Input files

RaCInG needs at least four input files to function. Two of these files define the quantifications, and two others define the compatibility matrices. A fifth optional file can be provided: the sign file for ligand receptor interactions. In this document, we will provide the structure of the csv input files.

Cell-type compatibility with ligand type (L-matrix)

	Ligand 1 name	**Ligand 2 name**	**Ligand 3 name**	…
Cell type 1	0	1	1	…
Cell type 2	1	0	0	…
Cell type 3	0	1	0	…
⋮	⋮	⋮	⋮	

Notes:

- Values should be 1 (secretion possible) or 0 (secretion impossible).

Cell-type compatibility with receptor type (R-matirx)

	Receptor 1	**Receptor 2**	**Receptor 3**	…
Cell type 1	0	1	1	…
Cell type 2	1	0	0	…
Cell type 3	0	1	0	…
⋮	⋮	⋮	⋮	

Notes:

- Cell type identifiers should be the same as in the previous csv-file.
- Values should be 1 (secretion possible) or 0 (secretion impossible).

Cell-type quantification (C-matrix)

	Cell type 1	**Cell type 2**	**Cell type 3**	…
Patient 1	0.143527	0.2314527	0.135624	…
Patient 2	0.1678542	0.3214521	0.0953214	…
Patient 3	0.1137845	0.264512	0.0512389	…
⋮	⋮	⋮	⋮	

Notes:

- Patient rows should sum up to one. Code ensure that even if this is not the case, the final matrix after reading it in does have this property. Input values are then transformed such that rows sum up to one, and relative cell type value compared with the row sum remains the same.
- Cell type identifiers should be the same as in the previous csv-files.

Interaction weights (LR-matrix)

	Lig_Rec 1	**Lig_Rec 2**	**Lig_Rec 3**	…

Patient 1	5.3975	3.4678	10.3457	…
Patient 2	0.7913	5.1379	1.3974	…
Patient 3	2.39456	1.37945	0.12345	…
⋮	⋮	⋮	⋮	

Notes:

- Column identifiers should consist of a ligand identifier, followed by an underscore and a receptor identifier. For example, “ITGB2_ICAM2”.
- Patient identifiers should be the same as in the previous csv-file.
- Ligand and receptor identifiers should be the same as in the previous csv-files.

Sign interactions (optional)

	Receptor 1	**Receptor 2**	**Receptor 3**	…
Ligand 1	0	-1	1	…
Ligand 2	1	0	-1	…
Ligand 3	0	-1	0	…
⋮	⋮	⋮	⋮	

Notes:

- Ligand identifiers should be the same as in the previous csv-files.
- Receptor identifiers should be the same as in the previous csv-files.
- Values should be 1 (promoting), -1 (inhibiting) or 0 (unknown/nonexistent).

Property .csv files

	Property 1	**Property 2**	**Property 3**	…
Patient 1	0.5348	2.13458	1.48632	…
Patient 2	1.02365	7.26589	0.314578	…
Patient 3	2.31546	10.1245	1.23564	…
⋮	⋮	⋮	⋮	

Notes:

- Patient identifiers are the same as in the previous files.
- Properties consist of cell types separated by underscores.
- Only one type of property per file (e.g. only wedges).

Feature .txt files (triangles, wedges)

```
Triangle_type

Weight_type,number_patients,number_cells,number_graphs,average_deg

Patient_number,number_cells,average_deg

['cell1' 'cell2' 'cell3' …]

Count,total_number_feature_average,total_number_feature_std

Composition - Average:
```

```
0,0,0,number_this_subfeature_average
0,0,1,number_this_subfeature_average
#Etc until all sub features are done.
Composition – Std:
0,0,0,number_this_subfeature_std
0,0,1,number_this_subfeature_std
#Etc until all sub features are done.
Patient_number,number_cells,average_deg
['cell1' 'cell2' 'cell3' …]
Count,total_number_feature_average,total_number_feature_std
Composition – Average:
0,0,0,number_this_subfeature_average
0,0,1,number_this_subfeature_average
#Etc until all sub features are done.
Composition – Std:
0,0,0,number_this_subfeature_std
0,0,1,number_this_subfeature_std
#Etc until all sub features are done.
#Etc until all patients are done.
```

