | all(TRUE, FALSE) | all([True, False]) |
| Sets | Create | unique(x) | set(x) |
|  | Union | union(x,y) | x|y |
|  | Intersection | intersect(x,y) | x&y |
|  | Difference | setdiff(x,y) | x-y |
| Matrices | Creation from rows | rbind(x,y) | np.array([x,y]) |
|  | Creation from elements | rbind(c(1,2,3),c(4,5,6)) | np.array([[1,2,3],[4,5,6]]) |
|  | Creation by dimension | matrix(0, nrow=2, ncol=3)  matrix(1, nrow=2, ncol=3) | np.zeros((2,3))  np.ones((2,3)) |
|  | Slice element | m[1,1] | m[0,0] |
|  | Slice row/col | m[1,] / m[,1] | m[0,:] / m[:,0] |
|  | Check dimensions | dim(m) | m.shape |
|  | Transpose | t(m) | m.transpose() |
| Dictionaries | Creation | x = c(‘a’=1, ‘b’=2) | x = {‘a’:1, ‘b’:2} |
|  | Creation from lists | x=c(1,2)  names(x)=c(‘a’,’b’) | x=[1,2]  names = [‘a’, ‘b’]  d = dict(zip(names, x)) |
|  | Check names | x  names(x) | x.values()  x.keys() |
| Strings | Pairwise paste | paste(‘col’, 1:3, sep=’.’) | [i+’.’+j for i,j in zip(x,y)] |
|  | Combine paste | paste(c(‘a’,’b’), collapse=”.’) | a+’,’+b |
|  | Template | sprintf(‘%s foo %d’, ‘hi’, 4) | ‘%s foo %d’ % (‘hi’,4) |
|  | Check substring | grepl(‘ab’, ’abcd’) | ‘ab’ in ‘abcd’ |
|  | Index of substring | grep(‘ab’, ‘abcd’) | ‘abcd’.index(‘ab’) |
|  | Length | nchar(x) | len(x) |
|  | Slice (front) | substr(x, 1, 3) | x[:3] |
|  | Slice (end) | substr(x, 5, nchar(x)) | x[3:] |
|  | Slice (middle) | substr(x, 2, 4) | x[2:4] |
|  | Uppercase | toupper(x) | x.upper() |
|  | Lowercase | tolower(x) | x.lower() |
|  | Leading whitespace | trimws(x, ‘l’) | x.lstrip() |
|  | Trailing whitespace | trimws(x, ‘r’) | x.rstrip() |
|  |  |  |  |
| Functions | Definition | my\_fun = function(x, y=4){  return(list(‘a’=a, ‘b’=b))  my\_fun$a / my\_fun$b | def my\_fun(x, y=4):  return(a,b)  a,b = my\_fun(x,y) |
|  | Anonymous |  | my\_fun = lambda x,y: x+y |
|  | Arbitrary # inputs | my\_fun = function(...){  return(list(...))  #always turn ... to a list  } | def my\_fun(\*args):  return(args) #tuple  # args is tuple  def my\_fun(\*\*kwargs):  return(kwargs) #dict |
|  | Access global var | var <<- var+1 |  |
| If loops | Ternary | if (condition){  } else if (condition) {  } else {  } | if condition:  elif condition:  else: |
| For loops | Basic | for (i in 1:10){  } | for item in list: |
|  | Enumerate |  | for i,item in enumerate(list): |
|  | Paired |  | for (i,j) in zip(x,y): |
| List comps | Map | f(l) | [f(x) for x in l] |
|  | Filter | l[l>0] | [x for x in l if x>0] |
|  | Multiple Filters | l[l>0 & l%2==0] | [x for x in l if all(x[i]>0 for i in range(4))] |
|  | Negative filter | l[!(l %in k)] | [x for x in l if x not in k] |
|  | Filter then map | f(l[l>0]) | [f(x) for x in l if x>0] |
|  | 2 args paired | x+y | [i+j for i,j in zip(x,y)] |
|  | 2 args all combinations |  | [i+j for i in x for j in y] |
|  |  |  |  |
| Data Frames | Empty Creation | data.frame(matrix(nrow=0, ncol=3))  colnames(df) = c(‘a’,’b’,’c’) | pd.DataFrame(index=[‘a’,’b’,’c’],  columns=[‘x’,’y’,’z’]) |
|  | Preview | head(df) | df.head() |
| Properties | Dimensions | nrow(df) / ncol(df) | df.shape[0] / df.shape[1] |
|  | Row names | rownames(df) | df.index |
|  | Col names | colnames(df) | df.columns |
|  | Tally col | table(df$a) | df[‘a’] |
| Slice | Element by index | df[1,1] | df.iloc[0,0] |
|  | Element by col name | df$a[1] / df[1,’a’] | df.loc[0,’a’] |
|  | by row | df[1:2,] | df.loc[0:1,:] |
|  | by col | df[‘a’] / df$a | df.loc[:,’a’] / df[[‘a’]] |
|  | by cols / rearrange | df[c(‘b’,’a’)] | df[[‘b’,’a’]] |
| Manipulate | Delete col | df$a = NULL | del df[‘a’] |
