scanpy.tl.filter_rank_genes_groups

`scanpy.tl.filter_rank_genes_groups`(*adata, key=None, groupby=None, use_raw=None, key_added='rank_genes_groups_filtered', min_in_group_fraction=0.25, min_fold_change=1, max_out_group_fraction=0.5, compare_abs=False*)

Filters out genes based on log fold change and fraction of genes expressing the gene within and outside the `groupby` categories.

See **`rank_genes_groups()`**.

Results are stored in `adata.uns[key_added]` (default: 'rank_genes_groups_filtered').

To preserve the original structure of adata.uns['rank_genes_groups'], filtered genes are set to `NaN`.

Parameters:

- **adata** : **`AnnData`**
- **key** : default: `None`
- **groupby** : default: `None`
- **use_raw** : default: `None`
- **key_added** : default: `'rank_genes_groups_filtered'`
- **min_in_group_fraction** : default: `0.25`
- **min_fold_change** : default: `1`
- **max_out_group_fraction** : default: `0.5`
- **compare_abs** : default: `False`

  If `True`, compare absolute values of log fold change with `min_fold_change`.

Return type: **`None`**

Returns:

: Same output as **`scanpy.tl.rank_genes_groups()`** but with filtered genes names set to `nan`

Examples

```
>>> import scanpy as sc
>>> adata = sc.datasets.pbmc68k_reduced()
>>> sc.tl.rank_genes_groups(adata, 'bulk_labels', method='wilcoxon')
>>> sc.tl.filter_rank_genes_groups(adata, min_fold_change=3)
>>> # visualize results
>>> sc.pl.rank_genes_groups(adata, key='rank_genes_groups_filtered')
>>> # visualize results using dotplot
>>> sc.pl.rank_genes_groups_dotplot(adata, key='rank_genes_groups_filtered')
```