Notes:

- Number_cells is the number of cells per generated graph
- The count line gives the total count of the feature (not subdivided into specific cell types).
- In the composition lines the numbers correspond to the cell types in the cell-type array.
- This is the structure used for only wedges, trust triangles and cycle triangles.
- An enter at the end of the file is required.
- Filename: $cancer_$feature_$average(_norm).txt

Feature .txt files (Direct communication)

```
D
Weight_type,number_patients,number_cells,number_graphs,average_deg
Patient_number,number_cells,average_deg
['cell1' 'cell2' 'cell3' …]
Count,total_number_feature_average,total_number_feature_std
Composition – Average:
```

```
0,0,number_this_subfeature_average
0,1,number_this_subfeature_average
#Etc until all sub features are done.
Composition – Std:
0,0, number_this_subfeature_std
0,1,number_this_subfeature_std
#Etc until all sub features are done.
Patient_number,number_cells,average_deg
['cell1' 'cell2' 'cell3' …]
Count,total_number_feature_average,total_number_feature_std
Composition – Average:
0,0,number_this_subfeature_average
0,1,number_this_subfeature_average
#Etc until all sub features are done.
Composition – Std:
0,0,number_this_subfeature_std
0,1,number_this_subfeature_std
#Etc until all sub features are done.
#Etc until all patients are done.
```

Notes:

- Number_cells is the number of cells per generated graph
- The count line gives the total count of the feature (not subdivided into specific cell types).
- In the composition lines the numbers correspond to the cell types in the cell-type array.
- This is the structure used for direct communication only.
- An enter at the end of the file is required.
- Filename: $cancer_D_$average(_norm).txt

Feature .txt files (GSCC)

```
GSCC
Weight_type,number_patients,number_cells,number_graphs,average_deg
Patient_number,number_cells,average_deg
['cell1' 'cell2' 'cell3' …]
Count,total_number_feature_average,total_number_feature_std
Composition – Average:
```

```
0,number_this_subfeature_average

1,number_this_subfeature_average

#Etc until all sub features are done.

Composition – Std:

0, number_this_subfeature_std

1,number_this_subfeature_std

#Etc until all sub features are done.

Patient_number,number_cells,average_deg

['cell1' 'cell2' 'cell3' …]

Count,total_number_feature_average,total_number_feature_std

Composition – Average:

0,number_this_subfeature_average

1,number_this_subfeature_average

#Etc until all sub features are done.

Composition – Std:

0,number_this_subfeature_std

1,number_this_subfeature_std

#Etc until all sub features are done.

#Etc until all patients are done.
```

Notes:

- Number_cells is the number of cells per generated graph
- The count line gives the total count of the feature (not subdivided into specific cell types).
- In the composition lines the numbers correspond to the cell types in the cell-type array.
- This is the structure used for direct communication only.
- An enter at the end of the file is required.
- Filename: $cancer_GSCC_$average(_norm).txt

Kernel files

Kernels are stored as .npz files with the name "kernel_$cancer.npz". The .npz file always contains two Numpy arrays. Both are 3-tensors with kernel values of a given patient on the first and second axis (indexed with cell-type numbers). The third axis indexes the number of the patient.

Kernels van be loaded in Python by running

```
Import numpy as np

kernelList = np.load("kernel_$cancer.npz")
```

Each kernel file contains two kernels derived from the same dataset. One kernel that has been computed normally, and one that is computed after making the LR-input matrix uniform. The latter kernel is needed for the normalization procedure. The two kernels are named "`kernel`" and "`unifKernel`" in the npz object, respectively. You can extract them as follows.

```
Kernel = kernelList["kernel"]

Uniform_kernel = kernelList["unifKernel"]
```

As an example, to extract the normalized direct communication feature of patient 0 from cell-type 1 to cell-type 2 one would need to run:

```
Kernel[1,2,0] / Uniform_kernel[1,2,0]
```