|  | Rename | df = rename(df, newcol=oldcol) | df.rename(columns={‘old’:’new’}) |
|  | Append new row | df = rbind(df, c(1,2,3)) | df.loc[len(df)+1] = [1,2,3] |
|  | Concatenate rows | rbind(x,y) | pd.concat([df1, df2], axis=1) |
|  | Concatenate cols | cbind(x,y) | pd.concat([df1, df2]) |
|  | Lookup | left\_join(df, lookup\_tbl, by=’col’) | df= df.merge(lookup\_tbl,  on=’col’, how=’left’) |
| Wrangling | Sort | arrange(df, a, desc(b)) | df.sort\_values([‘a’,’b’],  ascending=[True, False]) |
|  | Filter (and) | filter(df, a<2, b>5) | df[ df[‘a’]<2 & df[‘b’]>5 ] |
|  | Filter (or) | filter(df, a<2 | b>5) | df[ df[‘a’]<2 | df[‘b’]>5 ] |
|  | Filter (membership) | filter(df, a %in% c(3,4)) | df[ df[‘a’].isin([3,4]) ] |
|  | Filter (negative memb.) | filter(df, !(a %in% c(3,4)) | df[ ~df[‘a’].isin([3,4]) ] |
|  | Filter (string) | filter( df, grepl(‘foo’,a) ) | df[ df[‘a’].str.contains(‘foo’) ] |
|  | Select | select(df, b, c) | df[[‘b’,’c’]] |
|  | Select range | select(df, b:d) | df.loc[:, ‘b’:’d’] |
|  | Select by name | select(df, contains(‘foo’))  select(df, starts\_with(‘a’)) |  |
| Analysis | Group by | group\_by(df, a) | df.groupby(‘a’) |
|  | Window functions | mutate(df, new=mean(a)) | df[‘a’].apply(mean) |
|  | Window, custom fn | mutate(df, new=my\_fun(a,b) )  df$new=Vectorize(my\_fun)(df$a,df$b) | df[‘new’]=np.vectorize(my\_fun)(  df[‘a’],df[‘b’]) |
|  | Window, conditional | mutate(df, new=ifelse( a>0, 1, 0 )) | df[‘new’]=np.where(df[‘a’]>0, 1, 0) |
|  | Window (group-identical attribute) |  | df[‘new’]=df.groupby(‘a’)[‘b’]  .transform(mean) |
|  | Summary functions | summarise(df, new= mean(a))  df$new = sapply(df$a,mean) | df.groupby(‘a’)[‘b’]  .agg(my\_fun) |
|  | Filter by on group-summary attrib. |  | df.groupby(‘a’).filter(lambda x:  len(x[‘b’]) >2 )  mean(x[‘b’]) >0 ) |
|  |  |  |  |
| Read files |  | xlsx |  |
|  |  | feather |  |
| Manipulate |  | dplyr |  |
|  |  | tidyr |  |
|  |  | data.table |  |
| Formats | Strings | stringr |  |
|  | Dates | lubridate |  |
|  | Time series | zoo, xts |  |
|  | Spatial data | sp, maptools, maps, ggmaps |  |
| Cleaning | Imputation | MissForest, MissMDA |  |
|  | Outlier detection | Outliers, EVIR |  |
|  | Feature selection | Features, RRF |  |
|  | Dimension reduction | FactoMineR, CCP |  |
| Visualization |  | shiny |  |
|  |  | ggplot2 |  |
|  |  | ggvis |  |
|  |  |  |  |
| Modeling | Anova, regression | car |  |
|  | generalized additive models | mgcv |  |
|  | linear mixed effects models | lme4 |  |
|  | non-linear mixed effects models | nlme |  |
|  | random forests | randomForest |  |
|  | categorical data tests | vcd |  |
|  | lasso and elastic-net regression | glmnet |  |
|  | survival analysis | survival, Basta |  |
|  | regression and classification models | caret, BigRF |  |
|  | time series forecast | forecast, LTSA, prophet |  |
|  | clustering | CBA, RankCluster |  |
|  | ordinal regression | Rminer, CoreLearn |  |
|  | Deep learning | tensorflow |  |
| Validation | General models | LSMeans, Comparison |  |
|  | Regression | RegTest, ACD |  |
|  | Classification | BinomTools, DAIM |  |
|  | Clustering | ClustEval, SigClust |  |
|  | ROC | PROC, TimeROC |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Create new repository**

*on github.com: create a new repository*

*in terminal: cd to folder*

git init

git add –A

git commit –m “first commit”

git remote add origin https://github.com/elliotdl/REPO\_NAME\_HERE.git

git push – origin master

**Update repository**

*create notepad file:*

git add –A

git commit –m “blah”

git push

*save as “github.bat”*